Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät der Universität Bielefeld

BRIX$_2$
A Versatile Toolkit for Rapid Prototyping and Education in Ubiquitous Computing

Sebastian Zehe
August 2016

Supervisors:
Dr. rer. nat. Thomas Hermann
Ambient Intelligence Group

Prof. Dr.-Ing. Ulrich Rückert
Cognitronics and Sensor Systems Group

"It doesn't stop being magic just because you know how it works."
– Terry Pratchett, The Wee Free Men

Contents

1 Introduction

The constant progress of technological development has always challenged us humans to rethink and reshape the way we face and utilize new ideas, concepts and devices. A device which has rapidly developed and radically changed our everyday lives in the past decades is the computer. Already in the early 1950's, where Howard H. Aiken, designer of the Harvard Mark I, arguably one of the earliest general purpose computers, expressed the fallacy of the original growth predictions:

"Originally one thought that if there were a half dozen large computers in this country, hidden away in research laboratories, this would take care of all requirements we had throughout the country." [1]

In the late 1970's, the personal computer became affordable for household use and soon, millions of units had been sold. Today, Mark Weisers' prediction from the mid 1990's, introducing the 'third wave in computing', has become part of our reality:

"Ubiquitous computing names the third wave in computing, [...] the age of calm technology, when technology recedes into the background of our lives." [2]

Computers are no longer just grey boxes sitting on our desktops, but have been broken up into smaller devices and blend in with our environment. Additional to the traditional computers, tablets and smartphones in the focus of our attention, more and more invisible computers are becoming a part of our lives. They constitute critical parts of everyday technology such as buildings, cars, and even devices we wear on our bodies.

Yet ubiquitous computing as a discipline holds many challenges to researchers, who build smart, interactive devices, test concepts in human/machine interaction and explore ethical, social and legal implications of computers that are woven into our surroundings. Here, they often choose practical, hands-on approaches, functional prototypes that allow them to experiment with new concepts in an early design stage. Demonstrators help to present the work to fellow researchers, industry partners and the public. As illustrative examples, prototypes and demonstrators help students understand ubiquitous computing research and raise their interest to participate in this field.

While researchers implement their ideas, they integrate programmable microcontrollers, sensors and actuators in everyday objects, rooms and clothes, thereby turning them into interconnected, interactive devices. To avoid to reinvent the wheel, especially in an early design phase where prototypes are rapidly iterated, the use of toolkits instead of custom devices is necessary for working quickly and efficiently. Today, a wide variety of toolkits and platforms for different prototyping purposes are available. Many of them have originally emerged from research projects and are now in the hands of engineers, scientists, students, artists and tinkerers. But among all these platforms, is there one that stands out as *the* prototyping toolkit for ubiquitous computing?

When we try to answer this question, we first face a diverse ecosystem of combined hardware and software prototyping platforms for applications e.g. ubiquitous computing including wearable electronics, ambient intelligence and smart devices. These platforms can be arranged on a continuum from general purpose to specialized devices. We consider general purpose toolkits to be microcontroller development boards that usually have a minimum set of features such as a microcontroller, programming interface and electrical connectors. They can be adapted to virtually any application by adding the demanded functionality. This, however, requires experienced developers who have detailed knowledge about the added components. Other platforms provide additional components as building blocks that can simply be connected to form an application. Regarding both approaches, we find that there is always a compromise between versatility and ease of use.

Toolkits can be specialized in two different ways: on the one hand they can be variants of general purpose devices that are optimized towards a certain purpose. The Arduino Flora for example is a smaller, sewable version of the Arduino Leonardo, optimized for electronic textiles. On the other hand, they can be designed specifically to support a certain range of applications, such as inertial measurement units for measuring linear accelerations and angular velocity of rigid bodies. In this case, a set of components required for this application such as a microcontroller, motion sensors and data storage are integrated. Wearable devices add the challenge of interacting with the human body in a natural way, so in addition they have to be optimized for low weight and compactness.

The advantage of general purpose devices is their adaptability to a wide variety of applications. However, these modifications require experience with electronics if the platform is initially equipped with only minimal features. In the case of a building block type platform, adding extra features is easier, but usually results in a bulky device, impractical for applications like wearable electronics. Specialized platforms perform well within the intended range of applications, but it is not always possible, let alone easy to add additional features like wireless communication.

Building and testing prototypes is an essential part of our work as researchers and developers and we rely on a set of different platforms to face the variety of challenges that ubiquitous computing applications hold for us. Because prototyping scenarios in our field are so diverse, we lack an appropriate toolkit that suits our full spectrum of needs.

Goals

In this project we aim to overcome the above stated problem by building a versatile and unified platform that can be adapted to the requirements of a wide range of applications in ubiquitous computing and used in place of a number of different specialized platforms. At the same time we are going to design the platform in a way that it can be used for prototyping and building applications by developers with no or little experience in electronics and programming. This way we also create a valuable tool to teach topics like for example physical computing, electronics and sensor systems. On the one hand, our scalable and adaptable platform is expected to allow a low-threshold introduction to the topics and on the other hand to also suit experienced students by providing a wide range of possibilities and challenges.

Methods

In order to achieve these goals, we will first identify fields of research that represent the different aspects of ubiquitous computing applications in general and for which specialized platforms and toolkits already exist. By carefully analyzing those existing platforms we are going to be able to extract a set of primary properties and features. In order to integrate the aspects of usability and scalability, we mainly focus our attention on platforms that already implement these concepts. As a result we can then refine this set of features and integrate them into a single, unified toolkit.

Expected Outcome

In the first place, we postulate that a platform, which unifies the principal properties and functionalities of a set of specialized platforms is able to substitute those platforms in prototyping and teaching scenarios and as a result meets our demands in ubiquitous computing applications. Furthermore, we suppose that a prototyping toolkit with minimum threshold can be used by people with little or no experience and will empower them to create their own ubiquitous computing applications without the help of an expert. Finally we expect that a modular platform with optional abstraction layers in software and hardware represents an efficient tool for beginners as well as expert users.

Thesis Structure

After identifying five different fields of application that resemble the range of scenarios in ubiquitous computing, we analyze almost 30 devices and toolkits from these fields in Chapter 2. The result is a 'best of' list of desirable properties, balanced towards our requirements as described in Chapter 3. In the subsequent design and evaluation process, we shape a concept for our toolkit:
A module with integrated microcontroller, motion sensor, wireless interface and battery forms the base of our platform. Electrical connectors allow users to attach extension modules, which contain for example sensors or actuators, to the base. This way we provide a core set of crucial features in a compact device, which can easily be extended with additional components and functions to adapt to a given application.

We utilized a custom development platform to efficiently test and compare different concepts and components to in order to refine our platform design. Specifically, we looked into different options for computing, sensing and wireless communication. Based upon the concept and the results from this technology evaluation, we developed a suitable physical form for the realization of our system, which we describe in Chapter 4. Here, we first focus on mechanical, electrical and usability considerations for such an extensible system, before we proceed to technical details. The result is the BRIX$_2$ prototyping toolkit. A base module with a compact size of $50\times30\times15$ mm contains two microcontrollers, a 9-axis Inertial Measurement Unit (IMU), a wireless transceiver and a battery. Three electrical connectors on top of the module allow users to easily connect extension modules with an average size of around $15\times30\times15$ mm. For a solid mechanical attachment of extension modules to the base module, we implemented a friction-based connection based on the LEGO® system.

Our platform is designed to be compatible to the widely used, open source Arduino platform, which consists of a microcontroller board and an Integrated Development Environment (IDE). This way, we can use the well-proven IDE, programming toolchain, libraries and software examples for BRIX$_2$. A custom library for all components of our system allows users to tap the full potential of our hardware. To add to the capabilities of the BRIX$_2$ base module, we developed and implemented an exemplary set of extension modules which we present in Chapter 5. These additional sensors, actuators and communication interfaces allow BRIX$_2$ to adapt to applications especially in ubiquitous computing. However, our system is not limited to this particular scope of functions defined by the currently existing extensions. More and different extension modules can and have been designed and built on demand.

After we implemented the $BRIX_2$ platform and extension modules, we performed tests of all key components to verify that they function as we expected. In Chapter 6 we summarize the technical specifications of those components before we present the results of experiments based on real-life scenarios. Besides the technical evaluation, we were also interested to know how $BRIX_2$ facilitated actual applications as well as in the users' experience of the work with our platform. In the final part of Chapter 6 we take a closer look at selected $BRIX_2$ applications that were implemented by students and fellow researchers and point out which particular features of our platform. In the third part of Chapter 6 we present the results of a qualitative survey conducted among users of the $BRIX_2$ platform. Through a questionnaire, we gathered their opinions on features of our system, asked them to share their experiences while working with our platform and their wishes and suggestions for future revisions.

In Chapter 7 we summarize which contributions the $BRIX_2$ platform and its development process made not only to research in the field of ubiquitous computing but also to teaching electronics and sensor systems. Finally we provide an outlook on potential future developments of our platform regarding technical and conceptual aspects.

2 Related Work: A Survey of Existing Platforms

As a foundation of the design process that finally leads us to the $BRIX_2$ toolkit, we survey existing platforms, built as tools for different fields which lie within or overlap with the field of ubiquitous computing that our research focuses on.
When we look at typical ubiquitous computing scenarios, we find a lot of applications that are based on standard Personal Computers (PCs) or smartphones. This type of hardware is easily available, equipped with powerful processors, a display, keyboard, mouse and touchscreen input and network access. If the requirements of an application exceed the scope of these devices, the demand for custom hardware arises. Those applications can for example be: Smart objects, equipped with sensors and actuators that allow to interact with them. Intelligent textiles that are aware of their surroundings and can assist the wearer in everyday life. Grids of sensors distributed in a building or a room that are interconnected by a wireless link and allow the space to sense different modalities. Sensors attached to a human body that record motion patterns and trigger external applications. The combination of PC-based application and custom sensors like i.e. paper-based Force-Sensing Resistors (FSRs). [3]
There are numerous platforms that help researchers and students to implement these applications: Microcontroller platforms are used by electronics developers to build standalone applications involving sensors as well as actuators and can serve as a link between such hardware and a PC. Physical computing toolkits allow to experiment with different sensors and actuators without requiring skills in soldering, electronics or programming. Wireless sensor nodes can be deployed to sense different modalities in multiple spots at the same time. Compact inertial measurement platforms can record and stream motion data. Wearable electronics platforms can be integrated into textiles or worn as accessories.
However, none of the devices we found in these categories is capable to meet our demands as a prototyping and teaching tool in all of the above applications. As a result, researchers in ubiquitous computing were forced to use a variety of tools for different applications which requires an extensive scope of skills in programming and electronic hardware development, leads to longer development times as well as additional costs and reduces the maintainability of existing projects.

The first step in our approach to unify the properties and features of many different prototyping platforms is to analyze existing devices from the fields we mentioned earlier. In this chapter, we introduce each of these fields before we present a selection of platforms that were designed to support typical applications in each field. We closely examine and compare technical aspects of those devices to work out characteristic features that contribute to their use a ubiquitous computing platform.
The first type of platform we introduce is the microcontroller platform. Devices we present afterwards, like wireless sensor nodes or inertial measurement units are specializations of a microcontroller board, equipped with further functionalities and features. Moving towards more specialized platforms, we subsequently regard toolkits for physical computing, Wireless Sensor Network (WSN), inertial motion measurement and wearable electronics. After that, we present the original BRIX system that we designed in 2010 and that serves as a major inspiration for the proposed BRIX$_2$ system.
It is important to note that this survey is already outdated at the moment of publication, because technology advances rapidly. Companies and open source developers constantly present novel platforms, which clearly indicates a demand for these devices. However, some aspects of the following chapter are going to be still valid for some more years and can be regarded as a base for future surveys and market analysis.

2.1 Microcontroller Platforms

Microcontroller platforms are Printed Circuit Boards (PCBs) including a microcontroller as well as all necessary components like power management, communication interfaces and Input/Output (I/O) connectors. They allow users to learn about microcontrollers or start their electronics projects without the requirement of soldering skills or even detailed knowledge about electronics. Some platforms have become so cheap and small that they are not only used in the design and prototyping process, but often left in the finished project instead of developing custom electronics in a further iteration. Certain microcontroller platforms have become so popular that they are a de-facto standard with a solid, constantly growing user base. This community supports beginners as well as professionals by sharing knowledge, ideas and projects. Before we present selected microcontroller platforms in detail, we start with a short introduction of the core component, the microcontroller.

2.1.1 A Brief Introduction to Microcontrollers

A basic component for applications that connect the virtual world (Software) with the physical world (Hardware) is the microcontroller, a fully functional computer on a single Integrated Circuit (IC). They contain a Central Processing Unit (CPU), Random Access Memory (RAM), a non-volatile program memory and I/O pins. These pins allow the software, which runs on the microcontroller to interact with other electronic components and devices outside the microcontroller. I/O pins can usually be configured as inputs, for reading external signals or as outputs to generate an electric signal. Some I/O pins can not only read digital signals, but also read analog voltages using an internal Analog to Digital Converter (ADC). I/O pins can also have special, higher level capabilities like data transfer protocols implemented in hardware, for example dedicated pins for RS232 [4] or Inter-Integrated Circuit (I2C) [5] bus communication.

Another crucial aspect of a microcontroller is the programming interface. In order to make the microcontroller do anything, it needs a program, called *firmware*, which is usually written and compiled on a PC and then transfered into the program memory of the microcontroller through the programming interface. There are different techniques and standards for such an interface. Common implementations are serial interfaces like In System Programming (ISP) [6] or the Joint Test Action Group (JTAG) [7] standard to directly access the memory of the device. This requires a special hardware that connects to the PC, which is referred to as a programming adapter. In order to eliminate the need for such an adapter, microcontrollers can be equipped with a bootloader, a small program which is either installed as a factory default or later by the user. It is executed on reset of the controller and allows firmware updates via an alternative communication interface like RS232 or Universal Serial Bus (USB) [8]. This means that firmware can be transfered to the microcontroller using standard interfaces that every PC already implements and does not require any special additional hardware except a standard cable. After the bootloader has received the firmware and has written it to the internal program memory, it terminates itself and runs the firmware that was uploaded.

2.1.2 Common Microcontroller Parameters

Today, a wide variety of microcontrollers is available so that during design time, specific devices can be selected to precisely match the requirements of a given application. Common parameters to consider are the number of I/O pins, which also influence the physical dimensions if the chip, the clock speed, architecture (typically 8, 16 or 32 bit), power consumption and peripheral features like support for certain communication protocols, timers, ADCs, Pulse Width Modulation (PWM) generators or Floating Point Units (FPUs).

2.1.3 Microcontroller Communities

Even though a lot of specialized microcontrollers are available on the market, there are a few general purpose product families that are commonly used among professionals and hobbyists. Tremendous user communities have developed around products like the Atmel AVR series [9] or the Microchip PIC series [10]. In some cases, these communities only focus on a limited number of chips, for example the Atmel ATmega168/328 and the Atmel ATmega32U4. The reason for this is, besides the low costs and low complexity of the devices, is the availability of those popular chips integrated into development boards. Communities share their knowledge in forums, chats, blogs, magazines and dedicated project sharing sites like *MAKE: projects*[1] or *Hackaday Projects*[2]. Forums usually have different sub-boards dedicated to various aspects of a platform, microcontroller or topic (interfacing sensors, communication, controlling actuators, general electronics, etc.). It is not unusual that popular microcontroller community forums achieve thousands of posts containing questions and answers every day.

[1]http://makezine.com/projects
[2]https://hackaday.io/projects

2.1.4 Microcontroller Platforms: Development Boards

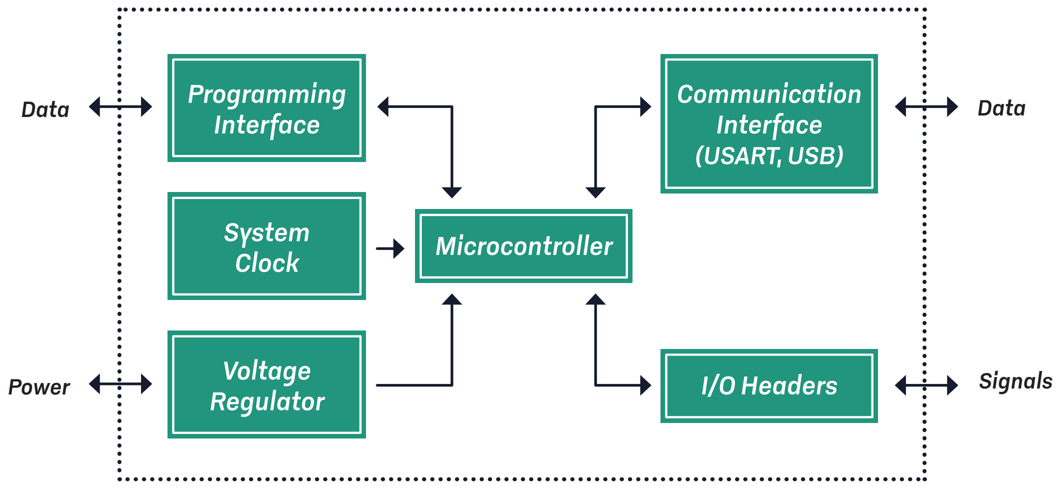


Figure 2.1: Diagram of a typical microcontroller development board.

In order to actually use a microcontroller, a number of additional electronic components are necessary, see Figure 2.1. First of all a power supply, often an external clock signal generated by an oscillator and an interface to transfer programs to the controller. Second, in order to connect the microcontroller to any other device, connection headers or solder pads for all or certain I/O pins are required. A microcontroller platform, or development board, integrates all those components. Most development boards are bare PCBs populated with a microcontroller, peripheral components, connection headers for I/O pins, a power supply connector and a PC compatible communication interface. Some development boards also include components like buttons, switches, Light Emitting Diodes (LEDs) or Liquid Crystal Displays (LCDs) that allow interaction with the microcontroller without connecting external components. This is convenient for users who are unfamiliar with microcontrollers, because they can focus on writing the firmware for existing, working hardware first, before altering and extending the hardware itself. It also reduces the potential for errors and thereby the level of frustration. In our survey, we summarized and compared the technical specifications of a total of six different microcontroller platforms. In the following, we introduce three of them in greater detail before we present a comparison of all six development boards.

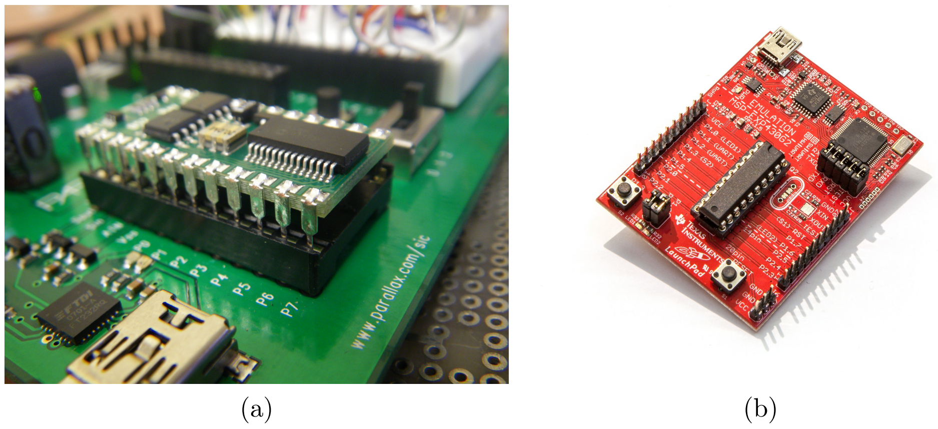


Figure 2.2: The Parallax Basic Stamp 2 on a carrier board ((a), photo by Marcin1988, GFDL) and the Texas Instruments MSP430 Launchpad (b).

2.1.5 A Pioneering Microcontroller Platform: The Basic Stamp

The Basic Stamp [11], see Figure 2.2 (a), is one of the first microcontroller platforms that gained a wider popularity in the late 1990s and the early 2000s. It is based an 8 bit microcontroller which runs an interpreter that executes firmware stored on an external Electrically Erasable Programmable Read Only Memory (EEPROM). The Basic Stamp is designed to be easy-to-use, sacrificing performance and complex compiler toolchains for a simple, interpreted programming language called PBASIC, a BASIC dialect. This made the Basic Stamp interesting for educational purposes as well as hobbyists and artists, because it allows users to implement microcontroller applications without expert knowledge. A flourishing community of users created documentation, examples and tutorials, increasing the platform's popularity. Although the Basic Stamp is still being developed and new boards are released, its popularity among professionals and hobbyists has decreased due to the rise of alternative platforms.

2.1.6 An Inexpensive Microcontroller Board: The MSP430 Launchpad

The Texas Instruments MSP Launchpad, see Figure 2.2 (b), is a development board based on the MSP430G2553 microcontroller. It was first released in 2010 for a price as low as 5 USD including shipping. [12] The MSP430G2553 is a 16 bit microcontroller optimized for low cost and low energy consumption. It runs at a clock speed of 16 MHz, has 16 kB of flash, 512 bytes of RAM and features hardware I2C, Serial Peripheral Interface (SPI) and Universal Asynchronous Receiver/Transmitter (UART). [13] An on-board emulator allows programming the microcontroller via USB and debugging it via JTAG. The Launchpad can be equipped with different MSP430 microcontrollers on the same board. All pins are broken out to pin headers on opposing sides of the PCB. Texas Instruments offers a number of add-on boards, called *BoosterPacks*, which can be stacked onto the Launchpad and extend its scope of functions. [14] The company also encourages users to design their own BoosterPacks. [15] TI originally offered two programming environments, Code Composer Studio 4 and the IAR Workbench Kickstart. [12] Later, the community driven Energia IDE was officially supported by Texas Instruments. Energia is based on the Arduino IDE, see Section 2.1.7, and tries to make the Launchpad as easy-to-use as an Arduino.[3] The documentation of the Launchpad is comprehensive and detailed. A user guide provides background information as well as schematics and an official wiki and several user forums provide technical support during troubleshooting and offer tutorials for a lot of different topics.

2.1.7 De-Facto Standard: The Arduino

The Arduino platform is currently the most influential and popular microcontroller platform. Arduino not only names a microcontroller platform but also an IDE for firmware development and a community of users. Because of its significance, we describe the Arduino toolkit in greater detail than other microcontroller platforms in this section.

A Short Arduino History

Arduino started as Massimo Banzi's fork of the Wiring project, which was originally developed by Hernando Barragán in 2003. [16] The goal of Wiring had been to make electronic prototyping tools, which by that time were mostly targeted on electrical engineers, available to students without extensive experience in electronics. The Wiring IDE, which is still used by Arduino today, was based on Processing by Casey Reas and Benjamin Fry. [17]. The Wiring microcontroller board was build around the Atmel ATmega128. Together with the IDE, it represented a powerful learning and prototyping kit that was used in multiple workshops and classes in 2004 and 2005. [18] In 2005, Banzi developed a board based on the Atmel ATmega8 as a low cost alternative to the Wiring board. [4] From there on, Banzi et al. successfully continued the project as Arduino, along with a fork of the Wiring IDE. Today, Arduino has become a very popular and influential physical computing platform.

[3]http://energia.nu/
[4]https://www.flickr.com/photos/mbanzi/172472136/in/album-72157594173657338/

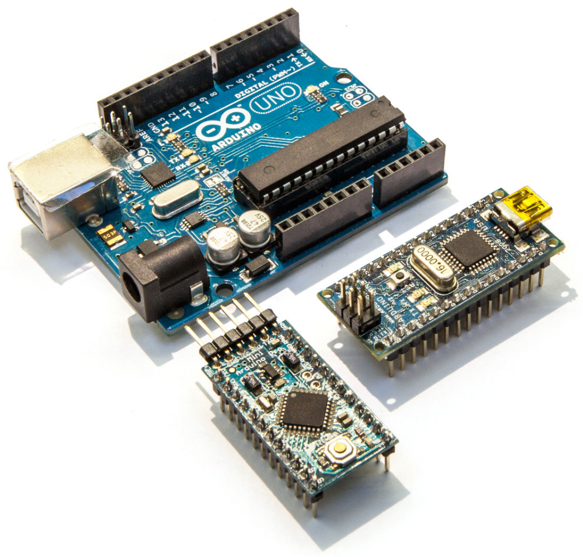


Figure 2.3: The Arduino Uno (top) along with an Arduino Nano and Mini Pro.

The Arduino Hardware

The first Arduino board was based on the Atmel ATmega8 microcontroller, but designs later changed to an ATmega168/328. This series of 8 bit controllers usually runs at 16 MHz, has 32 kB of flash memory that stores the firmware, 2 kB Static RAM (SRAM) and 1 kB Electrically Erasable Programmable Read-Only Memory (EEPROM). Of the 23 General Purpose I/Os (GPIOs), 8 are connected to an internal 10 bit ADC. The controller also features I2C, UART and SPI [19] in hardware. Most Arduino boards based on this microcontroller, like for example the Arduino Uno, see Figure 2.3, also contain a USB/Serial converter IC such as the FTDI FT232RL [20] to allow connecting the Arduino via USB instead of a serial port. Female pin headers provide easy access to GPIOs and power pins of the microcontroller. Besides the 8 bit Arduinos, there are also boards based on more advanced microcontrollers like the Atmel SAM3X8E ARM Cortex-M3 [21]. Yet the Arduino series offers the most variation in the 8 bit segment. Models like the Arduino Nano (45$\times$18 mm) or the Arduino Micro (48$\times$18 mm) represent a more compact alternative to the classic, 68.6$\times$53.4 mm board. The LilyPad Arduino series for example is optimized for wearables and electronic textiles, see Section 2.5. Its round PCB features contacts with bigger holes that can be sewn to fabric using conductive thread, creating a textile circuit board. The Arduino Mega series is based on an Atmel ATmega2560 microcontroller that offers more memory (256 kB flash) and a higher number of GPIOs (54) than the classical variants.

The Arduino IDE

The Arduino IDE is a crucial component of the toolkit and is designed to work with all available Arduino Boards. A firmware program, called *sketch* can be written in C or C++ directly inside the editor of the IDE and with a click of a single button, it is compiled and uploaded to the target platform. The IDE is operating system independent and available for Windows, Linux and OSX. At installation, the whole required toolchain is automatically set up and works out of the box. The Arduino IDE comes with more than 50 example sketches that serve as a starting point for basic applications like blinking an LED to complex tasks like setting up an embedded web server. Even fundamentals of the programming language like loops and conditions are covered by examples. Additionally, a language reference with examples, libraries and tutorials can be found on the Arduino Website.[21] An open source license allows all users to actively take part in developing the Arduino IDE as well as additional libraries that extend the functionality of the existing system. Thereby the product does not solely depend on the official developers, but is in fact in the hands of users. Compared with other IDEs like Eclipse [22] or the Atmel Studio [23], Arduino is rather simplistic and advanced developers might turn to alternatives. However, by reducing the scope of functions to a minimum, the Arduino IDE does not confuse beginners and keeps them focused.

Arduino Extensions: Shields

An Arduino alone is relatively limited in terms of possible applications. Only by connecting external hardware does the Arduino unfold its full potential. External components can either be connected to the pin headers of the Arduino using cables or in the form of readily available shields. Arduino shields have the same form-factor as the original Arduino board and are stacked on top of the Arduino. Many shields have female pin headers on top, so multiple shields can be stacked. The complexity of shields ranges from a simple motor driver to voice recognition shields or Local Area Network (LAN) and Wireless LAN (WLAN) implementations. Apart from the official Arduino shields, there are many third-party shields on the market.

The Arduino Community

The popularity of the platform gathered a massive community that exchanges knowledge through wikis and forums. Thousands of third-party tutorials and hundreds of libraries, each including their own examples, solve virtually any problem Arduino users could face. The official Arduino forum alone has over 2 million posts in over 270000 topics by around 270000 members (as of May 2015) [24]. Questions that have not been answered before will likely be answered by other users within minutes or hours. Topics are not only related to programming but also to hardware setups, for example how to connect certain devices to an Arduino.

The Impact of Arduino

In 2014, Banzi estimated that around 1.2 million Arduino boards were used around the world. [25] Apart from the original Arduino brand boards, there are a lot of clones[5] and "Arduino compatibles" on the market. [26] Several China based companies even offer Arduino boards for prices under 3.0 USD [27] which makes the platform affordable for almost anybody. Creating an Arduino compatible product means directly profiting from the giant community, documentation and knowledge base. Arduino has drastically reduced the effort that is necessary to start working with electronics. This makes it a valuable tool for people who want to implement a physical computing application without prior knowledge, but also students who intend to learn about embedded systems and electronics. [28] Arduino also became the electronics platform of the maker movement [29] which itself spawned a lot of startup companies and actually became a market. [30]

[5]http://www.Arduino.cc/en/Products/Counterfeit

2.1.8 Characteristics of Different Microcontroller Platforms

In this survery, we analyzed the technical specifications of six different popular microcontroller development boards, see Table 2.1. Of course this is only a small sample out of hundreds of boards on the market. We focused especially on platforms that target non-expert users, since that is also the same group of users we aim to design the proposed BRIX$_2$ system for.

Electrical Properties

Most platforms in our survey are based on 8 bit microcontrollers running on clock speeds around 20 MHz. Flash and RAM are usually small, but still sufficient for many applications. Multi channel ADCs with a resolution of at least 10 bit as well as hardware implementations of communication protocols like I2C, SPI and UART are common. The operational current is not critical in many applications microcontroller development boards are used for, but it is mostly under 20 mA in active mode. Sleep modes allow for reduced currents down to the µA range. Most controllers we analyzed have a wide operational voltage range, for example the Atmel ATmega328 (1.8 V - 5.5 V).
Apart from the microcontroller itself, development boards contain additional components. First, a programming interface to upload firmware or debug running code. Second, connectors that allow users to attach external electronics. Further components are voltage regulators, status LEDs or oscillators. For programming, most modern platforms use USB, either supported by the microcontroller itself (for example on the Atmel ATmega32U4) or through a USB to UART bridge controller. Older boards require a special programming adapter or cable to connect to a computer either through USB or a serial port. Two of the six platforms in our survey rely on a special adapter cable for programming.
When prototyping electronic circuits, breadboards [31] are commonly used to arrange and connect components like LEDs, sensors or ICs. In order to connect a breadboard circuit to a microcontroller platform, two basic techniques are common.

- **Pin Headers**: Boards with standard pin headers (2.54 mm pitch) on the bottom like the Basic Stamp or the Arduino Micro can be directly plugged into the breadboard, given the layout is compatible to the breadboard grid.

- **Female Pin Headers**: Boards with standard female pin headers (2.54 mm pitch) have two advantages. First, the leads of components like LEDs or resistors can be inserted directly into the headers without utilizing a breadboard. Second, add-on boards ("shields") with corresponding pin headers on the bottom can be stacked upon the microcontroller platform. Breadboards can be interfaced with single stranded wire, which is also used for the breadboard circuit.

Not all pins of the microcontroller are necessarily connected to a corresponding pin on the headers of the PCB. The platforms we analyzed connect an average of 90 % of all GPIOs to the headers. In addition to that, regulated and unregulated input voltage as well as GND are usually broken out.
Additional components on the platforms we surveyed are often voltage regulators that allow to power the board with voltages higher than the maximum supply voltage of the microcontroller. This gives users more flexibility when selecting a power source. LEDs are also common on microcontroller platforms as they allow a simple feedback from the running firmware, for example as a debug output.

Other Properties

Apart from the hardware, the software framework is a crucial element of a microcontroller platform. Experts are able to program most microcontrollers in different languages and different environments. For beginners and intermediate users however, many manufacturers supply a dedicated development environment along with their platforms. This consists usually of an editor along with a toolchain to compile and upload the program to the microcontroller. An operating system independent software maximizes the scope of potential users, so only one of six analyzed platforms necessarily requires Microsoft Windows. Most platforms we evaluated are relatively affordable with prices tags around 15 USD to 20 USD. Only the Basic Stamp is more expensive at 60 USD. Especially for beginners, the price tag is most likely a strong argument when selecting a platform to start with.

Microcontroller Platform	Arduino Uno	MSP Launchpad MSP-EXP430G2	PICAXE-18 Project Board	Teensy 2.0	Basic Stamp 2sx	ArduinoMicro
Clock Frequency (MHz)	16	16	32	16	50	16
Operational Voltage (V)	1.8 – 5.5	1.8 – 3.6	1.9 – 5.5	2.7 – 5.5	3.0 – 5.5	2.7 – 5.5
Architecture	8 bit	16 bit	†	8 bit	8 bit	8 bit
# ADC channels @ Resolution	6 @ 10 bit	8 @ 10 bit	10 @ † bit	12 @ 10 bit	-	12 @ 10 bit
RAM (kB)	2	0.5	0.5	2.5	0.03	2.5
Flash (kb)	32	16	2	32	3	32
Active Supply Current (mA)	10	4.5	†	13	60	13
Sleep Supply Current (µA)	6.5	0.1	†	160	500	160
Timers	2 x 8 bit, 1x 16 bit	2 x 16 bit	1 x † bit	1 x 8 bit, 2 x 16 bit, 1 x 10 bit	1 x 8 bit	1 x 8 bit, 2 x 16 bit, 1 x 10 bit
Communication	I2C, SPI, UART	I2C, SPI, UART	UART	I2C, SPI, UART, USB	UART	I2C, SPI, UART, USB
GPIOs (Accessible‡ / Existing)	20 / 23	18 / 24	16 / 16	25 / 26	16 / 20	25 / 26
Connector Type	Female Pin Headers	Female Pin Headers	Solder Pads	Pin Headers	Pin Headers	Pin Headers
USB	Yes	Yes	No	Yes	No	Yes
Programs over . . .	USB	USB	Adapter Cable	USB	Adapter Cable	USB
Peripherals on PCB	XTAL, USB bridge controller, LEDs, VREG	LEDs, Button, Debugger Controller	Programming Header, ULN2803	LED, OSC, USB Connector,	EEPROM, OSC, BOD	LEDs, OSC, VREG
Hardware Open Source	Yes	No	No	Yes	No	Yes
Toolchain Open Source	Yes	Yes	No	Yes	No	Yes
Toolchain OS	Linux, Windows, OS X	Linux, Windows, OS X	Linux, Windows, OS X	Linux, Windows, OS X	Windows	Linux, Windows, OS X
Size of Board (mm)	68.6×53.4	65×50×13	60×50×10	30 x 18	30.0×16.0×3.81	48×18
Price (USD)	25	10	12	16	60	20

† Data not available.
‡ Via pin headers or solder pads.

Table 2.1: Properties of different microcontroller platforms. Numbers taken from specifications and datasheets.

Conclusion

Our analysis of different microcontroller development platforms shows that success does not only depend on the technical specifications. The Arduino for example has an inferior microcontroller in terms of architecture, clock speed, memory and power consumption. However, many applications, especially in education do not require powerful microcontrollers and will most likely not even use the full potential of an 8 bit microcontroller like the Arduino's Atmega328P. On the other hand, there is also a clear demand for more powerful and complex or energy efficient microcontroller development boards, but rarely among beginners. For advanced and professional users, it makes a lot of sense to upgrade to a more capable microcontroller. This requires experience and knowledge that is usually acquired on less complex systems. In the past years, popular product lines like Arduino have started to offer powerful, ARM based boards with a user-friendly interface, but as of now, simple 8 bit boards are still far more common due to their compatibility with the existing code base, their simplicity and their lower price tag. Since the microcontroller is the central component of the BRIX$_2$ toolkit, the analysis of popular microcontroller platforms provides us with a set of principal features and functions. We find that functions like ADCs or timers are integrated in almost any microcontroller in our survey and that on-board LEDs and USB are common features for microcontroller platforms.

2.2 Physical Computing Toolkits

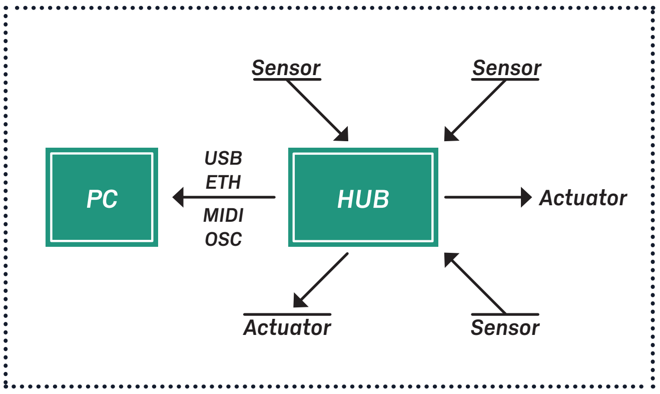


Figure 2.4: Diagram of a typical physical computing platform.

The term “physical computing” describes connecting the virtual world (software and networks) and the physical world. Microcontrollers are the link between the computer, representing the virtual world, and sensors as well as actuators that sense and manipulate the physical world. Physical computing is usually approached in two different ways: Either users intend to enhance their computer’s capabilities by connecting external custom devices or they intend to enhance their custom electronics by connecting them to a computer, for example to get easy access to the Internet or use the computer for heavy processing and mass storage tasks. Physical computing toolkits and for example microcontroller platforms can not always clearly be separated. In our analysis, we define a physical computing system as

1. A kit of different modules that are readily available.
2. The modules can be connected in order to build applications spanning the virtual and the physical world.
3. This process does not require knowledge in electronics and/or soldering.
4. Firmware programming is not required.

By this definition, physical computing platforms in contrast to pure microcontroller platforms require no special skills in order to build a working application. They consist of readily available building blocks that are tested and proven to work. This way they can just be combined into an application, even by non-expert users. Regarding the hardware, physical computing platforms usually implement hubs or interfaces between sensors as well as actuators and a PC, see Figure 2.4. They abstract those external components so they can be accessed by software through for example USB or ethernet. In our survey we analyzed a total of six selected physical computing platforms. In the following section we present four of them in greater detail.

2.2.1 Physical Computing in Education

We are surrounded by electronics in almost every aspect of our lives. Computers no longer just sit on our desktops, manifested in grey boxes or laptops but are integrated into basically any electronics device, from phones and watches to microwaves and power tools. Because of this close connection between the virtual world and our everyday environment, we consider it important to teach students about the fundamentals and inner workings of computers as well as embedded systems. Education is a primary aspect in the field of physical computing and some platforms are especially designed for that purpose. They are optimized on usability, transparency and simplicity. These platforms are often too limited to be used in actual projects and applications and can be regarded as a sandbox for physical computing. For actual applications, more generalized and complex platforms come into play, once a student or user has understood the basics.

2.2.2 Open Source Physical Computing: Grove Electronic Brick Kit

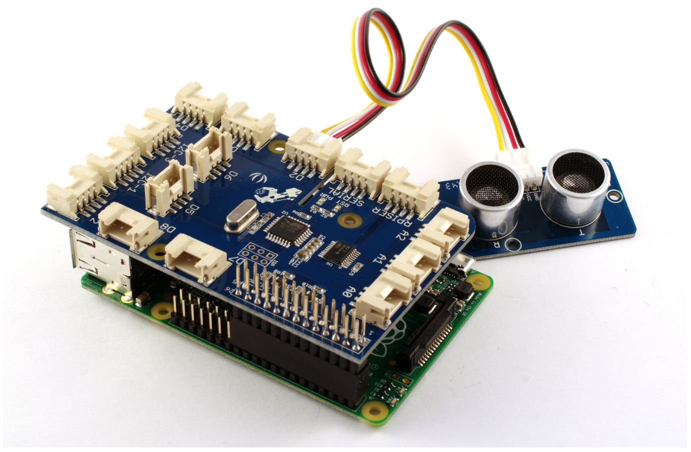


Figure 2.5: The Seeedstudio Grove shield for Raspberry Pi with attached ultrasonic distance sensor. Photo by Gareth Halfacree, CC BY-SA 2.0.

With the Grove Electronic Brick Kit, see Figure2.5, Seeedstudio offers a huge set of small PCBs that can be connected to a *Base Shield*. The base shield represents an adapter board that connects to different platforms like Arduino, Raspberry Pi, MSP Launchpad. This way, Grove creates a unified interface between several platforms and approximately 100 small extension boards. The boards are connected via four-pin cables to the I/O or bus pins of the host microcontroller platform, depending on their functionality. A basic kit with a base shield, 10 extensions and cables costs around 50 USD. [32] An extensive and detailed documentation is available in the Seeed wiki [33]. The whole platform is open hardware and open software.

2.2.3 Legos for Electrical Engineers: Tinkerforge Building Blocks

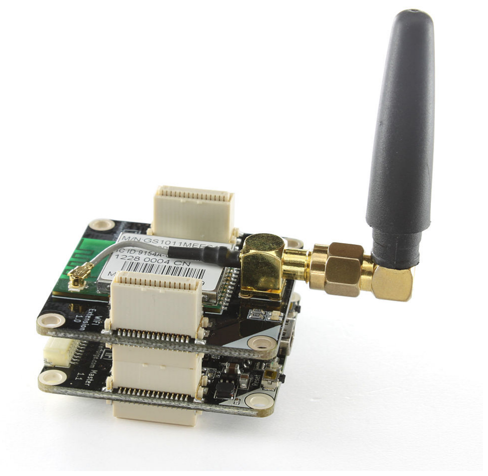


Figure 2.6: Tinkerforge Master Brick with FiFi Extension. Photo by Tinkerforge GmbH, CC BY 1.0.

The German company Tinkerforge[6] offers a variety of interconnecting boards that focus on physical computing, home automation and industrial applications. Different types of boards are combined to a stack that suits the desired application, see Figure 2.6. Fundamentals are *Bricks* and *Bricklets*.

Bricks are 40×40 mm stackable PCBs. They all serve different purposes such as motor control or measurement of inertial motion. Each Brick can individually be connected to a host system like a PC or Smartphone via USB. A *Master Brick* can be used to route the signals of a whole Brick stack through a single USB connection. *Master Extension Bricks* allow to use alternative communication protocols between host and stack like Ethernet, RS485 or WLAN. Among the currently (Feb. 2015) available Bricks are Direct Current (DC), stepper and servo motor drivers, IMU and Master Bricks. Prices tags range from 35 to 115 USD.

[6]http://www.tinkerforge.com

Bricklets are more simple extensions like sensors, buttons, displays that can be connected to Bricks using cables. Each Brick has connectors for up to 4 Bricklets. Currently (Feb. 2015) there are 40 different Bricklets in the Tinkerforge online shop, including potentiometer, motion detector, Radio Frequency Identification (RFID), humidity sensor, analog and digital I/O, LCD, LEDs, Global Positioning System (GPS). Prices range from 3 to 45 USD.

Although each brick contains an Atmel ATSAM3S4B 32 bit microcontroller, they are not designed to run an application on their own and always rely on a host system, which executes applications for the stack. These can either be the *Brick Viewer* or a custom software. Tinkerforge offers Application Programming Interfaces (APIs) for programming languages like C/C++, Java, JavaScript, LabVIEW, MATLAB/Octave, Perl, Python, Visual Basic .NET. Once a stack is assembled, the confguration can be detected by software. Even passive Bricklets contain a small EEPROM that identifies them. Instead of connecting the stack to a host computer, a *Red Brick* can be used to run the application software. This single board computer features an *Allwinner A10s* System on Chip (SoC) with integrated Cortex A8 CPU, Graphics Processing Unit (GPU) and Video Processing Unit (VPU), USB host, High Definition Multimedia Interface (HDMI) and 512 MB SDRAM. A Debian Linux system runs off an SD card. The Red Brick costs 80 USD not including cables or SD card. All Tinkerforge products have open source licenses and documentation is available for hardware and software. A medium sized but active community exists, which is mostly based in Germany. [34]

2.2.4 Physical Widgets: Phidgets

Phidgets was introduced in 2001 by Saul Greenberg and Chester Fitchett, University of Calgary, as a toolkit for rapid prototyping of physical user interfaces [35]. The project continued as a commercial product, which is still actively developed by Phidgets Inc.[7] Originally, the concept of the platform was to connect sensors and actuators to software that is running on a host system like a PC, hence providing a flexible, physical interface. The most recent platform (as of 2014), the PhidgetInterfaceKit 8/8/8 is designed around a Cypress enCoRe USB controller that connects to 8 digital inputs, 8 digital outputs and 8 analog sensor ports to a PC.The company offers over 100 sensor modules that can connect to these ports via cables. APIs for more than 20 programming languages allow easy integration of the Phidgets hardware into custom software projects. For standalone applications, dedicated single board computers are available for the Phidgets toolkit. Most of the documentation can be found in the Phidgets wiki. The hardware itself is closed source. An active user forum with around 4000 members is accessible through the Phidgets website.

[7]http://www.phidgets.com/

2.2.5 Teaching Physical Computing: littleBits

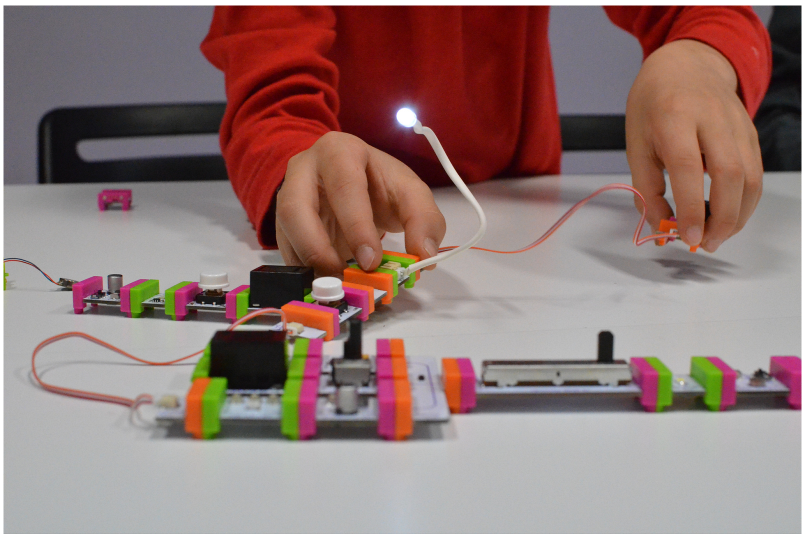


Figure 2.7: A child working on a littleBits circuit. Photo by Lisa George, CC BY-SA 2.0.

littleBits is not a typical physical computing platform that acts as a hub for sensors and actuators, but primarily a learning platform for basics in electronics and physical computing. Nonetheless it can be utilized to build all sorts of interactive applications, which is why we included it in this section. littleBits started as a project by Ayah Bdeir at the MIT Media Lab with idea to regard electronics as a building material just like paper, cardboard and screws. [36] The result is a library of mostly passive electronic boards that can be assembled into actual applications, see Figure 2.7. Three basic, color coded types of modules, inputs, outputs and wires snap together with magnetic contacts. An analog signal is routed through the whole circuit from the power source and every input module affects the downstream signal of the whole circuit. littleBits allow a tangible and hands-on way of designing electronic circuits which makes it a valuable learning tool. An optional Arduino module adds programmability to a circuit for more advanced applications and scenarios. Using the Arduino module, it is also possible to use littleBits the same way as any other physical computing platform in our survey. [37] The whole project is open source and the company explicitly encourages users to create their own modules for the system for example by providing design rules [38] and hosting design contests. Documentation on each module and kit is available and an active online forum allows users to exchange knowledge and experience.

2.2.6 Characteristics of Different Physical Computing Toolkits

Physical computing is a wide field and the toolkits we analyzed are designed for different needs from interactive art installations to prototyping of industrial applications. In general, most platforms in our survey basically connect sensors and actuators to an application running on a host computer. We summarized all properties of the analyzed platforms in Table 2.2.
The central elements of physical computing platforms are hubs. They connect the host PC to the sensors and actuators. The hubs in our survey are mostly based on microcontrollers that can either be programmed by the user or ship with a fixed firmware that allows the software on the host computer to access connected sensors and actuators. The utilized microcontrollers range from simple 8 bit controllers (littleBits) to powerful 32 bit ARM processors (Tinkerforge BRICKs), matching the individual target field of application. Most of the systems are extensible by custom electronics, but usually the companies offer a wide spectrum of different extensions so that many use cases are already covered. This allows users to just buy a component instead of making their own, which would require certain technical knowledge and skill. A characteristic property of every physical computing platform in our survey is the way that extensions connect to the base modules. There are three fundamentally different approaches:

Stacking: Modules stack on top of each other using pin headers or fine pitch connectors. This leads to a rigid and compact structure, but the form factor is fixed.

Cables: Extensions are connected to the hub using cables. On the one hand, this leads to a flexible structure which can, on the other hand get messy and chaotic.

Side-by-Side Connectors: littleBits uses a magnetic connector system that allows users to build circuits similar to using dominoes. Such structures are mostly arranged on a 2 dimensional plane and can take up a lot of space if they get complex.

The number and variety of different readily available extensions define the level of flexibility and ability of a physical computing platform to adapt to various applications, considering that most users are not capable or willing to extend the platform with custom hardware. Apart from the RFduino[8], all systems in our survey offer at least 50 different sensors, actuators and other extensions that are compatible with the corresponding hub and software. The prices of these extensions usually range from around 5 USD for simple passive modules to around 100 USD for very complex, active modules. The hubs themselves range from around 20 USD for a simple, Arduino based educational platform to almost 190 USD for the ICUBE-X[9], which al-

[8]http://www.rfduino.com
[9]http://infusionsystems.com/catalog/

lows easy integration of sensors and actuators into for example music performances. The system is compatible to software frameworks in this field. The hardware is also designed for stage environments.
Systems like X-OSC[10] or ICUBE-X allow artists to set up interactive installations and to focus on the creative aspect rather than on the technical details. They are reliable and compatible to common software frameworks by using protocols like Open Sound Control (OSC) [39] or Musical Instrument Digital Interface (MIDI) [40]. Both platforms are not reprogrammable, so they are by design limited to a certain range of applications. Programmability on the one hand drastically increases the adaptivity and therefore the range of potential applications, but on the other hand increases the complexity of an application and requires more time and skills. In an embedded electronics lecture, platforms like X-OSC and ICUBE-X would unnecessarily abstract and hide their inner workings. It becomes harder to learn something about the functionality and principles behind the platform. In this case, a toolkit with basic electronics and software is more applicable. Students can start building from the ground up and can later develop more complex applications on top of this basic knowledge. Toolkits like Grove and LittleBits are a good example for this. Beginners can start with the very basics, like for example blinking an LED or reading a potentiometer. As the application grows, so do the skills of the users. When they end up with a complex system, they have learned how every aspect of it works. Both platforms provide enough extensibility so that even advanced users stay motivated and are not limited by the system's capabilities. From the technical side, we found that most physical computing platforms we examined are not designed for performance or compactness. More important are usability aspects, for example the color coding and reverse protection of the littleBits system or simplicity of the Grove extension system. Educational platforms either have to be self-explanatory or well documented. Ideally they are both. A comprehensible documentation like the Grove wiki enables users to learn on their own and without institutionalized teaching. Our survey of physical computing platforms shows different design approaches for modular toolkits with readily available extensions for a variety of targeted applications. Since the concepts of the individual platforms are so diverse, it does not make sense to identify common parameters. However, we identified some outstanding features and concepts that can potentially be integrated into the BRIX$_2$ system. We specify those in the following chapter.

[10]http://www.x-io.co.uk/products/x-osc/

System	tinkerforge BRICKs	ICUBE-X	littleBits	Rfduino	Phidgets	Grove
Application Area	Industry, DIY, Education	Art, Music, DIY	Education, DIY	Internet of Things, DIY	Education, Research	Education, DIY
Programmable	Yes	No	Yes	Yes	No	Yes
Hub Microcontroller	Cortex M8	†	ATmega32U4	Cortex-M0	Various	Various
Extension Connectors	Cables, High Density Headers	Cables	Magnetic Connectors	Pin Headers	Screw Terminals	Cables
Hub Size (mm)	40×40	121×94×34	20×†	22.9×29	48×19	70×50
Available Extensions	50	54	60	10	> 100	100
Extension Price Range (USD)	4 – 110	6 – 206	6 – 50	10 – 30	7 – 150	2 – 100
Hub Price (USD)	35	187	35‡	40	80	9*
Operating Systems	Windows, Linux, OS X	Windows, OS X	Windows, Linux, OS X	Windows, Linux, OS X	Windows, Linux, OS X	Windows, Linux, OS X
Programming Interface	Misc. Languages, Host-Side	Not programmable	Arduino IDE	Arduino IDE	Not Programmable	Various
Host Software Interface	USB, APIs	USB, Serial, MIDI, OSC	USB, Serial, HID	USB/Serial	USB, APIs	USB, Serial
Documentation	Wiki, Website, Forum	Manuals, Website, Forum	Videos, Forums	Manuals, Examples, Forum	Wiki, Videos, User Guides, Forum	Wiki, Forums
Estimated Community Size and Distribution	Medium, mostly Germany	Small, International	Big, International	Medium, International	Big, International	Medium, mostly American

† Data not available

‡ Price refers to the Arduino Module: http://littlebits.cc/bits/Arduino

∗ Plus an additional Arduino or a similar platform, since the Grove base board is just a shield.

Table 2.2: Properties of different physical computing platforms. Numbers taken from specifications and datasheets.

2.3 Wireless Sensor Nodes

A wireless sensor node is a battery powered device that captures sensor data, which is then sent wirelessly to a remote host system. A network of several wireless sensor nodes is referred to as WSN. Typical WSN applications are for example tracking of environmental data in a building or monitoring of industrial processes inside a factory. More general, WSNs can be used to sense one or many modalities in multiple locations inside a certain area. This area can be a human body, a room, a building or even a whole country. Wireless sensor nodes, also called *motes*, usually consist of a microcontroller, a wireless transceiver as well as number of sensors and is powered by a battery, see Figure 2.8.

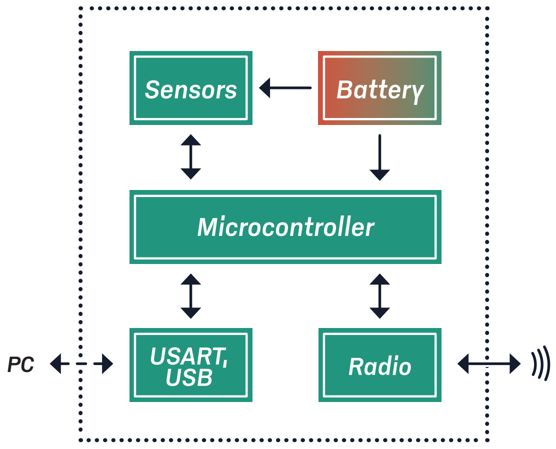


Figure 2.8: Diagram of a typical WSN mote.

In many scenarios, motes are supposed to operate for months or years, which requires a high degree of optimization towards energy efficiency. This is mainly achieved by choosing low power components and short duty cycles for microcontroller and wireless transceiver activities. The research field of wireless sensor networks became popular around 2005 and publications focused on networking aspects like efficient routing strategies and scalability of networks with a large number of nodes. Today, the principle of WSNs often also manifests in the field of the *Internet of Things (IoT)* [41], which is closely related to ubiquitous computing. But not only the wireless networking aspect of WSN motes is interesting for the design of BRIX$_2$. Battery powered, standalone operation is also a feature which would facilitate applications in ubiquitous computing such as smart objects which can be moved around freely. In our survey we analyzed the technical specifications and properties of five different popular WSN motes. In the following we present two of them in detail before comparing all five.

2.3.1 Scientific Sensor Networking: TelosB

The Telos wireless sensor mote was developed as a research platform by UC Berkeley in 2005 [42]. It was marketed by Crossbow Technology, Inc in the same year. [43] The low price of initially 99 USD made the platform popular for researchers and even today, numerous publications on WSN are based on this mote. TelosB is based on an Texas Instruments MSP430 microcontroller running at 8 MHz and powered by two AA batteries. The IEEE 802.15.4 [44] compliant wireless transceiver operates in the 2.4 GHz band and uses an on-board Planar Inverted Folded Antenna (PIFA). It achieves communication ranges from 75 - 100 m outdoors respectively 20 - 30 m inside buildings. Sensors for temperature, humidity and light are integrated into the TelosB platform. A 16 way pin header allows connecting external electronics such as additional sensors. TelosB can be programmed via USB and runs TinyOS, a small, open-source embedded operating system.[11] The documentation on tinyOS is detailed and accessible, however documentation on the mote itself is sparse, although it is an open source project. Several motes based on the TelosB are still being released, yet the hardware is similar.[12]

2.3.2 WSN Prototyping: Libelium Waspmote

Waspmote is a modular WSN platform by Libelium, designed for prototyping purposes. The base board only contains the microcontroller, an accelerometer, a Real Time Clock (RTC) as well as an SD card slot. Other components like the wireless transceiver, sensors or power sources are available as external modules. This allows developers to test different configurations while adapting the platform to a specific application. Libelium offers 15 different transceiver boards and 10 sensor boards with multiple sensors each. [45] Waspmote is designed around an Atmel ATmega1281 8 bit microcontroller clocked at 14 MHz and can be programmed via USB. The Waspmote IDE is based on the Arduino IDE and provides a simple programming interface for the motes. Apart from that, Libelium offers an API, sensor data aggregation software and mobile applications. The company provides detailed technical documentation, along with tutorials and a forum for the user community.

[11]http://www.tinyos.net/

[12]http://www.zolertia.com/ti

2.3.3 Characteristics of Different Wireless Sensor Nodes

In this section we presented two of the five WSN platforms we analyzed in total. We selected platforms that focus on research and prototyping applications in contrast to commercial industrial platforms. The reason for this is that a generalized platform like our proposed BRIX_2 system does not aim for professional industrial applications. All properties are summarized in Table 2.3

Microcontrollers on Motes

One of the key elements on every mote is the microcontroller. It runs the application which polls the connected sensors, buffers the data and relays it to the wireless transceiver. The microcontroller and the wireless transceiver are typically the most power consuming components on a wireless sensor mote. In order to save energy, the application operates on low duty cycles. This means the controller stays in sleep modes most of the time and is woken up in regular intervals to poll the sensors and transmit data. The shorter these active phases are in comparison to the sleep phases, the lower the over all power usage.
Most of the platforms in our survey are designed around low power, 8 bit microcontrollers running at clock speeds below 20 MHz. Their current draw does not exceed 11 mA in active mode and 12 uA in deep sleep. Flash memory to store the firmware as well as RAM are limited, which is not regarded critical because applications are usually rather minimalistic.
The Atmega128L is especially popular because of its support for tinyOS, an embedded operating system widely used on WSN motes. [46] Apart from short wakeup times from sleep modes that facilitate low power consumption, prominent features of a microcontroller on a wireless sensor node are hardware implementations of bus protocols like SPI and I2C. They allow to interface sensors and wireless transceivers. A low minimum operational voltage allows to drain batteries far below their rated voltage.

Wireless Networking

Platforms in our analysis mostly contain a fixed wireless transceiver chip. Only the Libelium Waspmote, as a modular prototyping platform, has an expansion slot that allows to use it with a number of wireless and wired communication interfaces. Built in transceivers usually operate in the Industrial, Scientific and Medical (ISM) band [47] on either 2.4 GHz or 868 MHz. The transceiver chip only provides the physical layer for wireless communication. The other layers [48] are implemented on the microcontroller.
For energy efficient applications like WSN low power transceiver are selected. Usually, idle, Receive (RX) and Transmit (TX) power are considered separately. Especially the idle power is critical because the transceiver, just like the microcon-

troller, remains in idle mode most of the time.[46] The motes we analyzed have idle transceiver supply currents well below 100 uA. While actively receiving or sending data, the transceiver wakes up from idle state and draws around 15 mA before returning to idle. At data rates of at least 76.8 kb per second, transmitting a data packet only takes fractions of a second. The range of the wireless transmission is, apart from external influences, defined by the antenna and the gain of the output amplifier for TX as well as the input sensitivity for RX. Many motes use chip antennas or trace antennas because they are inexpensive and easy to integrate. Encryption can also be an important feature for WSNs, especially when critical data is exchanged. Modern transceiver chips implement different encryption methods in hardware. [49]

Other Properties

The runtime of a mote is determined by its energy consumption and the power source used. The classical Berkeley mote design (TelosB, MicaZ, BTnodes) uses two AA batteries as a power supply. These batteries are widely available in a variety of different technologies, see Section 3.2.4. More modern motes are powered by Lithium Polymer (LiPoly) batteries, which have a higher energy density but can not be discharged to low voltages without permanent damage. For protection they require special charge and discharge controllers. In order to update the firmware on a mote, communicate with its application or download data from an internal flash memory, a serial connection is required. This is usually implemented via USB on modern platforms. Older WSN nodes require USB/Serial adapters for communication and ISP adapters for programming. All motes we analyzed feature extension headers that allow to connect external electronics. Since not even half of the devices in our survey contain on-board sensors, add-on boards are available for these platforms. The physical size of WSN motes is usually non critical, but does not exceed 73.5×51×13 mm in our survey. WSN nodes for research applications are sparsely documented. Technical descriptions and user guides may be present, but tutorials or application examples are hard to find. There is also no real user community that could provide reliable support. A different example is the Libelium Waspmote. It is intended for prototyping and a professional product at the same time and therefore well documented. The company also actively drives the development of a user community by providing a web forum.[13] The results of our survey of WSN motes show us which hardware and concepts are required to exchange data wireless as well as to increase the runtime of a battery powered device, two requirements for many applications in ubiquitous computing.

[13]https://www.libelium.com/forum/

System	Libelium Waspmote	Zigduino	MICAz	TelosB TPR 2420CA	BTnodes
Microcontroller	ATmega1281	ATmega128RFA1	ATmega128L	MSP430	ATmega128L
Clock Frequency (MHz)	14	16	8	18	8
Operational Voltage (V)	1.8 – 5.5	1.8 – 3.6	2.7 – 5.5	1.8 – 3.6	2.7 – 5.5
Architecture	8 bit	8 bit	8 bit	16 bit	8 bit
RAM (kB)	8	16	64	10	64
Flash (kB)	128	128	128	48	128
Active Supply Current (mA)	10	4.1	11	1.8	11
Sleep Supply Current (µA)	0.5	0.25	12	5.1	12
Wakeup Time (µs)	†	25	180	6	180
Wireless Transceiver	misc	ATmega128RFA1	CC2420	CC2420	CC1000
RF Band	misc	2.4 GHz	2.4 GHz	2.4 GHz	868 MHz
IEEE 802.15.4	Yes	Yes	Yes	Yes	No
RX Current (mA)	–	12.5	18.8	18.8	9.6
TX Current, 0 dBm (mA)	–	14.5	17.4	17.4	16.5
RF Idle Current (µA)	–	0.25	21	21	96
Data Rate	–	2 Mb/s	250 kbps	250 kbps	76.8 kbps
Typ. Range Outdoor (m)	–	†	75 – 100	75 – 100	†
Typ. Range Indoor (m)	–	†	20 – 30	20 – 30	†
Power Source	LiPoly, Solar Panel	LiPoly	2x AAA	2x AA	2x AA
Extension Connectors	Pin Headers	Pin Headers	Fine Pitch	Pin Headers	Fine Pitch
USB	Yes	Yes	No	Yes	No
Programming Interface	OTA, USB, ISP	USB, ISP	OTA, ISP	USB, ISP	ISP
Onboard Sensors	Temperature, Accelerometer	None	None	Light, Humidity, Temperature	None
Size (mm)	73.5×51×13	68.6×53.4	58×32×7	65×31×6	65×31×6

† Data not available

Table 2.3: Properties of different WSN motes. Numbers taken from specifications and datasheets.

2.4 Inertial Measurement Platforms

In ubiquitous computing, natural and intuitive interfaces between humans and computers can make technology disappear and blend into our environment. For most people, moving their body is an essential part of their everyday lives. Through motion based interfaces, users can interact with a computer without an abstract interface like a keyboard or a mouse. Inertial measurement platforms are used to sense orientation and movements of a rigid body, for example to track the motion of human limbs or other objects. They are usually designed around inertial sensors namely accelerometers, gyroscopes and magnetometers. To obtain and process data recorded by those sensors, a microcontroller is required. Depending on the application, the data is either streamed to an external computer over a wired or wireless connection or stored on a memory inside the device.
In the following we introduce the principles of inertial sensing along with the key components of an inertial measurement platform before we present three of the 8 platforms we surveyed in detail. Finally we compare all 8 devices to determine common properties and relevant features of inertial measurement platforms.

2.4.1 Inertial Sensing

An IMU is the key component of any inertial measurement platform. It allows to determine the three-dimensional orientation of an object such as an aircraft, which can be expressed as three angles of orientation, quaternions or rotation matrix. A typical IMU contains three different sensors:

Accelerometers measure the acceleration of the IMU in three orthogonal axes. The principle of operation is usually to measure the displacement of a damped proof mass on a spring caused by gravity or external acceleration.

Gyroscopes measure angular velocity of the IMU in three orthogonal axes. There are different principles of operation for gyroscopes. The most basic is the mechanical gyroscope that consists of a spinning disc mounted inside two gimbals. A rotation of the outer gimbal results in an according rotation of the inner gimbal which can be measured. These two sensors allow to measure the linear movements as well as the rotations of the IMU. A third sensor is often added in order to correct the measurements of the other two:

Magnetometers measure the surrounding magnetic field. Given the knowledge about the magnetic properties of the environment, such as the Earth's magnetic field, the magnetometer can be used as a compass. There are is a variety of ways to measure a magnetic field, for example utilizing the Hall effect.

2.4.2 Sensor Fusion

The readings of all three sensors can be fused by special algorithms in order to acquire a more precise orientation. All three sensors have different disadvantages when determining their orientation in 3D space which can be compensated by the advantages of the other two sensors. For example if used on the Earth's surface, a three-axis accelerometer can determine a gravitation vector and thus providing the pitch and roll angles. However it can not determine the yaw angle and it is affected by other accelerations beside gravity. A gyroscope can determine all angles by integrating the angular velocities, but tends to drift over time. The magnetometer provides an absolute orientation by measuring the Earth's magnetic field, but its readings are easily disturbed by other magnetic fields. A well-known algorithm for sensor fusion is the Kalman Filter.

2.4.3 MEMS Motion Sensors

Micro-ElectroMechanical System (MEMS) are devices in the micrometer scale. They are typically fabricated from materials like silicon or metals by using a combination of material deposit and etching processes closely related to manufacturing processes of integrated circuits. In the last two decades, advances in MEMS technology allowed an increasing level of integration for inertial sensors. Within five years, products have developed from analog single-axis gyroscopes [50] to the combined IMU devices available today, containing three axes gyroscopes, accelerometers and magnetometers along with a sensor fusion processor [51]. Both devices have equal package sizes whereas the price has halved. This is a significant step towards motion sensing devices that are smaller, more powerful and affordable than ever before.

2.4.4 Motion Capturing

In application fields like health and sports science, virtual reality as well as character animation, recording of body motion data is essential. Classical techniques are often based on optical observation of motions for example with a single or multi camera system. More advanced methods of optical Motion Capturing (MoCap) include active or passive markers that are attached to the body and tracked by the camera system. The result is data with a high temporal and spatial resolution that can be directly transformed into a virtual 3D space. However, aside from the high costs for stationary MoCap systems, they reach their limits when the subject moves greater distances and/or markers leave the camera's field of view.

2.4.5 Motion Capturing with Inertial Sensors

IMUs can be used in motion capturing as an alternative for classical tracking methods like optical or magnetic tracking. Almost any human body part can be equipped with a number of IMUs in order to capture either a full body posture or parts of the body such as single limbs or just the head. A disadvantage of inertial motion tracking is that it does not provide absolute 3D coordinates. The technique relies on a pre-defined skeleton on which the rotations of the single tracking devices are mapped. If the tracked person walks through a room, the virtual skeleton will still be at the same position. Position data can only be estimated or added by an optical tracker. Inertial motion tracking is less expensive than optical motion tracking, easier to use, the amount of data is smaller, but it is not as accurate as optical tracking. Apart from MoCap, IMUs can also be used in different applications like aviation, stabilization (of cameras or antennas), shipment tracking or Human Computer Interfaces (HCI).

2.4.6 Typical Components of Inertial Measurement Platforms

The central element of an inertial measurement platform are the sensors, see Figure 2.9. Today mostly highly integrated MEMS devices with digital interfaces are used. They contain sensor elements for all three measurement axes (x,y,z) and signal processing hardware. Some sensors are even capable of internal sensor fusion and directly provide orientation data, for example Euler angles or quaternions. [52]

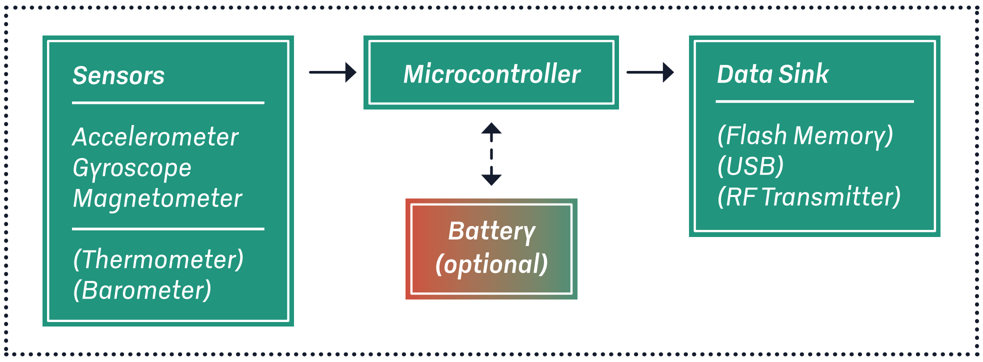


Figure 2.9: Diagram of a typical IMU platform.

Another key element of an IMU is a sink for the motion data itself. It can either be stored on the device, for example in a flash memory or streamed to an external device such as a PC via USB or a wireless transceiver. The third fundamental component is the microcontroller that connects the data sink and the sensors. It can also perform signal processing, sensor fusion, timestamping of data, etc. Many IMUs contain batteries or offer at least battery connectors so they can be used as standalone devices for recording motion data for hours or even days. In the following, we present three IMUs in detail. Some of them are highly optimized and target specific markets and applications whereas others are general purpose open source products.

2.4.7 Industry Grade Motion Sensing: Xsens MTi-10

The Xsens MTi-10 series IMU module is a professional motion capturing system. It is especially popular in the games and film industry, but is also used for measurement and active stabilization purposes in heavy machinery and aviation. The MTI-10 comes in a 41.0 x 56.5×21.3 mm aluminum enclosure and is accessed via a proprietary, 9 pin connector for data, sync and power. It can output acceleration data in a range of ±5 g and angular velocity data in a range of ±450 °/s with a rate of up to 2 kHz and a latency below 2 ms. Sensor data is fused internally using a Kalman filter [53]. The MTi-10 module costs around 1100 USD (2015).[14] Xsens also offers motion capturing sets based on the same technology that consist of 17 modules that are either attached to a body with straps or integrated into a full body suit.
Xsens modules are highly professional, commercial products. They are closed source and not extensible. Due to their price, they are mostly used in industrial, military and scientific applications, so there is no real open user community.

2.4.8 Semi-professional Motion Capturing: YEI 3-Space

The 3-Space Embedded IMU by YEI Technology [54] is a MEMS-based IMU optimized for motion capturing in a medium price range. The basic IMU board measures 23×23 mm and contains a 3-axis accelerometer, magnetometer and gyroscope as well as a temperature sensor. The sensor data is fused by a microcontroller which then provides raw data, Euler angles, rotation matrix or quaternions via SPI, USB or UART. YEI offers a number of fully encased variations based on the 3-Space Embedded that allow standalone operation by adding a battery and a wireless interface or logging memory. The battery runtime for the wireless version is rated for around 5 hours. Several software components like the YEI 3-Space MoCap Studio[15] and several plugins for third-party 3D software are available under open source licenses. The hardware itself is closed source.

[14]http://shop.xsens.com
[15]http://www.yeitechnology.com/yei-3-space-MoCap-studio

2.4.9 Open Source Inertial Sensing: IMUduino

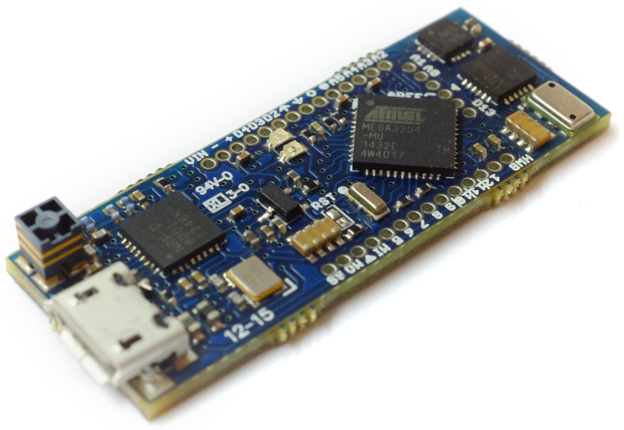


Figure 2.10: The IMUduino programmable IMU platform.

The IMUduino, see Figure 2.10, is a 39.8×15.72 mm Arduino compatible (Atmel ATmega32U4) board that includes an Invensense MPU6050 IMU, a Honeywell HMC5883L 3-Axis Digital Compass, a MS561101BA03-50 Barometer and a Nordic nRF8001 Bluetooth Low Energy (BTLE) transceiver. The product is designed and sold by Femtoduino and was founded via Kickstarter in November 2014. [55] It is now (2016) available in the femtoduino web shop for 129 USD.[16] The design allows to connect external hardware to the microcontroller using 1.27 mm pitch pin headers. Hardware documentation is sparse, only a pinout is available. The software is based on external, open source libraries and is accessible on Github. [17]. A special community for this device does not exist.

2.4.10 Characteristics of Different IMU Platforms

In our comparison of eight inertial sensing platforms, we identified a number of characteristic and common features as well as major differences between the platforms depending on their intended field of use.

Sensors

All platforms in our survey, see Table 2.4, use a complete IMU sensor set consisting of an accelerometer, a gyroscope and a magnetometer with 3 axes each. Usually, MEMS components are used as sensors, although more expensive and professional platforms

[16]http://femto.io/products/imuduino
[17]https://github.com/zrecommerce/imuduino-btle

use high quality components whereas more affordable platforms use consumer grade sensors.
Manufacturers of professional, closed source IMUs usually do not provide detailed information on what sensors are used in their products. However, the specifications are always provided. Accelerometers in almost any platform in our survey have a maximum range of ±8 g to ±16 g and a resolution below 1 mg. Gyroscope ranges are around ±2000 °/s max. with resolutions below 0.1 °/s. The magnetometer ranges are usually around 600 uT to 800 uT with resolutions well below 1 uT. The Invensense MPU6000/6050 series is used in at least 3 of 8 analyzed platforms as an integrated accelerometer and gyroscope sensor along with an additional magnetometer. Some platforms feature additional sensors such as barometers for altitude measurements. Temperature sensors are not always included in the systems descriptions but are in fact present on each platform for gyroscope temperature compensation.

Data Management

Most platforms stream sensor data to a host device during recording. This is either done through a wireless or a wired connection. Wireless transfers are problematic in terms of reliability, bandwidth and latency, so professional systems tend to use wired connections. If no proprietary protocol is used, data is usually streamed via USB to a host system. In scenarios that do not require real time recording, data can also be stored in a flash memory and read from the device afterwards. All analyzed IMU platforms are able to provide raw data as well as orientation data, which is calculated on the device, either on an internal microcontroller by a closed source firmware, a custom SoC [56] or directly on the motion sensor itself.
The maximum data rate varies strongly even among the professional systems between the APDM Opal[18] at 128 Hz and the X-Sens MTI-10 at 2 kHz. The FSM-9[19] and MTI-10 have maximum latencies of 2 ms which allows real time tracking applications. The data rates and latencies of ArduIMU[20] and IMUduino can not be listed because they depend on the firmware and communication method used in the application. Only two of the systems in our survey can store sensor data on an internal flash memory.

Application Specific Features

Most of the analyzed systems are designed for motion capturing, for example the Xsens, FSM-9, APDM, ProMove or Yei 3-Space. However, three of the systems in our survey, the ArduIMU, X-BIMU[21] and IMUduino are more general, flexible and

[18]http://www.apdm.com/wearable-sensors
[19]http://hillcrestlabs.com/product/fsm-9
[20]https://www.sparkfun.com/products/retired/11055
[21]http://www.x-io.co.uk/products/x-bimu

designed for prototyping motion based applications.
General purpose IMUs like the IMUduino, ArduImu or X-BIMU can be modified or extended in terms of firmware and hardware in order to suit a particular scenario. Those platforms are usually bare PCBs, batteries are optional and the documentation provides more technical details than for specialized solutions. Some products are open hardware and/or software. Two of the analyzed platforms are even Arduino compatible (IMUduino, ArduIMU) and based on 8 bit microcontrollers. More advanced and closed systems tend to use controllers with a much higher processing power. It is not always clear from the specifications provided by the manufacturer what exact hardware is used, but the professional systems obviously implement complex custom sensor fusion algorithms in order to supply optimal orientation data output. Other characteristics of professional IMUs are at least optional enclosures that allow mounting on a human body, an extensive documentation and compatibility with existing professional data processing software like MATLAB[22] or custom software packets supplied with the product. The prices for professional units are up to an order of magnitude higher than for the less professional, general purpose IMUs. Devices like the ArduIMU, IMUduino or X-BIMU are much closer to the platform we aim to develop regarding technical specifications, but also areas of application. They demonstrate that it is possible to build a powerful IMU using low-cost, consumer grade components.

[22]http://www.mathworks.com/products/matlab

System	FSM-9	YEI 3-Space	ArduIMU+ V3	ProMove mini	APDM Opal	X-BIMU	IMUduino	Xsens Mti-10
Accelerometer	BNO070	†	MPU6000	†	†	MPU6050	MPU6050	†
Accelerometer Range (g)	$\pm$ 8	$\pm$ 8	$\pm$ 16	$\pm$16	$\pm$6	$\pm$ 16	$\pm$ 16	$\sim$ 5
Accelerometer Resolution (mg)	$<$ 6	1 (14 bit)‡	0.5 (16 bit)‡	0.062	0.7 (14 bit)‡	0.5 (16 bit‡	0.5 (16 bit)‡	0.076 (16 bit)‡
Gyroscope	BNO070	†	MPU6000	†	†	MPU6050	MPU6050	†
Gyroscope Range (°/s)	$\pm$ 1833	$\pm$ 2000	$\pm$ 2000	$\pm$2000	$\pm$2000	$\pm$ 2000	$\pm$ 2000	450
Gyroscope Resolution (°/s)	0.04	0.06 (16 bit)‡	0.03 (16 bit)‡	0.007	0.24 (14 bit)‡	0.03 (16 bit)‡	0.03 (16 bit)‡	0.007 (16 bit)‡
Magnetometer	BNO070	†	HMC-5883L	†	†	†	HMC-5883L	†
Magnetometer Range (µT)	$\pm$ 600	$\pm$ 810	$\pm$ 800	$\pm$4912	$\pm$ 600	†	$\pm$ 800	80
Magnetometer Resolution (µT)	$<$1	0.2 (12 bit)‡	0.4 (12 bit)‡	0.15	0.07 (14 bit)‡	†	0.4 (12 bit)‡	0.002 (12 bit)‡
Additional Sensors	None	Temperature	None	Barometer, High-g Acc.	None	None	Barometer	None
Sensor Data Formats	Raw, Human Interface Device (HID), Orientation	Raw, HID, Orientation	Raw, Orientation	Raw, Orientation, Altitude	Raw, Orientation	Raw, Orientation	Raw, Orientation, Altitude	Orientation, Raw
Data Sinks	USB, SPI	USB, SPI	USB, I2C, SPI	USB, Flash, RF 2.4 GHz	USB, Flash, RF 2.4 GHz	USB, UART, Xbee	USB, BTLE	UART, USB
Max. Data Rate (Hz)	250	1350	$*$	1000	128	256	$*$	2000
Latency	1.8ms	†	$*$	†	30ms	$*$	$*$	2ms
Microcontroller	Cortex M0+	†	Atmega328P	†	†	PIC24FJ64GA	ATmega32U4	†
Supply Current (mA)	24	45		†	†	13	†	106
Power Source	USB	USB, optional Battery	USB, optional Battery	USB, internal Battery	internal Battery	USB, optional Battery	USB, optional Battery	USB
Size (mm)	19×18×4	23×23×2	38×25	51×46×15	49×36×13	32×25×10	40×16×5	57×42×24
Weight (g)	†	1.3	†	20	22	†	2.7	55
Enclosure	optional Strapmount	optional Misc	No	Yes	Yes	Optional	No	Yes
Extensible	No	No	Pin Headers	No	No	Pin Headers	Pin Headers	No
Programmable	No	No	Yes	No	No	No	Yes	No
Price (USD)	299	125	80	400	†	300	129	1,100

† Data not available

‡ Resolution is not specified, so we calculated it by $Resolution = \frac{Range_{max}}{2^{Resolution_{bits}}}$. The actual resolution might be higher at lower measurement ranges.

$*$ Depends on Firmware and Application.

Table 2.4: Properties of different IMUs. Numbers taken from specifications and datasheets.

2.5 Wearable Electronics Platforms

Electronic circuits embedded into textiles or worn on the body are often referred to as wearable electronics. Typical applications are for example wrist worn fitness trackers, sensors embedded into clothing to measure the performance of an athlete or actuators integrated into shoes or belts to provide a feedback to the wearer. We regard these applications as part of the field of ubiquitous computing, because they represent technology that merges with everyday objects. In this section we focus on systems that can be used to prototype or build wearable electronics. We surveyed three different platforms with different target areas of application such as integration into textiles or compact, wearable devices.

2.5.1 A platform for Body-Worn Devices: Xadow Kit

The Seeedstudio Xadow kit [23] consists of a main board as well as several extension boards, all 25.43×20.35 mm in size and connected by Flat Flex Connectors (FFCs).This allows to fold the resulting system and is more flexible than just a stack, especially for wearable projects. The main board is based on an Atmel Atmega32U4 and compatible to Arduino. Two electrically identical, 12 pin connectors on opposing sides of the board allow connections to other modules. The system is either powered through USB or an external LiPoly battery which can also be charged via USB through the main board. The main board costs 20 USD, including FFCs and a LiPoly battery. Currently (Feb 2015) there are 24 different extensions available at Seeedstudio, from a simple buzzer (7 USD) to a 9-Degrees-of-Freedom (DOF) IMU (40 USD), a BTLE module (30 USD) or a GPS module (44 USD). Xadow is well documented in the Seeedstudio Wiki[24] and hardware as well as software are open source.

2.5.2 A Sewable Arduino: LilyPad USB

The LilyPad Arduino USB is a product which is based on the Arduino LilyPad, see Figure 2.11, proposed by Buechley and Eisenberg of University of Colorado in 2008. [57] The LilyPad is basically an Arduino designed for integration into electronic textile applications. It has a round shape with a diameter of 50 mm and selected pins of the microcontroller are connected to through-hole pads on the edge of the PCB. The hole diameter is bigger than usual to allow sewing contacts to it using conductive thread. The LilyPad USB is based on an Atmel Atmega32U4 Microcontroller instead of the Atmega168/328 on the original LilyPad. This enables communication and programming of the microcontroller via USB instead of using a USB/Serial adapter board. The LilyPad USB also features a standard connector for LiPoly batteries

[23]http://www.seeedstudio.com/depot/Xadow-c-84_120
[24]http://seeedstudio.com/wiki/

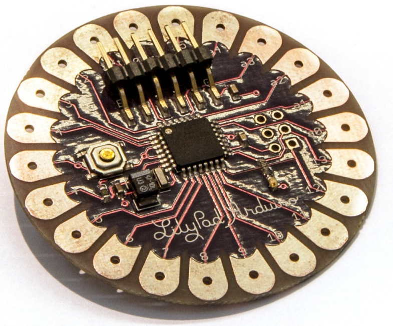


Figure 2.11: The Arduino LilyPad electronic textiles platform.

which can also be charged via the USB port. The microcontroller is supplied through an on-board 3.3 V regulator. As for all Arduino products, the LilyPad is open source and compatible with the Arduino IDE and most of the examples. Sewable extension boards containing sensors, actuators or switches are commercially available.[25] The LilyPad, along with the similar Adafruit Flora[26] is a popular platform for prototyping electronic textiles and often mentioned on websites and in publications especially focused on electronic textiles in an educational or fine arts context. [58], [59]

2.5.3 The LilyPad Alternative: SquareWear

SquareWear, see Figure 2.12, is a 44.8 mm^2, Arduino compatible microcontroller board including an Atmega328P controller, a rechargeable on-board 45 mAh coincell battery, a full color LED, buzzer, light sensor and temperature sensor. Three of the I/Os are wired to MOSFETs to allow switching high loads. All relevant pins of the microcontroller are wired to solder connectors on the edge of the board that are also optimized to be sewable like the Arduino Flora / LilyPad. The documentation consists of a manual and several tutorials. SquareWear is open hardware and costs 21 USD.[27]

[25]https://www.sparkfun.com/categories/135
[26]https://www.adafruit.com/products/659
[27]http://rayshobby.net/cart/squarewear/sqrwear-20

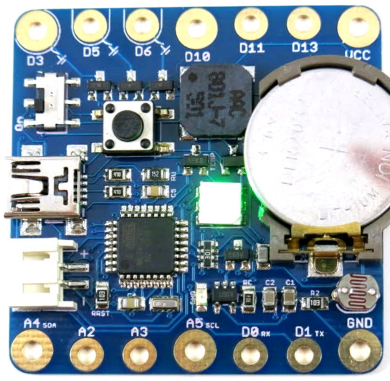


Figure 2.12: The SquareWear platform. Photo by Ray Wang, CC BY

2.5.4 Characteristics of Different Wearable Electronics Platforms

At the time of writing, not many dedicated wearable electronics platforms existed, so we could only survey a relatively small sample of 3 different devices. All systems we regarded are in fact regular microcontroller platforms optimized towards specific use-cases and use Arduino compatible 8 bit microcontrollers.
We identified two different main areas of application for the platforms we analyzed. SquareWear as well as the LilyPad Arduino are optimized for electronic textiles whereas the Xadow Kit is designed for Wearables. In the following, we present a closer look on those areas of application.

Electronic Textiles

Electronic textiles often use fabric as the base material for circuits. Connections are made with conductive thread sewn onto or woven into the fabric. In order to keep the complexity of those circuits low, thus reducing the amount of work and the error-proneness, groups of components are arranged on standard PCBs. Those building blocks can be an LED with a resistor, but also a microcontroller platform such as the LilyPad or SquareWear. There are special sewable extensions the LilyPad[28] that can also be used with SquareWear, which is probably the reason why there are no dedicated extensions for SquareWear. In order to be integrated into textiles, electronic textiles platforms have to be able to make contact by sewing conductive thread to through-hole pad. To make them more comfortable to wear when integrated into textiles, the PCBs of both the LilyPad and SquareWear are small and rounded. An integrated power supply, as found in the SquareWear is a valuable part of the elec-

[28]https://www.sparkfun.com/categories/135

tronic textiles building block because it eliminates the need for an extra battery. However, it can also limit the application in terms of power consumption, voltage or maximum current. In this case, attaching an external battery to a platform like the LilyPad increases the flexibility of the platform regarding the power supply.

Wearables

Apart from textile based solutions, electronics can be worn as accessories. The Xadow platform represents a prototyping solution for wearable devices like smart watches or fitness trackers. A flexible and compact design is achieved by separating blocks of components into smaller PCBs, connected by FFCs. Technically, the electronics are similar to the electronic textiles platforms, but the high density connectors allow a greater level of compactness.

Common Characteristics

In order to integrate electronics into clothes, they need to be small and compact and ideally contain a powerful battery. Wearable electronics, especially electronic textile projects are often only basic and implemented by designers and artists, not electronics experts. They do not require a powerful microcontroller, so all the analyzed platforms are based on simple, 8 bit microcontrollers running on clock speeds not greater than 16 MHz, see Table 2.5. Typical application areas are prototyping and education, so the platforms have to be easy-to-use, ideally Arduino compatible and well documented. The three wearable electronics platforms in our survey use the Arduino IDE for programming and a USB connection for uploading, communication between microcontroller and PC and charging a connected battery. All three platforms are relatively affordable and cost around 20 USD.

System	SquareWear 2.1	Xadow Kit	LilyPad Arduino
Microcontroller	Atmel Atmega328	Atmel Atmega32U4	Atmel Atmega32U4
Clock Frequency (MHz)	12	16	8
Flash (kB)	32	32	32
RAM (kB)	2	2.5	2.5
Application Area	E-Textiles	Wearables	E-Textiles
Power Source	Internal LiPoly, USB	Optional LiPoly, USB	Optional LiPoly, USB
Connectors	Solder Pads, Sewable	FFC	Solder Pads, Sewable
Onboard Sensors	Temperature, Light	None	None
Readily available extensions	None	14	~ 20
Size (mm)	44.8×44.8	25.43×20.25	18 Ø
Price	21 USD	20 USD	20 USD
Programming and Communication Interface	USB, ISP	USB, ISP	USB, ISP
Programming Language/Environment	Arduino IDE	Arduino IDE	Arduino IDE

Table 2.5: Properties of different wearable electronics platforms. Numbers taken from specifications and datasheets.

2.6 Former Work: The BRIX Toolkit

The BRIX system was developed by us as a compact, extensible and mobile physical computing toolkit in 2009. [60] Since the idea for the $BRIX_2$ platform is based on BRIX, we introduce our former work in the following section. BRIX already implements many features required for ubiquitous computing applications, but can still be significantly improved.

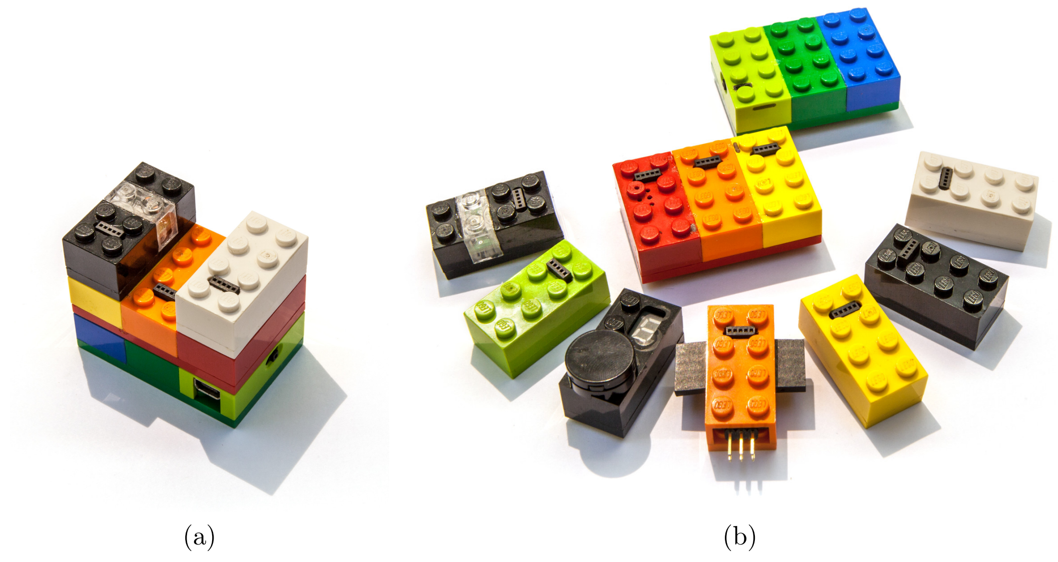


Figure 2.13: A typical BRIX stack (a) and the complete BRIX kit (b).

2.6.1 Vision

The system was originally designed for interactive applications that incorporate sensors to measure modalities like motion or pressure and actuators like vibration motors or speakers to provide instant feedback. Many of these applications included attaching the components to for example a musical instrument [61] or a human body [62], [63]. Therefore a compact and lightweight system was required, which operates on a battery and can be connected to a host device through a wireless interface, for example to record motion data. At the design time of BRIX, physical computing toolkits either lacked some of those required key features or were too expensive. For that reason, we decided to design a custom solution, the BRIX system.

2.6.2 Concept

BRIX consists of a base module (see Figure 2.13(b), center) and several extension modules, see Figure 2.13(b), which can be stacked onto the base module in order to extend the range of functions, see Figure 2.13 (a). The system is powered by a battery module (see Figure 2.13 (b), top) that mechanically and electrically connects to the base module. In order to operate the system, a host system like a laptop or smartphone is required, because BRIX is configured and controlled via Bluetooth. A server running on the base module manages data streams from and to BRIX and handles functions like auto-discovery of connected extension modules. Although the system is flexible and adaptable, standalone applications without a host system are not possible.

2.6.3 Hardware

The base module contains an RN41 Bluetooth transceiver module, an ATmega168 microcontroller, an ADXL345 3-axis accelerometer, an ITG3200 3-axis gyroscope and three 5-pin connectors for extension modules. All features we regarded as crucial in most applications we designed BRIX for were integrated in the base module, whereas other sensors and actuators were optional and implemented as extension modules.
A prominent feature of the BRIX system is the case design. In order to protect the electronics from impacts, we needed to encase them. As an additional requirement, since BRIX is a modular system, the enclosures had to be stackable. A mechanical connection can be implemented in many ways, for example by magnets, Velcro, snap-in or friction based. We decided for a friction based connection and found out that these need tight tolerances in manufacturing to be durable but also easy enough to part, which Lego has optimized throughout many years. Lego bricks are cheap and and can easily be modified, which made them a suitable material for our enclosures.
The extension connector has only 5 pins in order to save space on the PCB. The connector signals only include power supply and an I2C bus. All extension modules contain their own microcontroller which provides a uniform interface between the BRIX I2C bus and the component on the extension module, for example an Red, Green and Blue (RGB) LED with a driver chip.

2.6.4 Results

BRIX was a success for the applications it was originally designed for, namely streaming data from various sensors to a host and generating feedback of different modalities, controlled from the host. The modular structure of the system provided a lot of flexibility and BRIX was considered for more and more applications. However, although the hardware was adaptable to those applications, the firmware needed to be modified frequently. This required to open the enclosure of the modules and flash a custom firmware using ISP. This is a process that could only be performed by developers or expert users.
Regarding the electronics, we found that the protocol based extension connector leads to unnecessarily expensive and complicated extension modules, because they all require their own microcontroller. The wireless communication based on Bluetooth offered the advantage of direct compatibility to laptops and smartphones and did not require additional hardware on those devices. However, the paring process and communication handling in the host software caused problems. Also there is a limitation of a total of 8 devices in a Bluetooth network.
The enclosure design worked flawless, from a mechanical and also from a metaphorical point of view. Lego's friction based connection held all modules safely together but also allowed us to easily modify the stack. Although the system was usable in mobile applications, there was clearly room for optimization towards a more compact system. As soon as two or more extension modules were connected, the whole stack started to get bulky. The look of the system abstracted the inner workings and motivated even people without knowledge about electronics and microcontrollers to utilize BRIX for their projects.

2.7 Survey Conclusion

In our survey we have analyzed the technical specifications and implementation details of a total of 28 platforms from five different fields of application. Our goal was to systematically collect properties of platforms that were designed for applications related to the field of ubiquitous computing. We presented some representative platforms in greater detail, while we regarded only the technical specifications of others. For each field we compared the properties of all according platforms and devices in order to identify common characteristics, functionalities and features. It is to note that a unified comparison of all 28 platforms might be possible, however the outcome would not be significant because of the major differences between platforms from one field and platforms from another. Based on the refined lists of specifications we gained in this chapter, we are able to merge all relevant properties into a single concept for a platform that is applicable for teaching and prototyping in all five fields we covered.

3 BRIX$_2$ Design and Development

In this chapter, we derive the design space for the proposed BRIX$_2$ system from the results of our platform analysis in the previous Chapter. To achieve this, we first summarize and condense the results from Chapter 2. Subsequently we prioritize the properties and functionalities of platforms in each field by ranking them based on the number of occurrences. Accelerometers and gyroscopes for example are features that are present in every IMU device we surveyed. As a result, those sensors are mandatory for a platform that aims to serve as a prototyping and learning tool in the field of inertial sensing. By carefully balancing those features for each field we aim to address with our platform, we define the constraints and requirements for BRIX$_2$ regarding conceptual and purely technical aspects. Based on these specifications, we develop concepts for our platform that are integrated into a testing and evaluation platform, the BRIX$_2$ Development Kit (B2DK). With these prototypes, we are able to test and verify the technical aspects of our future BRIX$_2$ platform. The B2DK is designed as a modular platform, so we can easily test combinations of different microcontrollers, wireless transceivers and sensors to determine the optimal set of components for BRIX$_2$.

3.1 Defining the Requirements of BRIX$_2$ by Analysis of Other Platforms

For the different fields of application described in Chapter 2, we identified common properties and features among the designated platforms we analyzed. In the following we summarize those properties by field of application. After that we rate the importance of each property which allows us to prioritize them. As a result we can distinguish mandatory and optional features and functions, so in the design process, we are able to decide whether certain properties or features have to be included in a core set of functions, as optional extensions or can be left out completely.

Microcontroller Development Boards

Most microcontroller platforms in our survey were built around basic, 8 bit microcontrollers with peripheral features like ADCs, I2C, SPI, UART. A USB interface is common for programming and data transfer. GPIOs are broken out on either pin headers or solder pads and therefore accessible to users. On-board voltage regulators allow the board to be powered with different input voltages. LEDs display the status of the board and can be controlled via I/O pins on some platforms. Manufacturers usually provide an IDE and programming toolchain for their products. Detailed documentation and an active online community which shares knowledge and projects are also common among the microcontroller platforms we surveyed.

Physical Computing Platforms

In our survey we defined physical computing platforms as a central hub that connects external sensors and actuators to a PC or smartphone. Every platform we analyzed is a plug and play solution, so no soldering or electronics knowledge is required. All systems are modular and extensible by a high number of available add-on modules. Manufacturers support the integration of their platform into existing software frameworks. Comprehensible, non-technical documentation is available for beginners. Additionally, detailed technical documentation is often provided and allows developers to customize the platform for special needs.

Wireless Sensor Nodes

Among the motes in our survey, most devices are designed around a low power microcontroller with typical peripherals like ADC, I2C, SPI and UART. A low power wireless transceiver allows data transmission over distances of more than 30 meters. Internal batteries which are swappable or rechargeable are present in almost every WSN mote. Most of them are designed to be compact and feature extension headers for additional sensor boards, which are also supplied by the manufacturer. The technical documentation is often detailed but not suitable for beginners who are not familiar with certain terms and concepts.

Inertial Measurement Units

All IMU platforms we surveyed contain a microcontroller and accelerometers, gyroscopes as well magnetometers with three axes each. Their compact design facilitates applications like for example mounting the device to a human body. Some devices have integrated batteries and radio transmitters which allow completely wireless operation. Besides Radio Frequency (RF) transceivers, common data sinks are USB or mass storage like SD cards. Platforms in our survey were able to sample motion data with a rate of 150 Hz or greater.

Wearable Electronics and Electronic Textiles

The wearable electronics platforms in our survey are based on basic, 8 bit microcontrollers. The devices are small and and make the design of compact applications in the fields of electronic textiles or wearables possible. Some platforms have integrated power sources. Two of the devices we surveyed have sewable connectors that allows almost flawless integration into textiles.

3.1.1 Technical Feature Selection by Priority

From Chapter 2, we compiled a list of all features we could identify throughout all platforms we surveyed. For each feature and each field, we summed up the number of occurrences of a given feature in Table 3.1. We chose a star rating to provide a clear overview. One star means the feature is present in at least a single platform. Two stars mean that the feature is present in at least 50 % and three stars translate to 80 % or more platforms with that feature. From this rating, we conclude the priority of a feature or functionality and can sort them by relevance. In the following we split the most prominent properties into mandatory and optional features.

Mandatory Features

($\star\star\star$ in at least one category)

From our survey we conclude that our system needs to be based on an Arduino compatible microcontroller with an ADC and USB. A wireless transceiver is the key element of a wireless sensor mote and can also be used to stream motion data from an IMU. An inbuilt set of sensors consisting of accelerometer, gyroscope and magnetometer is mandatory for IMU applications and also interesting for wireless sensor motes and wearables. An internal battery provides power for the system in standalone operation, important for wireless sensor motes, IMUs and wearable applications. Extension headers enable access to the microcontroller and allow the system to adapt to specific applications by connecting it to external hardware. A compact design makes it possible to attach the system to a human body or embed it into textiles. An enclosure protects the device from external mechanical forces. Dedicated extensions for the platform enable users to easily tailor the system to their desired application. All software necessary for operating and using the platform should run at least on Linux, OS X and Windows. Open hardware makes it possible for users and developers to learn from the systems implementation and actively take part in developing it. Compatibility to the Arduino system makes the platform directly accessible to the high number of existing Arduino users. In addition, most of the documentation, programming examples and tutorials that already exists for the Arduino System also apply to Arduino compatible platforms.

Optional Features

(⋆ ⋆ in at least one category)

Optional features are not present in the majority of the systems we surveyed, but can still be considered for integration into our design. Wireless sensor nodes for example use a variety of different wireless transceivers. Some of them are IEEE 802.15.4 compliant, which provides the physical layer for protocols like ZigBee or 6LoWPAN. Other platforms also use a Bluetooth based wireless transceiver which is able to connect to mobile end devices like smartphones or tablets. A more mechanical feature which is solely present in electronic textile platforms are sewable connectors. However, since this property is rather exotic it is not a necessity for our design.

Feature	µC Devboards	WSN Motes	IMUs	PhysComp Platforms	Wearables
Programmable Microcontroller	⋆ ⋆ ⋆	⋆ ⋆ ⋆	⋆	⋆	⋆ ⋆ ⋆
ADC	⋆ ⋆ ⋆	⋆ ⋆ ⋆	†	†	⋆ ⋆ ⋆
USB	⋆	⋆ ⋆	⋆ ⋆ ⋆	⋆ ⋆ ⋆	⋆ ⋆ ⋆
Arduino Compatible	⋆ ⋆	⋆	⋆	⋆ ⋆	⋆ ⋆ ⋆
Wireless Transceiver		⋆ ⋆ ⋆	⋆	⋆	
IEEE 802.15.4		⋆ ⋆			
ZigBee		⋆			
Bluetooth		⋆	⋆	⋆	
Accelerometer		⋆	⋆ ⋆ ⋆		
Gyroscope			⋆ ⋆ ⋆		
Magnetometer			⋆ ⋆ ⋆		
Temperature Sensor	⋆	⋆	⋆		⋆
Barometer			⋆		
Mass Storage (Flash)		⋆	⋆		
Internal Battery		⋆ ⋆ ⋆	⋆		⋆
Extension Headers	⋆ ⋆ ⋆	⋆ ⋆ ⋆	⋆ ⋆	⋆	
Sewable Connectors					⋆ ⋆
Compact‡	⋆ ⋆	⋆ ⋆	⋆ ⋆ ⋆	⋆ ⋆	⋆ ⋆ ⋆
Enclosure			⋆		
Extensions Available	⋆	⋆ ⋆		⋆ ⋆ ⋆	⋆ ⋆
OS Independent*	⋆ ⋆ ⋆	⋆	⋆ ⋆	⋆ ⋆ ⋆	⋆ ⋆ ⋆
Open Hardware	⋆ ⋆	⋆	⋆	⋆ ⋆	⋆ ⋆ ⋆

† Data not available.

‡ smaller than 45×45 mm.

∗ Works on Linux, Microsoft Windows and Apple OS X.

Table 3.1: Occurence of platform properties in different areas of application.

3.1.2 Conceptional Properties of Analyzed Platforms

After we defined the required technical features and functionalities for BRIX$_2$ derived from our analysis of other platforms, we take a brief look on other properties that we can potentially adopt into our system. Conceptional properties refer mostly to aspects that concern the relationship of users and our platform. A concept would for example be to design a toolkit that allows for a motivating and positive learning experience by offering a low threshold entry and at the same time challenges and opportunities for skilled users. In this section we not only point out the properties of other platforms but also start to work them into our concept for BRIX$_2$.

Modularity

The proposed BRIX$_2$ system is expected to adapt to a great number of different applications and scenarios. To ensure this versatility, we can follow examples like the Arduino, littleBits or WaspMote, which are easily reconfigurable and/or extensible, so the full scope of features has not necessarily to be defined at design time. Since we also aim for a physically compact platform, parameters like size, weight and form factor, we are more restricted regarding the design of a modular system than for example the Arduino Mega. Its relatively big PCB offers a high number of easily accessible pin headers and a lot of space for potential add-on modules. But in our survey, we also identified examples for compact and yet modular platforms like the Xadow kit using FFCs or Tinkerforge Bricks using small PCBs and high density connectors. We often found that modular platforms consist of a base board along with smaller extension boards. For BRIX$_2$, the base board or module would contain mandatory functions like we identified in the previous section, so many application scenarios can already be covered without extensions. Several ports on the base module allow users to connect extension modules and thereby adding the function their application requires. This way, the system is organized into functional building blocks. The extensible structure also allows to add more features after the initial design by just building new extension modules.

Programmability

In contrast to platforms with a fixed firmware like we find among IMUs and physical computing platforms, reprogrammable systems like the Arduino can be modified and adapted to applications they were not specifically designed for. This is a crucial requirement for BRIX$_2$ so it can serve as a versatile tool for prototyping. Even though programability increases the flexibility of a platform, it also requires the user's ability to actually program it. Especially for people without previous experiences in programming computers, this can represent a significant challenge. If we still want to make our platform accessible to this group of users, we need to create an environment that motivates and facilitates their learning process, unlike for example the TelosB platform, which is only sparsely documented. Again the Arduino toolkit offers a great example for a low threshold learning experience for beginners and at the same time a rich set of features that also allows skilled users to implement advanced applications. In Chapter 2 we have identified a number of systems like Grove, littleBits, WaspMote, IMUduino or Xadow that are compatible to the Arduino IDE. They take advantage of the open source philosophy and adopt the well tested IDE, the existing knowledge base as well as the enormous user base. We also consider this a great opportunity and feasible option for the BRIX$_2$ system.

Openness

Our survey shows that open source systems like the Launchpad, Arduino or Tinkerforge Bricks allow users to learn and to profit from those implementations. By making schematics, PCB layouts and software available to the public, manufacturers also encourage users to participate in further developments of those components, create their own extensions or even fork their own project based on existing design. Platforms like LilyPad, IMUduino or Grove would most likely not exist if the Arduino project was not open source. The design process of BRIX$_2$ also profits from openly available information, schematics and software, so we will also gladly share our complete work with users and other developers by releasing it under an open source license.

3.2 Towards an Initial BRIX$_2$ Design

Moving towards an implementation of BRIX$_2$, our next step is to weave our own ideas as well as concepts and properties of platforms we surveyed in the previous chapter into our own design. We have already sorted, prioritized and regarded many features as relevant for our own application in the previous section. In this section, we clarify our intended fields of use for BRIX$_2$ and their implications on the design. Afterwards we discuss how we can fulfill the technical requirements for BRIX$_2$, beginning with mechanical and appearance aspects, followed by software and finally electronic aspects.

3.2.1 General Concepts and Usage Scenarios

In the following, we present some general concepts for the BRIX$_2$ platform which are the fundamental guidelines for the design of our implementation.

Learning and Teaching with BRIX$_2$

One of our major objectives is to develop an efficient tool for teaching and learning electronics, sensor technology and microcontrollers as well as topics that base upon this fundamentals. We define three key requirements for an efficient learning and teaching tool:

Getting started quickly and easily: Lecturers should not need to spend a lot of time introducing the tool and its usage. On the contrary the students should be able to explore the system on their own after a quick guided introduction or even all by themselves.

Constant Motivation: In order to keep students motivated, the tool needs to be adaptable to different skill levels. Whereas the entry level is supposed to be as easy as possible, lecturers should be able to use the tool to challenge even advanced students. If students use the tool on their own, their creativity and strive for more knowledge, they should not be limited by the tool. An approach to this might be the use of different abstraction layers which can be removed as the lecture progresses, revealing more and more complex details about the inner workings of a system.

Adaptability to Different Scenarios: To apply the toolkit in a variety of different lectures, it needs to be flexible and rich of features. We already broadly discussed this adaptivity in Section 3.1 when we derived requirements for our platform from multiple related fields.

Prototyping

Besides teaching, another area of usage for our system is prototyping. Here, some requirements are very similar to the ones stated in the previous paragraph. Besides the flexibility to adapt to numerous applications we already listed for learning and teaching scenarios, it is important for developers to on the one hand achieve what they want quickly and easily. However, on the other hand, in many cases a direct control over the hardware beyond all abstraction layers has to be possible in order to modify or customize the platform to match a certain application it was not originally designed for. We also aim to create a platform that can serve developers through most of the development process and not for example only in an early evaluation state. Ideally we can support the prototyping process up to a final product.

Existing Knowledge Bases and Transferable Knowledge

Vast amounts of knowledge about electronics, sensors, physical computing and the like have become available through the Internet and add to the knowledge that exist in the form of more classical media such as books or magazines. The less we specialize our system technically, the more of the common knowledge can be applied to our platform. This is applicable to an extent where our users could actually find most of their questions and problems regarding our system answered by external sources without relying on support by developers. The other way around, if our platform is general and not specialized, students can apply the knowledge they obtained using our platform on other problems and systems. This is an important aspect, since we aim to demystify technology by allowing easy access and understanding of a subset of that technology. If users then can transfer this knowledge and use it as a base to understand more and different subsets of technology, our system has had a key impact on the way the look at the technology that surrounds us every day.

3.2.2 Mechanical Design Aspects

In the following we present approaches to implement mechanical and appearance properties we selected in Section 3.1. We begin with the physical appearance of $BRIX_2$ and the aspect of modularity, which both define constraints for the later mechanical design of our platform.

Physical Appearance

With $BRIX_2$ we aim to supply users with building blocks that allow them to implement their desired application, similar to the idea of littleBits. The same way software libraries abstract complex functions in a programming language, we plan to abstract complex electronic functions to simple modules that can be used as a building material. This abstraction could be achieved by enclosing all electronics components so the electronics are not visible anymore. In order to simplify and unify the design, all extension modules should have a similar or even the same enclosure design. To distinguish between extension modules of different types, a color coding could be used.

Extensibility and Modularity

In Section 3.1.2, we pointed out that a key feature of the proposed BRIX$_2$ system is extensibility in order to adapt to as many applications as possible. From a mechanical point of view, the concept of modularity involves a number of challenges:

Mechanical Connections: In our survey we found mechanical connections between different modules on many platforms rely solely on pin headers. In order to ensure a stable mechanical connection, more than one row of headers is needed to reduce the leverage of the connected module. If only one header is present, a supporting mechanical connection is required. This would have to be considered when designing the enclosure.

Electrical Connections: In Chapter 2 we identified different kinds of electrical connections between base units and extension units. Cable solutions aside, the most prominent connectors are female/male pin headers with a 2.54 mm (0.1 in) pitch, which takes up a lot of space. An alternative are high density connectors like the Hirose DF17 series [64]. Their downsides are a weaker mechanical connection and poor accessibility of the single pins. A further parameter to consider for electrical connectors is the number of insertion/removal cycles. If it is too low, the connectors will break or become unreliable after only a short time of usage.

Enclosures for a Modular Platform: A unified mechanical and electrical interface allows to connect any extension module to the same port on the base module. This connection has to be mechanically stable, so the enclosures themselves have to mate. The connection should also be easy to part, so several different types of mechanical connections will have to be evaluated in order to find a suitable method. We already implemented a promising approach with the BRIX project, see Section 2.6, which is well worth reviewing.

Size, Weight and Form Factor

The physical size of BRIX$_2$ matters especially in mobile applications like wearable electronics or motion capturing, where the device must neither be too heavy, nor too big. In our survey we found several examples for this compact and light type of platform. IMUs for example are designed to be small, compact and light to facilitate especially motion capturing scenarios when the sensors are body-worn. Wireless sensor nodes also profit from a compact design so they can easily integrate into all types of environments. For IoT applications, a small and lightweight system can be embedded into a variety of everyday objects in order to monitor and enhance their functionality. [65] Compactness can be achieved through a high level of integration, which means to place a lot of functionality on a small space by efficient design. Unfortunately a modular system structure counteracts the principle of high integration, because it comes with the additional space requirements of electrical connectors and also additional enclosures. This is the main reason why we take great care to integrate as much functionality onto the base module as we can to keep the overall device required for a specific application rather small. Should further functionalities be required, adding them by attaching extension modules always comes at the price of increased size and weight of the BRIX$_2$ stack.

3.2.3 Software Aspects

Before we select electronic components for our proposed system, we take a closer look on solutions for a software framework, which will influence the design of our hardware. In the following, we consider the two major required software components for our platform. First, the framework and language that is used to write firmware for the microcontroller. Second a number of tools required to compile that code to a binary and transfer it onto the microcontroller. Since existing solutions for both aspects exist, we are likely to adapt one of them, given it fits the requirements of our platform.

Firmware Programming Language

On the lowest level, a microcontroller can be programmed in assembly language, which provides precise control over every hardware component of the controller. On the one hand, this enables programmers to write very efficient code. On the other hand, it is a complex, unintuitive task and requires expert knowledge and understanding of the inner workings of a microcontroller. Since the design of BRIX$_2$ is also focused on beginners, we need a language that is more abstract and easier to understand. In our survey, the C programming language is used for a major number of platforms, most prominent on the Arduinos and compatible platforms. Also outside of the microcontroller context, C is a widely used programming language. In order to run the code written in a high level language on a microcontroller, it needs to be compiled into a binary code that can then be transfered to and executed by the microcontroller. Each type of microcontroller needs its own compiler or compiler settings because the device architectures differ. This is one reason why the selection of a programming language and compiler toolchain affects the selection of hardware components like the microcontroller.

Microcontroller Programming Toolchain

After the code is compiled, it needs to be transfered onto the controller. This is done by special tools or even combination of such running on the development PC called a programming tool. It either connects to the microcontroller directly via USB or a USB/Serial adapter or through a dedicated piece of hardware, a programmer. The binary is copied into the flash memory of the microcontroller. After that, the controller is reset and starts to run the program.

Arduino IDE

The Arduino IDE is the software package supplied along with the Arduino microcontroller development boards. Not only does it contain an editor for writing firmware for Atmel ATmega microcontrollers, it also integrates an open source toolchain consisting of a C compiler and a flashing tool. The whole software is operating system independent and configured in a way that it allows even beginners to just use it our of the box without any microcontroller or programming experience. As we already mentioned in Section 2.1.7, the Arduino is a de-facto standard among the development boards and physical computing platforms [66]. This means that a lot of documentation, tutorials and code examples as well as a solid user community does already exist. By designing our proposed platform to be Arduino compatible, parts of the documentation, examples and project that are based on the Arduino platform will also apply to BRIX$_2$. This way our platform profits from many achievements the Arduino developers as well as the community made in the last 10 years.

3.2.4 Electronics Design Aspects

As a final step, we approach the electronic design of the BRIX$_2$ platform. This does not only include a selection process for key components like the microcontroller, IMU sensor, wireless interface and power supply, but also to solve the challenge of reliably connecting multiple boards of a modular system.

Connectivity

Common in the system design of almost any development board, physical computing toolkit or wireless sensor node are at least two types of connectors. A PC connection for programming and data transfer and one or more connectors to attach external hardware.

Host System Connector Almost any re-programmable device relies on a data connection to a PC in order to upload new firmware and debug or control firmware during runtime. This connection can either be a dedicated programming or debugging interface like ISP or JTAG as well as a serial data connection like RS232 or

USB that communicates with a bootloader on the target device. A serial data connection has two important advantages over a dedicated programing interface. First, no special hardware is required for connection, because the serial data connection is compatible with common PC interfaces. Second, the serial data connection is general purpose and can also be used for other communication with the target device and are not restricted to programming and debugging. In modern platforms, the host system connector is usually USB. A major reason for this is the compactness and versatility of a USB header, which is smaller than for example a standard, D-SUB-9 or DIN RS232 header. Also, USB also offers power supply, which RS232 does not, so the 5.0 V supply voltage of the USB host device can be used to power the connected system or even charge its internal batteries.

Extension Connectors Not only the mechanical type of connector has to be considered but also the organization of electrical signals. In Section 2.2, we found two different strategies for the pinout of extension connectors.

- **Full Breakout:** All or most of the GPIOs of the microcontroller as well as supply voltage and ground are broken out to a connection header. This was commonly found on most development boards. The full breakout is flexible because it can connect to any hardware in a lot of different ways. The downside is that the number of pins that have to be broken out is usually not less than 10 or 20. This requires a big standard connector like a 2.54 mm pitch pin header or a compact but fine pitch connector which is hard to solder and interface.

- **Protocol Based:** A different strategy that allows a low pin count on the connector are bus based extension connections. These can be found for example on systems like BRIX, littleBits or Grove. The connector usually contains supply voltage, ground and a small number of bus lines like I2C (data line and clock line) or a small selection of GPIOs (for example one analog input, one digital I/O). This kind of interface relies on a protocol which defines how external hardware must connect. All external hardware necessarily needs to be compatible to the connector standard. For example on the former BRIX system, all extension modules had to implement an I2C slave. Protocol based connectors are electrically and mechanically compact and simple but also cause a more complex design process for external hardware along with higher component counts and costs.

Microcontroller

To create a reprogrammable platform, we need to design it around a microcontroller. This requires a careful component selection, because the type of microcontroller will not only fix parameters like memory space and computing power, but also the quantity and quality of the documentation which differs between manufacturers and product lines. To ensure the latter, we already decided to be compatible to the Arduino platform which makes it possible for us to apply the already existing massive knowledge base to our product. In the following we briefly describe the technical requirements for a BRIX$_2$ microcontroller before we discuss different, Arduino-compatible microcontrollers.

Requirements for a BRIX$_2$ Microcontroller In Chapter 2 we observed that among the programmable platforms, 8 bit microcontrollers are still common. Especially for applications in education and prototyping, a rather simple type of microcontroller is more feasible than for example a complex, high speed 32 bit processor. In many cases, an 8 bit microcontroller with 1 kB SRAM and 32 kB program memory is totally sufficient and in fact it is surprising what those systems are capable to accomplish. [67], [68] Besides the performance of the microcontroller, a number of other features are crucial:

- **Communication protocols implemented in hardware** are necessary in order efficiently communicate with sensors, other microcontrollers or a host system. Required protocols are for example UART, I2C and SPI.
- **Internal analog-to-digital conversion** is needed to measure analog voltages on an input pin of the microcontroller. This functionality is frequently used in order to read certain sensors, for example analog accelerometers, light sensors or potentiometers via a voltage divider.
- **Native USB support** allows the microcontroller to communicate with host devices like a PC via USB. This eliminates the need for a serial-to-USB conversion and level shifting electronics. The USB communication can be used for data transfer as well as programming via a bootloader and is common on most modern microcontroller platforms we analyzed.
- **Low energy modes** are special states the microcontroller can enter to drastically reduce the power consumption. This is applicable for example when the microcontroller is waiting for an external input.

Mechanically, the chip itself needs to be reasonably small to facilitate a compact PCB design. The package should also be solderable without professional equipment in order to achieve a reproducible system. This excludes for example Ball Grid Array (BGA) packages. Another requirement is sufficient documentation of the microcontroller. This does not only involve a detailed and well-structured datasheet but also an online community centered around that particular controller or controller family. The more people use a certain microcontroller, the easier the troubleshooting online. For the most popular microcontrollers, virtually any problem a beginner will face has been solved many times and is well documented on the Internet. All those requirements are met by the Arduino platform, which makes it feasible to also design BRIX$_2$ as an extended, Arduino compatible platform. We can thereby conclude that Arduino compatibility is a major requirement for a BRIX$_2$ microcontroller.

Arduino Compatible 8 Bit Microcontrollers In general it is possible to use microcontrollers other than the ones used in the original Arduinos with the Arduino IDE. [69] However, this usually requires changes in the IDE that have to be made by the user. In order to be compatible to Arduino out-of-the-box, we need to use a microcontroller that is actually used in one of the original Arduino implementations. This leaves us with a small collection of three different microcontroller series:

- **ATmega168/328**, used in the Arduino Uno, (Pro) Mini, Nano, and Lilypad [70]. This 8 bit microcontroller series is produced by Atmel. The controller can run at clock speeds up to 20 MHz, has 32 kB flash memory, 1 kB EEPROM and 2 kB SRAM. The ATmega168/328 can communicate via native UART, SPI and I2C. It has 23 GPIOs, eight are connected to an internal, 10 bit ADC. The controller has three programmable timers/counters and five different power saving modes. It is available in Dual InLine Package (DIP)28, Thin Quad Flat Package (TQFP)32 (7×7 mm), 4x4 BGA32 (5×5 mm), Very Thin Quad Flat No leads (VQFN)28 (4×4 mm) and MicroLeadFrame (MLF)32 (5×5 mm) packages. [71]
- **ATmega16/32U4**, used in the Arduino Leonardo, Esplora and Micro is an 8 bit microcontroller produced by Atmel. It runs at clock speeds up to 20 MHz, has 32 kB flash, 1 kB EEPROM and 2.5 kB SRAM. It is a USB 2.0 full speed device and can also communicate via native UART, SPI and I2C. The controller has 26 GPIOs, twelve are connected to an internal 10 bit ADC. It has four programmable timers and six sleep modes. The ATmega 16/32U4 is available in a TQFP44 (12×12 mm) and Quad Flat No leads (QFN)44 (7×7 mm) packages. [9]
- **ATmega 1280/2560**, used in the Arduino Mega and Mega 2560 is an 8 bit microcontroller series by Atmel. It runs at clock speeds up to 16 MHz, has 128 kB flash, 4KB EEPROM and 8 kB SRAM. It can communicate through four UARTs, SPI and I2C. The controller has 86 GPIOs as well as a 16 channel internal 10 bit ADC and features six programmable timers and six sleep modes. It is available in TQFP100 (14×14 mm) and BGA100 (9×9 mm) packages.

Wireless Networking

From Table 3.1, it becomes clear that among the fields of application we surveyed wireless transceivers are mainly required in WSN and IMU applications. However, there is a major difference in the requirements that both of these fields have regarding the wireless transceiver. WSN applications generally transmit low volumes of data with a low update rate of $\leq 1\,\text{Hz}$. In IMU applications on the contrary, data is streamed with a high update rate, up to $1\,\text{kHz}$, see Section 2.4.10. Also, WSN applications require more flexibility regarding the network topology whereas IMUs usually just use a star topology to send data from n nodes to a central device. For WSN prototyping and IMU applications as well as IoT applications inside buildings, the range of the wireless transmitters should be at least 30 meters. In our survey we can not clearly identify any protocol layer that is commonly used, so we will subject this matter to a further evaluation. Besides choosing a suitable protocol for wireless data transmission, our application needs a small, inexpensive wireless transceiver module with a standard interface to the microcontroller (I2C, SPI, UART). Furthermore the technology should be transparent, well documented and have an existing user base.

Sensors

Some platforms in our survey include different kinds of sensors, ranging from simple ambient sensors like temperature and light sensors to complex, smart devices like MEMS motion sensors. Although most of those sensors do not appear in enough platforms to make that particular feature mandatory for our platform, motion sensors are crucial. By targeting applications in the field of IMUs, motion sensors become a central element of our system, since the field is based on a combination of those particular sensors. But inertial motion sensors can not only be found in the field of IMUs: there are extension boards containing those sensors available for platforms in the other fields as well, for example in development boards [72], WSN [73], electronic textiles [74] and wearables [75]. This clearly illustrates the demand for motion sensors in all targeted fields. In Section 2.4.10 we analyzed the properties of all IMU sensors in our survey. We found that more than 50 % of all systems use MEMS sensors that combine at least an accelerometer and a gyroscope into a single package. Prominently, Invensense[1] sensors are used. Magnetometers are either part of the integrated solution or a separate chip like for example the Honeywell HMC-5883L [76]. The manufacturers of the other IMUs do not provide information about the type of sensors in their product, which are most likely not consumer grade devices.
Since we aim for a compact, cost efficient and reproducible system, an off-the-shelf, single-chip consumer grade solution that includes all three sensors would be ideal. The data interface should be either SPI or I2C for easy data acquisition via the microcontroller. The average ranges and update rates of the sensors we analyzed are summarized in Section 2.4.10 and should be used as a guideline when selecting

[1]http://store.invensense.com/

a specific sensor for our implementation.

Power Supply

In many cases, an electronic application can be powered externally. However, 80 % of the platforms that we analyzed are either designed to operate independent of an external power source or at least allow this as an option. In areas like motion capturing, wearable electronics and wireless sensor networks, platforms usually rely on an internal rechargeable battery to power the application.

Power Requirements In order to use BRIX$_2$ in motion capturing and wearable electronics scenarios, we aim for a battery runtime of 3-5 hours at full load, which we regard realistic for prototyping purposes. This means for example high frequency wireless streaming of data and powering external components like LEDs or speakers. Because BRIX$_2$ is supposed to be small and lightweight, we need a power source that can also fulfill this requirement. The user should be able to charge BRIX$_2$ as simple as charging a smartphone. This means the battery should not have to be removed for charging. A high number of charge cycles before losing capacity and a low self-discharge rate is crucial.

Battery Runtime The runtime of a battery powered system is basically determined by two factors:

- **Battery Capacity:** The capacity defines how much energy can be stored in a rechargeable battery. The amount of energy stored in a battery with a certain mass is given by the energy density and varies for different battery technologies. Hence the higher the energy density, the smaller or more lightweight a battery can be with constant capacity.
- **Energy Consumption of the Application:** Active components like microcontrollers, displays and wireless transceivers consume most of the energy in a mobile system. In order to keep the runtime of the battery long, the energy consumption has to be as low as possible. This can be achieved by selecting energy efficient components and through intelligent use of those components. Microcontrollers often implement a number of different energy saving options like sleep modes that can be entered if the application is not using the microcontroller in that period of time. The same applies to wireless transmitters. Their energy consumption peaks when transmitting data. By keeping transmissions short and the transmission frequency low, a lot of energy can be saved.

Battery Technologies There is a number of rechargeable battery technologies that can be considered for an application like BRIX$_2$. The main parameters we consider are size, weight, capacity, number of charge cycles before significant loss of capacity, maximum charge and discharge currents and typical output voltage. The following information and data is taken from [77], [78], [79], [80] and [81].

- **Nickel Cadmium Batteries** usually come in an AA or AAA form factor. This limited number of shapes and their size require small applications to be designed around the battery for an optimal packed product, see for example Section 2.3.1. The AAA form factor with $44.5 \times 10.5\,\text{mm}$ Ø is the smallest common NiCd standard. A device containing such a battery will be at least $50 \times 15\,\text{mm}$ in size. Another problem with those batteries is the low output voltage of around 1.2 V. This requires the electronics to either operate on this voltage or to be powered through a boost converter circuit. Alternatively, more than one battery can be used in series. NiCd batteries have a relatively low energy density and are also relatively heavy, see Table 3.2

- **Lithium Ion Coin Cells** are relatively small and have a flat, round form factor. They are designed for low power applications and thus have only a low capacity. Because of their flat shape, they can easily be integrated into PCB layouts and treated like a regular electronics component. Their output voltage of around 3.6 V is sufficient to power most modern active components like microcontrollers and sensors.

- **Lithium Polymer Batteries** are often used in so called *pouch* form factors. This refers to a relatively flat and rectangular shape. They can be purchased in almost arbitrary sizes from the size of a stamp to the size of a VHS tape. It is easy to find a matching battery for a given application, so the design process of a product has not be centered around a certain battery size. Lithium polymer batteries are light and have a high energy density which makes them perfect for mobile applications with a high energy demand like mobile computers. Their high charge currents lead to relatively short charge times.

Power Management In the BRIX_2 system, we will most likely have two different power sources: USB and the battery. This requires some dedicated electronics for tasks like power selection, voltage regulation and charge management. BRIX_2 is extensible which means that external hardware can be connected that also has to be powered. This is why we have to potentially supply much more power than required by the BRIX_2 module itself. We should aim for a voltage regulator and a charge controller that can handle up to 500 mA, which we regard sufficient even for externally connected high power LEDs and motors.

Parameter	Ni-Cd AA	Ni-Cd AAA	Li-ion Coin Cell	LiPoly
Energy Density (MJ/L)	0.18 – 0.54	0.18 – 0.54	0.9 – 2.63	0.9 – 2.63
Typ. Density (g/cm^3)	0.7	0.99	0.6 – 0.7	0.5
Typ. Size (mm)	50×14 Ø	44.5×10.5 Ø	10 – 40 ×2 – 8 Ø	any
Typ. Voltage (V)	1.2 – 1.25	1.2 – 1.25	3.6	3.7
Typ. Capacity (mAh)	600 – 1000	300 – 500	10 – 250	50 – 600
Typ. Charge Rate (C)	1.0	1.0	0.5	0.5 – 2
Typ. Discharge Rate (C)	11.5 – 2.0	1.5 – 2.0	0.5	0.5 – 2
Typ. Number of charge cycles	500	500	500	500

Table 3.2: Comparison of different battery techologies.

Visual Feedback: LEDs

Apart from a serial data connection, LEDs are a valuable tool to provide simple feedback from a microcontroller to the user. This can for example be debug information about the running firmware or a status display. Using a single LED, there is a limited variation of different signals that can be distinguished and interpreted by the user. To provide a broader feedback, we can equip BRIX$_2$ with a full color LED which can display almost arbitrary colors by mixing red, green and blue components. Users should be able to turn off every LED in order to save energy in low power applications.

3.2.5 Initial BRIX$_2$ Design: Conclusion

In this section we have narrowed down our design space by defining detailed requirements and presenting approaches towards an implementation. We covered mechanical aspects like connectors and enclosures and discussed the Arduino IDE as a software framework for our system. Finally we discussed electronic components that fulfill the requirements we proposed earlier. This includes an Arduino compatible microcontroller, a versatile wireless transceiver, different sensors and rechargeable, internal power supply. In the next section, we take the next major step towards BRIX$_2$ with the B2DK, a system that will enable us to test our concepts and evaluate different electronic components.

3.3 The BRIX$_2$ Development Kit

To evaluate and test different technical aspects and concepts of a future BRIX$_2$ platform, we required a first hardware prototype. For this purpose we developed a modular system that allowed us to test various hardware and software components and configurations. The B2DK, see Figure 3.1, simulates a future BRIX$_2$ base module on a functional level. In this section, we present the design of the B2DK and list all components we put to the test. Subsequently we discuss the results of our evaluation, which leads to a final selection of electronic components and paves the way for the design of the final BRIX$_2$ implementation we present in Chapter 4.

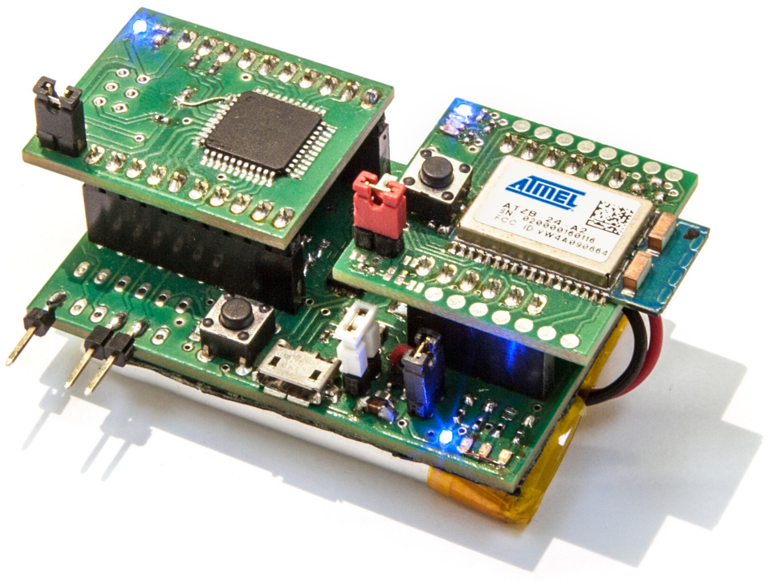


Figure 3.1: B2DK carrier board with Atmel ATmega32U4 microcontroller board and Atmel ZigBit board.

3.3.1 A Modular Platform

The B2DK has a modular design, so components like the microcontroller or the wireless transceiver can easily be swapped, which makes it a versatile development tool. By moving components that are to be evaluated to dedicated PCBs, we can test a variety of different configurations of the whole system without a complete redesign. At the same time, we still tried to keep a compact form factor. Of course we could have utilized evaluation boards that manufacturers offer for components like microcontrollers, sensors and wireless transceivers. However, all those boards would be different and not have a standard interface to integrate them all to one system. By designing custom boards, we not only gained flexibility, saved time during evaluation and were able to do tests with equal conditions but also could reuse the boards we developed in later projects after our evaluation, for example in [82].

3.3.2 Carrier Board

The carrier board contains all components of BRIX$_2$ that are either already fixed or non-critical, like for example the LED display or the power management circuitry. Three different ports on the carrier board can be used to connect microcontroller boards, wireless transceiver boards and external electronics or measurement equipment. Every active component can be detached from the power supply using jumpers which also allow to measure the current of each active component. LEDs indicate the status of different power lines and charge status of the battery. The carrier board was not optimized for a small size, but measuring 37×58 mm, it is still compact enough to test mobile applications. Power for mobile applications is supplied by a LiPoly battery which is mounted on the bottom. In the following, we will describe all elements of the B2DK carrier board in detail before introducing the extension boards.

Power Supply and Power Management

In the following, we introduce the power supply and management electronics of the B2DK, see Figure 3.2. Just like the planned BRIX$_2$ base module, the B2DK includes a battery which can be charged via USB. The system can also be powered by the USB connection while being charged. We chose a single-cell LiPoly battery with a capacity of 1000 mAh and a nominal output voltage of 3.7 V [83]. LiPoly batteries provide a high energy density and are thus small, lightweight and inexpensive (see Section 3.2.4), which makes them perfect for a compact, mobile platform like BRIX$_2$. The capacity of the battery, which is determined by the size, will most likely be different in the final application, but the electronics necessary to charge and discharge the battery are identical for any single cell LiPoly battery. Multiple cell LiPoly have a higher output voltage but require to balance the voltage of all cells in order to prevent irregular current flows. [84]

LiPoly Charging For LiPoly batteries, a constant current - constant voltage charging method is used. The battery is charged with a constant current at around 1 C, which is the total capacity. As soon as a cell voltage of 4.2 V is reached, the battery is charged with a constant voltage of 4.2 V until the charge current drops to almost zero. [85] This charging procedure can be automatically controlled by dedicated LiPoly charging controllers. In our application we chose a Microchip

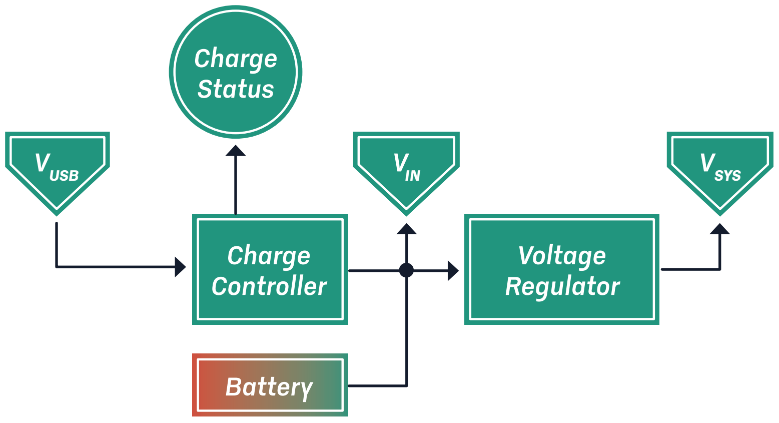


Figure 3.2: B2DK power management electronics block diagram.

MCP73831 single-cell Li-Polymer charge management controller [86] because its parameters match our application requirements and we already had a well-tested reference design. [87] The charge current is programmed to 500 mA using a fixed resistor. For the B2DK battery this is only 0.5 C, but in the final application, we will use a battery with a lower capacity. The only side effect of a lower charge current is a longer charge time. A red LED is connected to the $STAT$ signal to indicate the charge status of the battery. It can be separated by a solder jumper. The charge voltage is supplied by the USB port of the B2DK carrier board. A yellow LED indicates the condition of the USB input voltage and can also be separated via a solder jumper to cut the overall power usage or during power measurements.

Voltage Regulation The active components of the B2DK are selected for a 3.3 V system voltage. This means we have to regulate the battery (4.2 V max.) or USB (5.0 V) input voltage to match that requirement. We chose a Microchip MCP1603L switching regulator, because it is more efficient than a linear regulator and handle input voltages down to 2.7 V, which is far below the battery cutoff voltage of 3.0 V [88]. The MCP1603L supplies output currents up to 500 mA, which is sufficient for the BRIX$_2$ system and requires only small passive components, so it matches the size restrictions of our final application. In our circuit, two jumpers allow to measure the current before and after the regulator. LEDs indicate the condition of input voltage and output voltage of the regulator. The LEDs can be separated by solder jumpers to lower the overall power usage or during power measurements.

IMU Sensor

Since compactness is a major design aspect for BRIX$_2$, we favor a single chip solution for our IMU in contrast to separate ICs for each component. The Invensense

MPU60xx, used in almost 50% of all IMU devices of our survey, see Table 2.4, integrates a 3-axis accelerometer and a 3-axis gyroscope as well as a digital motion processor into a 4×4×9 mm QFN package [89]. The specifications of the MPU60xx regarding sensor ranges, resolution, update rate and power consumption are well within our requirements. The platforms in our survey add an external magnetometer to the MPU60xx to compensate the gyroscope drift in the sensor fusion step.

Invensense MPU9150 At the design time of the B2DK, Invensense just released the MPU9150, which is basically in MPU6050 with an integrated AKM AK8975 [90] magnetometer. This way, Invensense created the first 9-DOF IMU with integrated sensor fusion unit in a single, 4×4×1 mm QFN package. With a resolution of 0.3 µT / Least Significant Bit (LSB) and a range of ±1200 µT, the magnetometer of the MPU9150 lies within the average of the magnetometers in our survey. We regarded the internal sensor data fusion as a great opportunity to improve the performance of BRIX_2 in motion capturing scenarios. Since there were no other products at design time that offered this feature, we decided to integrate the sensor as a fixed part into our carrier board design.
The circuitry required to run the MPU9150 does only consist of a few capacitors to stabilize the supply voltage and to run the internal charge pump for the gyroscope. We connected the *"Frame Synchronization"* input ($FSYNC$) to a solder jumper just in case we have to interface that signal. The interrupt pin as well as the I2C interface are connected to the microcontroller expansion port. The I2C address selection pin ($AD0$) is connected to GND, setting the LSB of the device's I2C address to 0.

Expansion Ports

The B2DK carrier board contains three different extension ports: one for microcontrollers, one for wireless interfaces and an auxiliary extension port, see Figure 3.3. The ports are reverse polarity protected by design and offer a mechanically stable base for the add-on boards. The microcontroller and the wireless port have different sizes so the extension boards can only be plugged into their designated ports. Each of them consists of two rows of female pin headers with standard pitch of 2.54 mm and a spacing which is a multiple of the pitch. This allows us to plug the extension boards into a standard solderless breadboard for testing.

Microcontroller Port The microcontroller port consists of two 10-way female pin headers, so a microcontroller board can be mounted mechanically stable on top of the B2DK carrier board. The pinout of the headers is designed to prevent permanent damage to either board when the microcontroller board is connected in reverse. One header contains all communication bus signals like I2C, SPI and UART as well as the reset signal for the microcontroller. The other header contains the power supply, USB data and power lines, the interrupt signal of the MPU9150, a battery voltage

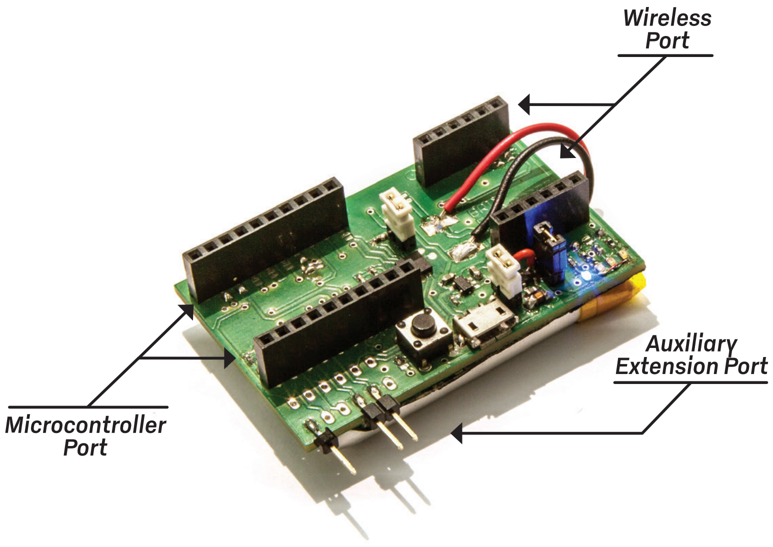


Figure 3.3: B2DK ports overview.

measurement pin connected to a voltage divider between V_{BAT} and GND, the charge status signal from the charging circuit as well as two general purpose I/O signals that connect to the extension port. The I/O signals also connect to two testpads which grant quick access for debugging and analysis, even when all extension boards are plugged in.

Wireless Port The wireless port connects a wireless receiver board to the carrier board. It consists of two 6-way female pin headers which are reverse polarity protected. One row of headers connects the I2C and the SPI bus of the microcontroller port to the wireless modules. The other row contains the UART signals and the power supply. The layout of these signals is designed to match a popular, off-the-shelf USB-to-serial converter board based on the FTDI FT232RL[2]. This way we can connect each of the wireless extensions boards directly to USB using the FTDI board, given that the wireless module can communicate via UART. This saves costs and also allows to use wireless modules in different projects without the carrier board, see Figure 3.4 (a).
In case of a conflicting layout of the RX and TX pins between the wireless port and the microcontroller port, we added a two-way solder jumper into each signal so TX and RX can be swapped.

[2]https://www.sparkfun.com/products/9873

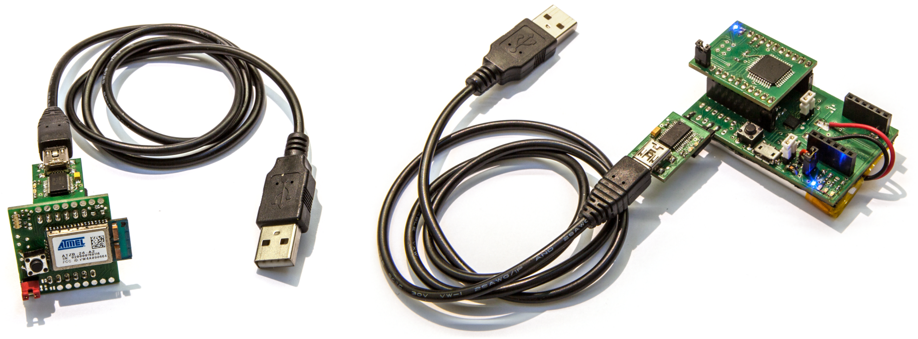

(a) (b)

Figure 3.4: Zigbit Module connected to USB via an FTDI board (a). Emulation of a wireless module on AUX port via USB (b).

Auxiliary Extension Port The third port on the B2DK carrier board can be used to connect external measurement equipment or a PC during regular operation. The signals are similar to the wireless port, except the Clear To Send (CTS) pin which is substituted by a GPIO of the microcontroller and an additional GPIO. The layout of the second pin header is again matching the FTDI adapter layout. This also allows to emulate either a wireless module or a microcontroller in software running on a connected host PC via USB, see Figure 3.4 (b).

Host Connections

The B2DK carrier board has a MicroUSB connector to connect the microcontroller module to a host PC. We decided not to integrate a USB-to-serial converter into the carrier board because we focus on microcontrollers with a native USB support. The battery of the carrier board is charged through the USB connection via the charger circuitry.

3.3.3 Microcontroller Extension Board

Microcontroller boards contain a microcontroller, oscillator, reset circuit and all necessary passive components to operate the controller. All relevant bus signals as well as USB can be broken out in order to connect to the base module. The only board we actually manufactured and tested is based on the ATmega32U4, because this is the most promising controller for BRIX$_2$. In Section 3.2.4, we listed all Arduino compatible microcontrollers. From the three controller families in the list, the ATmega16/32U4 series is the only one that natively supports USB, which we identified as a key requirement for BRIX$_2$. Our microcontroller extension board contains the

Atmel ATmega32U4 and a 16 MHz crystal oscillator. The passive components are capacitors to stabilize the input voltage as well as a pull-up resistor for the microcontroller reset and two termination resistors on the USB data lines. We added an ISP header in order to update the bootloader on the ATmega32U4, which ships with an Arduino incompatible default bootloader [91].

3.3.4 Wireless Transceiver Extension Boards

Wireless transceiver ICs and modules, protocols and stacks for a wide variety of applications and purposes are available on the market today. To find a suitable wireless transceiver for the BRIX$_2$ base module, we conducted tests with different wireless technologies. As a toolkit we developed four different wireless extension boards that can be connected to the B2DK base board via the wireless port. In this section we introduce the concept of a wireless transceiver module and point out which parameters have to be taken into account when selecting such a component. Subsequently we describe all modules we tested, present detailed background information on the wireless standard they use and finally present the results of our evaluation.

Background Information on RF Transceivers

A wireless transceiver translates a wired data transmission into a wireless data transmission and vice versa. In the most basic application, a wired connection is simply replaced by two RF transceivers as a so-called "Cable Replacement". However, various protocols for wireless transmissions offer more complex and flexible connections and allow for example network topologies such as broadcasts, meshes and tree structures. [92] A digital wireless transceiver is usually just a single chip solution which contains all components necessary to send and receive data wirelessly:

- **Digital Interfaces** for communication with an external microcontroller or other host system on the wired end of the transmission. Often a serial interface such as UART, SPI or I2C.
- **Frequency Synthesis** generates the analog wireless transmission signal from digital data.
- **Power Amplification** for the analog transmission signal.
- **Analog Signal Conditioning** of the incoming RF signal picked up by the antenna. This includes amplification and filtering.
- **Analog-to-Digital Conversion** of the analog signal for further digital processing.

Designing RF Circuits An RF transceiver needs an antenna and other passive components to be operational. The design of such an HF circuit is a complex engineering task that requires expert knowledge. When producing a PCB for a circuit like that, a number of constraints have to be considered, for example the antenna design, isolation of digital and analog domain, trace layout and impedances as well as grounding. [93]

RF Modules A way around the complex design of a custom RF circuit is to use an off-the-shelf wireless transceiver module. These small, shielded PCBs contain all necessary components and provide a communication interface to a microcontroller. Often, a chip or trace antenna is included on such a module. This dramatically reduces the complexity of the overall design because the critical parts are already tested and optimized by the manufacturer of the RF module. Since BRIX$_2$ is meant to be open, reproducible and understandable hardware, we wanted to keep our design as simple as possible, which is why we decided to use an off-the-shelf RF module in our platform.

System on a Chip RF Modules RF modules that also contain a microcontroller are referred to as SoC RF modules. [94] They allow to connect external components like sensors or actuators directly to theSoC RF module. The internal microcontroller either comes with a fixed default firmware and an API for its functions, like for example the XBee module [95] or is freely programmable. The manufacturer usually provides example firmware, for example for the Atmel ZigBit series [96]. SoC RF modules allow highly integrated applications, because almost no further components are required.

Wireless Technology Standards To ensure the compatibility of wireless devices across different manufacturers, technical standards for wireless networking exist. These standards define for example frequency bands, protocols or transmission power. There are many different wireless technology standards such as Global System for Mobile Communications (GSM), WLAN, Bluetooth, ZigBee or ANT. All of those standards have been designed for different purposes. WLAN for instance is optimized for high bandwidth, but has a high power consumption whereas ANT is optimized for low power consumption, but has a low bandwidth.
The first step in choosing a suitable wireless transceiver for a given application requires consideration of two basic parameters:

- **Protocol**: This defines how data is transfered from one module to the other in terms of data structure and networking functions. Today we can choose between a variety of different protocols such as Bluetooth(LE), WLAN, Zigbee, ANT. The protocol defines the organization and structure of wireless messages and networks and should match the desired application in terms of bandwidth, packet sizes and network topologies.

- **Carrier Frequency Band**: Most RF transceivers operate in ISM radio bands which include carrier frequencies like 433 MHz, 868 MHz, 915 MHz and 2.4 GHz, depending on national and international regulations. If the selected protocol does not have extensive error correction mechanisms, it is crucial to choose a carrier frequency that is not excessively used by other devices such as laptops or smartphones, because this will lead to interference with the RF transceiver. In the worst case, no data can be transfered at all, because the band is completely occupied by other devices. For example it is almost impossible to use an RF transceiver with a proprietary protocol (no error detection or correction mechanisms) operating in the 2.4 GHz band inside a regular office building because of the excessive traffic on the 2.4 GHz band caused mainly by WLAN and Bluetooth devices. The package loss in this case is most likely close to 100 %. The frequency band also affects the propagation range and possible bandwidth of a transmission. The higher the frequency, the lower the propagation range but the higher the bandwidth [97], [98]. However, not all wireless standards operate in every available frequency band. For example BTLE only operates in the 2.4 GHz band whereas the ZigBee standard defines operation in the 2.4 GHz, 915 MHz or 868 MHz bands.

ANT Module

The ANT wireless extension board allows us to test different aspects of the ANT wireless network protocol. In the following, we briefly introduce our ANT wireless extension module, the protocol and the transceiver hardware we use.

The ANT Protocol ANT is a low power wireless communication protocol. It was developed by Dynastream Innovations Inc. and is primarily used in wearable devices like fitness trackers or heart rate monitors. ANT devices operate in the 2.4 GHz ISM band and transmit short messages with a high bitrate, which leads to small duty cycles and thereby saves energy. [99] The ANT protocol is designed for short-range applications up to 30 meters. [100]

Network Topologies: ANT supports Peer to Peer (P2P), star, tree and mesh networks. Each node can transmit and receive, which allows a bi-directional communication. Up to 65533 nodes can join an ANT network using shared channels.

Channels: ANT nodes connect to each other using *“channels“*. A channel can connect two or multiple nodes (shared channel). Usually, bi-directional channels are used, but transmit/receive only channels are also possible. A channel has one master and one or more slave devices. Each node can open several channels in order to address different remote nodes or to broadcast in shared channels. Users can define the message rate for each channel individually. [101]

Data Types: ANT supports three basic message types: *"Broadcast Data"*, *"Acknowledge Data"* and *"Burst Data"*. Each of these data types can be used in both directions with some restrictions. Regardless of the data type, data packets are always 8 bytes.

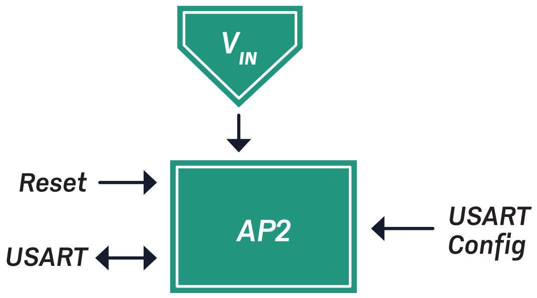
- Broadcast Data: This is the default data type and sent from the master to the slave on every channel period. If a slave device broadcasts, the message is only sent once and can be lost if any interferences occur.

- Acknowledge Data: If a packet is sent as acknowledge data, the receiver will automatically send an acknowledge message back to the sender. This uses more bandwidth but gives an indication whether the packet was received and can thereby prevent a loss of data.

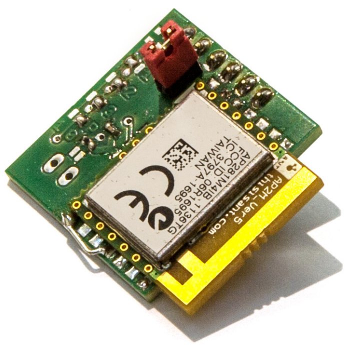
- Burst Data: This is a mechanism for the transmission of large amounts of data and consists of a series of acknowledged messages. The maximum data throughput is 20 kB/s. Burst transmissions do not have a maximum length, so they can be used to stream data.

ANT AP2 Transceiver Module The ANT AP2 RF transceiver module [102] sold by the ANT+ Alliance is an drop-in module that contains the Nordic nRF24AP2-8CH RF network processor [103] and on-board F-trace antenna. The CC2571 supports up to 8 simultaneous ANT channels and all three transmission types. It can be interfaced via UART with data rates up to 57600 baud. The physical dimensions are 20 x 20 mm and the peak TX current at 0 dBm is 28.8 mA at 3.0 V.

ANT Wireless Extension Board In order to connect the AP2 module to the B2DK, we integrated it into a wireless extension board, see Figure 3.5 (b), that matches the wireless port pinout. The ANT AP2 module requires only two external components, a capacitor to stabilize the power line and a pull-up resistor on the reset signal. A status LED indicating power was added for easier debugging. For power measurements, the LED can be disconnected with a solder jumper. Four 2-way solder jumpers allow to configure the UART interface of the AP2 module. The ANT wireless extension module connects only power and RX/TX to the carrier board, see Figure 3.5 (a).



(a)



(b)

Figure 3.5: B2DK ANT wireless extension board block diagram (a) and wireless module (b).

ZigBit Module

In order to test the ZigBee protocol and hardware, we developed a wireless extension board based on the Atmel ZigBit ATZB-24-A2 [104] module. Before we present our ZigBee board, we briefly describe the ZigBee protocol in detail.

A Closer Look on ZigBee The ZigBee specification was standardized in 2004 by the ZigBee Alliance and is compliant with the IEEE 802.15.4 standard. ZigBee devices operate in 868 MHz (European ISM), 915 MHz (US/CA ISM) and 2.4 GHz (Worldwide ISM) frequency bands. Raw data rates up to 250 kBits/s can be achieved at 2.4 GHz, but the protocol is typically used in low data rate scenarios like home automation. Star and tree networks are natively supported and are established and managed by a coordinator device. [105]
In order to communicate between two ZigBee devices, several device parameters must be considered:

Device Role: Every device in a ZigBee network has one of 3 device roles:

- Coordinator: The first device in the network that actually forms and sets up the network. There can only be a single coordinator in a ZigBee network at any given time.
- Router: Joins a network and can send, receive and relay messages through the network. Because the network relies on routers, it is not possible for this device type to enter a sleep or idle mode.

- End Device: Communicates with a parent device. It is able to send and receive, but not route. It can enter sleep or idle modes.

PAN ID: The 16 bit Personal Area Network (PAN) ID is shared by all devices in the same PAN. While several devices can communicate on the same RF channel, they are virtually separated into PANs. The PAN ID is set by the coordinator when forming the network.

Extended PAN ID: If two PANs conflict, meaning that two intentionally independent acpPAN share the same PAN by accident, the 64 bit *"Extended PAN ID"* can be used as a fallback to communicate to the designated devices to change their PAN ID in order to separate the two conflicting networks again.

Extended (MAC) Address: Each device can be identified by a 64 bit Media Access Control (MAC) address which is supposed to make it unique from all other ZigBee devices. Generally, this address is assigned at manufacturing and should/can not be changed to ensure uniqueness.

Short (Network) Address: The *"Short Address"*, *"Network Address"* or *"Node ID"* is a 16 bit value that is unique only in a PAN. It is randomly assigned at join-time. If conflicts occur, the MAC address is used as a fallback.

Forming a Network: The coordinator chooses a channel (RF-Frequency) to communicate on. The number basically offsets the communication frequency from 2.4 GHz. After that, the coordinator chooses also a PAN ID and an extended PAN ID.

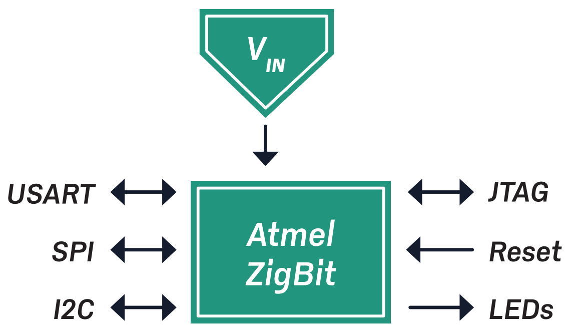
Now other devices can join the network. A device can only enter a network via a coordinator or router, not via an end device. The join process works as follows: The new node scans for available networks. It sends a request to all channels and all coordinators or routers answer the request by submitting their network parameters. The node selects one of these devices and sends a join request. If the request is permitted, the new node obtains a network address and becomes part of the PAN.

Atmel ZigBit ATZB-24-A2 For our ZigBee experiments we used the Atmel ATZB-24-A2 module. It contains an Atmel ATmega1281 microcontroller and an Atmel AT86RF230 RF transceiver along with a dual chip antenna. The physical dimensions of the module are 13.5×24 mm and it can be interfaced via UART, JTAG, I2C and SPI. The maximum TX current at 0 dBM is 18 mA at 3.0 V. For firmware development, Atmel supplies the BitCloud SDK[3], which offers a full ZigBee Pro compliant stack and an API to control a ZigBee network. Applications written in

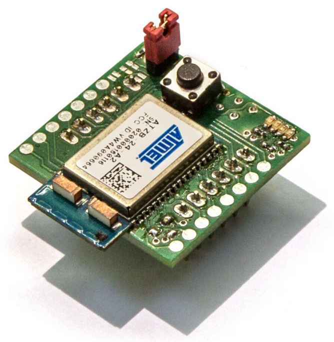
[3]http://www.atmel.com/tools/BITCLOUD-ZIGBEEPRO.aspx

BitCloud can be transfered to the ZigBit module via a serial bootloader and run on the internal ATmega1281.

ZigBit Wireless Extension Board



(a)



(b)

Figure 3.6: B2DK Atmel ZigBit wireless extension board block diagram (a) and wireless module (b).

The ZigBit extension board, see Figure 3.6 (b), contains a capacitor for input voltage stabilization, a power indicator LED and 3 LEDs which are connected to GPIOs 0-2 of the ZigBit module. All LEDs can be separated via solder jumpers during power measurements. We also added a pull-up resistor for the reset line and a JTAG header to program and debug the microcontroller on the ZigBit module. All three interfaces, SPI, UART and I2C as well as GPIO1 are broken out connect to the microcontroller extension port via the carrier board, see Figure 3.6 (a).

XBee PRO Module

XBee is a brand of wireless transceivers sold by Digi International Inc.[4]. The company offers a variety of SoC RF modules that are mostly based on an IEEE 802.15.4 compliant hardware layer and the ZigBee protocol. XBee modules contain a microcontroller which runs the XBee layer and allows P2P as well as point to multi-point wireless networks.

The XBee Protocol XBee module can communicate with an external microcontroller or host computer via UART. Its wireless transmitter, GPIOs and bus lines can be controlled either by AT commands or in an API mode. AT commands are human readable and easy to use whereas the API mode uses packets that consists of raw bytes. This is more efficient, but harder to debug and maintain.

Network Types: There are two network types for wireless XBee networks, *"Non-Beacon mode"* (P2P) and *"NonBeacon with coordinator"*, where one module acts as a ZigBee coordinator [106].

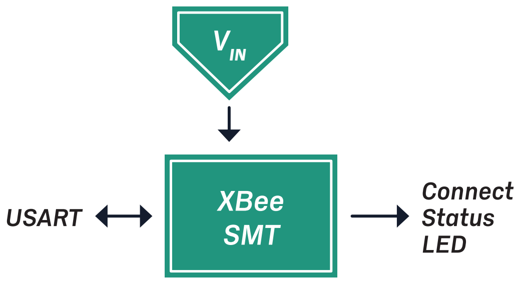
Transmission Modes: The XBee protocol implements two transmission types, unicast and broadcast. Unicast messages are acknowledged and an automatic transmission retry can be configured. Broadcast messages are not acknowledged and all receiving modules will accept the transmission.

Adressing: Every XBee module has an individual, 64 bit address as a factory default and can also be configured to accept messages with a short, 16 bit address. This address can be selected by the user. Each message packet contains sender and receiver address.

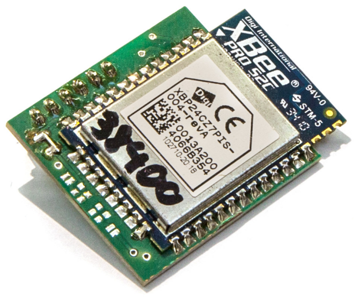
XBee-PRO S2C For our experiments, we use the Digi International XBee-PRO S2C SMT [107], a surface mount drop-in module. It measures 22×33.8 mm and an estimated TX current of 14 mA at 0 dBm and 3.0 V [108]. The XBee-PRO S2C also offers 15 controllable GPIOs, ADCs and SPI. XBee PRO supports transmission data rates up to 250 kb/s. XBee modules are especially popular for DIY projects involving wireless data transfer from one microcontroller to another due to its simple and easy-to-use protocol design and robust wireless link.

XBee PRO Wireless Extension Board The XBee-PRO S2C provides a convenient serial interface. On our XBee wireless module, see Figure 3.7 (b), we connected RX/TX as well as the Data Terminal Ready (DTR) and CTS signals, but in practice

[4]http://www.digi.com



(a)



(b)

Figure 3.7: B2DK XBee PRO wireless extension board block diagram (a) and wireless module (b).

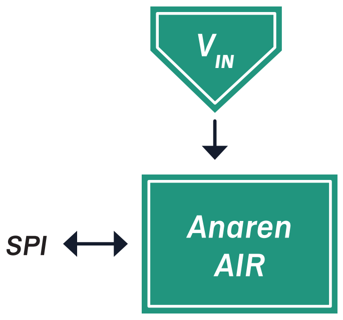
it turned out that only RX/TX (UART) were required, see Figure 3.7 (a). The module also features an association indicator LED which lights up when a connection is established. Like the power indicator LED, it can be separated with a solder jumper during power measurements.

Proprietary RF Module

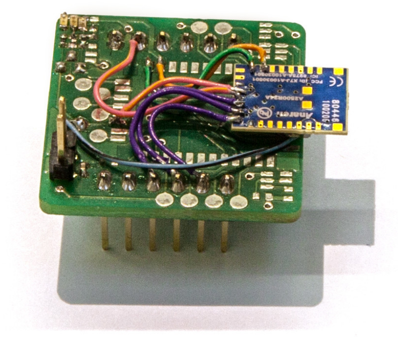
In contrast to drop-in RF modules with a complex stack like ZigBee or ANT, we decided to also test a transceiver module that is reduced to just the hardware layer. We find this kind of wireless solution frequently in WSN motes (see Section 2.3.3) , which use for example the Chipcon CC1000 RF transceiver chip and run the protocol stack on the application microcontroller. This allows a lot more flexibility in the stack design because developers have full control over the hardware layer.

Anaren AIR Modules As we already mentioned earlier, it is more convenient to use off-the-shelf RF modules than to design the RF circuitry from scratch because such modules are already tested and optimized. We decided to use an Anaren AIR module [109] for our proprietary RF wireless extension module because it is small (9 x 12×2.5 mm), inexpensive and based on the widely used Texas Instruments CC1101 [110] wireless transceiver IC. Anaren offers the AIR modules in a variety of pin-compatible devices with different carrier frequencies and antenna solutions. This provides us with a lot of flexibility because we can still choose the frequency band or antenna technology without a PCB redesign.

Texas Instruments CC1101 The CC1101 low power sub-1 GHz RF transceiver was released by Texas Instruments in 2008 and is still in active production state (as of 2015). The transceiver is designed for the 315/433/868/915 MHz ISM bands and supports data rates up to 600 kb/s. The TX active supply current varies with the carrier frequency and is 16.8 mA for the 868 MHz version at 0 dBm and 3.0 V. The CC1101 is interfaced via SPI and features sync word detection, address check, flexible packet lengths and automatic Cyclic Redundancy Check (CRC) handling.



(a)



(b)

Figure 3.8: B2DK Anaren AIR wireless extension board block diagram (a) and prototype wireless module (b).

Anaren AIR Wireless Extension Board The extension module for the Anaren AIR proprietary RF series modules is simple, because all necessary components are integrated on the drop-in module, see Figure 3.8 (b). We added a capacitor for input voltage stabilization and a power indicator LED which can be separated via a solder jumper for current measurements. The SPI pins connect to the microcontroller extension port, see Figure 3.8 (a). The GPIOs on the CC1101 can be configured as notification interfaces, for example as a Clear Channel Indicator or for FIFO Status Signaling.

3.4 Evaluation Using the BRIX$_2$ Development Kit

The BRIX$_2$ Development Kit was used for multiple purposes. First, we needed to practically test different wireless modules in order to select the most appropriate solution for the BRIX$_2$ platform. Second, we needed to try out initial designs for the fixed circuits like voltage regulation, charge electronics and the motion sensor. Third, we tested the microcontroller for Arduino compatibility to verify that our platform would work with the IDE. In the following we will present our experiments and the results.

3.4.1 Evaluation of Different Wireless Transceiver Modules for BRIX$_2$

All wireless modules we selected for our test fulfill the basic technical requirements for BRIX$_2$, see Section 3.2.4. Apart from just comparing the technical specifications and documentation of different wireless transceiver modules and protocols, it was important for us to test how easy the devices can be integrated into our platform, from a hardware as well as from a firmware perspective. In an exploratory process, we interfaced the transceiver module first with a PC and then with the B2DK. After setting up a wired communication between the host and the transceiver module, we established a wireless connection between two transceivers. However, we did not conduct broad RF communication tests with complex topologies or varying data packet sizes, since this would exceed the scope of this project. We start our evaluation process with a technical comparison of all transceiver modules we built for B2DK.

Comparison of the Technical RF Transceiver Properties

To give a short technical overview about the modules we used in our tests we summarized all facts that are relevant for our application in Table 3.3.
Three of the four modules we considered for testing are SoC modules, which means they contain a microcontroller and an RF transceiver IC, see also Section 3.3.4. Apart from the Atmel ZigBit module, the firmware of the internal microcontrollers can not be altered by the user. The maximum data rates range from 250 kb/s to 1 Mb/s which is sufficient for our applications. The TX current at 0 dBm is similar on all devices, whereas the maximum TX gain varies dramatically. The TX gain is the main factor that limits the range of the wireless transmission, given a well-matched antenna design. The XBee-PRO module is designed for high gain applications and ranges up to more than 3 kilometers. On the contrary, the Anaren ANT module is optimized for Body Area Networks (BANs), a common use for ANT technology. It has only a low maximum TX gain and therefore a limited range for wireless transmissions. The Anaren AIR module looks promising regarding the RF parameters because of its average to low TX/RX currents, its large link budget and high RX sensitivity.

Parameter	ANT AP2	ATZB-24-A2	XBee-PRO S2C	AIR A1101R08*
Protocol	ANT	ZigBee	Xbee/ZigBee	None
Frequency Band	2.4 GHz	2.4 GHz	2.4 GHz	868 MHz
Type	SoC	SoC	SoC	RF only
Transceiver IC	nRF24AP2-8CH	AT86RF230	EM357	CC1101
Microcontroller	MSP430F1232	ATmega1281	MC9S08QE32CFT	none
Max. Data Rate	1 Mbps	250 kBps	250 kBps	600 kBps
TX Current‡ (mA)	18	18	14	16.8
RX Current‡ (mA)	23.7	21.8	31	15.7
Idle Current† (µA)	1	6	1	0.5
Max. TX Gain (dBm)	0	+3	+18	+12
Max. RX Sensitivity (dBm)	-86	-101	-101	-112
Communication Interface	UART	UART, I2C, JTAG, SPI	UART	SPI, UART
Antenna	F Trace	Dual Chip	Trace	Ext. Monopole, Trace
Size (mm)	20×20×3.1	24×13.5×2	22×33.8×3	9×12 x 2.5
Price (USD)	20	28	18	13

† Lowest power mode where wakeup on RX is still possible.
‡ Input voltage 3.0 V, 0 dBm (only applies to TX).

Table 3.3: Parameters of different RF transceivers.

The physical size of the modules, which is a relevant design parameter for a compact device, varies between 108 mm^2 and 744 mm^2 surface area. The most expensive module is with 28 USD more than double the price of the least expensive module.

Exploratory Evaluation

Since BRIX$_2$ targets education and prototyping scenarios, it has to be adaptable and flexible, which means that it is not optimized towards a specific field of application, unlike most of the platforms we analyzed in Chapter 2. For that reason we can not optimize the parameters of our wireless transceiver technology towards a specific application but rather towards transparency and versatility to allow an easy understanding and usage of the technology as well as to adapt it to a wide number of application scenarios. With this in mind, the focus of our experiments and analysis of the different wireless modules rather lies on usability and flexibility and not on a detailed technical evaluation. In the following, we first present a number of exemplary wireless data transmission scenarios from the different fields of applications we intend to cover with BRIX$_2$ and derive experiments to test the transceivers in that kind of scenario. After that we describe the experiment setups before we present the results of the experiments and draw a conclusion.

Wireless Networking Testing Scenarios

In order to cover the main applications for the wireless transmitter of the BRIX$_2$ system, we designed the following scenarios:

1. Smart Environments: Different sensors and actuators inside a single room or a number of rooms inside a building communicate wireless and form a distributed, interactive application. In this scenario, data rates are low, but a solid connectivity is required even across rooms and in locations with heavy wireless activity such as office buildings and laboratories. Since the modules are static in this scenario, we assume that they can be permanently supplied with an external power source.

2. Sensor Networks: Similar to the smart environments scenario, sensor networks operate with a low data rate, but in order to cover a large area with a low number of motes, high range transceivers are necessary. This scenario is especially interesting outdoors. Although a permanent external power source may not be available, in outdoor applications, the BRIX$_2$ module could be powered with a larger, external battery. Short bursts of data with a low rate reduce the power consumption to a minimum.

3. IMU Data Streaming: Data from a BRIX$_2$ module attached to a human body is streamed wirelessly to a host application. In order to capture rapid motions, this scenario relies on a high data rate. The distance to the receiver is small. The sender is used without an external power supply and should be able to run at least 3 hours.

Experiment Setups

All of our experiments involved sending data packets wireless from the TX side to the RX side. Using the B2DK, we were were able to implement the experiment setup quickly and efficiently. On the TX side, the microcontroller generates data packets with a payload of 8 bytes, including a packet number, the TX gain and the data rate. Those packets are sent over the air via each of the four transceiver modules we put to the test. On the RX side, the packets are picked up by the corresponding RF transceiver and sent to the microcontroller on the B2DK, see Figure 3.9.

In general we vary three parameters: The data rate, the distance between RX side and TX side and the TX gain. The latter is either 0 dBm or the maximum the transceiver can accomplish, see Table 3.3.
The controller calculates the time between the current and the previous packet as well as the number of lost packets between two received packets. This data is then sent to a laptop that is connected via USB, where it is logged and stored in separate

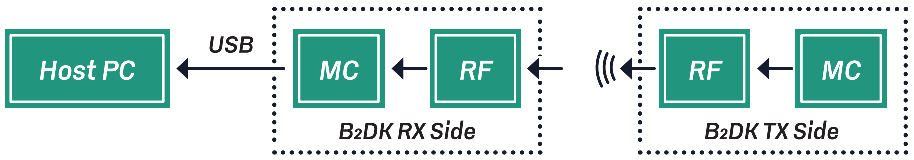


Figure 3.9: General setup for our wireless transmission experiments.

files for each experiment. We recorded a three minute data sample in each trial and used the following parameters:

1. Smart Environments: The experiments took place in a lab environment with various computer and audio setups in place. Several WLAN networks on the 2.4 GHz band were active during the experiments. The RX and TX side were located in separate rooms, without line of sight and a distance of around 15 meters. Both RX side and TX side rested on a table. The TX gain was set to maximum for each transceiver. We tested data rates of 10 Hz and 1 Hz.

2. Sensor Networks: The experiments were conducted outdoors in a remote location presumably with low interference on the RF bands we used. RX side and TX side were positioned 1.5 m above ground statically with a distance of 50 meters and 100 meters, a clear line of sight and the antennas facing each other. We adjusted the TX gain to maximum and set the data rate to 10 Hz.

3. IMU Data Streaming: The experiments were conducted in the same environment as scenario 1. We configured all transceivers to a TX gain of 0 dBm because we were testing a mobile application without external power supply. The distance between RX side and TX side was 5 meters with a line of sight. The receiver rested on a table next to the laptop while the transmitter was handheld by a person that moved constantly. This way the person's body interrupted the line of sight regularly, simulating a real-life motion tracking scenario. Each transceiver was tested with a data rate of 100 Hz and 150 Hz.

RF Module Setups

In this paragraph, we briefly describe how the four wireless modules that we tested using the B2DK were set up and configured. This makes it possible to reproduce our tests and verify our results.

ANT AP2 The AP2 transceiver module can be accessed through UART and is set up sending raw byte messages. We configured the receiving node (RX side) to create a *“Bidirectional Master Channel”*. All other settings are left at default. On the transmitting node (TX side), we opened a *“Bidirectional Slave Channel”* and set the data transmission rate using the *“Channel Period”* parameter to 200 Hz (0xAA) in order to cover all our tests.

Atmel ZigBit The ZigBit module contains a microcontroller that implements the ZigBee PRO stack and can also hold a custom application. Firmware can be compiled using the Bitcloud Software Development Kit (SDK), which also comes with several examples. We uploaded the SerialNet example to the controller using Atmel’s FLIP bootloader utility via UART. SerialNet allows to control the ZigBit module using a number of AT commands[111]. To set up a communication between the RX side and the TX side, we first configured several parameters of the RX side. The *“Extended Address”* was set, followed by the node role, in this case *“Coordinator”*. To create the network, we set the WPAN ID and the node’s *“Short Network Address”*. Subsequently the network was joined and thereby created. On the TX side, we defined the device role to *“Router”*, assigned a *“Short Network Address”* to the node and joined the network created by the coordinator. The TX side can also modify the gain of its transmission amplifier by using the *“WTXPWR”* command followed by a warm reset.

Digi XBee-PRO The XBee-Pro transceivers implement a stack that is based on Zigbee. The network parameters are therefore similar to the ZigBit parameters and the transceiver can also be configured by AT commands. The RX side is set up as a *“Coordinator”*, defines the *“PAN ID”* and creates the network. The TX side is configured as a *“Router”* and joins the network created by the coordinator.

Anaren AIR A1101R08 The transceiver module implements no dedicated network protocol stack, only two modes of transmission: broadcast or addressed. Receivers can perform an address check and data packets are tested for integrity using a checksum. Texas Instruments offers a tool that can generate initial register setups for several of their wireless devices[5]. These register setups can be exported for example in a C header file format and easily integrated into a microcontroller firmware which configures the module via SPI on startup. After this initial configuration, we set a four-byte *“Sync Word”* which is identical on the RX and the TX side. Optional the TX side configures the TX gain. To communicate, a packet has to be assembled with a payload and a receiver address. The checksum is calculated automatically.

[5]http://www.ti.com/tool/smartrftm-studio

Analysis and Results

In our experiments, the log files contained an incremental number of each packet that was sent from a transmitter, the time between two received packages in milliseconds as well as the number of lost packets between two received packages. Using these numbers, we were able to calculate two measures:

Packet Loss: This measure expresses the percentage of total lost packets during the whole transmission time. Packets are usually lost, or "dropped" because of incomplete transmissions or queuing issues. We consider this measure a basic indicator for the quality of a wireless link.

Jitter: Especially for real-time applications it is crucial that data arrives at the receiver with an even timing to ensure that the application runs smoothly. Mechanisms in wireless stacks such as retrying to send packets when no acknowledgment occurs lead to varying transmission times or 'intervals'. With the jitter, we express the variance of intervals around the average. In order to obtain a measure which is comparable between different wireless networking standards, we chose a normalized version of the variance. We calculate the jitter J as the coefficient of variation, which equals the standard deviation of the measured intervals divided by the average interval A:

$$J = \frac{\sqrt{Var(intervals)}}{A(intervals)}$$

For our first scenario, smart environments, we selected a location with a significant level of activity especially in the 2.4 GHz wireless band. In the trial with 1 Hz data rate, only the ANT transceiver pair drops packets, see Table 3.4. The jitter of the CC1101 transceiver pair is much lower than the one of the other pairs. The reason might be that those transceivers operate in the 2.4 GHz band, which in the experiment environment is likely to be more occupied than the 868 MHz band the CC1101 operates on. This can lead to more corrupted packages which are then retransmitted, leading to varying transmission times. The same behavior is expected in the second trial with a data rate of 10 Hz, but this time, the jitter of the CC1101 is comparable to the other transceivers. Looking at the dataset, we find that there are only two of around 180 packets with a transmission time different from the average. In the data sets of the other transceivers, those abnormal transmission times occur more often, but with a smaller difference to the average. In the second trial we also noticed a packet loss which is comparable throughout the ANT, CC1101 and XBee transceiver pairs but significantly higher for the ZigBit pair.
In the wireless sensor network scenarios, see Table 3.5, we first notice that the ANT transceiver has a high packet loss even at only 50 meters distance, most likely due to its low TX gain of only 0 dBm in comparison to the other transceivers. Only

Transceivers	Scenario	Rate	Distance	TX Gain	Packet Loss	Jitter
ANT	SE Trial 1	1 Hz	10 m	0 dBm†	4.62 %	0.215
ZigBit	SE Trial 1	1 Hz	10 m	+3 dBm†	0.00 %	0.378
XBee	SE Trial 1	1 Hz	10 m	+ 18 dBm†	0.00 %	0.499
CC1101	SE Trial 1	1 Hz	10 m	+ 12 dBm†	0.00 %	0.001
ANT	SE Trial 2	10 Hz	10 m	0 dBm†	5.17 %	0.262
ZigBit	SE Trial 2	10 Hz	10 m	+3 dBm†	24.64 %	3.209
XBee	SE Trial 2	10 Hz	10 m	+ 18 dBm†	1.45 %	1.675
CC1101	SE Trial 2	10 Hz	10 m	+ 12 dBm†	3.10 %	0.351

† Maximum setting for this transceiver.

Table 3.4: Experiment results for the smart environments scenario.

four of expected 180 packets were received, which is not sufficient to calculate a representative jitter, so we excluded the ANT transceiver pair from the second trial. All other transceiver pairs however show no packet loss in the first trial and also low and comparable jitter, since the experiment environment is expected to be free of any wireless traffic on the 2.4 GHz band. In the second trial, only the ZigBit transceivers are robust enough to successfully transmit every packet. The XBee and CC1101 transceivers show a packet loss of 11 % respectively 18 % as well as a much higher jitter than in the first trial. The ZigBit transceivers can maintain their low jitter throughout both trials.

Transceivers	Scenario	Rate	Distance	TX Gain	Packet Loss	Jitter
ANT	WSN Trial 1	1 Hz	50 m	0 dBm†	> 90 %	–
ZigBit	WSN Trial 1	1 Hz	50 m	+3 dBm†	0.00 %	0.002
XBee	WSN Trial 1	1 Hz	50 m	+ 18 dBm†	0.00 %	0.008
CC1101	WSN Trial 1	1 Hz	50 m	+ 12 dBm†	0.00 %	0.001
ANT	WSN Trial 2	1 Hz	100 m	0 dBm†	–	–
ZigBit	WSN Trial 2	1 Hz	100 m	+3 dBm†	0.00 %	0.002
XBee	WSN Trial 2	1 Hz	100 m	+ 18 dBm†	11.24 %	1.484
CC1101	WSN Trial 2	1 Hz	100 m	+ 12 dBm†	17.58 %	0.965

† Maximum setting for this transceiver.

Table 3.5: Experiment results for the WSN scenario.

In our third scenario, the simulated streaming of inertial sensor data, we focused on high data rates at a rather low distance. From the data, see Table 3.6, it is remarkable that the ZitBit module shows high packet loss rates as well as jitter in both trials. ANT and XBee both show a comparable packet loss around 10 % in the first trial but differ significantly in the second trial, where the packet loss of ANT is higher, whereas the XBee packet loss is almost the same as in the first trial. In both trials, the CC1101 has by far the lowest packet loss and the lowest jitter.

Transceivers	Scenario	Rate	Distance	TX Gain	Packet Loss	Jitter
ANT	IMU Trial 1	100 Hz	5 m	0 dBm	10.03 %	0.304
ZigBit	IMU Trial 1	100 Hz	5 m	0 dBm	85.48 %	40.238
XBee	IMU Trial 1	100 Hz	5 m	0 dBm	8.1 %	4.182
CC1101	IMU Trial 1	100 Hz	5 m	0 dBm	0.43 %	0.095
ANT	IMU Trial 2	150 Hz	5 m	0 dBm	43.93 %	0.368
ZigBit	IMU Trial 2	150 Hz	5 m	0 dBm	90.26 %	74.483
XBee	IMU Trial 2	150 Hz	5 m	0 dBm	7.65 %	7.143
CC1101	IMU Trial 2	150 Hz	5 m	0 dBm	0.60 %	0.09

Table 3.6: Experiment results for the IMU data streaming scenario.

Please note that for all scenarios there is a potential for optimization for each technology which would probably lead to different results. In our tests we used basic setups, because the optimizing process can be lengthy, complex and complicated. This means effort that is not feasible for the basic tests we conducted in order to find out if all technologies are usable in our targeted applications and scenarios.

Discussion

In our analysis of four different wireless transceivers, we have taken a look on the technical specifications, set them up for practical tests and conducted experiments in three different scenarios with two different parameter variations each. In the following we briefly conclude the results for each transceiver before we discuss which transceiver we select for our actual BRIX$_2$ implementation.

ANT AP2

Technical Specifications: The ANT AP2 transceiver module with integrated Texas Instruments MSP430 microcontroller operates in the 2.4 GHz band, which is already heavily occupied by WLAN and Bluetooth devices in many environments. The maximum output power of 0 dBm is by far the lowest of all modules in the test while the data rate of 1 Mb/s is the highest. The peak current consumption is relatively high in TX as well as RX with respect to the other three transceivers. From a mechanical point of view, with a size of 20×20 mm it could mechanically well be integrated into a compact application like BRIX$_2$.

Handling and Programing: Technically, the module can easily be interfaced through a configurable UART. There is no intended update option for the microcontroller firmware on the device, so users rely on the scope of functions that are implemented by the manufacturer. The ANT protocol offers several different modes and topologies which are all optimized towards special purposes. This makes it difficult to find a setting that covers a wider area of applications. This additional complexity would have to be carefully wrapped in a final BRIX$_2$ implementation to not confuse non-expert users.

Practical Tests: In our experiments, we observe significant packet losses in medium and long range scenarios which are most likely caused by the low maximum TX gain. At a distance of 50 meters, only 3 of expected 180 packets were picked up by the receiver. In a low range and high packet rate scenario, the packet losses ranged from 10 % to almost 45 %. Since the ANT protocol and devices are intended to be used in BANs with low data rates, these results are not surprising and show clearly that the technology is not capable and flexible enough to perform well in our required scenarios. The jitter, caused by protocol inherent mechanisms such as packet resends is relatively low for the ANT transceiver in most scenarios and still acceptable for potential BRIX$_2$ applications.

Atmel ATZB-24-A2

Technical Specifications: The Atmel Zigbit module also features an integrated, programmable microcontroller and operates on the 2.4 GHz band like most of the transceivers we tested. It has a maximum data rate of 250 kB/s and average power consumptions for TX and RX. The maximum TX gain if 3 dBm is relatively low, compared to the other modules. The footprint of the transceiver is the second smallest in our comparison and could be easily integrated into the final product. With a price of 28 USD, the transceiver is by far the most expensive in our test.

Handling and Programing: Since the integrated Atmel microcontroller is fully programmable, the ATZB-24-A2 is a versatile platform. However, the Bitcloud SDK is quite complex which makes developing custom firmware for the module difficult. For basic networking scenarios, the pre-compiled SerialNet firmware allows a quick configuration of the ZigBit module by issuing AT commands through a UART connection. Several possible network topologies, modes and parameters offer a lot of potential to optimize a certain application.

Practical Tests: The ZigBit transceiver pair performed well in the WSN scenario and did not lose a single package even at 100 m distance even though it only operated on a TX gain of 3 dBm. In the more noisy lab environment, we observed a significant packet loss of almost 25 % even at a low data rate of 10 Hz and a distance of 10 meters. In the high data rate scenario, the ZigBit pair lost 85 % to 90 % of the packets which also went along with high level of jitter. This is not acceptable for us because the wireless streaming of IMU data is clearly stated as one of our target applications.

Digi XBee-PRO S2C

Technical Specifications: The XBee module features an internal microcontroller which is not designed to be reprogrammed by developers or users. The device operates on the 2.4 GHz band and its 18 dBm maximum TX gain is the highest in our test. Surprisingly it has the lowest TX peak current and also the highest RX current of all four modules. With a physical size of 22×34 mm, it would be difficult to integrate into the compact device we plan to implement.

Handling and Programing: The XBee module, despite of its internal microcontroller can not be programmed with a custom firmware but only be reconfigured. Through the UART interface, communication with the module is possible either through AT commands or in an API mode, a more compact command format. Once the modules are set up, XBee behaves like a cable replacement with the option of addressing different modules, however more complex network topologies are also possible.

Practical Tests: In our long range experiments, the XBee transceiver pair performs well on the 50 m range without any packet loss and low jitter. Surprisingly, regarding the 18 dBm TX gain, the packet loss on 100 meters is more than 11 % along with much higher jitter than at 50 meters. In the smart environments scenarios, the XBee modules have the lowest packet loss, but also almost the highest jitter. The high data rate scenarios show a packet loss around 8 % in both trials, also with relatively high jitter. This would be a problem especially in real-time applications, for example when motion data is processed and fed back to the user in a closed interaction loop.

Anaren AIR A1101R08* / CC1101

Technical Specifications: The Anaren AIR module is the only one in the test that does not contain an internal microcontroller and basically consists of the Texas Instruments CC1101 wireless transceiver along with an antenna. The transceiver operates in the 868 MHz band without any specific protocol. It has a rather high maximum TX gain of 12 dBm and a low power consumption in RX and TX. The maximum data rate is with with 600 kB/s the second highest in our test. Due to its simplicity, the Anaren AIR module is small and is therefore ideal for the compact BRIX$_2$. Besides that, the price of 13 USD is the lowest of all four modules.

Handling and Programing: Once the CC1101 is set up by writing all configuration registers, it can be issued with packets for broadcast or address based wireless transmissions. There is no stack that already implements for example network topologies, which reduces the complexity on the one hand and increases the flexibility on the other hand.

Practical Tests: First of all, due to the lack of any protocol, the Anaren AIR transceiver pair shows low jitter in almost all scenarios because the data is just passed on without any processing. Obviously, this can also lead to lost packets because no mechanisms that could increase the stability of the transmission are in place but could still be implemented later on in software. In the smart environment scenario, the packet loss is good to acceptable. In the WSN scenario, the module works flawlessly at 50 meters without packet loss and with a low level of jitter. On 100 meters it still works but with a packet loss around 18 %. In the high data rate scenarios, the transceiver pair beats all others. We observed a packet loss around 0.5 % in both trials and a low jitter compared to the other modules.

Conclusion on the Wireless Networking Evaluation

Now that we have evaluated all wireless transceivers that we considered for our BRIX$_2$ system,we are able to decide which of the technologies is most suitable for our purpose. Our requirement is a general purpose transceiver that works reasonably well in all scenarios we identified in Section 3.2.4. Since we are building a platform also for usage in education, we need an understandable, well documented technology. Also, we have to consider technical parameters like wireless properties,

size, energy consumption, availability and price. Among the modules we tested were more specialized like for example the ANT AP2 and ZigBit but also general purpose devices like XBee or the Anaren AIR based on the Texas Instruments CC1101. The flexibility of those technologies becomes especially clear in the experiments we conducted. We find that only three of the four devices are actually usable in all scenarios we tested. The TX gain of the ANT transceiver is simply too low to work well in wireless sensor networks that require transmission distances greater than 20 meters. Other modules lose packets when required to transport data with a high rate, such as the ZitBit modules in the motion data streaming experiment. Added to these disadvantages are the high costs for both types of modules, especially the ZigBit module with almost 30 USD. The XBee and the Anaren AIR devices show an equally well and consistent performance throughout all the tests with the XBee performing slightly better when signal strength matters and the CC1101 for packet loss at high data rates.
Regarding their wireless networking capabilities and properties, both are suitable for our BRIX$_2$ platform. However, there are three facts to consider. First, the XBee module is too big to be easily integrated into a compact application like BRIX$_2$. In contrast, the AIR module is by far the smallest in our test and perfectly suited for integration. Second the internal microcontroller is not programmable with a custom firmware, which limits its functions to the scope that the manufacturer has implemented. Third, the XBee radio is 65 % more expensive than the Anaren AIR modules.
In conclusion we identified various strong arguments for the Anaren AIR module and its CC1101 wireless transceiver chip. The device does not have any significant disadvantages that would render it problematic for our use in BRIX$_2$, is well documented and widely used in other applications. It shows good wireless networking performance, is small and relatively inexpensive. The CC1101 is a wireless transceiver with a low level of abstraction, which makes it a perfect example for education purposes and is flexible and adaptable in prototyping scenarios. To compensate for the microcontroller that is missing on the Anaren AIR module in contrast to all other modules, we can equip BRIX$_2$ with an additional controller that not only handles the RF interface but can also control other parts of the system like for example battery voltage monitoring, since it is freely programmable for us and also potentially for users, which results in increased flexibility.

3.4.2 Charge Electronics

To test the charge circuit, we recorded the charge and discharge times of a 450 mAh LiPoly battery via the B2DK equipped with a microcontroller and an RF module that continuously sent data to a PC. This way we could tell by timestamp of the last received packet when the battery of the B2DK was empty and the device stopped operating. Additional to the microcontroller and RF module, we also connected a number of LEDs to the output of the voltage regulator on the B2DK to increase the discharge current to an average around 300 mA.
As we can see in Figure 3.10, both charge and discharge times are consistent over 57 trials. This shows that the capacity of the battery does not significantly decrease after around 60 charge / discharge periods, just as we expected from the specifications of LiPoly batteries in Section 3.2.4. In our experiments, we lost the measurement data of some trials by accident, so we left them out in the plot.

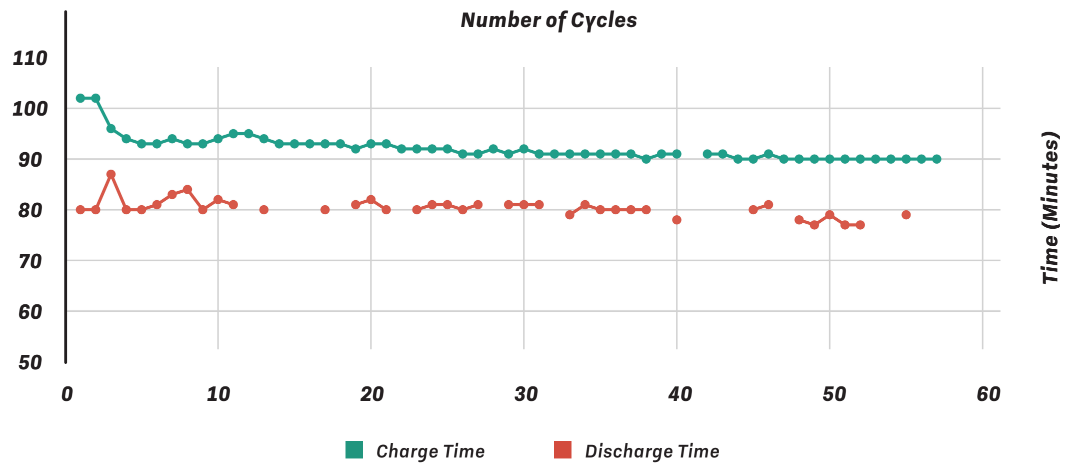


Figure 3.10: B2DK Charge/Discharge experiments over 57 Cycles

During testing we became aware that in our current setup, the battery always supplies the device while it was turned on, even when it was connected to USB. This leads to a constant cycle of discharging and charging and thereby shortens the lifetime of the battery. To overcome this issue, we are going to include a power multiplexer into the final implementation of BRIX$_2$. This component sources the input power from USB if connected and otherwise from the battery.

3.4.3 Microcontroller

Regarding the microcontroller, we were mostly interested whether it could be programmed using the Arduino IDE. The ATmega32U4 ships with the Atmel FLIP bootloader installed which had to be removed. Via ISP, we erased the flash memory of the microcontroller and installed the *"Caterina"* bootloader, which is also used on the Arduino products based on the ATmega32U4 like the Arduino Leonardo. After the bootloader was in place and the fuse bits were set accordingly, we could upload firmware to the microcontroller through USB via the Arduino IDE with "Arduino Leonardo" selected as a board. The bootloader takes up 4 kB of flash memory on the controller, leaving the users with 28 kB of memory for their firmware. We can conclude that the microcontroller works together with the Arduino IDE as expected and is suitable to be integrated into the BRIX$_2$ platform.

3.4.4 Inertial Motion Sensor

The MPU-9150 inertial motion sensor we integrated into B2DK has features that outperform any other device available at the design time of our platform. Its Digital Motion Processor (DMP) provides Euler angles and quaternions by fusing the sensor data on chip. This is a unique feature, so there is basically no alternative to compare it against. However, we conducted some basic tests to verify that the sensor performance itself matches our expectations and the device is feasible for our target applications. We recorded the raw data of accelerometer and gyroscope to determine the noise of the accelerometer as well as the drift of the gyroscope while the sensor is motionless. Subsequently we did tests to find out how fast data can be read from the sensor. Finally we examined the DMP data in a static positions and in a scenario that involves motions.

Accelerometer and Gyroscope Noise

Raw sensor data can be acquired directly from the sensor's internal registers via the I2C bus. For easier handling and programming, we used the MPU6050 library of the I2Cdevlib[6]. By the time we conducted our tests, there was no library for the MPU9150 yet, but since both devices are compatible regarding the features we use in our experiments, we could also use the existing one. To quantify the noise of the accelerometer and the gyroscope, we recorded 20000 data packets with a rate of 300 Hz. The sensor was not in motion at the time of recording and rested on a table. All recordings took place at room temperature.
It is noticeable that the noise on the z-axis on the accelerometer is higher than on the x- and y-axis, see Table 3.7. There is no information about accelerometer

[6]http://www.i2cdevlib.com/devices/mpu6050

Sensor Axis	Standard Deviation	Percentage of full Range
Accelerometer X	62.62 LSB	0.096 %
Accelerometer Y	53.67 LSB	0.081 %
Accelerometer Z	80.69 LSB	0.123 %
Gyroscope X	13.32 LSB	0.020 %
Gyroscope Y	13.80 LSB	0.021 %
Gyroscope Z	9.94 LSB	0.015 %

Table 3.7: Accelerometer and gyroscope noise levels.

noise in the datasheet of the MPU9150, but slightly higher noise levels on the z-axis appear to be common among 3-axis devices. The Analog Devices ADXL330 datasheet[112] for example states a 25 % higher noise density for the z-axis than for x and y. The gyroscope shows a much lower noise levels than the accelerometer and interestingly enough, the z-axis has the lowest noise. This might be, as well as for the accelerometer caused by properties of the implementation of multi-axis MEMS devices, however we could not find sources to verify this theory. Please note that we only tested a single device in this experiment.

Speed Test

To find out if we could stream data at high rates to a PC through the USB interface, we performed a speed test. In practice, there certainly is room for optimization, for example by using different streaming formats or a HID device instead of a virtual serial port. In our test, we streamed data from the serial port of the B2DK to a virtual comport via USB with a baud rate of 115200, the maximum default speed that is supported by Arduino. We compared two different types of data packets. First, a human readable acceleration and angular velocity data vector read from the sensor. The vector contains six comma separated numbers with up to 5 digits plus sign. Additionally we included a byte which indicates the time from one packet to another in milliseconds, typically only 1 digit plus the separation comma. We constantly moved the sensor and recorded 10000 data packets with an average length of 39.24 bytes, including carriage return and line feed. The average time from one receiving event to the next is 3.74 milliseconds and equals a data rate of 267.38 Hz. Next, we sent the same data as a raw byte stream which consists of 2 bytes per value and one byte for the packet time. This sums up to fixed length of 15 bytes including carriage return and line feed. With this method, we calculated an average packet time of 2.1 milliseconds which equals a data rate of 476.2 Hz.

DMP Data

To verify the performance of the DMP algorithm on the MPU9150 sensor, we conducted a test in which we moved the device in free motion containing linear movements as well as rotations around all axes for 3 − 4 seconds. After that, the device was returned to a fixed position on a table and the roll, pitch and yaw angles were read from the sensor. This process was repeated for three minutes. As a reference, we recorded another data set in which the sensor remained in the fixed position to determine the drift around each axis, also for three minutes. In Figure 3.11 we can

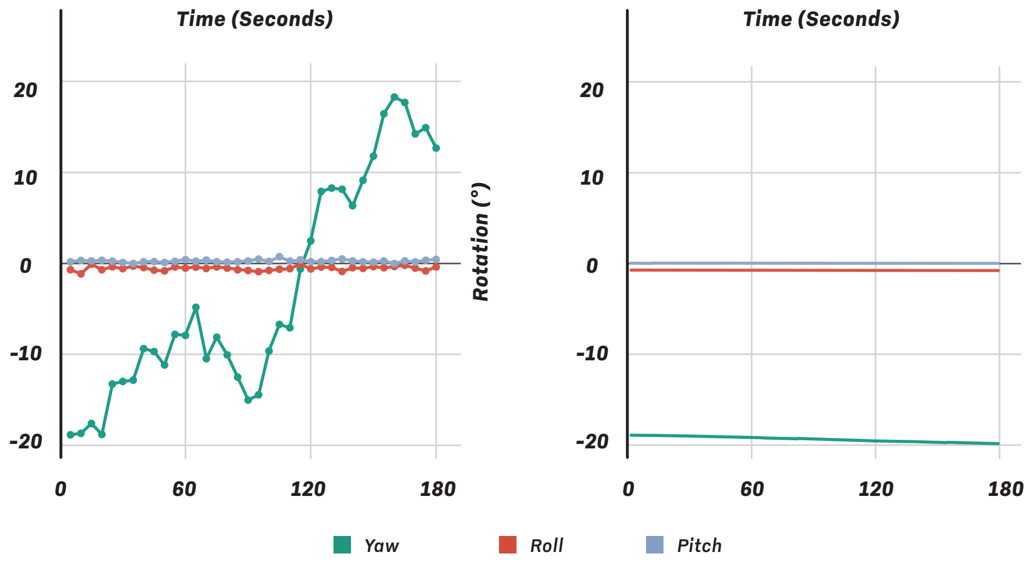


Figure 3.11: Drift of the DMP Euler angles after repeated motion (left) and in fixed position (right).

clearly see that in both cases, the yaw angle, which is the rotation around the z-axis of the sensor, tends to drift while the two other angles stay stable. This is especially significant in the first experiment that involves movement of the module, where it drifts by 11 % in three minutes versus 0.3 % in the static test. Because of the drift we observed and although the MPU9150 is advertised as a 9-axis sensor with an inbuilt sensor fusion algorithm, we assume that the DMP firmware does not incorporate the magnetometer data into the sensor data fusion.

Conclusion on the IMU Sensor Evaluation

As we already stated, the MPU9150 was a unique device at the time we designed the B2DK. However, regarding the technical specifications we find that it is equal to or better than consumer grade sensors in platforms examined in Section 2.4. Our measurements for the accelerometer and gyroscope raw data showed tolerable levels of noise. In a second experiment we have confirmed that we can extract data from the device and stream them to a computer at much higher rates than required for our applications. The most interesting feature of the MPU9150 is the integrated DMP. Unfortunately we discovered that the orientation data drifts around the z-axis, which is most likely caused by a lack of magnetometer date integration into the sensor fusion. Since the fusion algorithm only exists as a binary file, successful modifications without support of the manufacturer are highly unlikely. In general we can conclude that despite some flaws, the MPU9150 is a suitable device for the BRIX$_2$ platform. It fulfills all requirements of our potential applications and even adds features like the DMP, which will likely be of great use for many scenarios.

3.5 Conclusion

In this chapter we started with summarizing and ranking properties, functionalities and features of the platforms we surveyed in Chapter 2. In the next step we identified functionalities that are mandatory for the BRIX$_2$ system if we want to use it for a broad range of applications in ubiquitous computing. They are going to be included into the central element of our modular platform, the base module. Functionalities with lower priority were considered optional and are to be implemented as extension modules. We then started to develop a design concept for BRIX$_2$ inspired by several other platforms we had analyzed earlier. In a second iteration. we narrowed down our concept and worked towards an actual implementation.
Since some aspects of our toolkit like the selection of the wireless transceiver or the motion sensor had to be tested in practice, we developed the B2DK. This modular test platform allowed us to rapidly change the hardware configuration and and test the system with different wireless transceiver modules or microcontrollers. We performed extensive experiments in order to select a suitable wireless transceiver for BRIX$_2$. All four potential technologies were tested in three different scenarios with different parameters each in order to cover most use cases that are found in ubiquitous computing applications. Although we had already selected an IMU sensor because of its superior features and properties, we also tested it in practice to verify that it would suit our requirements. Furthermore we evaluated the charge electronics for the internal battery as well as the microcontroller, which we also selected earlier. As a result of our experiments and testing, we selected the Texas Instruments CC1101 as the wireless transceiver module for BRIX$_2$ and verified our selection of other core components. At this point we were ready to implement the first iteration of BRIX$_2$ modules, which we present in the next chapter.

4 Implementation

With the results we obtained from our experiments described in Section 3.4.1 and based on the early design concept we presented in Section 3.2, we are now able to implement the BRIX$_2$ platform. In this chapter, we first revisit the initial design concept we worked out in Chapter 3 and refine its key aspects up to an extent where they can be included in a first implementation. After that we describe how we implemented the three major aspects of BRIX$_2$ namely the hardware, separated into electronics and mechanics (enclosure), the software components as well as the documentation. Regarding the hardware, we only present the base module in this chapter and discuss the implementation of the different extension modules separately in Chapter 5.

4.1 The BRIX$_2$ System Design

We begin this section with discussing the physical appearance of BRIX$_2$ regarding technical as well as aesthetic aspects and briefly describe how the product is supposed to look. Subsequently we present more detailed technical concepts for the electronics as well as the software components of the BRIX$_2$ platform. Each of these design aspects is discussed with an increasing level of detail, so they directly lead over to the implementation sections.

4.1.1 Physical Appearance

The physical appearance of BRIX$_2$ is not only determined by purely technical requirements, but has also an aesthetic component. In the following, we will discuss different aspects of the physical concept and look of BRIX$_2$.

Appearance Aspects: What We Intend to Convey

We build BRIX$_2$ for several different groups of users. For some, the system will be the first physical computing or even electronics platform they encounter. Intermediate users might have seen similar systems and can compare them to BRIX$_2$. Professional developers are interested in a tool that allows them to efficiently and quickly fulfill their tasks. For the first group, we consider it relevant that a certain level of abstraction is part of the product's appearance. Beginners might be repelled by the look of a PCB, because it conveys something complex, technical and unknown. Handling a device like that appears far beyond their scope of abilities. Intermediate users might want to be able to take a glimpse of the underlying technology of the device they are working with, while expert users are interested in an efficient and functional design. The looks of existing products range from bare PCBs like we found in Section 2.1 to abstract, sealed blackboxes like the for example MTI-10, see Section 2.4.7. On the one hand we intend to design a product whose outward appearance is abstract and playful enough to invite beginners to use our system and facilitate their curiosity. On the other hand, we aim to provide a clean and functional look for advanced and expert users that transports the concept of an efficient and capable system whose capabilities allow them to implement their applications quickly and easily.

Electronic Building Blocks

Modular systems can have different granularities by design. littleBits for example only features a single function per module or building block. On the one hand, this allows to design applications without overhead, because only functions that are required in the application will be included. On the other hand though applications get physically bulky because electrical and mechanical connections between all elements can use up more space than the actual components. As a major goal we try to design BRIX$_2$ as compact as possible, which is why we are going to include a basic functionality in a base module which represents the central component for any application. This way we risk potential overhead of functions, but reduce the physical dimensions of the system. In order to keep this overhead as small as possible, we already determined this fundamental set of functions in Chapter 3. As a result, users will be able to add functions to the base module in order to adapt BRIX$_2$ to their desired application. This concept is close to the former BRIX system, which is why we next take a look at our experience with that system to determine which of its features is going to be carried in the current implementation.

Former Experiences with BRIX

The BRIX system consists of three different types of modules: base module and battery module, each the same size (48×32×13.5 mm) as well as extension modules which are a third of the size (16×32×13.5 mm). Base and battery module are stacked onto each other to form the minimal functional unit. Extension modules can be stacked onto one of the three identical extension ports on the base module. When several extension modules are used, three extension connectors allow a greater level of integration than a single one, which would mean stacking all modules on top of each other, resulting in a bulky device. This proved to be practical in actual applications and could be kept as a concept for BRIX$_2$. The concept of a separate battery module was initially developed to allow quick swapping of the battery instead of charging the whole device. This way, for example a data recording session could continue without any waiting time. It turned out that this feature was actually never used because our mobile applications did not required operation times longer than a single battery lasted. On the downside, the separate battery used a lot of space because it had its own enclosure and its own PCB with charge electronics. With BRIX$_2$ we should be able to integrate the battery into the base module while keeping it at the same size.

Experiences with Lego Bricks as Enclosures

As we already stated in Section 2.6, a prominent feature of the BRIX system that also coined its name are the enclosures made of Lego bricks. People also sometimes even referred to the BRIX system as “Legos”, because the electronics were perfectly integrated into the bricks. While using the BRIX system, the friction based connection between the modules proved to be sufficiently reliable. Also the bricks allow a high level of integration, because enclosures can be scaled precisely, regarding for example the height of extension modules. When regarded as a raw material, Lego bricks can easily be machined in subtractive processes like Computer Numerical Controlled (CNC) machining and glued together to form bigger, rigid structures. Also they are inexpensive and widely available in a great variety of sizes and colors. Apart from their properties as a material, Lego bricks are also a metaphor for playful and exploratory construction, flexibility and modularity. Building with Lego bricks is a practical, hands-on experience that many people are already familiar with. We can use that metaphor and positively associations to motivate especially beginners to use BRIX$_2$ and to take the edge off the impression of a technically complex system. In general, reactions to BRIX’ visual appearance were positive among all kinds of users. Because of our encouraging experiences with Legos as enclosures for BRIX, we are also going to incorporate this distinct design element in BRIX$_2$.

Case Study: An Enclosure Concept for BRIX$_2$

In the following we briefly introduce the enclosure concept for BRIX$_2$ based on our former experiences. Since the general concept of the system changed, so does the enclosure. Higher production volumes planned for BRIX$_2$ in contrast to BRIX do also affect the enclosure concept which has to be optimized towards manufacturability.

Base Module The physical size of the BRIX base module in terms of Lego units is 4 x 6 studs. This design allows to put three extensions on top next to each other, each encased in 2×4 stud bricks. The height is a standard brick height plus the bottom plate, which sums up to 13.5 mm. The battery module is the same size. We intend to integrate the battery into the base module while keeping it the same size as before. This is possible through the use of low profile extension headers and a different battery form factor. The result is a size reduction to 50 % while we increase the number of features at the same time. In the BRIX system, the power switch and USB socket were located on the battery module and have to be included into the new BRIX$_2$ base module. To achieve that we have to use smaller components and a more dense PCB layout. Also all interactive elements, the indicator LED, power switch and USB connector should be located on the same side of the module. This is relevant for scenarios where the module is only partially accessible, for example integrated into a sewn pocket in an electronic textiles project.

Extension Modules As already implemented in BRIX, the extension modules are going to be encased in single, 2×4 stud Lego bricks, closed on the bottom with a 2×4 Lego plate. Other sizes like 4×4 or even 4×6 for extension modules with increased complexity should be compatible as well. The height of an extension module can easily be adapted to its contents. For example an extension module which only contains a hall sensor can be much flatter than a module that contains a vibration motor. This flexible height means we can use space more efficiently. However, in the final design, we should stick to multiples of base plate heights. This allows us to level out extension modules of different sizes using standard Lego 2×4 plates, should the application require this.

Making Enclosures out of Lego Bricks In the former BRIX project, we used only Lego bricks to build the enclosures, which requires a lot of precision machining and gluing. In the recent years, the capabilities of 3D printers rapidly advanced, so we might be able to incorporate this manufacturing technique in the cases for BRIX$_2$. This reduces the amount of work for each enclosure, making it more feasible for larger scale productions, which we plan for around 100 devices. However, the cases for the extensions are still simple to build as they do not require a great amount of machining. This is why we plan to keep this concept in the current project.

4.1.2 BRIX$_2$ Technical Concepts

In the following we present the technical concept of the BRIX$_2$ system, regarding mechanical as well as electronic aspects. First we describe the concepts for the base module with a focus on the electronic system design. After that, we introduce the design challenge for a modular system and present some technical details on the interconnection between modules and an outlook on the design of our extension modules, which are covered in the next Chapter.

BRIX$_2$ Base Module

The base module contains several blocks of electronic components that have to be interconnected and placed on the PCB. The central components are the two micro-controllers, user controller and system controller. In the next paragraphs, we present our microcontroller concept along with some details on the LED and the power management. Other components like the sensor or the RF interface have either already been covered or are straight forward in terms of design, which is why they are not included in this conceptual section. First, we take a look on the constraints for the PCB design process that emerge from mechanical considerations, for example the enclosure.

Mechanical Constraints for the Base Module PCB The first and most important mechanical constraint for the base module PCB is the physical dimension. In Section 4.1.1 we defined the physical size of the enclosure and thereby the maximum size of the PCB with 44.6×28.7 mm. The height is limited to 9 mm for the PCB populated with components on both sides plus the battery. This constraint is given by the space inside a standard Lego brick when closed with a Lego plate from below. We decided for three extension connectors that allow to attach up to three extension modules next to each other, since their size is a third of a base module. If we construct the enclosure in the classical way as we did with BRIX, it would be ideal if the switch and the USB connector are located in the center of one of the 2×4 bricks that form the enclosure. This way, the cutouts for the components can be machined before assembly. The LED has to be placed at a translucent area of the enclosure. This can either be achieved by using a transparent/translucent Lego brick or simply be leaving a hole for the LED. The wireless module has to be placed at the edge or corner of the PCB according to the design rules published by the manufacturer [113]. Instead of a wireless module with a PCB antenna, we can also consider the version with an antenna connector. This allows us to place a wire antenna inside or outside the module. The motion sensor should be placed in the middle of the PCB to keep the axes symmetrical and the center of rotations close to the center of the module itself. The battery should fill the remaining space as efficient as possible in order to maximize the capacity. Since the top side of the PCB has to be as close to the top plate of the enclosure as possible and most of the space is occupied by the extension connectors, the most feasible spot for the battery is below the PCB. This means we have to organize the components on the bottom of the PCB in a way that results in a maximized space for the battery. For the extension headers, we should use the female variant of the fine-pitch headers for two reasons. First, the female connectors have a lower profile as the counterparts and second, the pins are less accessible mechanically on the female header, so they are protected from mishandling resulting in short circuits.

Microcontroller Concept A fundamental part of the BRIX$_2$ platform is the microcontroller. It allows users to program the system and adapt it to their individual applications. We already discussed which microcontroller we were going to use as our main application controller (in the following called *"user controller"*), however we also mentioned in Section 3.4.1 that we are going to incorporate a second microcontroller which is going to be part of our wireless transmitter unit and fulfill peripheral tasks (in the following called *"system controller"*). One might argue that the whole BRIX$_2$ system, including the RF stack can well run on the main microcontroller along with the application users implement. However, we already identified that there is not much flash memory and SRAM available on the user controller, the Atmega32U4 we already selected in Section 3.3.3. Using a second controller, we do not only double the total available memory but also have a second CPU that can handle additional task parallel to the application running on the user controller, such as reading and configuring the motion sensor, reading the battery cell voltage, establishing wireless connections and displaying the status of the module using an LED. All this can be accomplished without the user even noticing that those processes are running in the background. If those functions are not required in a particular application, the second controller can spent most time in sleep mode where it consumes only little energy. Both microcontrollers can also share the same external clock signal, so we do not need additional active components for the second controller. The footprints of both microcontrollers are rather small, so they do not take a lot of space on the PCB.

User Controller The user controller makes the BRIX$_2$ platform programmable and provides the flexibility required to use it to a wide variety of applications. It is the only component that is supposed to be modified by the user. We decided for the Arduino compatible ATmega32U4 mainly because of its built in implementation of a USB stack. It allows direct access to the device from a host computer via USB, which means increased convenience for users. The controller has two internal serial ports, one is connected to USB, the other one can independently communicate with other devices through a UART, see Figure 4.1. The ATmega32U4 has 26 GPIOs of which we try to connect the maximum possible amount to the extension header. The ability to do so is limited by the routing space on the PCB and the number of pins on the extension connector.

System Controller Although the system controller is not supposed to be maintained by users, the option to do so, for example in order to update the firmware would be practical. The easiest way to achieve this is to keep it Arduino-compatible as well. This would also allow us to use Arduino libraries for the system controller firmware during development and would not require additions to our already existing toolchain. Since BRIX$_2$ is already USB compatible through the user controller, the system controller does not require an individual USB interface. We can simply use the most popular microcontroller used for Arduinos, the Atmel ATmega328. It is available in a smaller package than the ATmega32U4 and the features are basically identical, see Section 3.2.4. One crucial task of the system controller is to configure and control the RF module and offer an abstract interface for the user controller to access the device. Besides that, it is able to measure the battery voltage through a voltage divider and inform users about the status of their BRIX$_2$ module using an individual LED. If required, the system controller can also access the sensor and some of the extension modules via the I2C bus.

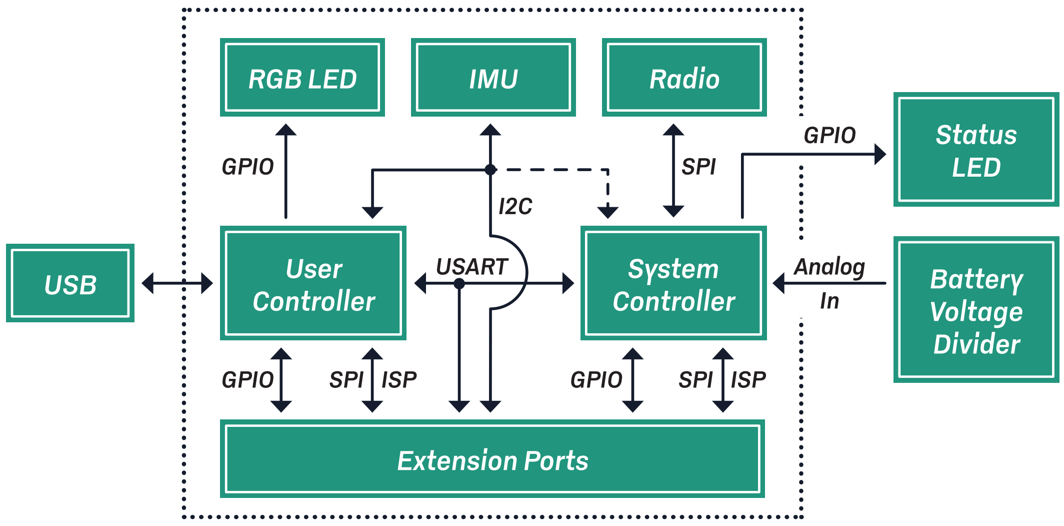


Figure 4.1: BRIX$_2$ base module block diagram.

Interfaces on the Base Module In order for all components on the base module to work together flawlessly, they need to be interconnected by a number of different signals. In this paragraph we list and describe these signals and point out their role in the communication concept on the base module.
USB: The USB connection is established by the user controller. It can be forwarded to the system controller by bridging the internal serial ports *"serial"* and *"serial1"* of the user controller, see 4.2.2.
ISP: Both microcontrollers need to be initially flashed with a bootloader or their default firmware. This is done through the ISP interface which mostly uses the SPI pins of each microcontroller. To allow access to both ISPs, we have to connect them to the extension header.
I2C: This bus allows both microcontrollers to communicate with the motion sensor and compatible devices that are connected over the extension header. Since the microcontrollers can both be masters on the I2C bus, this scenario has to be taken care of in software.
UART: A serial connection between both microcontrollers allow users to send requests and receive data from the system controller. This way it is possible to exchange data wirelessly between two separate BRIX$_2$ modules the same way as using

a serial connection between both microcontrollers. Through abstraction, the wireless interface behaves the same way.
Both the user controller and the system controller are connected via UART in order to exchange data. This allows users for example to read the battery status from the system controller or to submit packets for wireless transmission. The UART signals are also connected to the extension port in case an application requires to communicate with an extension over serial. For doing so, the UART of the system controller needs to be deactivated temporarily to prevent short circuits.
SPI: The SPI interfaces of both controllers are connected to the extension header separately. The system controller and the RF transceiver are connected via SPI.
GPIOs: Analog as well as digital GPIOs from the user controller are connected to the extension header. Three I/Os with PWM capability are connected to the RGB LED. If there are free pins left on the extension header, GPIOs of the system controller might also be connected to the extension header.

Power Management The BRIX$_2$ electronics are supplied by two power sources, the USB port and the battery. A power multiplexer is required in order to supply the system via USB if plugged in or via the battery otherwise. In order to provide a stable and constant input voltage for the system, a voltage regulator is necessary. Although more external components such as inductors are required, a switching regulator is to be preferred to a linear regulator because of its higher efficiency. The BRIX$_2$ base module has to be powered with a voltage level of 3.3 V, because there are components like the motion sensor and the RF transceiver which are not 5.0 V tolerant. For charging the battery, a LiPoly charge controller is required to not damage the battery during the charge cycle, see Section 3.3.2.

LEDs The purpose of the LEDs is to display the status of the device itself and the application that is running. In Section 3.2.4 we already proposed the use of an RGB LED which is completely programmable by the user. Independent of the RGB LED which is controlled from the user controller, we also propose another LED connected to the system controller. This can reliably indicate whether the BRIX$_2$ module is running and in which state it is, independent of the user application. For applications where low power consumption is crucial, users can actively override the system controller and turn off the status LED, see Section 4.4.2. A third LED indicates the charge status of the battery. It is connected directly to the charge controller, thus is independent of any programming. This way the user is directly informed if the charge process of the battery is still running or if the battery is fully charged and the device can be disconnected. Both the system status LED and the charge status LED can be low current, single color LEDs. The status LED is preferably green for optimal visibility at low current whereas the charge status LED should be red to indicate that a critical operation is in progress.

Physically Extending BRIX$_2$

Systems like littleBits, Arduino and Tinkerforge Bricks are good examples for extensible platforms. They consist of a central element which can be combined with additional elements to increase the functionality. That way, the system becomes a kit of different modular units, or "electronic building materials", as Ayah Bdeir calls it in [36].
In our survey in Chapter 2 we have presented different approaches towards physically extensible electronics, summarized in Section 3.1.2. In order to keep a system compact, even when extended with multiple additional modules, a stacking method is superior to implementations like littleBits. FFCs or cable based connections also allow high levels of compactness, like for example the Seeedstudio Xadow system. However cables also represent a weak element in terms of electrical connectivity and are also potentially prone to wrong handling, which may result in possible damage to the system.
Stacks with fixed, board-to-board connectors are common whenever bare PCBs are used, like for example the products we presented in Section 2.1.4. Many platforms use extensions that are the same size as the base board and simply stack as many as required onto each other. Often, this arrangement wastes a lot of space, since the extensions contain less components than the base board, for example [114] or [115].
In our prior work on BRIX, we implemented an extension port system with three parallel connectors that allow to put three extension boards on top of the base board next to each other. Should an extension board be too complex, it is still possible to make one that requires two or even three slots. This way, the available space is separated into smaller units and can be filled more efficiently.
The general concept of BRIX worked well, but had a strict limitation of the pin count. Also the connectors were not rated for a significant number of insertions, which resulted in unreliable electrical connections. As a conclusion, we are going to roughly adapt the mechanical extension module concept of the BRIX system with major changes on the connector and the signal organization itself, which we present in the following.

Signal Organization In Section 3.2.4 we already discussed two common types of extension connectors we identified while analyzing different platforms: full breakout and protocol based connectors. The former try to connect as many microcontroller signals as possible to the extension connector, which results in a high number of signals and pins on the connector. The latter use an abstraction layer in form of a protocol like I2C that adds complexity but drastically reduces the pin count of the connector. From our experience with BRIX we found that the protocol based connector unnecessarily complicated the communication with the extension boards, because the extensions need to be able to comply with that protocol. This can create a lot of overhead when the components on the extension are rather simple, for

example just an RGB LED. In the case of BRIX$_2$, we decided to use Arduino as a base. A lot of examples and libraries for connecting external components like sensors and actuators to an Arduino, which features a full breakout extension connectors do already exist. We would not be able to pass this knowledge on to our users directly if we abstracted our extension connector. Instead we would have to adapt all these examples to our platform.
By using a more advanced PCB production process than for the BRIX modules, we are able to route most signal of the user controller and even some of the system controller to the extension connector. This leads to less complex and thereby less expensive extension modules with a lower energy consumption, a transparent interface between base module and the extensions, greater Arduino compatibility and increased flexibility.

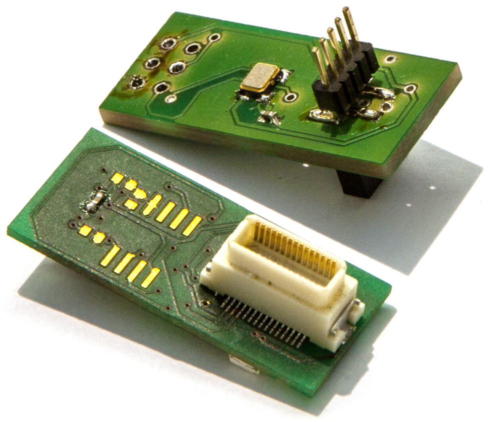


Figure 4.2: Comparison of connection headers on BRIX and BRIX$_2$ extensions.

Extension Connectors In Chapter 2, we identified two prominent types of board-to-board interconnects: female/male pin headers with a 2.54 mm (0.1 in) pitch and dual-row fine pitch connectors. Standard female pin headers are more accessible than fine pitch headers, which can only be interfaced with the designated receptacle and not pin by pin with for example a simple wire. Also they are widely available and inexpensive. The downside of standard pin headers is their rather big footprint on the PCB as well as the low number of insertion times before they wear out. The footprint can be reduced by using finer-pitch pin headers such as 2 mm or 1.5 mm. However, this is a deviation from the standard and again reduces the accessibility and compatibility of the connector and combines the disadvantages of both solutions. In the previous section we argued for implementing a full breakout extension connector. In a compact platform like BRIX$_2$, this can only be accomplished using fine pitch connectors simply because of their smaller footprint. The disadvantage of a poor accessibility of the single pins by external hardware is actually not as relevant for BRIX$_2$, because users are given a library of extension modules to work with, which reduces the demand for custom external hardware. Should it be necessary to connect such hardware, they can use an adapter board that breaks out all signals of the extension connector to more easily accessible solder pads. See Figure 4.2 for a comparison of a 5-way 2 mm pitch pin header on a BRIX extension module (top) and a 30-way fine pitch connector on a BRIX$_2$ extension module (bottom).

Extension Modules At this point, we only predefine a rough concept for our extension modules, since we designed a number of devices that are quite diverse and different from each other. In general, the mechanical size of the PCB is determined by the inner size of a single 2 x 4 Lego brick, 29×23 mm and the placement of the extension connector on the lower side. Also, the lower side should not contain high profile components which would collide with the studs of the 2×4 Lego plate that seals the module at the bottom, see Figure 4.3. Extension module electronics need

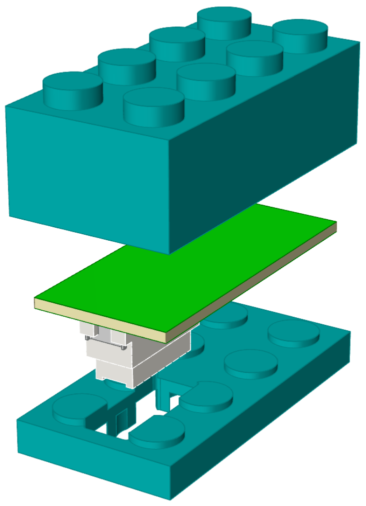


Figure 4.3: Exploded view of an extension module: PCB (middle) between the enclosure top part and bottom plate.

to be 3.3 V compliant and must not draw more current than the regulator on the base module can provide. The maximum height for components on the top side is determined by the inner height of a single Lego Brick minus the thickness of the PCB itself.

4.1.3 Software Concept

Similar to the Arduino platform, the BRIX$_2$ toolkit is not only a hardware but also incorporates software components for tasks like writing firmware, compiling and uploading of binaries to the flash memory of the microcontroller. In this section we introduce the software components we aim to integrate into our kit in order to allow easy and accessible learning and prototyping with the system, even for users without any experience in microcontrollers and programming

Programming Toolchain

In Section 3.2.3 we already introduced the concept of toolchains for microcontroller firmware development and the Arduino IDE. By using the Arduino software package for our project, we are offered an all in one solution: editor, file manager, library manager, compiler toolchain, uploading tool and language reference. All these components are open source and free for download as well as tested by hundreds of thousands of users. Since Arduino explicitly aims at the same groups of users as we do (beginners, students, advanced and expert users who want a quick prototyping cycle), it is perfectly suited for BRIX$_2$ as well. In order to make our platform and the Arduino IDE fit together seamlessly, we can do two things:

- **Match the IDE to our hardware**: Since the Arduino IDE is open source, we are allowed and able to fork our own version of it and redistribute it to our users. We would be able to carefully modify it exactly to fit our needs, but the level of maintenance required is high. Alternatively, we could have our users download the standard IDE and patch it themselves. This would at least require a detailed description in our documentation and basic knowledge about computers and software on the user side, which might represent a basic hurdle for users and could potentially lead to a rejection of our toolkit.

- **Adapt our hardware to the IDE**: This means we have to be 100% compatible to one of the boards that is supported by the IDE natively. The major advantage is that we do not need to modify the IDE and maintain our own version of it.

As we can see, modifying the IDE for our purpose is not applicable. This is why we design our hardware in a way that it is 100 % compatible to common Arduino boards that are currently supported by the IDE. Users can simply download the latest version of the IDE from the Arduino website and use it with BRIX$_2$ out of the box.

The BRIX$_2$ Arduino Library

The Arduino IDE ships with a number of libraries that simplify a lot of tasks such as controlling an LCD or using buttons as input. Beside the libraries that ship with Arduino, there are numerous third-party libraries for all kinds of devices and purposes that can be added to the Arduino IDE. Usually, libraries are supplied along with example sketches (firmware projects for the Arduino), which users can just upload to their device, connect the hardware they want to use with the library and it just works. This is a major feature of Arduino, because implementing the functionalities of the library usually requires profound knowledge about the microcontroller and the principles of the connected device. Using the library and the examples, users have a base to work from and a fallback in case their customized version of the example sketch breaks. If users want to know more about the underlying principles of how the hardware or library works, they can just look at the code, since every library is supplied as open source. This means that knowledge is not initially needed to succeed, but is always accessible. For our BRIX$_2$ system, we aim to supply an Arduino library as well, which abstracts things like reading the motion sensor, communicating wireless and controlling the extension modules we offer. All these functions of course need example sketches so users can try them out, experiment and modify the examples towards their own customized sketches and projects. Some functionalities like for example support for the motion sensors are already covered by third-party libraries, which are constantly being improved. Instead of implementing and maintaining those functionalities ourselves, we can also integrate those components into our own library given that the license of the external library allows it.

4.2 Implementation of the BRIX$_2$ Electronics

In this section we describe the electronics of the BRIX$_2$ base module in greater detail. The fundamental components such as microcontrollers and the sensor were already selected in the earlier chapters. In the beginning of this chapter, we presented the general concept for the electronics in terms of communication between the components. Now we take a closer look on how we practically implemented the base module electronics. First, we describe the structure and communication channels of the base module, the power supply and the extension port in detail, followed by the physical organization of the components on the actual circuit board. Finally, the description of the PCB leads over to a section about the enclosure implementations for the BRIX$_2$ base module.

4.2.1 A Side Note on the Selection of Components and Technologies

It is to be kept in mind that the BRIX$_2$ base module was designed and developed in 2011/2012. Especially in the fields of microcontrollers and wireless interfaces, standards have developed further. This is why some components used in this project might seem outdated at the time of writing. As technology evolves more and more rapidly, systems will most likely be obsolete to some point as soon as a design is finished. This however does not make a device less capable than at the beginning of the design process, but it offers designers greater opportunities for the next revision of BRIX$_2$.

4.2.2 Core Circuit: Microcontrollers, Sensor and Wireless Interface

For better overview, we split up the detailed description of the base module electronics implementation into three parts: core circuit, power supply and extension ports. Since the core circuit contains almost all functionality of the BRIX$_2$ base module, we present it first.

Structure and Communication Channels

Apart from the extension connector, the BRIX$_2$ base module has two main communication channels to the outside world: USB and the wireless interface. The first is certainly most significant, because it allows users to upload custom software to the user controller and exchange data between a host device and the BRIX$_2$ application. Thanks to its native USB implementation, the user controller can be directly connected to the USB header, using termination resistors on the data lines. The system controller handles the wireless interface, which it connects to using the SPI bus, as well as other system functions. To allow the applications running on the user controller to communicate with the system controller, the secondary UART of the first is connected to the primary UART of the latter, see Figure 4.4.

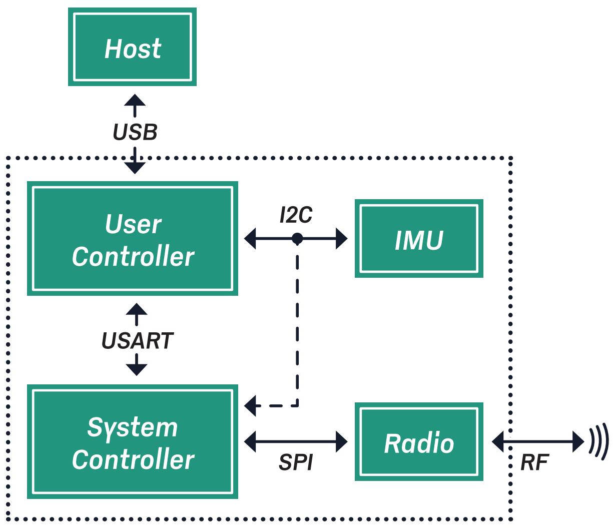


Figure 4.4: Core components and their communication structure on the BRIX$_2$ base module.

USB Access for the System Controller Although users are only supposed to communicate with the system controller indirectly through their application on the user controller, there are two scenarios in which direct USB access to the system controller is necessary. First, in order to access the wireless interface directly from the host system, turning the BRIX$_2$ base module into a USB/RF bridge and second to update the firmware running on the system controller without additional ISP hardware. The first scenario can be implemented in software. A simple sketch on the user controller could relay data from the secondary UART (connected to the system controller) to the primary UART (USB) and vice versa. This way the system controller is accessed via USB as if it was directly connected. The second scenario also uses this sketch, however there are two more requirements. First, a bootloader on the system controller that makes it compatible to the Arduino IDE. Second, the reset line of the system controller needs to be controllable by the user controller. Usually, the ATmega328 that we use for our system controller is programmed through an FTDI USB/Serial converter and the reset line is handled by the DTR signal of the serial port. The sketch on the user controller could capture this signal from the USB side and control the reset pin of the system controller via a GPIO accordingly. This way users and developers can easily upload sketches to the system controller as well.

Sensor Access via I2C The MPU9150 IMU sensor can be accessed via a standard I2C interface with a bit-rate of up to 400 KHz (*"Fast Mode"*). We connected the bus lines Serial Data Line (SDA) and Serial Clock Line (SCL) to the I2C interface of the user controller. This way, the user's application can access the sensor directly. We did not connect the system controller to the same bus, because it would also act as an I2C master when trying to access the sensor at the same time the user controller does. This would lead to illegal bus states if not taken care of in software. Special applications however might require to read the IMU sensor from the system controller. In this case, two $0\,\Omega$ resistors can be populated on the PCB to connect the system controller to the I2C bus.

RF Interface Access via SPI The Anaren AIR A1101R08 RF module is connected to the system controller via SPI. The chip select pin is connected to a GPIO of the controller. This way the SPI bus is not fixed to the RF module but can also access external SPI devices that can be connected via the extension header. The RF module also has two pins that can be configured for various functions and are connected to two GPIOs on the system controller. The user controller can not directly access the RF module without external hardware. In theory, an extension module could be created that connects the SPI buses of both controllers, allowing the user controller to access the RF module, should that ever be necessary.

Clock Signal Generation

Both microcontrollers require a 16 MHz clock signal to operate. This can either be generated by a passive crystal resonator connected to both clock pins of the microcontroller along with two capacitors to GND or by an active external oscillator connected to only a single clock input pin of the microcontroller. Crystal resonators are usually less expensive than crystal oscillators and also have a smaller footprint, however a single crystal resonator can not supply clock signals for two microcontrollers, thus both would need their own crystal, as for example on the Arduino Uno board [116]. A single crystal oscillator can supply its clock signal to several devices, so we can use it for both microcontrollers. We selected a TXC TD 16.000 MEMS oscillator [117], which is 3.3 V compatible and available in a rather small, $2.5{\times}2$ mm LGA package.

USB Connector

In order to connect a BRIX$_2$ module to a host via USB, different connectors can be used. On BRIX, we decided for a MiniUSB-B connector, since it was one of the most popular connectors for mobile devices at design time. This means cables to connect Mini-B to a regular USB port on a host system were common and easily available. In 2009, microUSB became an EU standard for cellphone charging connectors [118].

As a result, this connector was quickly widespread not only for cellphones, but for mobile devices in general. MicroUSB has two more advantages. First, the connector has a lower profile than the MiniUSB-B (3 mm versus 4 mm). Second, the MicroUSB connectors are more durable and rated for at least 10000 insertion cycles, which is 100% more than MiniUSB (5000 cycles) [119]. For those reasons, we chose microUSB-A/B receptacles that are compatible to any MicroUSB cable. This way BRIX$_2$ is chargeable with the same equipment as almost any smartphone or other mobile device.

4.2.3 Power Supply: Voltage Regulation, Charging and Battery

The power supply circuitry consists of three major parts: The voltage regulation, the battery along with its charge controller and the power multiplexer, see Figure 4.5.

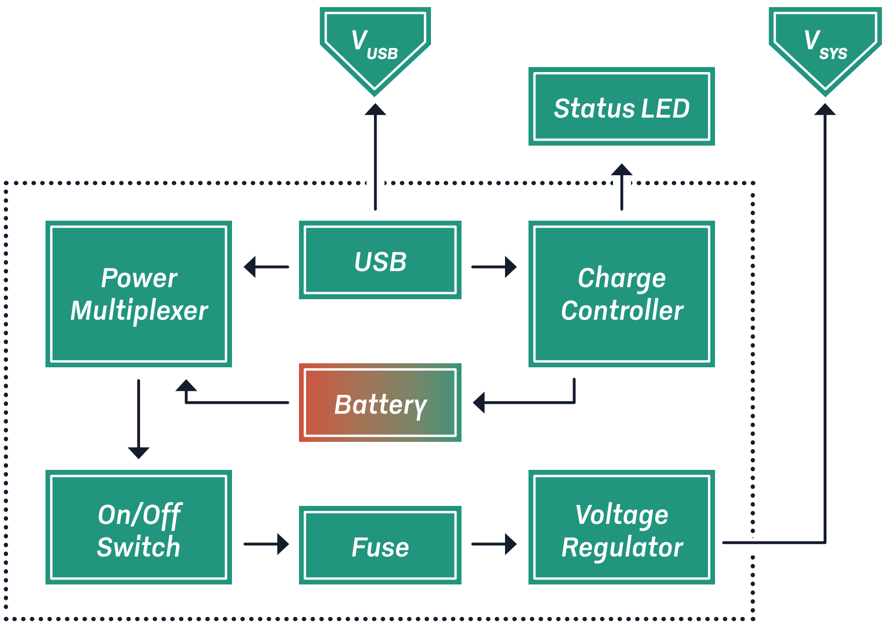


Figure 4.5: BRIX$_2$ power management block diagram.

Voltage Regulation

We use the voltage regulation that we already tested with the B2DK, because it performed according to our requirements in our tests. The regulator is rated for 500 mA maximum current and has an over-current and over-temperature protection.

For additional protection, we added a 500 mA fuse on the input line of the regulator. This way we provide redundant safety mechanism against over-current.

Power Multiplexer

As we already concluded in Section 3.4.2, the existing circuit encounters problems while the application is running and the battery is charging, because power is constantly supplied by the battery. This leads to longer charge times and on the long run to a reduced lifetime of the battery. To solve this issue, we need to add a component that selects the most feasible one of the two input power sources to supply our application while the battery is charged via USB. Such a component is called power multiplexer and is common in mobile applications. We selected the Texas Instruments TPS2111 [120] autoswitching power MUX because it is widely available, inexpensive and has a small footprint. Besides a single resistor to configure the current limiter feature, it requires no additional passive components. The device basically connects the input source with the highest voltage to the output. This way our application is directly supplied by the USB when connected to a USB port, while the battery charges independently.

Charge Controller and Battery

Like the voltage regulation circuit, the charge controller was already tested on the B2DK and performed as required. An element we had not selected before was the battery. LiPoly batteries are sold in a great variety of shapes, sizes and capacities, so we could almost certainly leave an arbitrary space for our battery and select one that fits. However, the Product Life Cycle (PLC) of many LiPoly batteries is rather short, so the products might no longer be available after months or even weeks. This is is serious issue when developing a device like BRIX$_2$, which is supposed to be reproduced by other people and not just designed for a single production run. If vital parts are no longer available, BRIX$_2$ modules can no longer be made without design changes. This problem can be avoided by selecting a battery that will be available for a long time, because there is an ongoing demand for it. This also tends to lower the prices as larger volumes of the device are produced. We chose to base our design on a battery which is available as a replacement part for a recent consumer product with a large production volume. Regarding the shape, size and capacity, we found that the battery of the Apple iPOD classic (EC-008) fits perfectly into the BRIX$_2$ enclosure and already contains a circuit that protects it from over-currents and deep discharge. EC-008 compatible batteries are widely available from different suppliers and relatively inexpensive. For optimal use of space inside the BRIX$_2$ enclosure, we have to adapt our PCB layout in a way that components on the bottom side leave room for the battery, which takes up around two thirds of the surface area. The capacity of the battery is 450 mAh, which is even more than the battery in BRIX had and totally suitable for our application. We estimate a runtime of at least 5 hours for an average, mobile application.

Battery Voltage Monitoring

In many applications, it is interesting to know the current charge of the battery in order to estimate how long it will last. The cell voltage can be used as a base for an estimation of the remaining charge, since it drops when the battery discharges. LiPoly batteries have a characteristic discharge curve that is almost linear for around 70% of the discharge process. In practice it is probably easiest to use a lookup table to determine the battery charge based on the cell voltage. The cell voltage can be measured by an analog input pin of one of the microcontrollers. Since this task is more related to the whole system itself, it should be performed by the system controller. This way we can easily wrap all measurement functions and provide the user with either the correct voltage or the remaining charge. However, we can not directly connect the positive lead of the battery to the analog input of the microcontroller because the maximum cell voltage of 4.2 V exceeds the maximum input voltage on an analog input pin, which is in our case $V_{cc} + 0.5\,V = 3.8\,V$. To overcome this problem, we measure the output voltage of the power multiplexer, which in mobile operation equals the cell voltage and when connected to USB equals the USB voltage of 5.0 V through a voltage divider to decrease the voltage to a tolerable level for the analog input pin. We chose high resistor values of 220 kΩ and 100 kΩ for the voltage divider to ensure that the power consumption of the divider is reasonably low.

4.2.4 Extension Port

The extension port is a prominent feature of the BRIX$_2$ platform and allows users to expand the capabilities of the system by adding extension modules. In this section we first present some hardware decisions about the extension port, followed by a description of the microcontroller signals that we chose to connect to the extension port and thus make them accessible for users.

Extension Connector

Earlier in this chapter we argued for a full breakout connector in favor of a bus-based connection to be more compatible to Arduino, allow for easier and simpler extension module designs and to increase the overall flexibility. Since we decided for three parallel connectors, we can not fit standard pin headers on the PCB. Instead we have to go for fine pitch connectors, which have the clear disadvantage to be unable to interface without the matching receptacle. We observed in our survey in Chapter 2 that the Hirose DF17 series board-to-board connectors are often used as an alternative to standard pin headers in case a higher pin density is required. The DF17 connector series has two rows of pins with a pitch of 0.5 mm and is designed for mechanical stability and good electrical contact. The connectors can be obtained for various stacking heights from 4 mm to 8 mm and numbers of contacts from 20 to 80. Given this wide variety of configurations, we can carefully scale our connector to exactly fit our needs. We decided to use a 30 pin connector with a stacking height of 6 mm (board-to-board distance), since it is the maximum size we can fit on the PCB.

Extension Port Signals

Since we are limited to 30 signals on the connection header, our pinout can not be fully Arduino compatible, because for example the Arduino UNO has 29 signals, but we have to keep some signals on the header reserved for different purposes. We present the pinout strategy of the extension header in greater detail in the following paragraph, see also Figure 4.6.

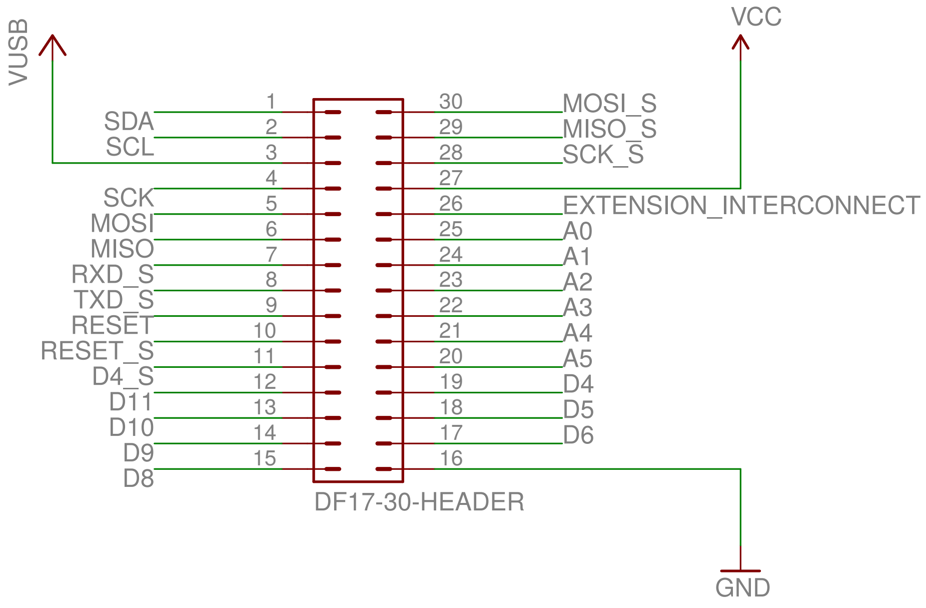


Figure 4.6: $BRIX_2$ extension port signal mapping.

Bus Signals As we already mentioned in Section 4.1.2, there are different buses on a microcontroller and we need access to at least some of them. Crucial, but in general not frequently used is the ISP which allows to program a new firmware on the controller initially and for updates on the bootloader. After that, three of the four data pins of the ISP, Master In, Slave Out (MISO), Master Out, Slave In (MOSI), and Clock Signal (CLK) and can still be partially used as SPI or GPIOs whereas the

RESET signal is only required during programming and is not intended to be utilized by users. The four data pins occupied by the ISP of the system controller are basically lost for user applications and can, apart from the programming scenario, only be used in special purposes. However, they allow extension modules to communicate with the system controller in case that should be necessary. Another signal that covers both microcontrollers is the UART interface with two data lines, RX and TX. The controllers are connected using those two lines and the UART represents the main communication channel that is available for users to address the system controller, for example in order to send wireless packets or read the status of the battery. Both signals are also connected to the extension header, but can only be used in special purposes, because UART is technically not a bus, since it does not support addressing. In case an extension module intends to communicate to one microcontroller via the shared UART, the other microcontroller has to turn its UART off actively to not cause illegal electrical states such as shorts. A bus that is commonly used for extensions is the I2C bus. It consists of two signals, SCL and SDA, and is de-facto standard when communicating between microcontrollers and other active components like digital sensors. The bus supports multiple devices simultaneously. This is why we can also connect the sensor on the base module to the bus at all times and are electrically safe as long as components on extension modules use a different I2C address than the sensor.

GPIOs We have connected 13 GPIOs of the user controller to the extension header as well as a single one of the system controller. From the 13 user controller GPIOs, 6 can be used as analog inputs and 4 can be configured as PWM pins. On these pins, users can generate a PWM signal by simply using the *analogWrite()* function in the Arduino IDE. Other special functions such as interrupt inputs are also implemented on some GPIOs we selected.

Extension Interconnect A single signal on the extension header is reserved as an extension interconnect. The pin can be used to communicate between two extension modules without interfering with any component on the base module. So far no extension module has ever used this feature, but there are certainly applications that might require it, for example for an extension to check if another extension module of the same type is also connected. In this case, the extension module would first check the extension interconnect and if it does not read a specific signal, it would occupy the interconnect with its own signal.

Power Lines We connected two different power lines to the extension connector. First, the system voltage of 3.3 V from the regulator. This is limited to around 400 mA, since the regulator and the fuse have a maximum current of 500 mA and there needs to be some room for loads on the base module. This 400 mA limit is not actively regulated and just a convention between developers and users. Should users draw more than 500 mA, the regulator will turn off or the fuse will blow. The other power line on the extension connector is the USB voltage that is only available when the base module is plugged into USB. The connection also works the other way around, so if USB is not connected, the module could in theory charge via the extension port. The USB voltage of 5.0 V is required for some extensions for example the servo extension. The servos need 5 V as a power source to run stable. This way the extension can only be used if a USB connection is present. For mobile scenarios involving such extension modules, we recommend using a USB power bank to power the application.

4.2.5 Physical Structure: The Printed Circuit Board

The PCB holds all components and forms the electrical connection between them. We decided to use only double layer PCBs for two reasons. First, one of our primary goals is to make technology graspable and understandable. This includes not only the use of our finished product but also its design, meaning that people can learn from our design and use it as a base or reference for their project. In this terms, double layer PCB designs are less complex and easier to understand than 4+ layer designs. We also use the Computer Aided Design (CAD) software Eagle by CadSoft[1] to lay out our PCB. CadSoft offers a free version that is restricted to only 2 layers. A four layer design could not be viewed or modified in the free version, rendering the design files useless for people without access to an expensive, full version of the software. Secondly, BRIX$_2$ is supposed to be reproducible by non-professionals for a reasonable price. Manufacturing double layer PCBs in contrast to for example four layer boards is generally cheaper and offered by virtually any PCB manufacturer. The size of the BRIX$_2$ PCB is defined by the design target of a 4×6 Lego form factor which restricts it to 44.6×28.7 mm. Due to the limited space inside the BRIX$_2$ base module, we were forced to reduce the thickness of the PCB from the standard of 1.6 mm to the slightly more expensive 1 mm.

Board Layout and Integration into an Enclosure

In this section, we present our arrangement of components which is partially coupled with the enclosure design. We first regard the components that are fixed by design, then discuss different placements of constrained components before introducing routing and arrangement issues. Because of the coupling of PCB and enclosure design process, we already touch on some aspects of the enclosure before we focus on the enclosure in detail in the next section.

[1]http://www.cadsoftusa.com/

Fixed Components In Section 4.1.2, we already mentioned some components whose position on the circuit board is fixed by design, namely the motion sensor and the extension headers. We decided to have three extension ports, like in the former BRIX design. The spacing between those headers is determined by the form factor of both the base module and the extension modules and is exactly the width of a standard, 4×2 stud Lego brick. The headers obviously need be on the top side of the board so the extension modules can be stacked upon the base module. We decided to put the Hirose fine pitch extension headers as close to the edge of the PCB as possible to maximize the available space for other components on the base module as well as on the future extension module PCBs. The motion sensor is supposed to be in the center of the PCB in order to keep the rotational axes close to the center axes of the base module itself. Because the extension connectors already cover almost half the PCB, we have to shift the motion sensor slightly away from the center, see Figure 4.7. The Hirose headers have a base which is 2 mm high and is wider than the rest of the header. If we want the header to stick out of the top plate with only the smaller top part, the top plate of the case are going to be flush with that base. This means all other components on the top side must not be higher than 2 mm or otherwise they collide with the top plate.

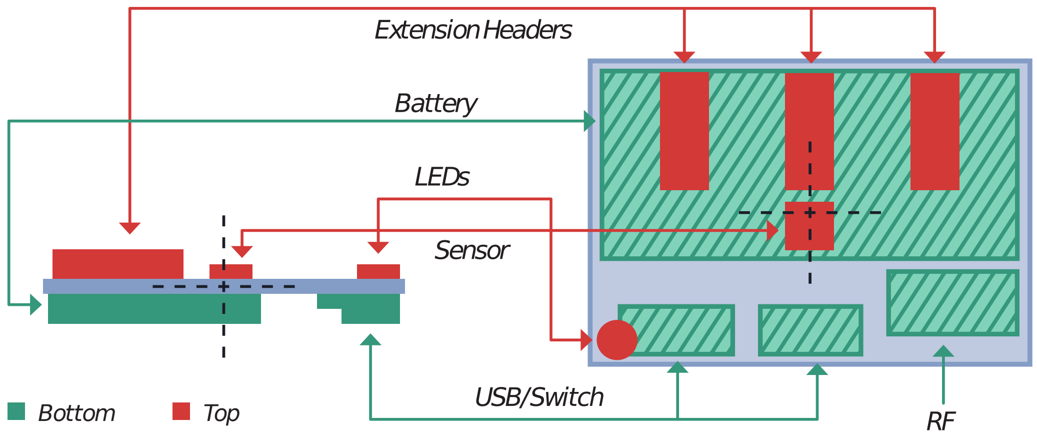


Figure 4.7: Component placement on the BRIX$_2$ PCB.

Constrained Components Since routing all traces of the extension headers takes up all space on both layers, the bottom side of the PCB contains no components in the header area. This is ideal to place the battery, which is a high profile component and therefore needs to be placed on the bottom side, where it takes up almost 70 % of the PCB surface. Obviously we need to place all components higher than 2 mm on the bottom side as well, namely the RF module, the MicroUSB header, the power switch and possibly large passive components like capacitors. Since the antenna end of the RF module needs to be at the PCB edge by design rules, we can place it in one corner, see Figure 4.7. We decided to place the switch and the USB port within the Lego grid again, so it matches up with the Lego bricks the case is made of.

4.3 Implementation of the BRIX$_2$ Enclosures

Since we base the enclosure design on the previous BRIX system, early BRIX$_2$ enclosures look similar to it. During several design stages, we developed them towards a more reproducible concept involving 3D printing technology. In this section we briefly touch on two concepts before we present different iterations of enclosure designs as well as the most recent design.

4.3.1 Appearance and Manufacturability

Our enclosure concepts are based on the one hand on Lego bricks and on the other hand on decentralized manufacturing techniques like 3D printing.

Lego as a Material and a Message:
As we already mentioned in Section 4.1.1, the Lego aspect of the BRIX and BRIX$_2$ modules is not only chosen for practical reasons like durability, price and mechanical connection but also to convey a certain message to the user. Lego bricks are a symbol for creativity and modularity, a message that we would also transport with the appearance of BRIX$_2$.

Rapid Prototyping and Manufacturing:
By making our platform open source regarding software and hardware, we encourage others to produce their own BRIX$_2$. As manufacturing techniques such as CNC machining, laser cutting and 3D printing become increasingly available through fablabs[2] and online services like Shapeways[3], we can combine them into a decentralized assembly line. Our long term goal is to design a product that anybody can have manufactured through publicly available services. A good example for this concept is the Thingiverse website we introduce in Paragraph 4.3.3.

4.3.2 Case Design 1: Lego Only

Our first approach was basically an adaption of the BRIX case design with cutouts in different places and a translucent Lego brick to keep the LED visible. We constructed the enclosure from two standard, 2×4 stud bricks, a 2×3 and a translucent 1×2 brick. We first glued all bricks together using plastic glue. In the next step, we used a CNC machine to cut out holes in the front for the switch and USB connector as well as in the top for the extension connectors. Finally we hollowed out the entire structure leaving only the outer walls of the Lego bricks. This step is done last because it significantly destabilizes the structure, making it difficult to clamp the workpiece

[2]https://www.fablabs.io/
[3]http://www.shapeways.com/

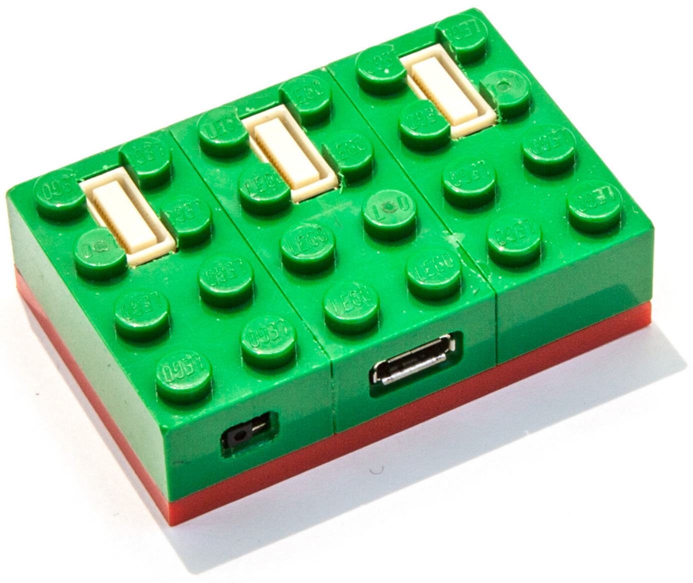


Figure 4.8: Case design I for the $BRIX_2$ base module.

inside the machine without breaking it. After inserting the electronics, we sealed the bottom with a regular, 6×4 Lego plate which we glued down.

CNC Machining Lego Bricks

CNC machines are computer controlled milling machines that allow to precisely cut workpieces in an subtractive process. A spindle, equipped with a rotary cutter, can be moved linear by motors along several perpendicular axes, typically X, Y and Z and sometimes rotated around additional axes. This allows the computer to place the tool anywhere in the workspace and remove material from the workpiece. The process is controlled by a special programming code, the G-code, which is interpreted by a software running on the computer that drives the machine. G-code is basically a sequence of configuration and motion commands which can be either simply in a text editor for less complex operations or generated by special Computer Aided Manufacturing (CAM) software. For all our CNC operations, we used a 3-axis CNC, different end mills and a standard vise for clamping. The software we run is LinuxCNC on a real-time Ubuntu Linux computer which interprets our hand-written G-code and controls the machine.
Although the hardware setup is common for well equipped hackerspaces and fablabs, the process requires experience and skill in order to get the clamping procedure and the tool setup right. However, once the machine is set up, it can process multiple

$BRIX_2$ enclosures in a single run without additional effort. Still we would not consider this manufacturing technique feasible for large scale production or especially reproduction by other people. It involves a lot of work steps and numerous things can and will go wrong in the first try. Also for us, it was a process of trial and error to an extent like for example finding out that different colored Lego bricks have different material properties which require distinct machining parameters for spindle speed and feed rate.

Conclusion of Case Design I

The enclosure made only from Lego bricks perfectly blends into the Lego world and easily carries its metaphors, see Figure 4.8. The mechanical connection to extension modules is stable and reliable. Using Lego bricks we can build an inexpensive and light case to protect the $BRIX_2$ electronics, however the manufacturing process involves a lot of steps that have to be carried out with great precision and require expensive machinery and experience to use it. This does not comply with our requirement for manufacturability and reproducibility.

4.3.3 Case Design II: Lego plus 3D printing

In earlier enclosure designs, one reason to use Lego bricks was the excellent friction based mechanical connections between those bricks which we utilized to hold extension modules in place on top of the base module. This means only the top part of the case actually has to match this demand for low tolerance and precise fit. The rest of the case can be made from a different material. Since 3D printers are widely available today and much easier to use than a classical CNC machine, we designed an enclosure which could be mostly 3D printed and only consists of a single Lego piece that requires some traditional CNC machining, but much less than the previous version. Before we describe the actual implementation, we briefly introduce the concept of 3D printing.

The Current State of 3D Printing

There are different techniques to generate a workpiece in an additive process which is referred to as 3D printing. The approach we focus on involves the extrusion of a thin stream of polymer material, usually nylon or ABS, through a heated nozzle onto a build surface to create the workpiece layer by layer. The extruder head can be moved in perpendicular 3 axes, similar to a CNC machine. The layer based technique allows to print parts that are not possible to produce with subtractive processes like CNC machining, for example a hollow sphere. When we made the first BRIX version in 2009, 3D printing was already available to us, but the process was not sufficiently precise to replicate a reliable and durable friction-based connection

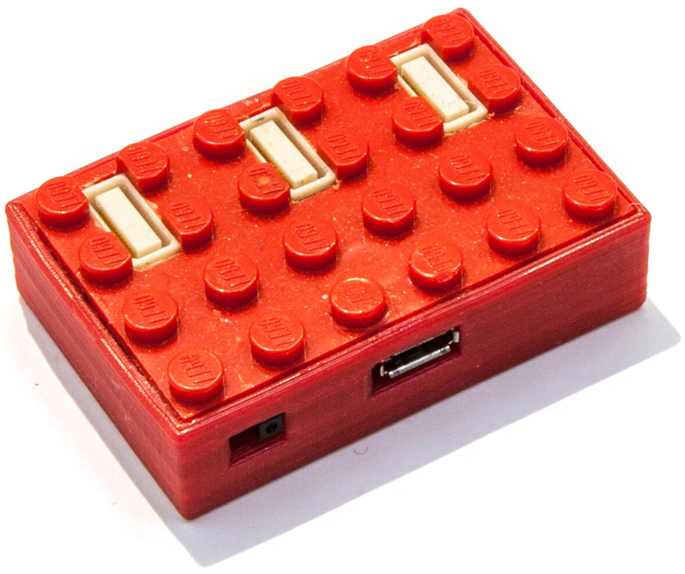


Figure 4.9: Case design II for the BRIX$_2$ base module.

like Lego implements. Since we also had a CNC machine available and only built a small quantity of devices as a prototype, we had decided to use the more complicated process we described earlier. Now 6 years later, 3D printing has become more precise, inexpensive and far more common, even in the consumer market. The popularity of this technique among hobbyists and makers has massively increased and meanwhile, online services and stores provide 3D printed parts made from a variety material, ranging from different plastics to metals and ceramics. Users either supply their own 3D models or choose one from a database of objects like Thingiverse[4]. This way people do not even need to have personal access to a 3D printer, but can just order the parts they want. For our prints, we used a Makerbot Replicator 2X[5] with different filament materials. The dual print head of the machine allows to print two different filaments in one process. Not only two colors, but also two different materials can be combined in a single print. The Replicator 2X is a machine from the higher consumer price segment and costs around 3000 USD.

Combining Lego and 3D Printing

For our next enclosure design iteration, we decided to keep the original Lego only for the most crucial part and produce the rest of the case with a 3D printer. To connect

[4]https://www.thingiverse.com/
[5]https://store.makerbot.com/replicator2.html

to the extension modules, only the top Lego studs are required. We used a 6×4 stud Lego plate and cut out the holes for the extension connectors as well as cleared the bottom with a conventional CNC machine. The part can be fully processed using only a single clamping operation and two cutting operation, with a fine tool for the holes and a bigger tool for the surface clearing. The rest of the enclosure is done in a single printing operation. We chose a flexible TPE material called NinjaFlex[6] in order to better protect the electronics inside from mechanical shocks, caused by for example dropping the module on the floor. On the top, the material wraps around the Lego plate, which is glued on after the electronics are inserted, see Figure 4.9. The flexible material also allows for lower precision during the gluing process and makes inserting the electronics much easier. After the first version of this combined enclosure, we also started to use translucent NinjaFlex material and added cutout for the LED into the top plate. This way the LED is visible through the 3D printed material on the sides and the top, see Figure 4.10. In the same design change, we added a well around the switch cutout, so the switch can be accessed even with a fingertip instead of a fingernail as in the early version.

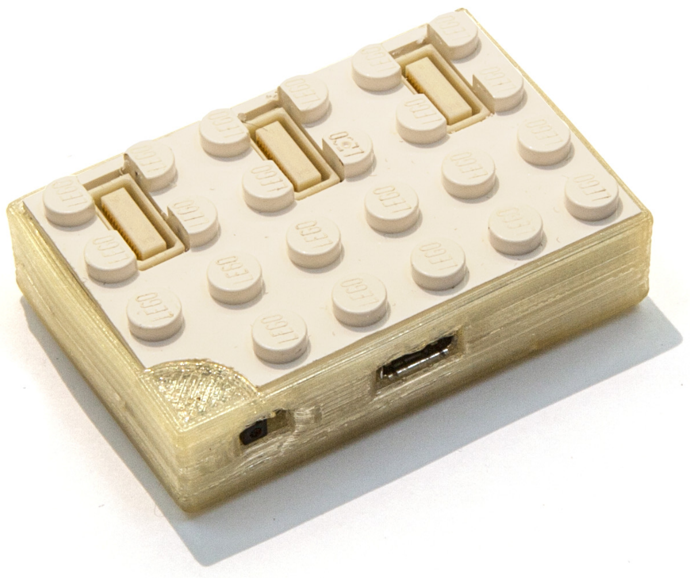


Figure 4.10: Case design II for the BRIX$_2$ base module with improvements.

[6]http://www.ninjaflex3d.com/products/ninjaflex-filaments/

Conclusion of Case Design II

Compared to the Lego-only version of the enclosure, the new approach has distinct advantages. First, it is much easier and faster to build. It involves no high precision gluing process, no multiple clamping operations in the CNC machine and consists of only two parts instead of five. The printing works extremely well and takes around 10 minutes per piece. Second, we can add support structures on the inside of the enclosure which fit the electronics precisely and fill hollow spaces to increase the overall stability of the module. The flexible material allows for higher tolerances in the manufacturing process, is shock-proof and does not feel as edged as the Lego only enclosure. This might be interesting especially for applications in wearable electronics or electronic textile scenarios, where the modules come into close contact with the wearer's skin. The Lego plate on top still provides an excellent fit for the extension module.
A severe problem we encountered after using the new case in several applications is the loss of the Lego bottom plate. We realized that the plate did not only seal the lower side of the module, but also offers superb mounting options for wearable as well as static scenarios. With the old design, users could simply glue a Lego brick to an object and fix a BRIX$_2$ module on top. The connection is solid enough even for wearable electronics scenarios and more importantly non-permanent. Another problem is that a CNC machine is still involved into the process. If we could produce BRIX$_2$ enclosures without this complicated and expensive tool, we would be much closer to our proposed, easy to reproduce enclosure. To solve these issues we created a third prototype which we describe in the following.

4.3.4 Case Design III: 3D printing onto a Lego Plate

In order to optimize the enclosure for manufacturing with a 3D printer, we conducted further experiments with 3D printed Lego studs in order to create a suitable friction based connection between the extension modules, which are still encased in a standard Lego brick and the base module. We found that given the precision of the printer and the low mass of the extension modules, a connection between a Lego part and printed Lego studs is robust enough when the printing parameters are selected carefully, so we can 3D print the top plate instead of cutting it from a Lego plate. That way the process is sped up and more importantly the CNC machine is no longer required. As mentioned earlier, a drawback of the former case made of a 3D printed part and a Lego part was the lack of a bottom Lego plate. If we also printed this, the mechanical connection would not be stable enough to hold the mass of the base module in scenarios where a BRIX$_2$ module is for example attached to a wearers wrist during a recording of body motions, see also Section 6.3.2. High accelerations would simply tear the module off the fixture. This is why we decided to again include an original Lego plate into the design, this time at the bottom. A

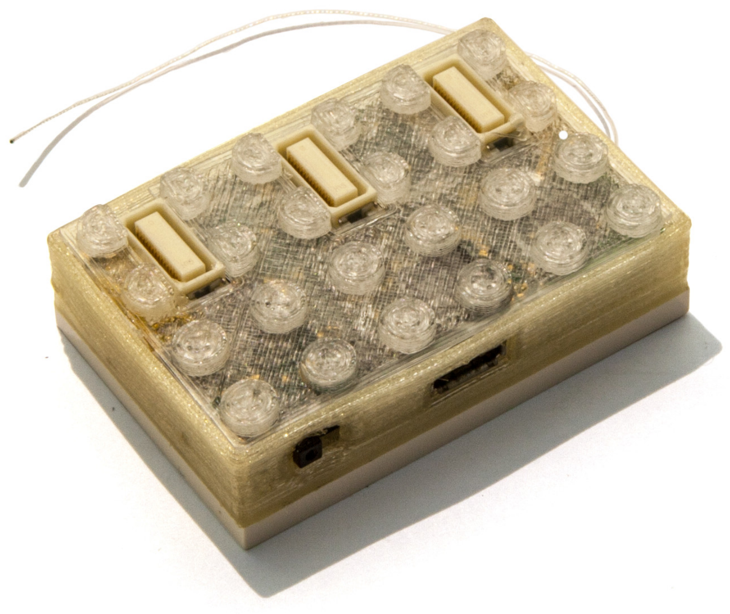


Figure 4.11: Case design III for the BRIX$_2$ base module.

major improvement this time is that we only need remove the studs on top of the plate, which can be done with several tools, for example a sharp knife, a file or, as we did it using an end mill and a drill press. Instead of 3D printing parts of the enclosure and glue it to the Lego part, we decided for a more elaborate strategy. We printed the NinjaFlex material directly onto the Lego plate whose studs we removed earlier. This way we can still have support structures for the electronics and the bonding between both components is even stronger than in the glued versions.
A challenge in this process is that the position and rotation of the Lego plate needs to be known before the printing process in order to place the part on the right spot in the building volume. To solve this problem, we define the position of the Lego plate by simply printing a plate with Lego studs inside the printer first. This structure also has a big bonding structure, called "raft" that keeps it from separating from the build platform of the printer. The printed plate structure remains inside the machine and serves as a mounting point for the real Lego plate we intend to print on, see Figure 4.12. With this method, we can on the one hand fix the Lego plate on the build platform and on the other hand know its exact position. Now that the Lego plate is mounted inside the printer, we can print the rest of the case on top of it. Again we use NinjaFlex material and a similar geometric shape for the walls as in the previous version. After the printing process, we can simply remove the finished enclosure and put another clean Lego base plate onto our mounting plate

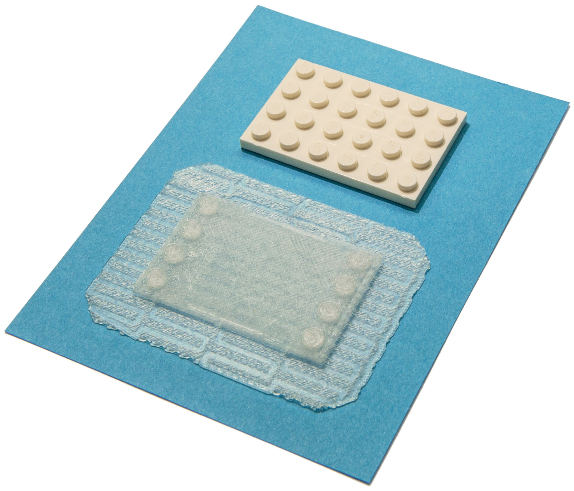


Figure 4.12: 3D printing raft with Lego studs next to a Lego base plate (before removing the studs).

to print more enclosures. The top plate is printed with translucent ABS material which allows the RGB LED to shine through the material, see Figure 4.11.

Conclusion of Case Design III

Since no CNC is needed, all processes are easy to handle with just basic tools and a 3D printer. This way we drastically reduced the tool and experience requirements for creating a $BRIX_2$ enclosure. We have replaced the top Lego plate with a 3D printed copy, which leads to a slightly less stable mechanical connection, however regarding the low mass of the extension modules, they still connect tight enough. Only in the long run, we will be able to find out if the 3D printed material will wear out, decreasing the quality of the connection, but so far we did not encounter any problems. What we noticed is that on some modules, the connection works better than on others, which is explained by inaccuracies of the printing process. If the parts are printed by a professional service on a more precise machine, the results will probably be significantly better. By using the original Lego plate as a base for 3D printing, we still maintain the excellent mounting options as well as the metaphor that Lego bricks provide. Through translucent, flexible material for the sides, the electronics are well protected from shocks and impacts and the LED can shine through the enclosure. In addition to that, users can take a glimpse at the electronics on the inside. This weakens the “black box” effect which causes users to completely forget about the inner workings of a device and allows them to see the actual components included in $BRIX_2$.

4.3.5 Outlook on Future Case Designs

Although the current state of the enclosure works well and can be produced relatively easy in smaller scales, we would not consider it to be the optimal solution. As a first step towards a new iteration, we can further optimize the process for using 3D printers and other nowadays widely available tools like laser cutters to allow a greater group of people to make their own BRIX_2. Optimizations might also include reducing the printing process to a single print. Once the Lego base plate is in place and the sides of the enclosure are printed as in case design III, we could insert the electronics while the print process is paused and then print the top plate including the studs, all in one print. This would also eliminate the procedure of gluing all parts together. A support structure on the inside to prevent the top plate from sinking in during printing could be implemented by inserting a thin laser-cut film of translucent plastic material before continuing the print process. In the long run, we can expect 3D printers to become even cheaper and more available and at the same time also faster and more precise. This offers great possibilities for decentralized manufacturing.
Secondly, we can optimize the enclosure for mass production in large volumes. In this case, one would typically use techniques like injection molding to fabricate the whole workpiece in two parts. We would also get rid of the Lego plate on the bottom because the precision of the injection molding process is sufficient for creating a well working friction based connection. After all, Lego bricks are fabricated with the same process. A disadvantage are the significant initial costs for injection molding, because a mold has to be designed and built, which involves experience, skill and complex machinery. This approach would be the opposite to the 3D printing approach because it is centralized and can only be performed with very expensive tools and facilities.

4.4 Implementation of the Software Components

Similar to the Arduino, BRIX$_2$ is not only a hardware but also a number of software components, some developed and maintained by us, some already implemented for other projects. In this section we present the *LiBRIX*$_2$, a firmware library for BRIX$_2$, discuss how it helps users to get started with our platform and how it facilitates a more efficient prototyping flow.

4.4.1 An Arduino Library for BRIX$_2$

In the following, we briefly introduce the concept of Arduino libraries, point out why they are important and explain how they are integrated into the Arduino IDE, before we present details on the implementation of the different components of the LiBRIX$_2$.

Arduino Libraries

When users purchase an Arduino board, they are supposed to download and install the Arduino IDE in order to write firmware for the device and upload it to the flash memory of the controller. The IDE not only contains an editor for programming and the necessary chain of tools to compile and upload firmware, it also comes with a variety of libraries that are basically software wrappers for different functions of the microcontroller and external components. These include libraries to handle the controller's SPI and I2C interfaces or to control components like servo- and stepper motors, displays, WLAN and GSM shields.

Using Libraries By including the according libraries into their code and using the functions they implement, users can add external hardware components to their projects without much effort. The drawback of this is that the concept hides a lot of the internal workings of the controller and external components. One might argue that this abstraction can not lead to a deep understanding of the principles and details of said components and functions. However, this is on the one hand not important when getting started, in fact if it was required to deeply understand every aspect of the system before successfully writing their first program, users would be quickly demotivated and lose interest. On the other hand, intermediate and advanced users can look at the documentation and source code of those libraries and even modify them or write their own. All libraries for the Arduino are open source and mostly written in C++. When users or companies create new products for the Arduino like shields or breakout boards for sensors, they usually also supply a library (see for example [7] or [8]) that can be added to the Arduino IDE in order to

[7]https://github.com/sparkfun
[8]https://github.com/adafruit

immediately start using those products without previous knowledge.

Example Sketches An Arduino library does not only contain the implementation of functions and wrappers that can be used to interface a functionality or external component, but also comes with example code. These sketches are accessible through the *"Files > Examples"* menu. Every standard library and most of the external libraries offer at least one, usually multiple example sketches that demonstrate different aspects of the library and the component it refers to. This way, users never have to start from scratch and always have a reliable base to develop from. Frequently, a project can be successfully implemented by combining the examples of all components that are used and only writing a minimum of custom code. This way of approaching a task is especially useful in prototyping scenarios with an iterative design process where it enables quick results and thereby a high number of iterations in a given time. But also in learning and teaching scenarios, examples are valuable, because students can learn single-handedly from the example code by using and modifying it. A sequence of exercises might start with simply combining different examples into a single application and then moving the tasks further away from the scope that the provided example code covers, thereby increasing the amount of code that the students have to implement on their own. This is again rewarding in the beginning and later on, students always have a functioning base of code that they can fall back to in case they get stuck with their own application.

Making Arduino Libraries Arduino libraries are archive files (.zip) that contain the implementation (.cpp, .h), an optional *"keywords.txt"* that defines keywords of the implementation such as function names that are supposed to be highlighted in the Arduino IDE and the *"examples"* folder, containing one or multiple example sketches (.ino). The Arduino IDE offers a function that installs a library .zip file by moving all files to the correct locations. After this process, the library can be included into custom code and all of its examples are available in the IDE. The implementation can contain multiple files which have to be written in C++. Usually a library is just a class or object for the device, for example for a light sensor. The class then implements functions for reading from the sensor, configuring it, etc. In the user's code, this the class is then instantiated, like for example:

```
#include <LightSensor.h>
LightSensor mySensor;
mySensor.initialize();
int brightness = mySensor.getData();
```

4.4.2 LiBRIX$_2$

To provide beginners with a low threshold entry into the word of microcontrollers and to offer developers a flawless prototyping tool, we implemented LiBRIX$_2$, an Arduino compatible library for all components of the BRIX$_2$ system such as the wireless interface, motion sensor, RGB LED and extension modules. When we implemented LiBRIX$_2$, we considered two key aspects:

Modular Code: There are many functions of BRIX$_2$ that require wrappers, but not all of them are necessarily used in a particular application. This applies especially to the extension modules. If we included all this into a single implementation file, there would be a lot of overhead in the code and in the resulting binary. Since space on the flash memory of our microcontroller is a sparse resource, this is not an option. Instead we decided to split up the library into the single aspects and only keep the core functions in a common file, so the structure of the library matches the structure of the BRIX$_2$ hardware. Installing the library is still simple, because everything is contained in a single .zip file. However, users have to include the correct files for their application. For example if an application requires the motion sensor, the RGB LED and a proximity sensor extension module, the sketch would include the following:

```
#include <BRIX2.h>
#include <ProximityExtension.h>
#include <InertialSensor.h>
// Generate Sensor Objects and BRIX2 Object
ProximityExtension myProx;
InertialSensor myIMU;
BRIX2 myBrick;
```

Another advantage of multiple sub-libraries instead of a single, general library is that by looking at the include files, it is directly obvious which hardware configuration of extension modules is required for this sketch.

Existing Libraries: For some of these components, for example the motion sensor and some sensors on extension modules, libraries were already implemented and maintained by either the Arduino project or external developers under open source licenses. There are two ways in which we can include those external libraries into our project. On the one hand, users could download and install the external files themselves. This will eventually lead to problems in case the libraries are updated and therefore different from the configuration that we tested and verified. Second, we download those libraries, test them with our own libraries, integrate them into the LiBRIX$_2$ and thereby redistribute the external libraries. Most licenses allow redistribution of the software, which enables us to simply integrate them into our own set of libraries instead of implementing our own. This reduces the maintenance effort and we can rely on tested and user-proven code.

Organization

As we already mentioned, we have separated several functions from the main library in order to save flash memory space when some features are not required in the application. Obviously, all extension modules have their own library object because they differ significantly in their scope of functions and are in general treated as optional components. This leaves us with only the functions on the base module. Here we separated the inertial sensor, because this part of the library is based on the extensive and large I2Cdevlib. If the sensor is not required in an application, this separation saves a significant amount of space.

BRIX$_2$ Core Library

The core library offers functions that control the wireless interface, the LEDs and various other components. Besides the two functions to set the color of the RGB LED, which is directly connected to the user controller, all other functions are just requests to the system controller which are sent via UART. The firmware on the system controller then interprets these commands and reacts accordingly by either adjusting the requested setting or providing the requested response. In the following, we briefly list all functions of the BRIX$_2$ core library.

- **void Initialize()** Is to be called in the setup of the sketch and configures the output pins for the RGB LED.
- **void setRGB(unsigned int R, unsigned int G, unsigned int B)** Sets the red, green and blue component of the RGB LED
- **void setHSV(unsigned int hue, unsigned int sat, unsigned int val)** Sets the hue, saturation and value components of the RGB LED
- **void statusLEDOn/Off()** Two functions to request the system controller to turn the status LED off.
- **void wirelessOn/Off()** Two functions to activate or deactivate the wireless transceiver in order to save power.
- **void RXTXOff()** Disables the UART of the system controller in case it is required for a different purpose.

- **void resetSystemController()** Is required to re-establish the communication with the system controller in case the UART was disabled.
- **float getVoltage()** Returns the battery cell voltage in Volts.
- **int getAddress()** Returns the module's individual wireless address
- **int checkNode(byte address, int timeOut)** Checks if a remote node is available in the wireless network. If so, it returns the Received Signal Strength Indicator (RSSI), else -1.
- **void sendAscii(byte address, char* text, byte length)** Sends arrays of characters to a remote node via the wireless transmitter.
- **void sendInt(byte address, int* numbers, byte length)** Same as sendAscii for integers.
- **void sendFloat(byte address, float* numbers, byte length)** Same as sendAscii for floats.

Inertial Motion Sensor Library

The inertial sensor library implements functions to read sensor data from the device. Functions that read raw data also offer a scaling option that maps the raw values to a certain range. Configuration functions allow users to calibrate the sensor and select measurement ranges. In order to read DMP data from the device, we recommend the MPU6050 component of the I2Cdevlib [9] or the implementation by Kris Winer [10] which also incorporates the magnetometer.

Example Sketches

The example sketches for the base module are separated into four different subcategories.

- **Basics:** These examples are meant to be the starting point for novel users. They demonstrate how to control the RGB LED, how to communicate with a host computer using the serial port and how the functions of the system controller can be queried.
- **HID:** The HID examples demonstrate how to use BRIX$_2$ directly as a USB mouse or keyboard. This is especially interesting for applications in human/-computer interfaces.

[9]https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
[10]https://github.com/kriswiner/MPU-9150

- **Sensor:** Sensor examples show users how to read sensor data from the internal motion sensor and how to stream that data to a host computer.
- **Wireless:** The wireless examples implement basic communication between multiple nodes. Sending and receiving of text and numeric values are demonstrated. Usually, these sketches require at least two BRIX$_2$ base modules to try them out. The wireless example sketches can be configured to compile for either sender or receiver.

Automated Packaging and Testing of LiBRIX$_2$

In order to ensure a stable and working LiBRIX$_2$ at every release, we developed a set of tools for packaging and testing the library. As we already mentioned, this is especially relevant in order to verify that the external libraries work flawlessly together with our own. In the following paragraph, we briefly describe the process of compiling and releasing a new version of the LiBRIX$_2$ in case some of the files are updated.

Packaging: We use a shell script that downloads the latest stable versions of LiBRIX$_2$ and all external libraries we utilize from their repositories. The files are then copied into a single folder and selected files are automatically patched in order to avoid conflicts between certain libraries. The result is an archived file (.zip) that is uploaded to our webserver after the testing procedure. Users then only need to download and install a single, merged library to start working with BRIX$_2$ and all extension modules.

Testing: With a second shell script, we can verify that every example sketch we provide with the LiBRIX$_2$ actually compiles for our platform. As a prerequisite, the LiBRIX$_2$ that requires testing is installed inside a clean Arduino IDE setup. The script then compiles every single example file of the library and reports back a logfile with the compilation results. Only if all examples compile without problems, the LiBRIX$_2$ can be deployed on our website. We automated testing with continuous integration in mind. By using a Continuous Integrated Testing (CIT) server such as provided by Center of Excellence in Cognitive Interaction Technology (CITEC) [11], we could constantly test our development code base and deploy working parts of it to our website automatically.

[11] https://toolkit.cit-ec.uni-bielefeld.de/

4.5 Documentation

In this section, we explain how the informations and documentation on the BRIX$_2$ system is structured and to which level of detail every source of information covers our platform. The BRIX$_2$ documentation is arranged hierarchically, see Figure 4.13.

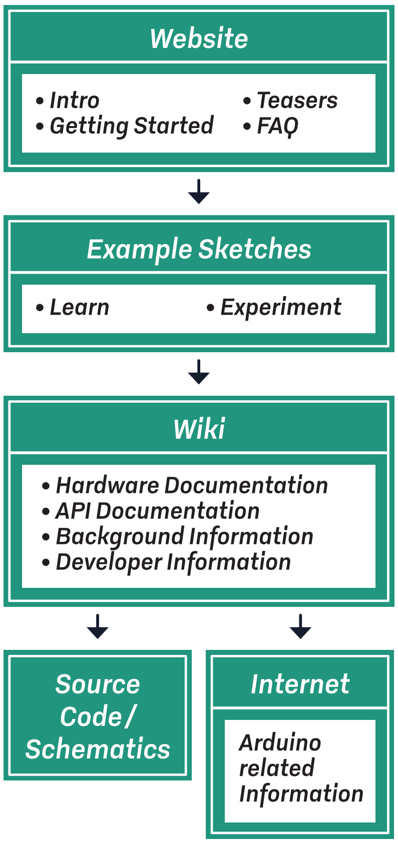


Figure 4.13: BRIX$_2$ documentation hierarchy.

The first source of information is the website[12]. Here, users are presented basic technical aspects but also the philosophy and motivation of the BRIX$_2$ project. After that, the website guides through the installation process of the IDE, LiBRIX$_2$, basic settings of the IDE and trying out the first example. Should users encounter problems that are not already covered by the Frequently Asked Questions (FAQ) on the website, they can find more detailed information in the BRIX$_2$ wiki[13]. The wiki can be regarded as a manual and a technical description for our platform in which we try to cover every aspect of the system, including all extension modules. Since our

[12]http://tiny.cc/brix2
[13]https://opensource.cit-ec.de/projects/brix2/wiki

platform is based on the Arduino, a lot of questions regarding programming are not BRIX_2 specific, but also apply to other Arduino boards. In these cases, the official Arduino programming reference, the user forum, numerous tutorials on the Internet and several books also support our users. The lowest level of our documentation structure is the source code of the LiBRIX_2, the schematics and board layout of our electronics as well as the technical drawings of our enclosures. Expert users and developers can look into these files to solve problems and to gain further knowledge about our system.

4.5.1 Website

As we already mentioned, the BRIX_2 website is the first support level and made to catch the attention of potential users. It contains basic informations about the system and guides users up the point where they are able to upload an example sketch to their base module.

Introduction and Features: In the introduction, we motivate why to use BRIX_2 by stating that it helps understanding some of the technology that surrounds us and that we use every day. The focus of the introduction on the website is the learning/teaching aspect of our system. Subsequently we summarize the key features of BRIX_2 in a list of photos with captions to provide a quick overview of the capabilities of our platform.

Getting Started and FAQ: The user is guided through the first steps of getting started with BRIX_2, from the download of the Arduino IDE, installation of the LiBRIX_2, configuration to uploading sketches. After this short introduction, we present some FAQ that are supposed to help users with the most common problems they can encounter in the beginning while trying out the examples and implementing their first applications.

4.5.2 Wiki and Repository

The $BRIX_2$ wiki is part of the Redmine environment that also contains the repository and issue tracking for our projects. The landing page leads to two main subcategories: The $BRIX_2$ user guide and the developers corner. Similar the website, the wiki is constantly extended and updated by developers.

$BRIX_2$ User Guide

The user guide contains in depth information about our platform and is roughly separated into three categories:

$BRIX_2$ Base Module: This page provides a technical overview about the base module, followed by some remarks about handling. The list of electrical specifications and limitations of the system helps users to efficiently use the base module during their experiments. The page also lists all signals of the extension port.

$BRIX_2$ Extension Modules: On the landing page of this category, users find a table that shows which extension modules use which pins on the extension port. This information is crucial to discover possible interferences when combining certain extension modules into a single application. Each extension module has its own wiki page that is linked on the landing page. These pages contain technical details, example codes and backgrounds on the working principles of the given extension module such as for example what physical property a certain sensor measures.

$LiBRIX_2$: This page provides an overview over the functions of the $LiBRIX_2$, including the libraries for the extension modules. Examples and usage tips for many functions are given if applicable.

Developers Corner

This category contains several tutorials and in depth information for developers. Here they learn for example how to set up a $BRIX_2$ module that has just been assembled, how to build enclosures or how to use the developer tools. The source files of our project are also accessible from this site.

4.5.3 Arduino Community

In Section 2.1.7, we already briefly covered the Arduino Community. There are several big user forums, FAQs and tutorials about virtually any topic that is related to the Arduino platform. Should $BRIX_2$ users come across a problem that is not inherent of our system but also applies to Arduinos, they will most likely find several sources of information through an online search engine. Especially forums usually have extensive categories that are dedicated for problem solving and have short reaction times due to the huge number of active participants.

4.5.4 Source Code

As an open source project, all parts of our design are public, including software, firmware, schematics, layouts and CAD design files. These documents help developers and advanced users to fully understand BRIX_2 and to actively take part in a further development process, but also to use parts of our system as the base for novel projects. Our sources are available in a git repository on the CTEC open source server[14].

[14]http://openresearch.cit-ec.de/git/brix2.git

4.6 Conclusion

In this chapter we have presented a final design concept for the BRIX$_2$ system before we discussed the implementation of hardware, software and the documentation in detail. We have successfully integrated all functionalities we identified as requirements Section 3.1 into the BRIX$_2$ base module. As a result we created a mobile microcontroller platform that is powered by a rechargeable battery, reprogrammable, extensible and equipped with an IMU as well as a wireless transceiver. Multiple iterations of different enclosure designs led to a robust enclosure for the base module that is easy to produce. Since we designed our platform to be Arduino compatible, our users only require the Arduino IDE, which is available for free and under an open source license. To allow an easy handling of all the features BRIX$_2$ offers, we developed the LiBRIX$_2$, a custom Arduino library for our platform. Not only does LiBRIX$_2$ implement software components for accessing different functionalities of the base module and the extension modules, it also includes example sketches that demonstrate how to use different aspects of our library. Our documentation is hierarchically structured and a website as well as a wiki provide information on our platform ranging from a brief overview to a detailed description of the inner workings of BRIX$_2$.
Regarding the hardware of our system, this chapter has only covered the base module. We present the concept of extension modules as well as their implementation in the next chapter.

5 BRIX$_2$ Extension Modules: Towards an Adaptable and Open-Ended Platform.

As an addition to the BRIX$_2$ base module and the corresponding software components, extension modules are a key aspect of our platform. They allow to add functionalities and features to a BRIX$_2$ application rapidly and easily. This is especially valuable in prototyping scenarios where the hardware requirements are not yet fixed and can change from iteration to iteration. Our modular approach allows us to design compact and inexpensive individual components that can be assembled to a complex, tailored application.
In this chapter, we first discuss which types of extension modules we considered to develop in order to compile a total set of functionalities that prepare BRIX$_2$ for as many different applications and scenarios as possible. Our initial kit of extension modules includes 12 different types, but we would like emphasize that we explicitly encourage users and developers to add their own designs to our existing kit in order to make BRIX$_2$ even more versatile. Moving on to the implementation of our extension modules, we first describe how we build enclosures for the modules since this process applies to all of them. After that we present each module we build in detail, regarding electronics aspects as well as the corresponding software component in the LiBRIX$_2$.

5.1 Selecting Features for a Library of Physical BRIX$_2$ Extensions

In Chapter 3 we summarized all features that we found during our survey of over 30 different platforms and carefully selected the ones we later integrated into the BRIX$_2$ base module. All features we did not include were either to be implemented as extension modules for the BRIX$_2$ system or not relevant enough to include them for now. If we follow the list of optional features, see Section 3.1.1, there are three features to be implemented as extension modules: Bluetooth, temperature sensing and a flash memory. In addition, we have selected further functions to be part of the BRIX$_2$ kit, which we structure and describe in the following paragraph. Again

please note that the library of extension modules we present in this chapter is only a temporary snapshot and has most likely been extended at the time of publication, see also Section 7.2 or the BRIX$_2$ wiki [121].
In order to determine what features might be demanded by future applications, we split them into two main categories: Input and output, with a particular respect to ubiquitous computing and interactive scenarios. As a third category, we list features that do not fall in the two main categories. In the following, we characterize the categories with features. We derive the feature requirements from personal experience and observation of extension boards for existing products like littleBits, Grove or Arduino. As a result, we extract a feature list for our initial selection of BRIX$_2$ extension modules.

5.1.1 Input: Interfaces and Sensors

In a broad definition, every device that picks up and quantifies an external event or state is a sensor. If we arrange electrical and electronic devices that fall into this category from simple to complex, the list begins with the pushbutton, which is basically a pressure sensor with a resolution of a single bit. Buttons allow to input a binary state, however the time between two pushes can vary. Usually they are connected to a GPIO which is configured as an input. Pushbuttons can be useful for example to control the flow of an application or trigger certain events. More complex and flexible devices for that purpose are rotary dials and potentiometers, which can be used to adjust certain features of an application. Especially in early development phases, hardware and firmware are carefully adapted to the given application or scenario. A prototype whose parameters can be altered at runtime saves time and effort.
Touch-free interaction with an application is possible through distance sensors. They quantize the distance to an object in front of the sensor. Compact devices based on triangulation or quantization of infrared light are integrated in almost any smartphone and are thereby affordable and widely available.
Applications that react to their surrounding require sensors that measure environmental and ambient modalities like air temperature and humidity or light intensity. There is a variety of compact, affordable devices for this purpose to choose from. By offering users those sensors and input devices as extension modules, we cover a major portion of the sensing capabilities of the devices we surveyed in Chapter 2.

5.1.2 Output: Actuators and Feedback

Besides sensors, an application might require actuators to close the interaction loop. For this purpose we need to design modules that generate signals which can be sensed by humans through different sensory modalities. The most basic ones are visual, auditory and haptic perception. Simple visual feedback is already possible using the RGB LED of the base module. To convey more information visually, a

small display or array of LEDs could be used.
Auditory feedback can range from simple beeps to complex sounds such as speech or music. Fundamental sounds such as square wave signals can easily be generated on a microcontroller and output by a small piezo speaker. For other waveforms, music or speech samples, more complex and additional hardware might be required.
Haptic feedback is picked up by humans through their skin. It allows us to provide a sensation which is locally defined in contrast to for example audio, but does not require direct attention like for example visual feedback. This way we can directly guide the user's attention to a certain part of their body. Basic haptic feedback can be generated by vibration motors, which represent an inexpensive and compact alternative to dedicated, expensive haptic actuators.
A ubiquitous computing application such as a smart object or wearable device can also provide information to users or attract their attention by physically moving or changing shape. In order to move or actuate objects, electric motors can be used. In model making, especially servo motors are popular. A servo motor includes a geared motor and a position feedback. This way it can adjust itself to and maintain a certain position that is defined by an input signal, which can easily be generated by a microcontroller. To connect servos to the BRIX$_2$ system, we only need an adapter board that fits the electrical connectors that are common on almost any servo motor.

5.1.3 Wireless Communication, Storage and Interfaces

Although the BRIX$_2$ base module provides means for wireless communication, the RF standard we chose is not directly compatible with laptops or smartphones. Although the former can interface the 868 MHz RF network through USB and a BRIX$_2$ base module that acts as a bridge, the latter are not always capable to act as a USB host. A common standard for short-range wireless communication among mobile devices is BTLE. An extension module that makes BRIX$_2$ compatible to BTLE allows users to build prototypes that can be connected to smart phones and interact with existing and custom mobile apps. Another wireless standard, common in entertainment electronics for decades is infrared communication. An infrared transmitter extension can enable BRIX$_2$ users to easily interface TVs, radios or ACs for example in home automation scenarios. By also integrating an infrared receiver, we also allow to control a BRIX$_2$ application using any infrared remote control.
Data storage resources on the BRIX$_2$ base module are sparse. Both controllers only have several kilobytes of memory, which is not sufficient in scenarios where for example motion data is recorded with a high data rate. Some platforms we surveyed in Chapter 2, especially in the fields of wireless sensor nodes and IMUs feature a flash mass storage for such purposes. Micro SD cards are a prominent storage solution that provides gigabytes of memory in a compact device that is compatible to modern smartphones and laptops. Equipped with an extension that can hold an SD card, BRIX$_2$ is capable to store large amounts of data which can later be processed on a

different device.
Advanced developers will most likely reach the limits of functionality that the BRIX$_2$ base module and provided extension modules offer. Since we use fine pitch headers as connectors for our extension modules, they are not as accessible as for example standard pin headers. A small adapter board with accessible solder pads and optional pin headers makes it possible to connect custom electronics to a BRIX$_2$ base module.

5.2 Extension Module Concept

In the former BRIX system, see Section 2.6, we encased the extension modules in a single 2 x 4 stud Lego brick that we hollowed out. We closed the bottom with a matching Lego plate that had a cutout for the connector. We are going to keep concept for now, because it is easy to implement and has different advantages which we present in the following.

5.2.1 Mechanical Properties and Handling

We found that Lego based enclosures were robust, inexpensive and easy to produce. The mechanical connection is reliable, especially in combination with the electric connector. The Lego grid can also be used as a way to provide reverse polarity protection for the connector by putting it in a slightly asymmetric location. This way the electrical connection and the mechanical connection can only line up in the correct way. Using a single Lego brick as enclosure for an extension module restricts the size of the PCB and the components to roughly 13 x 28 x 8 mm. Should the implementation of an extension module require more PCB space, designers could build an oversize extension module that is twice or thrice the size, encased in two or three combined Lego bricks and more than one extension slot of the base module.

5.2.2 Appearance and Metaphor

Apart from the purely mechanical aspects, the enclosure made from Lego bricks carries the Lego metaphor perfectly. Extending the hardware capabilities of the BRIX$_2$ base module is as simple as adding another brick. The integration of the electrical connection into the mechanical connection supports this, because there are no additional actions required such as plugging in cables and the system appears to be less technical. Lego bricks are available in a wide variety of colors [1], including also translucent materials, which allows us to color code our extension modules, sometimes even matching to their function, for example to chose blue bricks to encase a Bluetooth extension module.

[1]http://Lego.wikia.com/wiki/Colour_Palette

5.2.3 Constrains and Limitations

As a consequence of the compact design of the BRIX$_2$ platform, the size of extension modules is restricted. Although we chose a compact and fine pitch electrical connector, it takes up almost 25 % of the PCB, which is half of the space on the bottom layer, see Figure 5.1. Developers can work around this by designing extension modules that occupy two or even three of the extension slots of the BRIX$_2$ base module. This of course blocks those slots and thereby restricts the use of additional extension modules in that particular application. The extension modules of the former BRIX

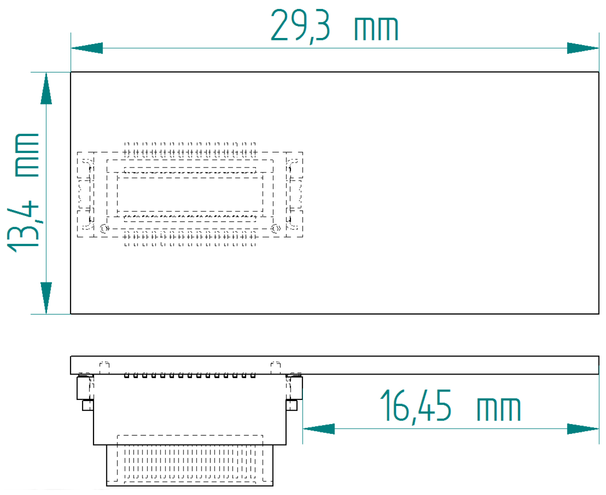


Figure 5.1: Basic extension module PCB layout: The extension connector takes up almost half of the bottom layer space.

platform provided a header on top, which allows to stack multiple extensions onto each other. This was possible because the extension headers had a much smaller footprints than the ones we used for the BRIX$_2$, see also Figure 4.2. This time, we decided against this concept for two reasons. First, the headers would cover almost 50 % of the total PCB area, which leaves only around 180 mm^2 of double layer PCB space for the actual electronics. Secondly the resulting stack would just be to bulky for most wearable applications.

5.2.4 Pin Collisions

Another restriction for the extension module electronics is the number of available pins on the extension connector. Although many of the components on our extension modules are connected to the I2C bus and can share this resource, some extension modules connect to multiple GPIOs or special purpose I/Os. When two different extension modules are used in a single application and try to use the same pin for different purposes, we call this a pin collision, which leads to corrupted signals or in the worst case to short circuits. Since three extensions can be attached to a BRIX$_2$ base module at a time, the chance of pin collisions between different extension modules grows with the number of extension module variants we develop. Pin collisions can hardly be avoided in designs with a full breakout extension connector, see Section 3.2.4 and also appear for example for Arduino shields [122]. In order to approach this problem, we considered two design guidelines:

Thoughtful Distribution of Pins: Some collisions can be avoided by thinking ahead and trying to imagine which types of extension modules might frequently be combined into a single application. If no pattern of combination is obvious, pins are to be distributed evenly so that no GPIO is used much more frequently than others throughout all extension modules. We composed a list of the pins used by all our extension modules in the documentation wiki [121].
Flexible Pinouts: If possible, extensions should be configurable to alternative pinouts. This is especially feasible if no special purpose I/Os such as PWM outputs are required. Reconfiguration can take place at the time of population, for example by placing a $0\,\Omega$ resistor to a different location. In some cases, this also allows to manufacture two versions of an extension module that can be used simultaneously on a single BRIX$_2$ base module using the same PCB layout.

5.2.5 A Template for Extension Module PCBs

BRIX$_2$ is extensible, so it can adapt to new requirements even after design time. The scope of functions can be expanded by simply implementing additional extension modules. As a starting point for developers who aim to create a new extension module, we provide a template file. We used CadSoft Eagle as a design tool throughout the whole project, so we only provide the template for this particular software. However, it can be imported in other design tools like Altium Designer as well. The template predefines the shape and size of the PCB, which fits into a Lego brick as well as the position of the extension connector on the bottom layer. We provide the template along with some design guidelines in the documentation wiki to encourage developers to create new BRIX$_2$ extension modules.

5.3 Extension Modules Implementation

After we have defined the scope of extension modules we were to design as well as the concept for enclosures in the previous sections, this section is dedicated to the implementation of the BRIX_2 extension modules. In the following we first describe the implementation of the extension module enclosures. Subsequently we present details on all extension modules we implemented from three different categories, see Section 5.1. Finally we briefly touch on some conceptual aspects of the LiBRIX_2 extension module support.

5.3.1 Implementation of the BRIX_2 Extension Module Enclosures

To protect the extension module electronics from mechanical stress and in order to mount features like rotary knobs, we enclosed the extension modules in custom cases made from Lego bricks. In the conceptual part of this chapter, we already described implementation and manufacturing process for the BRIX_2 extension module enclosures.

Two-Piece Lego Cases

In general, our extension module enclosures consist of two parts. The bottom plate, a standard 2 x 4 stud Lego plate provides a secure mechanical connection of the extension module to the BRIX_2 base module. It is identical for any type of extension module, see Figure 5.2. A cutout for the extension header and slightly trimmed studs allow a perfect fit of the PCB. The PCB is then glued to the bottom plate. The top part is individually designed for each type of extension module to fit its particular requirements. In the most simple case, it consists of a standard 2 x 4 stud Lego brick which is hollowed out and trimmed to fit the height of the components on the PCB. Should the PCB contain components that stick out of the case like buttons or rotary knobs or require an opening in the case like infrared sensors, further cutouts in the Lego brick are required. The finished top part is then glued to the bottom plate, which closes the extension module and protects the inside.

Making Extension Module Enclosures

We utilize a 3-axis CNC machine to process the Lego bricks and turn them into an enclosure in a process identical to the one described in Section 4.3.2. In theory one could also accomplish this with more simple tools like a drillpress, however that would mean much more effort in time and labor. In the following, we describe our process of manufacturing BRIX_2 extension module enclosures.

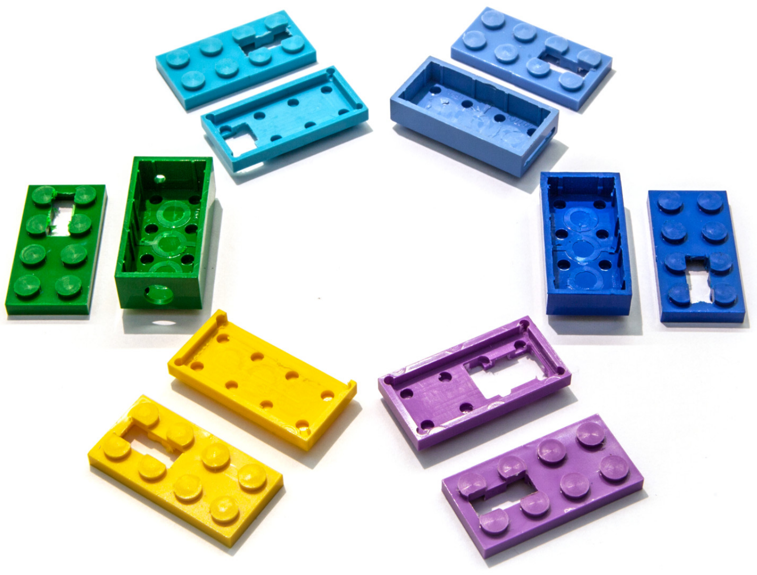


Figure 5.2: Various $BRIX_2$ extension module enclosures.

Bottom Plates Since all bottom plates are identical, they all share a common manufacturing process. We divided it into two stages. In the first stage, the CNC mill trims the studs of the bottom plate in order to match the heights of the electrical connectors. If the studs are not trimmed, both connectors do not match completely, resulting in an unreliable electrical connection. For this step we use an 8 mm end mill. In the second stage, we cut out a hole for the extension connector using a 1 mm end mill in order to achieve corners with a small radius. To save time, the first stage should be performed on all bottom plates before changing the tool and proceeding to the second stage. This way an average of around five bottom plates can be processed in a minute by a skilled CNC operator.

Top Parts These parts are individual for each type of extension module, so we only describe the general process first and cover some details. The manufacturing process depends on the complexity of the part. In general the stability of the workpiece should be maintained as long as possible throughout the process. This means the operations that weaken the structure of the brick the least are to be performed first. Smaller cutouts on the top side for example tend to have only little effect on the stability of the workpiece. The hollowing operation, performed with a 8 mm end mill should be the last operation, because it drastically reduces the stability of the brick and leads to problems with clamping the workpiece for following operations. Cutouts with square shapes should be performed with a 1 mm end mill, again in order to achieve corners with a small radius. If the components on the top side of the extension module are low profile, the bottom side of the brick can be trimmed in order to reduce the overall height of the module. However, we recommend to trim it only in increments of 3 mm, which is the thickness of a Lego base plate. This way the resulting extension module remain compatible to the Lego system.

Suggestions for a Future Manufacturing Process

We envision two different approaches to manufacture extension module enclosures in the future, similar to the ideas stated earlier in Section 4.3.5. Here we presented on the one hand a mass-manufacturing approach based on injection molding for high-volume production. On the other hand, a modified 3D printing approach allows decentralized manufacturing and does not require expensive machinery and tools. In the following, we focus on the latter, because it is more feasible for the near future of the BRIX$_2$ system.

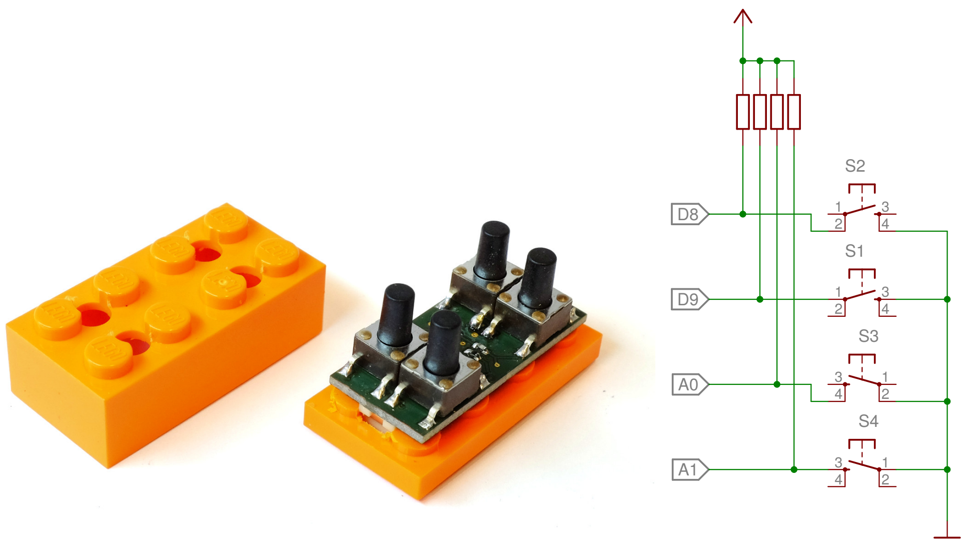
3D Printing and Lego For the base module, we already presented an approach that combines existing Lego parts and 3D printing. Since all extension module bottom plates are identical, printing the top parts on the bottom plates while inserting the electronics during the process is promising. Some of the top parts require much more tooling than the simple bottom plates, which means that there is a significant potential for optimization. For this reason we are going to experiment with process of combining Lego and 3D printing in the near future.

3D Printing Only As the capabilities of 3D printing grow, we will no longer rely on injection molding in order to produce highly precise plastic parts. Within a few years we will be able to print a reliable friction based connection like Lego parts provide today. Using off-the-shelf 3D printers, people will be able to print their own enclosures for BRIX$_2$ base modules as well as extension modules.

5.3.2 Input and Sensor Extension Modules

In order to sense the physical world or react to the actions of a user, any computing system requires inputs or sensors. To add to the sensors that are already integrated into the BRIX$_2$ base module, we designed five different extension modules for input and sensing that we present in the following.

Button Extension Module



(a) (b)

Figure 5.3: Opened BRIX$_2$ button extension module (a) and schematic (b).

To generate a basic, binary input for a BRIX$_2$ application, we built the button extension module, see Figure 5.3 (a). It features four pushbuttons that are connected to GPIOs on the user controller. This way they can easily be accessed in the firmware by just reading the state of the corresponding pin. We use APEM MJTP SERIES [123] pushbuttons which have a short actuation travel and a distinct tactile feedback. The buttons are rated for a minimum of 100000 cycles and have a contact bounce time of less than 10 milliseconds.

Direct Connection vs. Multiplexing Our current implementation of the button extension module requires 4 GPIO pins of the user controller, each connected to one of the buttons. If a button is not pressed, a resistor pulls the signal to VCC. As soon as the button is pressed, the pin is connected to GND, see Figure 5.3 (b). To reduce the number of pins required, several ways of multiplexing are possible. Using a resistor network, time-multiplexing or a dedicated multiplexer IC connected for example via I2C, we could reduce the required number of pins to a single input. All these solutions would require a special firmware that reads the buttons. Of course this could be wrapped in a library, but this abstraction layer would remove certain aspects such as contact bouncing, which are interesting in teaching scenarios. Also reading a signal from an input pin is one of the most basic programming exercises on a microcontroller. By not using a hardware and software abstraction layer, we allow users a direct access to the button and stay compatible to many tutorials on that topic which mostly address beginners.

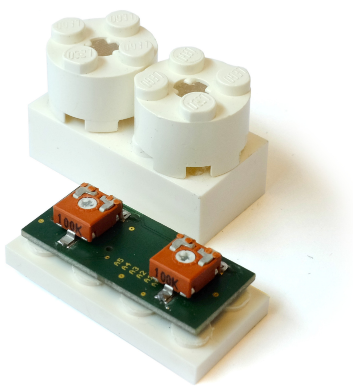
Enclosure Features When designing the module we placed each button exactly in the center between two studs of the Lego enclosure. The knobs of the buttons stick out slightly more then the studs. A 2 x 1 stud Lego plate can be placed right above a button to keep it pressed. This way a BRIX$_2$ application can be held in a certain configuration, which can also be changed at any given time. The knobs are guarded by the studs protecting them from accidental actuations, for example when the application is carried in a pocket.

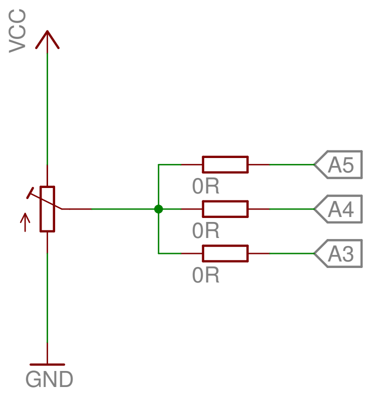
Examples in LiBRIX$_2$ At the time of writing, we supply a single example for the button module that demonstrates how to read all four buttons and map them to LEDs. The code does not support debouncing at the moment because it was intentionally kept simple. For more details on reading buttons, we refer to the already existing Arduino examples and tutorials [2].

Potentiometer Extension Module

The potentiometer extension module, see Figure 5.4 (a), offers users two rotary controls that can for example adjust parameters of the application while it is running. There is also an almost linear mapping between the rotation angle of the knob and the output data of the module, so angles can be measured directly. On this extension module, two ACP CA6 series potentiometers [124] with a linear taper and 100 kΩ resistance form two voltage dividers. The center taps can each be connected to three different analog inputs of the user controller. The configuration is done with 0 Ω resistors during population of the board, see Figure 5.4 (b). This way, we can build three completely independent potentiometer modules with the same PCB. They can be used together on a single BRIX$_2$ base module, giving users a total of six controls.

[2]https://www.arduino.cc/en/Tutorial/Debounce


(a)



(b)

Figure 5.4: BRIX$_2$ potentiometer extension module (a) and schematic (b).

Enclosure Features In order to operate the potentiometers, they need to be equipped with a shaft and a knob. We decided to utilize round, 2 x 2 stud Lego bricks for that purpose. They are located on top of the potentiometer extension module. The potentiometers can only rotate around 230 ° mechanical and the shaft is sensitive to sheer stress due to its small diameter. For this reason we have to protect the devices from wrong handling, which can easily occur. We do this by using one of the studs on top of the extension module as an end stop for the knob. This requires only a small modification on both Lego parts which can easily be done with a file, drill press or even a sharp knife. This modification makes the extension module mechanically robust and safer for use.

Examples in LiBRIX$_2$ Since the potentiometer is simple and straightforward to use, we only provide a single example for it. In the example sketch, we map the rotation angle of both knobs to the hue and brightness of the RGB LED. This demonstrates how the output data range of the potentiometers can be applied to different data ranges that are to be controlled.

AmbiSense Extension Module

In applications related to smart environments and wireless sensor networks it is frequently required to measure ambient properties like temperature or light intensity. We designed the AmbiSense extension module to integrate sensors for ambient modalities such as temperature, brightness and humidity into a single module, see Figure 5.5. This has been done for multiple reasons. First, we found out that the actual sensors are highly integrated and do not require a lot of external components to run. This allowed us to fit multiple sensors onto a single extension board PCB.

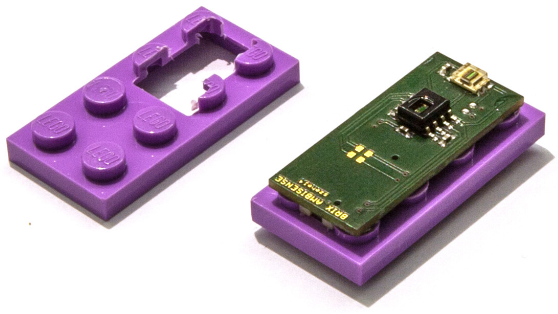


Figure 5.5: Opened BRIX$_2$ AmbiSense extension module.

Second, the power consumption of both sensors is so low that even if one of them is not used, the total consumption is still negligible. Third, in applications that measure ambient properties, often more than one of these properties will be relevant. Splitting all sensors to separate extension modules would increase the costs (PCB, board-to-board connector, case) and reduce the amount of available extension ports on the BRIX$_2$ base module. The AmbiSense extension module is equipped with two different sensors. The Taos TSL2561 ambient light sensor [125] and the Honeywell HIH6130 combined temperature and humidity sensor [126]. Both sensors can be accessed via the I2C bus.

Light Sensor The TSL2561 is a digital light sensor that incorporates two independent photodiodes, one for broadband and one for infrared light. An integrated ADC samples the outputs of the photodiodes with a resolution of 16 bits. The internal microcontroller implements an I2C interface so the sensor values can be read over a bus. Additionally it features different, user-programmable interrupts that trigger for example as a reaction to certain light changes. In the board design we connected the address select pin for the I2C bus to a solder jumper, which allows us to configure one of three different addresses during population of the board. We selected this particular sensor because of its capabilities to sense light in everyday scenarios, including infrared light. Also, the device is commonly used in physical computing projects and breakout boards for that sensor are available for different platforms. This also means that documentation on this sensor and how to use it is also widely available. [127], [128], [129]

Temperature and Humidity Sensor For measuring ambient temperature and relative humidity, we use the HIH6130 digital humidity/temperature sensor. The device is inexpensive, has a measurement range of 5°C to 50°C (temperature) and 0% - 100%

(relative humidity) and a convenient I2C interface. The internal MEMS sensors are sampled with a resolution of 14 bits, which leads to resolution of 0.025°C respectively 0.04%. The device also features two alarm outputs which are controlled by configurable temperature and humidity thresholds. They can be optionally connected to the pins D9 and D10 of the user controller during board population. The reason why we selected the device for our extension module is on the one hand because it integrates two sensors into a single chip. On the other hand, the device is, similar to the light sensor we utilize, used in multiple existing projects. Different libraries for the sensor were readily available, for example [130].

Integration into LiBRIX$_2$ In order to use the AmbiSense extension, users can include the ambisense class of the LiBRIX$_2$ into their sketch. This component offers functions to read out different light spectra, the temperature and the relative humidity. We provide four different example sketches that demonstrate different aspects of the extension module.

Proximity Extension Module

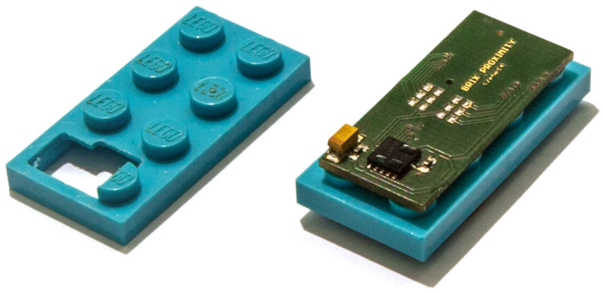


Figure 5.6: Opened BRIX$_2$ proximity extension module.

The proximity extension module, see Figure 5.6, uses an infrared distance sensor to detect the proximity of objects. Use cases are for example non-touch interfaces or proximity detection. Our extension module provides the proximity value, which gives an indication of how close an object is to the sensor, via the I2C bus. The range of the sensor is 0 to 200 mm, however the mapping between output value and distance to an object is highly non-linear and also depends on ambient light conditions. For this reason we do not call the device a distance sensor.

Proximity Sensor In our extension module we use the Vishay VCNL4010 fully integrated proximity sensor [131], which uses infrared light to measure the proximity of objects. Its typical applications are for example smartphones, where the sensor is used to detect when users hold the phone to their heads and also to measure the ambient light level. The infrared LED current of the VCNL4010 is configurable between 10 and 200 mA. The LED is only active for a short period of time when a measurement is performed. An internal processor supplies proximity and ambient light measurements via the I2C bus. Several interrupts can be configured by the user, such as thresholds or a certain value measured for a configurable period of time. We designed the PCB in a way that the interrupt pin can be optionally connected to one of six different pins of the user controller. The internal pull-up resistors of the user controller have to be activated in that case, because the interrupt line of the sensor is an open drain output.

Integration into LiBRIX$_2$ The LiBRIX$_2$ includes a class for the proximity extension, which users can integrate into their code. It offers two functions to obtain the raw proximity reading from the device or a reading scaled to a certain range. Two example sketches demonstrate how to use the extension module.

MakeyMakey Extension Module

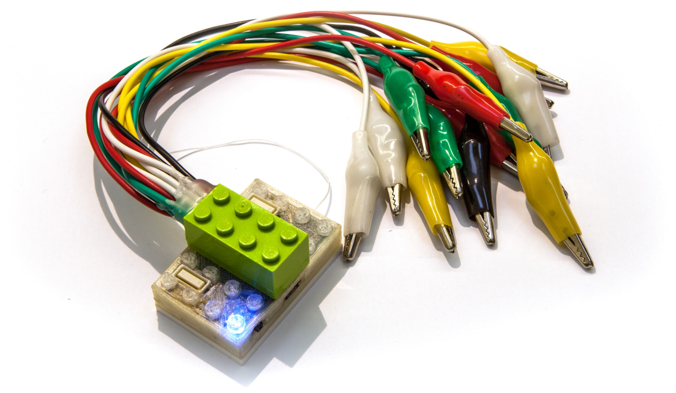


Figure 5.7: BRIX$_2$ equipped with a MaKey MaKey extension module.

In 2012, Jay Silver and Eric Rosenbaum of MIT developed a device that could turn any conductive object into a switch. Their platform called "MaKey MaKey" [132] uses high resistance switching and signal filtering on a microcontroller to detect if

a circuit is closed, even if its resistance is in the range of MΩ. The microcontroller then sends keypress events to a computer via USB and basically acts as a keyboard. In an application, users would connect the object that is supposed to be a switch, for example an apple to the board using alligator clips. The ground connection of the board is connected to the user. If the apple is now touched by the user, the circuit closes and is detected by the MaKey MaKey, which then triggers a predefined key, for example the left arrow. Thus the MaKey MaKey makes it possible to turn basically every surface into a switch, which offers great prototyping possibilities for interactive applications such as human computer interfaces. Since MaKey MaKey is open source, it is possible for us to utilize many parts of the project for our own implementation and turn it into a $BRIX_2$ extension module.
The MaKey MaKey is based on the Arduino platform and the only additional components are pull-up resistors for each sensing channel. By integrating those resistors into an extension module, we can turn $BRIX_2$ into a compact MaKey MaKey. We decided to connect all 16 GPIOs of the user controller (13 dedicated GPIOs plus three ISP GPIOs)to alligator clips on the MaKey MaKey extension module, see Figure 5.7, skipping only the I2C and UART lines. Due to the high resistance pull-ups, pins not used as a switch can still be used as an GPIO, for example to read a button extension or control motors via the servo extension. Given some changes in the sketch used for the MaKey MaKey, the touch events might not only trigger keypresses via USB, but also other functions within a standalone $BRIX_2$ application. This allows for example to build interactive textiles with touch-sensitive surfaces.
Our extension is compatible to the MaKey MaKey product[3], so our users can get inspired by projects based on the original product or use the official documentation to find support, for example to determine which materials work with the system.

[3]http://www.makeymakey.com/

5.3.3 Feedback and Actuator Extension Modules

Interactive applications require inputs and outputs. We already covered the possible inputs the BRIX$_2$ extension module kit provides in the previous section. In the following, we present the output and feedback extensions we developed to allow BRIX$_2$ to affect the physical world.

Servo Extension Module

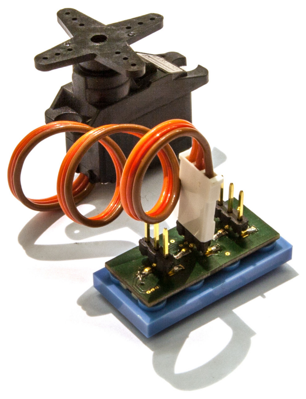


Figure 5.8: Opened BRIX$_2$ servo extension module with connected servo motor.

When attempting to make small and light things move in a prototyping situation, servo motors are the actuator of choice for many projects. They are commonly used in model making and physical computing because they are inexpensive, powerful and easy to control. With the servo extension module, we enable users to directly connect servo motors to a BRIX$_2$ module and use it to reach into and manipulate the physical world, see Figure 5.8.

On Servo Motors Servo motors consist of a geared DC motor and a position feedback sensor, usually a potentiometer. The setup allows the servo to approach and maintain a certain angle of rotation, defined by the input signal. The total range of rotation is usually around 200 degrees. Servos are available for a wide variety of applications from highly integrated to big powerful devices with significant torque. They are usually supplied with a 5.0 V input voltage and a pulse width modulated control signal. The advantage of servos in contrast to regular DC motors are the integrated controls and driver electronics as well as a standardized 3-way connector.

The BRIX$_2$ Servo Extension Module The servo extension module features three independent servo connectors. Each signal line can be connected to two different GPIOs of the user controller each during populating using $0\,\Omega$ resistors. The pin layout matches most common servo models. Power for the servos can not be supplied by the battery but only from USB. This means that the extension module can only be used if a USB power source is available. For mobile applications, we recommend using a USB power bank.

Servo Arduino Library To control servos with the servo extension module, we refer to the standard Arduino servo library [4], which we also include in our own servo extension module example. It demonstrates how to map acceleration data to the movement of the servo motor and provides a starting point for many similar servo control applications. Since servo motors are widely used in physical computing, there are already many tutorials, examples and project logs that BRIX$_2$ users can turn to in order to obtain more informations on the topic.

VibroSound Extension Module

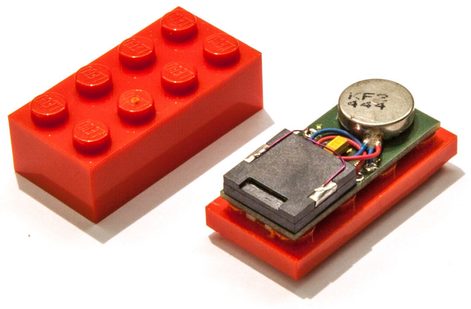


Figure 5.9: Opened BRIX$_2$ VibroSound extension.

Auditory and haptic sensations and stimuli have proven useful as an alert and notification signal in products of our everyday lives, from cellphones to microwave ovens. However, vibration has gained increasing popularity with the rise of mobile devices, because it is unobtrusive and spatially confined. The vibrating device has to be close to the body in order to be perceived. Sound on the contrary addresses all listeners in close range and can be regarded as a “broadcast” signal.
The VibroSound extension module, see Figure 5.9, covers both modalities in a single device and is capable to provide haptic and auditory feedback. The former is

[4]https://www.arduino.cc/en/Reference/Servo

generated by a small vibration motor similar to those used in cellphones and other mobile devices. The latter is generated by a piezo speaker and can generate simple sounds with controllable pitch.

Vibration Feedback To generate haptic feedback, the VibroSound extension module contains a Parallax C1026B series coin vibrating motor [133]. The device is designed for mobile applications and operates on a voltage of 3.3 V. It is rated for vibration frequencies of 10 to 55 Hz an can achieve an acceleration of $22\,\mathrm{m/s^2}$. The starting current of the vibration motor can be as high as 120 mA, which can not be handled by the GPIOs of the user controller. This is why we integrated an ON Semiconductor LV8413gp dual channel motor driver [134], which can handle peak currents of up to 600 mA. Using two GPIOs, users are able to control the direction of rotation. The rotation speed can be set using PWM and directly relates to the intensity of the vibration.

Auditory Feedback To generate sound with our extension module, we integrated an EKULIT PA12A03 piezo buzzer [135]. The pitch of the output sound can be controlled by different frequencies of the input square wave signal, ranging from around 1 to 10 kHz. However, the resonant frequency is only rated from 3.5 to 4.5 kHz. Due to the nature of such devices, the frequency response is highly non-linear and has no plateau, which means the amplitude of the output sound strongly depends on the pitch. This is why piezo speakers are not suitable for sounds that are more complex than just a beep or a simple melody. Nevertheless they are sufficient for many applications that just need to inform the user about a certain event or convey a one-dimensional parameter using the pitch of the sound.

VibroSound Component in LiBRIX$_2$ For sound as well as vibration, we each offer functions that a generate a continuous or single event signal. The intensity of the vibration can be defined by an 8 bit value. If it is set to zero, the vibration stops. Another function called "Shake" allows to also set a duration, so the vibration stops after a defined period of time. Similar to the vibration signal generation, for the sound component the library offers to set the pitch of the tone permanently or to beep at a certain pitch for a defined period of time. Using the beep function, users can for example generate short alert sounds or even play melodies, as we show in the "Tetris" example for the VibroSound module.

AudioAmp Extension Module

As we already pointed out, the VibroSound extension is only capable of generating basic sounds. Some applications however require more complex and richer sounds, for example when multidimensional data has to be sonified. Usually, such tasks are performed on a host PC or other external device with sophisticated computing power. The AudioAmp extension module allows to synthesize complex sounds or even play back audio samples on a microcontroller, see Figure 5.10. It also includes an amplifier circuit so speakers can be directly connected. This allows users to create portable, interactive sound applications using only the BRIX$_2$ system.

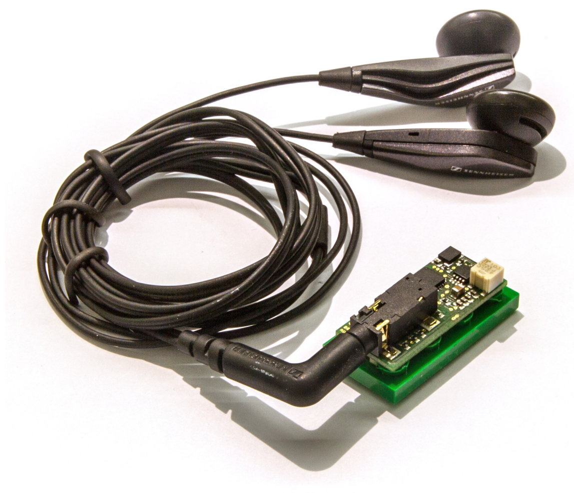


Figure 5.10: Opened BRIX$_2$ AudioAmp Extension Module connected to earphones.

System Overview Compared to other extension modules, the AudioAmp module is the most complex one we developed. It features an Atmega328P microcontroller to which the sound synthesis can be outsourced, a Texas Instruments TPA2005D1DGN class-D audio amplifier [136], a headphone jack and several configurable modes of operation.
The microcontroller is equipped with an Arduino compatible bootloader and can be accessed the same way as the system controller on the base module, see Section 4.2.2. This way users can upload dedicated sound synthesis sketches to the AudioAmp controller and offload that task from the user controller. The sketch on the user controller then controls the synthesis via UART. This requires the system controller to deactivate its UART.
The audio amplifier is supplied by V_{USB}, which means that it can only be used when connected to a USB power source. This is done for similar reasons as for the servo extension module, see Section 5.3.3: the amplifier is designed to work with a

5 V power source at an output power of typically 1.2 W. Using the power from the USB connection, our integrated power supply electronics do not risk damages due to wrong handling. The amplifier is capable of driving speakers with 8 Ω and 4 Ω internal resistance with an efficiency of up to 84 %. A trimmer allows to adjust the output volume of the amplifier.
The headphone jack is used for two purposes. First, users can connect headphones or an external amplifier to make the output of the module audible. Alternatively, speakers can be connected to the audio jack using a custom cable.

Modes of Operation The AudioAmp extension supports a number of different modes of operation which allows it to adapt to a high number of different applications. All configurations can be selected via solder jumper. The sound synthesis can either be performed on the AudioAmp microcontroller or on the user controller of the BRIX_2 base module, which is the default. It is more accessible for beginners and sufficient for many basic applications. Only as a step towards application optimization should users consider to swap the synthesis to the AudioAmp controller. The amplifier is optional and as a default, the module is configured for headphones without amplification. If the amplifier is switched on, the headphone jack will output the amplified signal. Combinations of all modes are possible.

Sound File Playback Despite of their limited computing power and memory, 8 bit microcontrollers are able to play back sounds and music from an external memory, for example an SD card. When combined with the SD card extension module, see Section 5.3.4, the AudioAmp module can be used for that purpose. Although we do not include these functionality into the LiBRIX_2, there are several external libraries that can be used, for example [137] or [138]. If the sounds that are played back are supposed to be influenced by for example sensor data, they need to be parameterizable. Usually the only thing that can be adjusted during playback are volume and playback speed. For more flexibility, sounds need to be generated at runtime in a process called sound synthesis.

Mozzi Synthesis Framework The Mozzi sound synthesis library [5] for Arduino is an extensive collection of different software synthesizer components that run on almost any Arduino compatible microcontroller. A large number of examples demonstrates how to generate, filter and parameterize even complex sounds. The library is explicitly designed to work with different sources of data that influence the sounds such as light sensors or motion sensors. For synthesis, common techniques such as oscillators, filters, delays and envelopes can be used. The library is open source and

[5]https://sensorium.github.io/Mozzi/

widely used. We redistribute it as a part of LiBRIX$_2$. By including libraries like the Mozzi framework, BRIX$_2$ becomes a compact platform for embedded sound and music generation as well as sonification, in other words: a musical instrument whose function and shape is totally controllable by the user.

5.3.4 Communication and Interface Extension Modules

After we have presented our input and output extension modules, we take a closer look on extensions that fall in neither category. Their purpose is communication with external devices, increasing accessibility to the base module or providing a mass data storage.

Breakout Extension Module

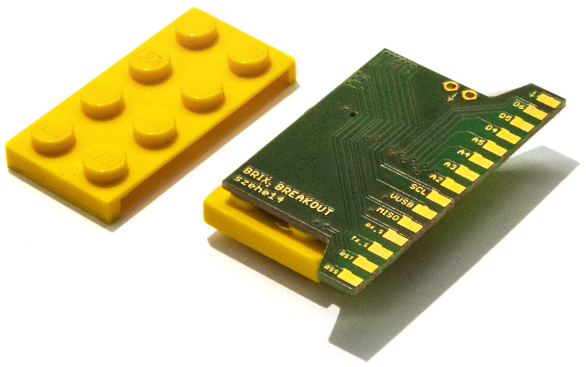


Figure 5.11: Opened BRIX$_2$ Breakout extension module

In the previous chapters we already discussed the accessibility of the BRIX$_2$ extension headers. Especially in prototyping scenarios, it is frequently required to attach custom hardware to a given system. For this purpose, we developed the breakout extension module, see Figure 5.11, which connects all signals from the fine pitch header of the BRIX$_2$ base module to solder pads with a standard pitch of 2.54 mm (0.1 in). A two-row pin header can be soldered to the edge of the board and connect for example to an Insulation-Displacement Connector (IDC). The breakout extension significantly increases the flexibility and adaptability of the BRIX$_2$ system because users are no longer limited to the existing scope of extension modules and system features. Now they are able to combine custom hardware with the existing capabilities of the base module and other extension modules.

Infrared Extension Module

In some smart environments applications, it is required to communicate with entertainment electronics like TVs or radios, for example to turn off the radio when sensors distributed in a room detect that nobody is present. The other way around, TV remotes are omnipresent and can be used to control other devices, given they are able to receive infrared signals and interpret the protocol. For purposes like these, we developed the infrared extension module, see Figure 5.12. Working with infrared

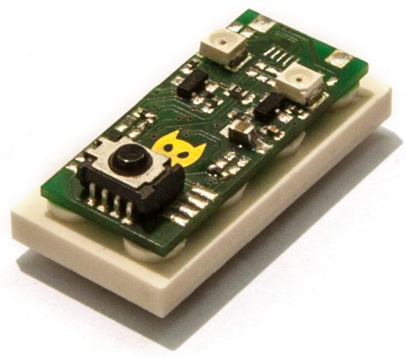


Figure 5.12: Opened $BRIX_2$ infrared extension module.

signals is common in physical computing. Numerous tutorials, libraries and examples online explain how to receive and send infrared commands. In order to send infrared signals, an LED with a wavelength between 800 and 1000 nm wavelength is used. On the infrared extension, we use two VISHAY VSMS3700-GS08 [139] infrared emitters that operate at a peak forward current of 120 mA. Both are driven by transistors and controlled by one of two selectable GPIOs of the user controller. Infrared control signals are pulses of infrared light that encode a series of bits. The signal is modulated with a carrier frequency, mostly to reduce the effects of ambient lighting [140]. Using an infrared decoder, a component that filters the signal and provides the raw bitstream, it is possible to read the infrared transmission with a microcontroller. We use a VISHAY TSOP6438TR [141] infrared receiver which is optimized for a carrier frequency of 38 kHz. The PCB layout of the infrared extension allows to populate the receiver facing to four different orthogonal directions. The output of the receiver can be connected to one of three optional GPIOs of the user controller via 0 Ω resistors.
The $LiBRIX_2$ offers examples and wrappers for sending and receiving infrared signals. It is possible to interact with devices of several different brands. Signals received from an infrared remote control can be read as raw bytes for example trigger events in a custom $BRIX_2$ application.

SD Card Extension Module

Applications like sensor data logging or playback of audio samples require much more memory space than available on the microcontrollers of the BRIX$_2$ base module. To provide mass storage capabilities for our system, we designed the SD card extension module, see Figure 5.13. It serves as an adapter between the user controller and a micro SD card. For example after data recording, the card can be removed and accessed in a different device such as a smartphone or laptop.

SD cards are accessed via SPI. Since the user controller supports SPI in hardware,

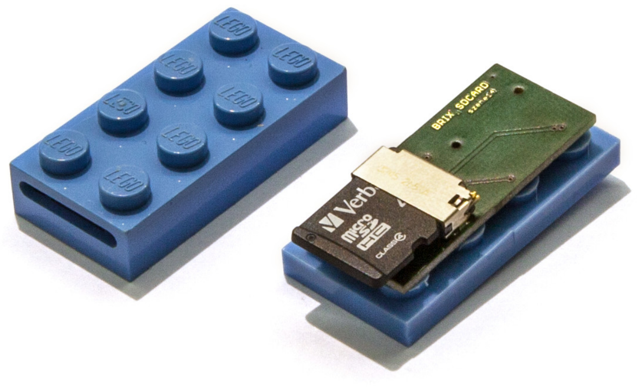


Figure 5.13: Opened BRIX$_2$ SD card extension module.

no additional components are required to interface the SD card. While the SD card is connected, other SPI devices on the bus can operate at the same time, but each of them is selected using an individual chip select pin. To allow maximum flexibility, the chip select pin on the SD card extension can be selected from six different options using $0\,\Omega$ resistors. The file system of the SD card should be FAT32, which supports cards with a capacity of up to 32 GB. The Arduino IDE provides a library that allows to read and write from SD cards. Our examples demonstrate some of these functions. Should users require assistance for more advanced applications, they can consult the Arduino documentation and other resources online.

BTLE Extension Module

Many modern mobile and wearable devices like smartphones and fitness trackers communicate via BTLE. The BTLE extension module, see Figure 5.14, allows users to design their own custom Bluetooth Low Energy devices using the BRIX$_2$ system. The core of the BTLE extension module is a BlueGiga BLE113 module [142] that contains all RF parts including a chip antenna as well as a programmable microcontroller. The module is controlled via UART and can be initially programmed through a two-wire serial debug interface. BTLE peripheral devices share data wirelessly through *services*. A device can advertise several services at the same time. Each service contains one or more *characteristics*, which contain the data. Central devices like smartphones can access those data fields by connecting to the correspondent service.

Services and characteristics are defined in the firmware of the BLE113, which users

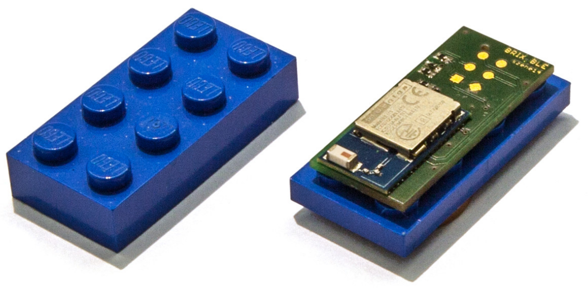


Figure 5.14: Opened BRIX$_2$ BTLE extension module.

are not required to alter for their application. We prepared a Generic Attribute Profile (GATT) for the BTLE extension that covers most applications. Our predefined services include for example the data from the IMU, but also several general purpose services that can be utilized for custom data types.

5.3.5 BRIX$_2$ Arduino Library for Extension Modules

To allow users to easily access to the functionalities of the extension modules, we integrated top level functions for selected extensions into the LiBRIX$_2$. In this section we describe how the code is organized and which features are available for extension modules in the LiBRIX$_2$.

Structure of Extension Module Libraries

We included all code for the extension modules into the LiBRIX$_2$ package so everything is available with just a single installation process. However, we separated the code for different extension modules into individual classes which have to be included into the sketch that utilizes that particular extension module. This modular structure provides more clearly arranged and the hardware configuration required for a particular sketch becomes directly obvious in many cases.

Wrappers and Feature Depth

We developed code with varying complexity for different extension modules. Some have a number of wrapper functions to initialize the device and use its features. Some do not require any wrapper code at all, for example the button or potentiometer extensions. We only provide special code for extensions if it is beneficial for easier understanding and quicker usage of an extension module. This is done to avoid unnecessary abstraction layers that hide trivial functions of the language or the microcontroller. For example if we wrapped the reading of the analog inputs to determine the angle of rotation of the knob, sketch would not be simpler. But at the same time, the underlying principle of reading analog ports would be obfuscated. Finally, the abstraction layer would make our extension non-compatible to other Arduino examples on reading analog ports or button presses if users are not familiar with the hardware side of the implementation. In this case, an example code on using the extension along with some helpful comments is more feasible.

Examples

For every extension module we developed, we also created example sketches that show how to use that particular module. Some extensions have multiple examples demonstrating different aspects or functions of the module or combinations with other functions of our system. Example sketches are an intuitive way to learn the programming language, although not a particularly structured one. They only show some random aspects of the language at a time and should not be the single source to learn a programming language. In our case they have two important functions:
Fallback: Example codes are designed to work out of the box when the hardware setup is given. Users can try out an extension module without having to write a

single line of code. This allows them to get a feeling for the capabilities of a certain extension without having to know how to program it. Once they get familiar with the example code, they will start to modify it towards their application needs, by adding or changing aspects of the code. Should an error occur during development, which breaks the application, they can always have a look at the original example code and analyze what is wrong with their own code.
Quick Reference: Example code shows how an extension module is used in a practical and condensed way. Of course there is an API and function documentation on the whole LiBRIX$_2$, but examples directly show how a certain functionality is to be integrated into a custom sketch. Especially in prototyping, it is faster and easier to just copy parts of a programming example into a custom sketch to make things work.

5.4 Conclusion

In the previous Chapter 4 we presented the implementation of the BRIX$_2$ base module, software components and documentation. We were able to integrate all functionalities and features we defined as requirements in Chapter 3 into a compact and robust devices that is compatible to the existing Arduino IDE. A custom software library allows users to easily integrate all functionalities of BRIX$_2$ into their applications. Our example sketches demonstrate how to use LiBRIX$_2$ and allow quick initial results even for users with little experience in programming. In this chapter, we presented another substantial component of BRIX$_2$: the extension modules. The allow users to adapt our platform to their application by adding all functionalities they require. Extension modules can just be stacked onto the base module, so no special skills like soldering are required to expand the range of features. Although we only designed 12 different extension modules that include sensors, actuators and communication interfaces, our kit is not limited to that scope. Even after the initial design time, novel extension modules can be build that further increase the flexibility of the BRIX$_2$ toolkit, which makes it an almost open ended platform.
In the following chapter, we analyze our platform regarding technical specifications and properties. We also take a look on the way BRIX$_2$ was used in actual applications by students as well as researchers and present the results of a user survey.

6 Analysis

This chapter is dedicated to a detailed analysis of the BRIX$_2$ platform in order to evaluate to which extend we succeeded in developing the system we aimed for. First, we focus on purely technical aspects and provide a detailed specification of all components that are integrated in our platform. Subsequently we focus on the performance of selected key components in real life scenarios. For this we conducted experiments and measurements to verify the data acquired by our sensors or the power consumption of active electronic components in BRIX$_2$. The results are presented in Section 6.2.
In the second part of this chapter, we report our experiences with BRIX$_2$ as a teaching platform in several lectures before we discuss selected projects of researches and students that illustrate the performance of our toolkit in fields like motion capturing, WSN or smart environments.
Finally, we present the results of a user survey conducted among 20 students who were asked about their experiences with BRIX$_2$ and opinions on the system. This helps us to answer questions of usability and user motivation and provides us with further ideas for further versions of BRIX$_2$.

6.1 The BRIX$_2$ Toolkit: Technical Specifications

In the following we sum up all technical properties of the BRIX$_2$ base module and extension modules to give the reader an overview about the capabilities as well as the limits of our system. In Table 6.1 we cover the mechanical and electrical properties of the base module and both microcontrollers, followed by information on the IMU sensor and the RF transceiver in Table 6.2. In Table 6.3 to Table 6.5 we cover the specifications of all BRIX$_2$ extension modules. A more detailed analysis of selected components is provided in the next section.

Parameter	**Value**	**Unit**	**Condition/Note**
General			
Base Module Enclosure	32×47.8×14.1	mm	
Base Module PCB	28.6×44.6×8	mm	w. battery
Base Module PCB Weight	14.4	g	w. battery
Base Module Total Weight	21.5	g	
Battery Capacity	450	mAh	
Charge Time	80	Minutes	on USB port
System Voltage	3.3	V	
Extension Ports			
Number of Pins	30		
GPIOs	25		
Analog Inputs	6		
Max. Current per GPIO	40	mA	
Available Buses	SPI, I^2C, UART		
3.3 V Rail Current (max.)	500	mA	limited, fused
5.0 V Rail Current (max.)	1000	mA	via USB
User Controller			
Operating Voltage	3.3	V	
Active Supply Current	10	mA	
Idle Supply Current	3	mA	
Clock Frequency	16	MHz	
SRAM	2.5	kB	
Available Flash Memory	28.672	kB	w. bootloader
EEPROM	1	kB	
Available Digital GPIOs†	20		
Available Analog Inputs†	6		
System Controller			
Clock Frequency	16	MHz	
Supply Voltage	3.3	V	
Active Supply Current	7	mA	
Idle Supply Current	1.5	mA	
SRAM	2	kB	
Available Flash Memory	30.720	kB	w. bootloader
EEPROM	1	kB	
Available Digital GPIOs†	6		

† Connected to Expansion Port.

Table 6.1: $BRIX_2$ base module specifications (I)

Parameter	**Value**	**Unit**	**Condition/Note**
Wireless Transceiver (Texas Instruments CC1101)			
Operating Voltage	3.3	V	
Idle Supply Current	1.7	mA	
Power Down Supply Current	0.001	mA	
TX Power Range	-69.2 – 10.7	dBm	
TX-Mode Supply Current	16.8	mA	0dBM
TX-Mode Supply Current	30	mA	+10dBM
RX-Mode Supply Current	15.6	mA	
RX Sensitivity	-112	dBm	
Transmission Range	10 – 100	m	indoor, 0dBM
Transmission Range	50 – 250	m	outdoor, 0dBM
Bitrate (max.)	500	kB/s	
IMU (Invensense MPU9150)			
Operating Voltage	3.3	V	
Idle Supply Current	0.01	mA	default mode
Active Supply Current	3.7	mA	IMU only
Active Supply Current	3.8	mA	IMU & DMP
Sampling Rate	1	kHz	
Accelerometer Ranges	±2, ±4, ±8, ±16	g	
Gyroscope Ranges	±250, ±500, ±1000, ±2000	°/sec	
Magnetometer Range	±1200	µT	

Table 6.2: BRIX_2 base module specifications (II)

Parameter	Value	Unit	Condition/Note
AmbiSense Extension Module			
Size of Enclosure	32×16×10	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	3.4	g	
Light Sensor Active Supply Current	0.24	mA	
Light Sensor Idle Supply Current	3.2	µA	
Light Sensor Dynamic Range	1000000:1		
Light Sensor Resolution	16	Bit	
Humidity Sensor Active Supply Current	0.65	mA	
Humidity Sensor Idle Supply Current	0.6	µA	
Humidity Sensor ACD Resolution	14	Bit	
Temperature Measurement Range	-25 - 85	°C	
AudioAmp Extension Module			
Size of Enclosure	32×16×16.1	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	4.9	g	
Internal Microcontroller	ATmega328p		
BTLE Extension Module			
Size of Enclosure	32×16×11.7	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	4.4	g	
Sleep Mode Supply Current	500	nA	
TX Active Mode Supply Current (max)	18.2	mA	
TX Power Range	-23 - 0	dBm	
RX Active Mode Supply Current (max)	17.9	mA	
RX Sensitivity	-93	dBm	
Breakout Extension Module			
Size of Enclosure	38.3×23.3×10	mm	
Size of PCB	38.3×21.5	mm	
Total Weight	4	g	
Solder Pad Pitch	2.54	mm	
Button Extension Module			
Size of Enclosure	32×16×17	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	5.7	g	
Number of Buttons	4		
Contact Bounce Time	<10	ms	
Switch Travel	0.25	mm	
Actuation Force	160	grams	

Table 6.3: Extension module specifications (I)

Parameter	**Value**	**Unit**	**Condition/Note**
Infrared Extension Module			
Size of Enclosure	32×16×10	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	4.5	g	
Range Transmitter (approx.)	5	m	using TV as receiver
Range Receiver (approx.)	13	m	using a standard TV remote
Current(burst)	230	mA	Transmit Mode
Current	0.15	mA	Receive Mode
MakeyMakey Extension Module			
Size of Enclosure	32×16×16.1	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	54.2	g	
Number of Sensing Channels	8		
Cable Length	200 – 250	mm	
Potentiometer Extension Module			
Size of Enclosure	32×16×25.8	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	7.4	g	
Number of Potentiometers	2		
Rotation Range	270	°	
Proximity Extension Module			
Size of Enclosure	32×16×10	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	3.4	g	
Active Supply Current (average)	4	mA	Max. Sampling Rate and LED Current
Measurement Range	15 – 100	mm	
SD Card Extension Module			
Size of Enclosure	32×16×11.7	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	4.2	g	
Active Supply Current	30 – 60†	mA	Write Operation

† Depends on SD card used.

Table 6.4: Extension module specifications (II)

Parameter	Value	Unit	Condition/Note
Servo Extension Module			
Size of Enclosure	32×16×18.1	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	4.8	g	
Number of Channels	3		
VibroSound Extension Module			
Size of Enclosure	32×16×16.1	mm	
Size of PCB	28.6×12.6	mm	
Total Weight	5.6	g	
Motor Active Supply Current (max)	63	mA	
Motor Vibration Intensity (max)	±2.5	g	Tight Coupling on Human Skin
Buzzer Active Supply Current (max)	2.5	mA	
Buzzer Resonant Frequency	4000	Hz	(±500 Hz)
Buzzer Sound Output (min)	75	dB	10 cm Distance

Table 6.5: Extension module specifications (III)

6.2 BRIX$_2$ in Practice

Technical specifications provided by the component manufacturers do often not fully reflect the behavior or performance of that particular component in an actual application. In order to provide a more illustrative impression of the performance of the BRIX$_2$ platform, we recorded measurements in exemplary applications and observed how different features and aspects of BRIX$_2$ perform and behave in practice. From the data we recorded, we can make qualitative statements about the system's capabilities and also the limitations of sensors, actuators and other components. In the following section we begin with the base module and its features and subsequently move on to the BRIX$_2$ extension modules.

6.2.1 The Base Module in Practical Use

The key features of the BRIX$_2$ base module are the microcontrollers, the IMU sensor, the RF transceiver and the battery that enables mobile applications. In this section we take a closer look at these components in real life applications. We regard quantitative parameters like current consumption or battery charge time, but also qualitative aspects, for example the overall performance of the IMU as well as potential opportunities for improvement we encountered while using the BRIX$_2$ platform.

Microcontrollers

Arduino compatibility was a key requirement for the microcontrollers used on the BRIX$_2$ base module. Both controllers are equipped with the same bootloaders as the original Arduino products they are used in (Arduino Leonardo and Arduino Due respectively) and work accordingly well with the Arduino IDE. However, they also share some issues with those original products:

User Controller Serial Port Issue: The ATmega32U4 implements a USB stack and can be connected directly to USB, in contrast to the ATmega328p, which requires a USB/Serial converter like the FTDI FT232RL. If the ATmega32U4 is reset, for example during the upload of a sketch, the USB device is removed and re-attached to the computer virtually. As a result, the number of the virtual serial port is changed sometimes, so the device is no longer available in the software. Usually, unplugging and reconnecting the BRIX$_2$ helps to overcome this problem.

User Controller Bootloader Issue: At startup of the BRIX$_2$ base module, it enters the bootloader for 8 seconds, indicated by the "breathing" blue LED. After that period is timed out, the sketch is started. Within the 8 seconds after the reset, the IDE can access the controller and upload a sketch. In some rare cases, the bootloader is ignored after reset and the sketch starts immediately. This can render the device unprogrammable through the IDE. The problem is persistent, after uploading different sketches, the bootloader stays inactive. We could not reproduce the behavior sufficiently and the only solution appears to be erasing the controller and to apply a new bootloader via ISP. In other similar Arduino products, this issue can be resolved by pressing the reset button of the board[1]. However, in the design of $BRIX_2$ we did not include a reset button. The reset line is available on the extension header, which makes it possible to integrate the reset button on an extension board.

Invensense MPU9150 IMU Sensor

The MPU9150, including a three-axis accelerometer, gyroscope and magnetometer as well as integrated sensor data fusion (DMP) is a key element of the $BRIX_2$ base module. In Section 3.4.4 we already presented measurements of the accelerometer and gyroscope noise as well as drift of the DMP data under different conditions. Now we focus on data recorded in a real life scenario in order to give an impression of the performance and limitations of the sensor. In the exemplary application we recorded acceleration data of a person while walking. We wanted to find out whether the recorded data was usable for gait detection in order to count the steps the person made. The $BRIX_2$ base module was placed in different locations of the body and in random orientation. We recorded the data of all 3 axes of the accelerometer and summed up the absolute sensor values.
As we can see in Figure 6.1, the quality of the resulting signal depends on the location of the sensor. When placed directly at the ankle using a flexible strap, we get a clean signal with a distinct pattern for each step. If the sensor is placed in the pocket of a persons pants, the signal gets a little more noisy, but the pattern is still visible. However, if the sensor is placed far from the body inside a shoulder bag, the pattern almost disappears in the noise. This example illustrates the signal quality of the accelerometer in the $BRIX_2$ base module in a dynamic scenario when the module is in constant motion.

[1]https://learn.sparkfun.com/tutorials/pro-micro–fio-v3-hookup-guide/troubleshooting-and-faq#ts-reset

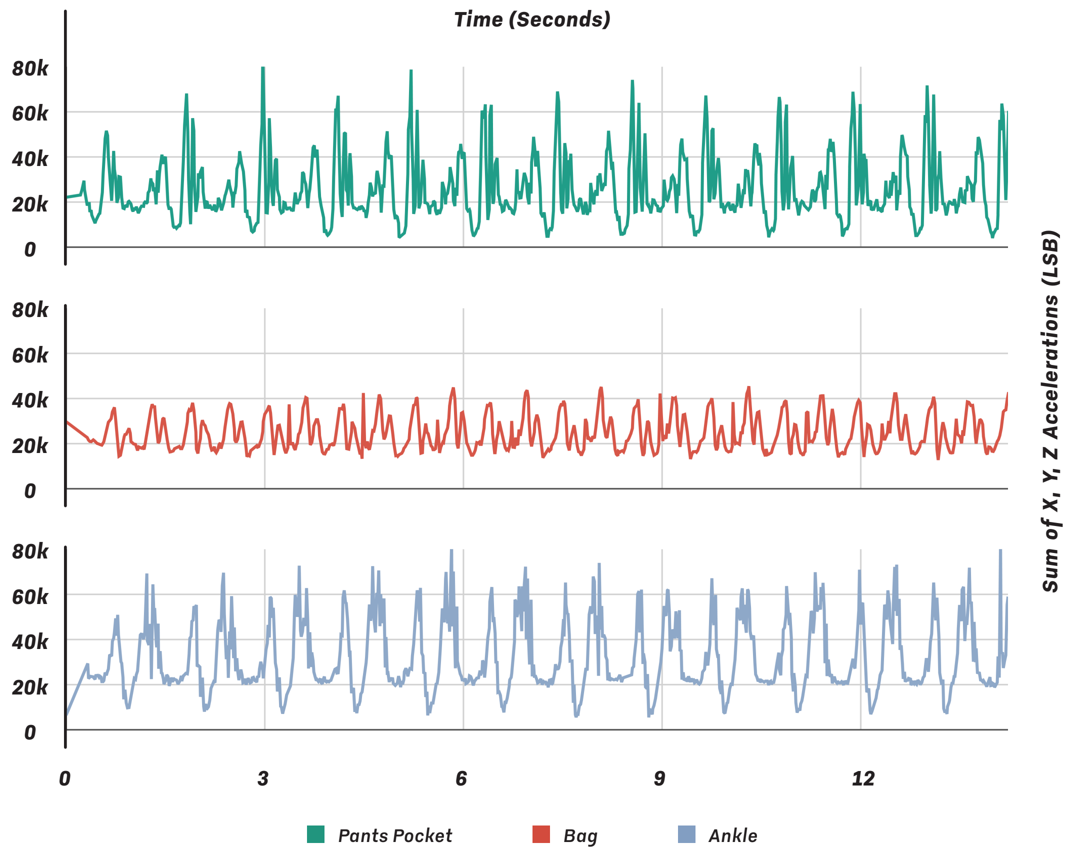


Figure 6.1: Summed up absolute x,y and z-axis accelerations for a walking scenario with 3 different sensor locations.

RF Transceiver

The RF transceiver allows a BRIX$_2$ application to communicate with other remote devices wireless. For most scenarios, two key parameters are critical: power consumption and transmission range. In the following, we provide measurements and calculations on both of these parameters.

Power Consumption There are basically two settings that affect the power usage of the Texas Instruments CC1101 RF transceiver used on the BRIX$_2$ base module: The device state (*Receive*, *Transmit*, *Idle*) and the TX power setting. In idle state, the typical supply current is 1.7 mA. The device can additionally be sleeping (500 nA) and woken up by radio or the crystal oscillator can be turned off (165 µA). In receive mode, the supply current is almost constant at around 15.2 mA, depending on the strength of the input signal. In transmit mode, the supply current depends on the transmit power settings.

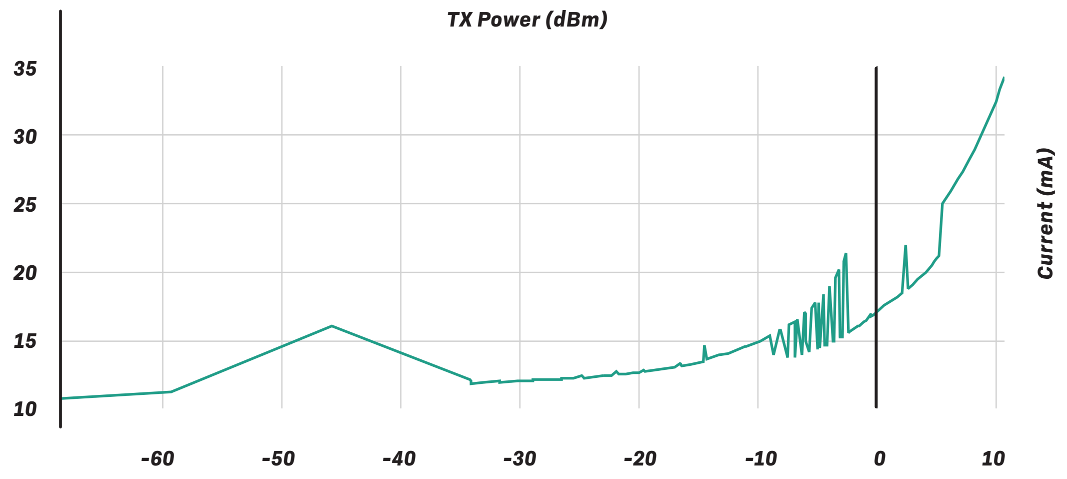


Figure 6.2: CC1101 transmit power vs. active supply current. Data taken from [143].

In Figure 6.2, we can see that the with increasing TX power, the current changes highly nonlinear. This is caused by the design of the power amplifier of the CC1101, which uses multiple stages and voltage ramping on some stages in order to optimize its efficiency. As a result, there are more and less efficient TX power settings, for example the current is the identical at -47 dBm and -3 dBm. This has to be considered when optimizing an application towards energy efficiency.
The library we use for the CC1101 RF transceiver keeps the device in RX mode unless it is in TX mode, thus makes no use of any idle or sleep mode. In order to determine the effect of the payload length on the average transceiver supply current under the assumption that no sleep modes are used, we can calculate the total transmission time (pt) based on our default data rate of 38400 baud (see [110], p.78) plus the fix transition time from RX to TX state and back rounded to 0.8 ms (see [110], p.54) as a worst-case estimation. Besides the payload, packets include a 4 bit preamble, a 16 bit sync word, 8 bit each for packet length and address as well as a 16 bit CRC:

$$pt = switch_{RX/TX} + (\frac{1}{baudrate} \cdot header + payload + CRC) + switch_{TX/RX}$$

Based on the total transmission time (pt) for a given payload size, we can calculate an average supply current (apc) for a given data rate in Hz and given TX supply currents for different output power levels:

$$apc = datarate \cdot pt \cdot i_{TX} + (1 - datarate \cdot pt) \cdot i_{RX}$$

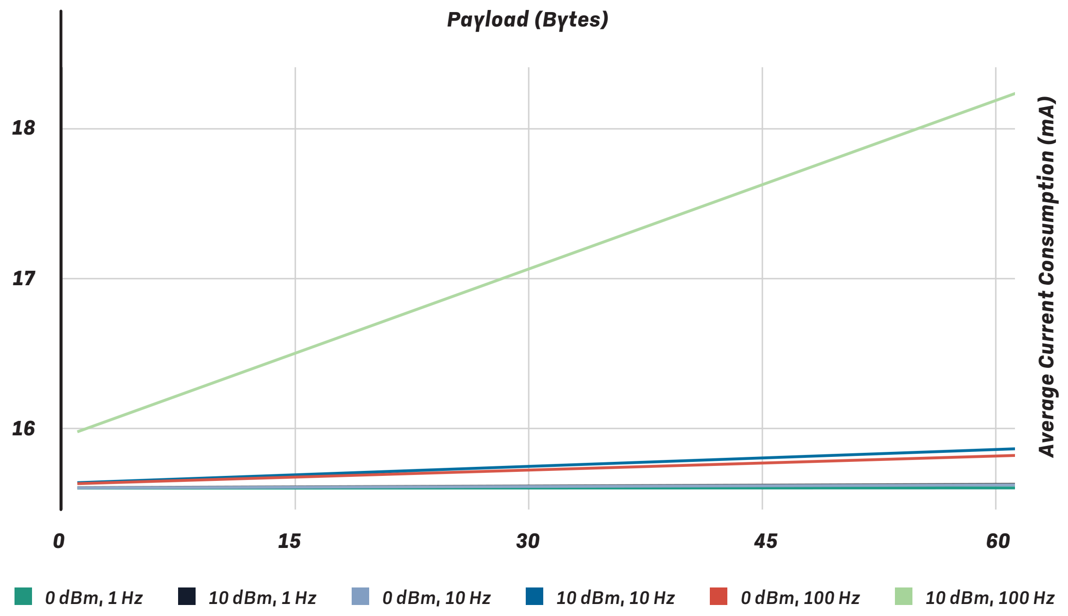


Figure 6.3: RF module active supply current as a function of payload size under different conditions.

The results for some parameter configurations are presented in Figure 6.3, which shows that the supply current varies between 15.5 mA and 18.3 mA. If we restrict the output power to 0 dBM, the supply current is even below 16 mA. As a result we can conclude that the data rate and output power of the wireless transceiver have only little effect on the power consumption. The packet length only affects the power consumption in a significant way if the output power and data rate are at a high setting.

In order to reduce the average operating current, users can actively put the RF transceiver into a sleep mode using a command in the LiBRIX$_2$ while it is not in use. Especially in scenarios with a low data rates, the device will spend most of the time in a power down mode and consume only 0.001 mA. If the RF feature of BRIX$_2$ is not used, the transceiver should be deactivated at all times.

Transmission Range In Section 3.4.1 we presented an experimental approach to evaluate different wireless transceivers. Among the four technologies we tested using the B2DK was also the CC1101, which is now integrated into BRIX$_2$. We tested three different scenarios in order to cover as many potential applications as possible, so the measurements should give the reader a solid impression of the capabilities of the BRIX$_2$ wireless transceiver. However, we would like to point out that there are certain factors besides the distance between RX and TX that can have significant effects on the transmission range and should be considered when planing a wireless application with BRIX$_2$:

- **Surrounding:** The quality of a wireless link depends on the surroundings and is influenced by signal reflection, refraction, and scattering [144]. In an open field, the link can behave completely different than in a corridor inside a building, regardless of the fact that both applications having a line of sight.
- **Band Occupation:** If in a surrounding the RF band around the carrier frequency of the wireless link is already heavily used by other devices, packet losses are more likely to occur than in areas with little wireless traffic. This especially applies to transmissions that do not rely on a sophisticated protocol which offers error correction mechanisms.
- **Obstruction:** Certain materials tend to block RF signals, whereas others have no great effect on the transmission range when placed between RX and TX. A link from one room to another might work great through a drywall but not at all through a reinforced concrete wall. The human body also absorbs RF signals [145], so in a wearable wireless application, it can be important at which location on the body the transceiver is placed.

We generally recommend to test a wireless link in practice under different conditions during the design of an application.

RGB LED

The ROHM SMLP34 Series PICOLED™ [146] with a size of 1×1×0.2 mm was one of the most integrated RGB LEDs we could find on the market in 2012. The forward current of all channels is approximately 10 mA each, which is low enough for a battery powered application. The LED has a common anode and we provide the power for each channel through a GPIO of the user controller. The common anode requires an inverse logic for controlling the LED, high means the channel is off, low means the channel is on. All this is is already wrapped in our LiBRIX$_2$, so users do not have to take this into account. The library also covers dimming of the color channels using PWM.
At design time, we specified the current limiting resistors for each color channel in a way that when all channels are fully lit, the resulting color appears as white. Since the relative response of the human eye to colors is non-linear [147], the forward currents of the LEDs are not equal but are especially low on the green channel. At this wavelength (555 nm), the human eye is more sensitive, so we had to adapt the brightness of the green LED in particular. In PWM dimming, the voltage - and thus

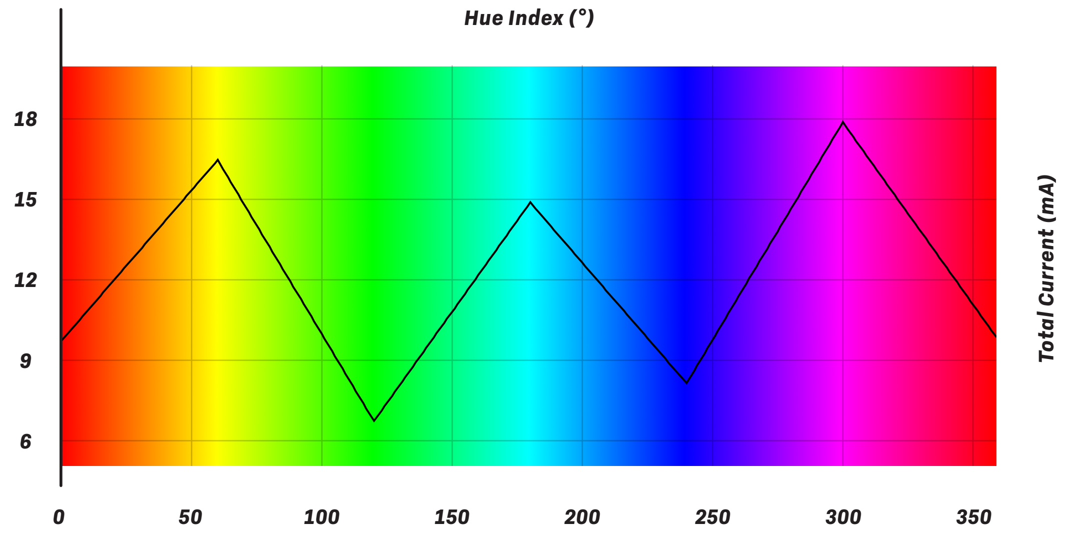


Figure 6.4: Total forward current vs. color. of the BRIX$_2$ RGB LED.

the forward current of the LED - is proportional to the PWM duty cycle [148]. In Figure 6.4 we can see the total forward current of the RGB LED for each color in the Hue, Saturation, Value (HSV) color model, which is also covered in the LiBRIX$_2$, see Section 4.4.2. At full saturation and full brightness, the total RGB LED supply current peaks at mixed colors yellow, cyan and pink in the hue value (0-359) and is lowest at the pure colors red, green and blue.

System Status LED

The green LED that displays the system's status is a Kingbright KPT-1608SGC [149] controlled by a GPIO of the system controller. The LED is designed to operate on a forward current of 20 mA. However, to save energy we operate it at around 10 mA. The brightness is still sufficient to give a basic status information, for example whether the device is turned on or off. Users can deactivate the system LED via the LiBRIX$_2$ in order to reduce the total power consumption.

Current Consumption by Feature and Potential for Optimization

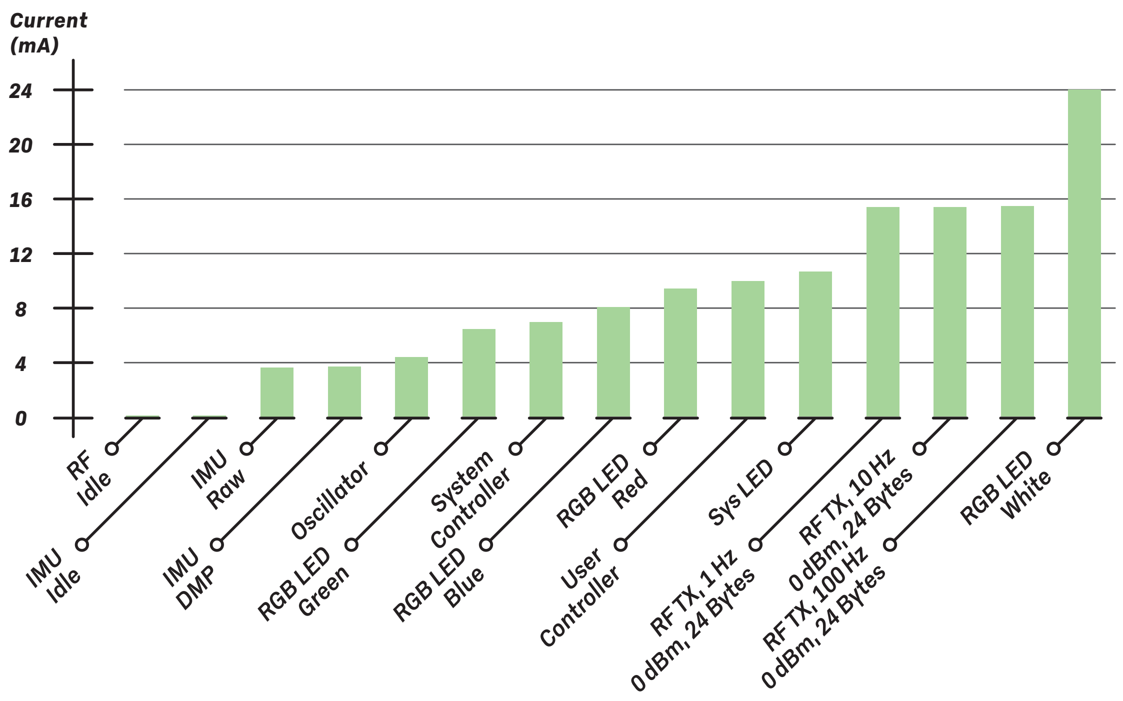


Figure 6.5: Overview of the active supply currents of different BRIX$_2$ base module features.

To provide an overview of the power consumption of each individual feature of the BRIX$_2$ base module, we arranged them next to each other. As we can see in Figure 6.5, the LEDs and the wireless transceiver have the highest active supply currents, followed by both micro controllers. Especially the LEDs offer a significant potential for saving energy. The system status LED for example is not required to be constantly lit only to inform the user that the BRIX$_2$ base module is turned on. Alternatively it could flash for 100 ms and stay off for a period of 900 ms, thus reducing its average supply current by 90 %. The same applies to the RGB LED.

Depending on the application scenario, the RF transceiver can also be turned off most of the time, for example while it is neither sending nor required to receive data. To illustrate the impact of such measures, we consider an example application where a $BRIX_2$ module, mounted to a door reports the opening angle of the door, measured by the IMU wireless to a remote PC in the same room. The status of the door is displayed by the RGB LED.

Before Optimization: By default, both LEDs are constantly lit, the RGB LED occasionally changes color. The RF transceiver is always active by default. The average total supply current is around 60 mA, which results in a total runtime of 7.5 hours.

After Optimization: Both LEDs are set to flash with a 10 % duty cycle. The RF transceiver is turned off while not sending. The average total supply current drops to around 30 mA. These simple optimizations do not restrict the performance of the module in any way but doubles the battery runtime to 15 hours.

Battery Charging

During the design phase, we already briefly tested the battery charging circuit by recording charge times of the battery. These proved to be consistent, which means on the one hand, the charging process is stable and on the other hand that the capacity does also not decrease significantly. For a more detailed analysis, we recorded cell voltage and charge current during a full charge cycle. As we can see from the plot in Figure 6.6 the charge current drops constantly while the charge voltage is constant. The Microchip MCP73831 charge controller we use on our boards has basically two charge modes, the *"Fast Charge Mode"* and the *"Constant Voltage Mode"*. It transitions from the first into the latter as soon as the cell voltage of the battery exceeds the regulation voltage of the charge controller. In our case (MCP73831T-2ACI/OT) this voltage is 4.2 V, which is almost immediately reached at the beginning of the charge process. For this reason, most of the charging is done in fast charge Mode with constant voltage, which leads to decreasing current as the battery charges. A reason for this atypical behavior of the battery, which should normally charge at a constant current for at least half the charge process might be the battery protection circuit, which is integrated into the batteries we use. Since we have no influence on that circuit, we can not fix this issue and can only consider to use different batteries without internal protection along with a custom battery protection circuit on future revisions of BRIX$_2$.

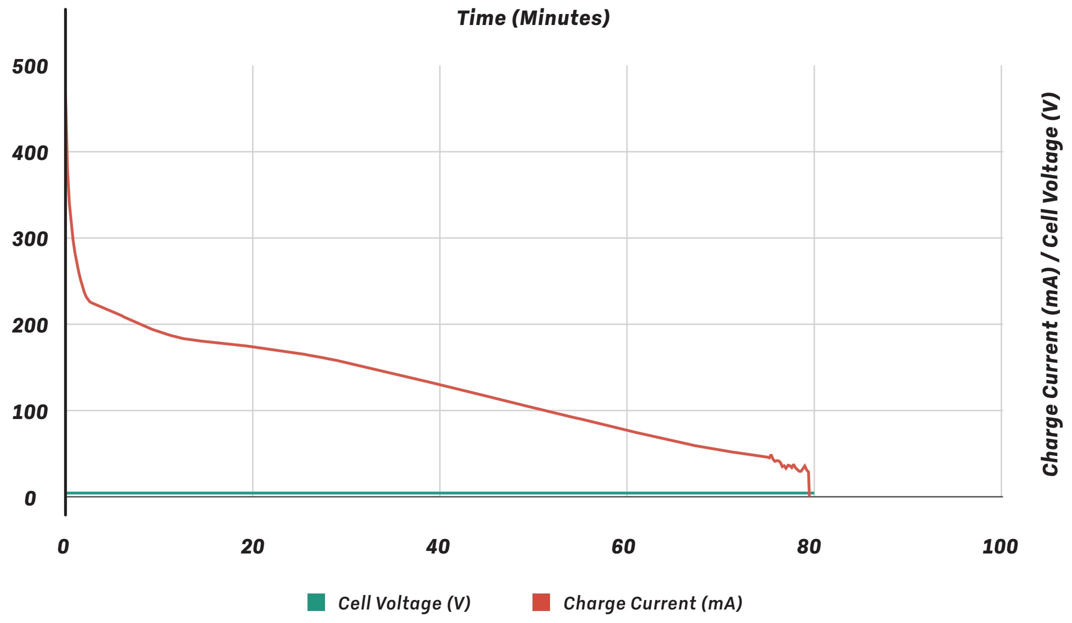


Figure 6.6: Charging the internal battery of a BRIX$_2$ base module: Cell voltage and charge current vs. time.

6.2.2 BRIX$_2$ Extension Modules in Practice

In Chapter 5 we have described the development process and implementation details of the BRIX$_2$ extension modules. In the following, we present typical scenarios and measurement data for selected extension module as an addition to the technical specifications summed up in Section 6.1. Depending on the type of extension module, we focus on typical sensor output data, current consumption or individual characteristics. This way we aim to provide an overview about the capabilities and limitations of individual extension modules. However, we exclude some extension modules from this section because they are either rather simple and passive, like the button or the servo extension module or they are already covered by external documentation, like the MakeyMakey extension module.

AmbiSense Extension Module

The AmbiSense extension module allows users to measure ambient light levels, temperature and humidity. To give an impression of the capabilities and limitations of the AmbiSense extension, we set up two experiments. First, we tested the reaction time of the temperature and humidity sensor under rapidly changing ambient conditions. Second, we recorded the ambient brightness level using the light sensor on a BRIX$_2$ module attached to a persons wrist while walking around inside and outside a building.

Temperature and Humidity Measurements
If temperature and humidity data is logged in a mobile application, e.g. when the sensor is worn on the wrists, the conditions can rapidly change. For example if the person leaves a house in the winter, the temperature changes from around 22 °C to 5 °C instantly. To determine how quick the temperature and humidity sensor can adapt to such rapidly changing conditions, we recorded data at room temperature, placed the sensor outside the window for around 2 hours and then placed it back inside. The room temperature was around 22 °C and the outside temperature around 8 °C. From the plot in Figure 6.7 it becomes clear that it takes around 15 minutes until the sensor shows the correct reading after being placed outside at minute 6. At minute 119, the sensor is placed inside the room again and takes around 20 minutes until it shows a correct reading. The reason for this high reaction time is the thermal mass of the sensor and the surrounding electronics. Since the sensor does not measure the temperature of the air but merely the temperature of the die inside the device, its thermal mass causes a certain inertia in the temperature fluctuation. As a consequence we do not recommend the sensor for measurements of rapidly changing temperatures. However, it appears to pick up small variation of the outside temperature from minute 20 to 119. This shows that the device can be used to measure ambient temperature in a static scenario, for example to acquire weather data.

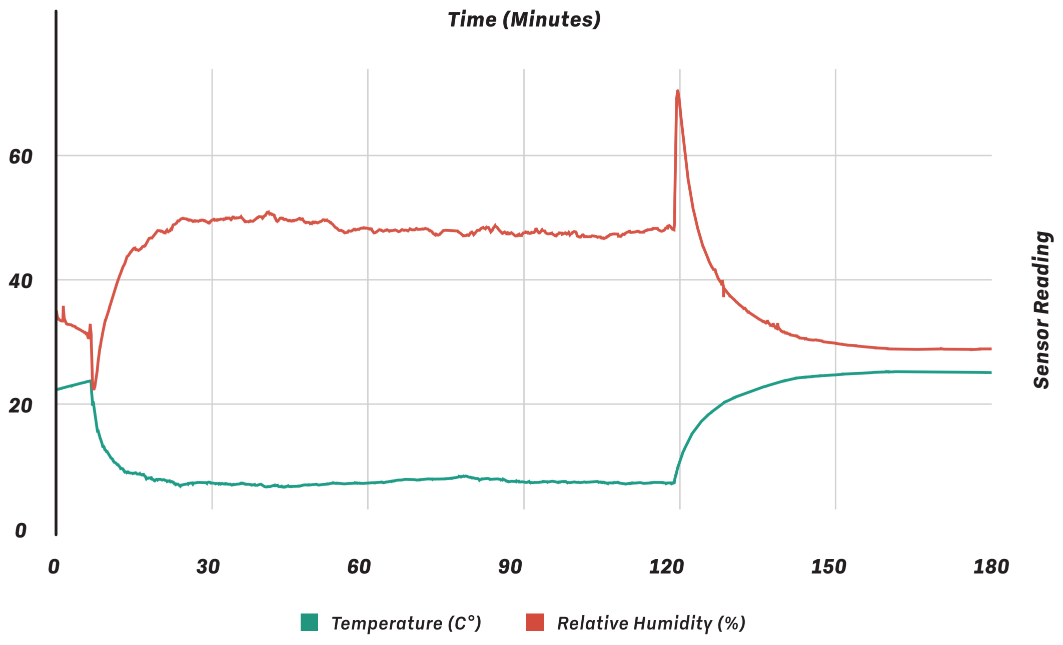


Figure 6.7: Reading of the temperature and humidity sensor under rapidly changing conditions.

Light Intensity Measurements

To test the light sensor of the AmbiSense extension we logged brightness data using a wrist-worn BRIX$_2$ module while walking around inside and outside a building. The walk starts in a bright corridor, proceeds through a relatively dark corridor until second 25. Then the person walks down 7 sets of stairs in alternating directions, which is visible in the brightness data, see plot in Figure 6.8, because of the windows on one side of the stairwell. At second 70 the person exits the building and walks outside in direct sunlight. The sensor is set to high-gain mode, which is why clipping occurs at this point. The person enters the building again at second 100 and proceeds through a dark corridor into another stairwell which is brightly lit by the sun through windows on one side. Again, we observe the alternating patterns of light intensity as the person walks up the 7 sets of stairs and enters a brightly lit corridor at second 210. The person walks on through a darker part of the corridor to the starting point. The experiment shows that the light sensor can be used not only to measure ambient light in a static scenario, but is also an instrument for activity measurement in mobile applications. Combined with other sensors, localization inside a building with known lighting conditions might also be possible.

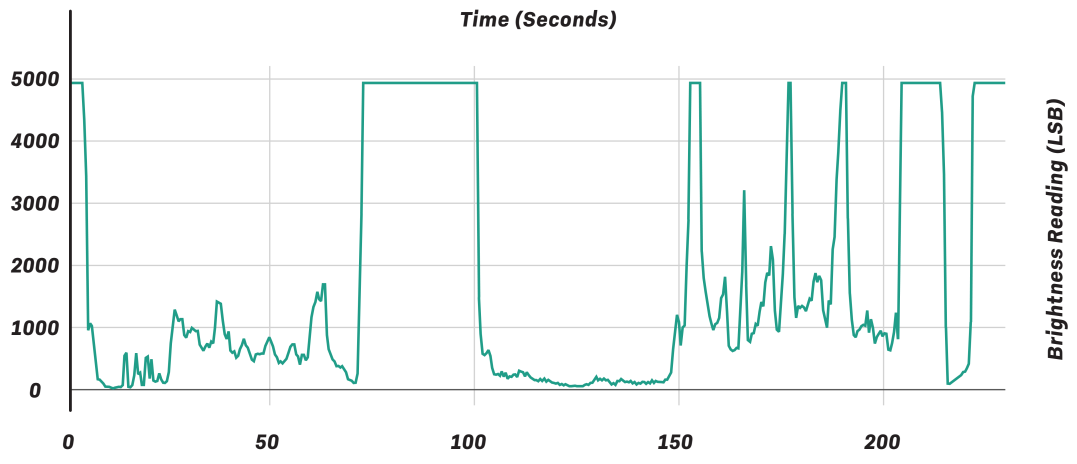


Figure 6.8: Brightness level measurements while walking around.

BTLE Extension Module

Equipped with a BTLE extension, BRIX$_2$ becomes compatible to a wireless standards available smartphones and tablets. The BRIX$_2$ GATT provides services that allow to read sensor data from the BRIX$_2$ module as well as to push data onto the device from a mobile App, for example to control the color of the RGB LED.

Power Consumption

BTLE communication is more complex than simple RF communication using the BRIX$_2$ wireless transceiver, so we only give an estimation of the average energy consumption of the BTLE extension module at this point. We distinguish three different states: advertising, connecting and communicating. While advertising, the BTLE extension sends out a short data packet which indicates its presence and available services. The transmission time for such a packet is 2.7 ms in which the transceiver consumes 13.5 mA [82]. At one advertising per second this calculates to an average current of 0.04 mA. When the BTLE extension maintains a connection to a mobile device, it sends out a connection event 100 times each second. With a current of 10.36 mA and a transmission time of 0.8 ms, this results in an average of 0.8 mA. As an example for the communication state, we suppose a scenario in which 16 bytes of payload are sent from BRIX$_2$ to a mobile app with a frequency of 100 Hz. The transmission time for each packet is 1 ms. At a supply current of 11.64 mA during transmission, we calculate an average of 1.2 mA.

Infrared Extension Module

To quantify the performance of the infrared extension module, we present measurements and calculations for transmission ranges and power consumption. Since both depend on different parameters, we only consider a single, exemplary scenario each.

Communication Ranges
Infrared transmissions as used by TVs and other entertainment electronics are unidirectional. There is no acknowledgment mechanism, so we can regard reception and transmission ranges separately. Our tests were performed with a Samsung television and the according remote control. To determine the transmission range of the $BRIX_2$ infrared extension, we sent commands to the TV and increased the distance until the signal was not received anymore. The result was a maximum distance for transmissions of 5.2 m. Other receiving devices might react differently. To determine the reception range, we used the Samsung remote control to send commands to $BRIX_2$. The signal could be received at distances up to 13 m. Again the result might be different with other remote controls.

Power Consumption
The TSOP6438TR infrared receiver on the infrared extension module is always active and draws an average current of 0.15 mA. The two transmitter LEDs are operated at a current of 130 mA each. If we transmit data encoded in the RC-5 standard, a single command is 14 bits long.[150] The transmission of each bit takes 1.778 ms. Half of that time, the signal is high. During this period, the LEDs are on but modulated so they are only active a third of that time. This calculates to duration of 5 ms per command in which the LEDs are actually powered. At 10 commands a second, the average current consumption is 1.3 mA. Actual applications however will most likely send commands event-based, so no continuous stream of commands will be transmitted.

Proximity Extension Module

In Section 5.3.2 we already pointed out that the proximity extension module is not a suitable sensor for measuring exact distances. To support this statement, we conducted an experiment in which we place the sensor in front of a white paper at varying distances with different lighting conditions.

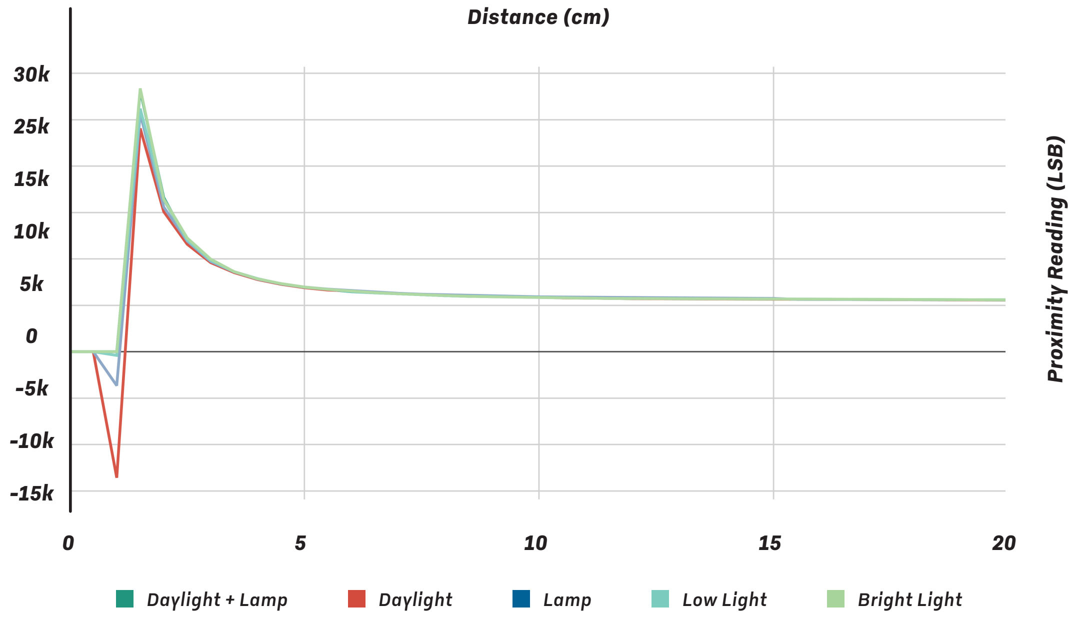


Figure 6.9: Proximity sensor data at varying distances and different lighting conditions.

In the plot in Figure 6.9 we can see that the characteristic of the sensor is hardly affected by the lighting conditions. The usable range of the sensor is from around 100 mm to 15 mm. Below the minimum distance, the sensor provides no useful data. Above the maximum distance, the output data does not change anymore. To demonstrate the temporal resolution of the sensor, we had a person wave all fingers of one hand across the sensor back and forth at a distance of around 30 mm. The sampling rate was 500 Hz.

The four fingers can be clearly identified in the plot two times, once each waving motion, see Figure 6.10. The high temporal resolution of the sensor could for example be used to identify different waving gestures with a varying number of fingers. By incorporating the distance of the single fingers, it is also possible to distinguish gestures in which the hand is tilted towards one side while waving, see Figure 6.10, 500 ms to 1500 ms.

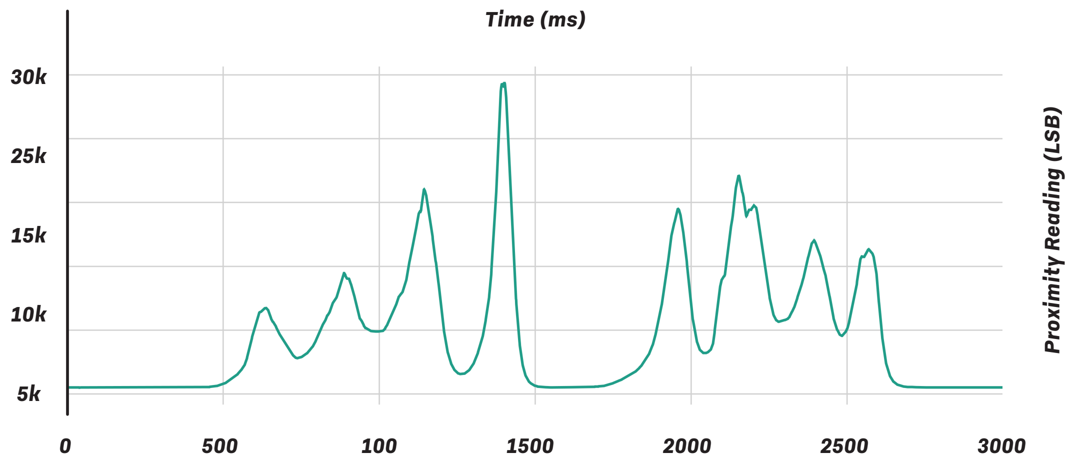


Figure 6.10: Waving a hand in front of the proximity sensor.

SD Card Extension Module

The SD card extension is mainly designed for data logging. In many scenarios, such an application is required to run several hours and sometimes even with high data recording rates. The power consumption of the SD card extension is mainly affected by two parameters: the type of SD card and the data rate. Peak currents are reached during writing operations. Depending on the type and brand of card, the supply current varies between 30ṁA and 80 mA for write operations and 15 mA to 30 mA for read operations. In our experiments we used a Verbatim 4 GB Micro SD card and the standard SD card library that is included in the Arduino IDE. We measured around 200 ms duration for a write operation at different payloads between 8 and 512 bytes. During that time, the card consumes 50 mA. Since flushing the data is not necessary for every data point, data can be buffered and written to the card for example once a second, which would lead to an average current consumption of around 10 mA.

VibroSound Extension Module

In this paragraph we present performance measurements of both aspects of the VibroSound module, vibration and sound generation.

Vibration Intensity In order to quantify the vibration strength of the VibroSound extension, we used the inbuilt accelerometer of the $BRIX_2$ base module the extension was connected to. The intensity can be adjusted in the sketch to a value between 0 and 255. We ramped up from minimum to maximum in around eight seconds and then ramped down again. The whole application was attached to a persons wrist using a custom strap. The VibroSound module was connected to the center extension header of the base module, directly above the sensor.

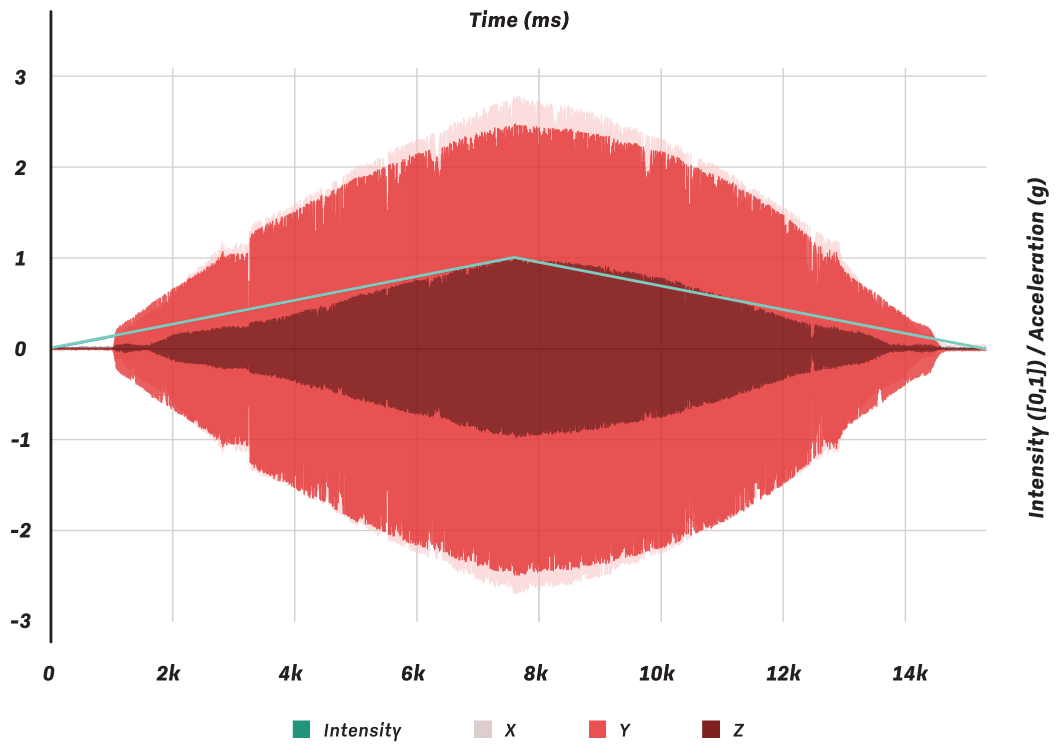


Figure 6.11: Vibration intensity measured by internal accelerometer.

In Figure 6.11 we can clearly see that the relation between intensity (normalized to the interval $[0, 1]$) and the acceleration is almost linear. On the x- and y-axis, the acceleration reaches a maximum of around $\pm$ 2.5 g. The z-axis is also affected, but only up to $\pm$ 1 g. This is caused by the orientation of the vibration motor in the x/y plane. However, the flexible surface the module was attached to most likely leads to crosstalking between the axes. It is also notable that the vibration motor only starts to move at around 15 % of the vibration intensity given in the firmware.

Sound Intensity The VibroSound extension uses a piezoelectric speaker which has a highly non-linear frequency response, meaning that the output volume depends on the pitch of the sound. The speaker on the VibroSound extension is only designed to generate simple alarm and notification sounds and not for actual audio playback. The LiBRIX$_2$ component for the VibroSound module allows users to set the pitch for a square wave sound output by the piezo speaker. We recorded a sweep of input values from 0 to 7000 in order to determine which frequency produces the most intense output. For recording we used a RØDE NT3 condenser microphone with an almost linear frequency response [151].

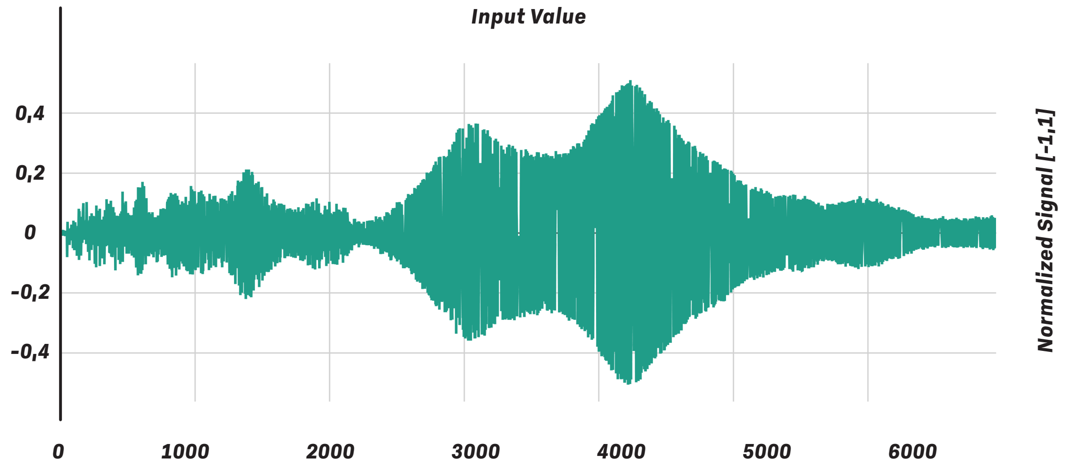


Figure 6.12: Frequency response of the piezo speaker.

Our plot of the frequency response, see Figure 6.12, indicates two main peaks at input values of 3000 and 4200 as well as a smaller peak at 1400. In order to generate the most audible signals, those input values should be considered. It is to note that the human hearing does not have a linear frequency response either, so the volume of a sound might be perceived differently by the human ear in contrast to a technical instrument. [152]

6.2.3 Conclusion

In this section we have presented some examples that demonstrate the capabilities and limits of the BRIX$_2$ base module and extension modules. Although we did not cover every single aspect of our platform, we provided a qualitative overview about its major functions that should help the reader to estimate the potential of the BRIX$_2$ platform. Though most components of our platform perform as we expected, others have room for improvement, like for example the temperature sensor. In the next section, we present our experiences with using BRIX$_2$ as a teaching platform in lectures as well as selected projects that were implemented using our toolkit.

6.3 BRIX$_2$ in Different Applications and Scenarios

BRIX$_2$ is designed as a teaching and prototyping platform for applications in different fields. In this section we report how our system performed in both of these use cases. First, we introduce three different lectures in which we used BRIX$_2$ as a teaching platform. We share our experiences regarding practical and didactic aspects and report opinions of the students. In the second part, we present different projects implemented by students and researchers. The selection of projects represents the fields we analyzed in Chapter 2, so we can evaluate whether our system is applicable in all of these fields.

6.3.1 BRIX$_2$ as a Teaching Platform

During and after the development of the BRIX$_2$ platform, we introduced the system in various lectures and workshop to students of different fields of study. They gained hands-on experience with the system while they learned about sensors, microcontrollers and physical computing. For us as developers, their feedback and experiences helped us to locate and fix flaws in our system and finally to determine if we met the goal of creating a valuable learning, teaching and prototyping platform. In this section, we share our experiences with using BRIX$_2$ for teaching in three different lectures. In the subsequent section, we discuss different projects of which some were implemented as exercises by the participants of those lectures.

Lecture and Workshop: Ambient Interfaces

"Ambient Interfaces" provides an overview of the field of ambient intelligence or ubiquitous computing. The students learn about a different aspect of the field each week, including physical computing and the BRIX$_2$ platform. In a 2-day workshop after the term, the students are supposed to design and implement a small application in the field of Ambient Intelligence. Most of them use the BRIX$_2$ system as a tool to realize their ideas. In the following we report our experiences with using BRIX$_2$ in the lecture before we introduce two of the projects as examples.
The group of students that attended the workshop had different fields of study. Some were computer science students whereas others studied sports science or design. Therefore the majority of the participants had only little or no previous knowledge about physical computing, microcontrollers or programming in general. At the beginning of the workshop, we gave a short, hands-on introduction to BRIX$_2$ and demonstrated the capabilities of the system. After the design process of their project, the students installed Arduino and LiBRIX$_2$ on their personal laptop and got familiar with the workflow. They tried some of the example sketches with a focus on their own application. Gradually they started to modify the example code and merging different examples until the sketch worked as they planned. In this phase,

questions regarding details of the programming language or the hardware platform arose and were either answered by the lecturers or the students found a solution on the Internet. Most of the projects that were built by students consist of multiple BRIX$_2$ base modules and extension modules. Frequently used features were the RF transceiver, the IMU, the proximity, VibroSound and AmbiSense extension. In almost all cases the students were able to achieve their goal in the given time. Many of them stated that they were surprised how easily they were able to implement their application without having previous knowledge about how to approach the problem. Some of the participants stated that the experience motivated and inspired them to use their newly gained skills in other projects beyond the scope of our workshop and lecture.

Lecture: Sensor Systems

This lecture is designed mainly for Master's students and provides theoretic knowledge as well as hands-on experiments with different sensors and sensor platforms. Among other platforms, students worked and learned with the BRIX$_2$ system in several of the lectures.
In one of the first lectures, the principle of a sensor platform, including sensors, microcontroller and a data interface was introduced using BRIX$_2$ as an example. In hands-on sessions, students were given some basic exercises to get familiar with the system and the principles. With Processing [2] and the Arduino IDE already installed on their PCs, they only had to install the LiBRIX$_2$ and tried some example codes. After that they were supposed to implement data streaming from BRIX$_2$ to a Processing sketch, the base for the following exercises. In later lectures on inertial sensors and sensor fusion, the students were given tasks like rotating a square on the screen using the gyroscope data. This worked as expected using a single integration of the angular velocity data. The next task was moving a square on the screen according to the movements of a BRIX$_2$ module on the table in two dimensions. In theory, this can be achieved by a double integration of the acceleration data. When they implemented this algorithm, the students quickly found out that in reality, noise and sensor drift lead to poor results. In our experience, learning by practically exploring the limits of a sensor is a much more memorable than just learning about it in a theoretical lecture, because it becomes a personal experience. BRIX$_2$ was also used to demonstrate the principle of reading resistive sensors like a thermistor using an RC circuit.
Despite some problems with the Arduino IDE under Microsoft Windows, mostly related to the virtual serial port, the students did not encounter technical problems. Again, the majority of class was surprised by how easy applications could be implemented and how accessible sensors can be using BRIX$_2$. The question how and

[2]https://processing.org

whether the modules can be obtained for personal use came up several times.
As part of the lecture, the students were supposed to do a project involving the hardware platforms introduced in the lecture. Some topics were already prepared by the lecturers, but ideas from students were also welcomed. Four groups of students decided to base their project on BRIX$_2$.

Lecture: Informationstechnik im Sport

The lecture *"Informationstechnik im Sport"* (Information Technology in Sports) was mainly aimed at students in the field of sports science. The topics ranged from technology used for motion and body activity tracking to signal processing. One of the lessons was a hands-on workshop with the BRIX$_2$ system. Most of the students had no programming experience at all, so the basics of programming were taught by the lecturer initially. Within two hours, all students were able to modify and combine the examples that are provided with the LiBRIX$_2$. The students were surprised that they could accomplish complex-looking physical computing tasks as for example moving the mouse pointer with their breath, using the humidity sensor on the AmbiSense extension, with no previous knowledge.

6.3.2 BRIX$_2$ as a Prototyping Platform in Research and Students Projects

After we have reported our experiences with BRIX$_2$ as a teaching and learning tool, this section is dedicated to prototyping, our second main application scenario. In Chapter 2 and 3 we have analyzed platforms from five different fields: microcontroller boards, physical computing platforms, wireless sensor motes, inertial measurement units and toolkits for wearable computing. We have designed our platform to be applicable as a teaching and prototyping toolkit in all those fields. In the meantime, BRIX$_2$ has been used in research, Bachelor's and Master's theses, and semester projects that were part of the lectures we presented in the previous section. In this section we take a look at some selected projects that are related to these five fields and illustrate how our platform performs as a prototyping tool.

BRIX$_2$ as an IMU: Measuring Body Postures with Low-Cost Inertial Sensors

In his thesis *"Ein inertiales Messsystem zur ganzkörperlichen Bewegungserfassung"* [153], a student explores the capabilities of inexpensive, IMU-based body motion trackers. As a contrast to costly systems like XSens (see Section 2.4.7), he developed an offline body tracking based on the BRIX$_2$ system. In his experiments, he attached up to 10 modules simultaneously to a wearer's body using custom flexible straps, see Figure 6.13. The modules were equipped with SD card extensions in order to store the

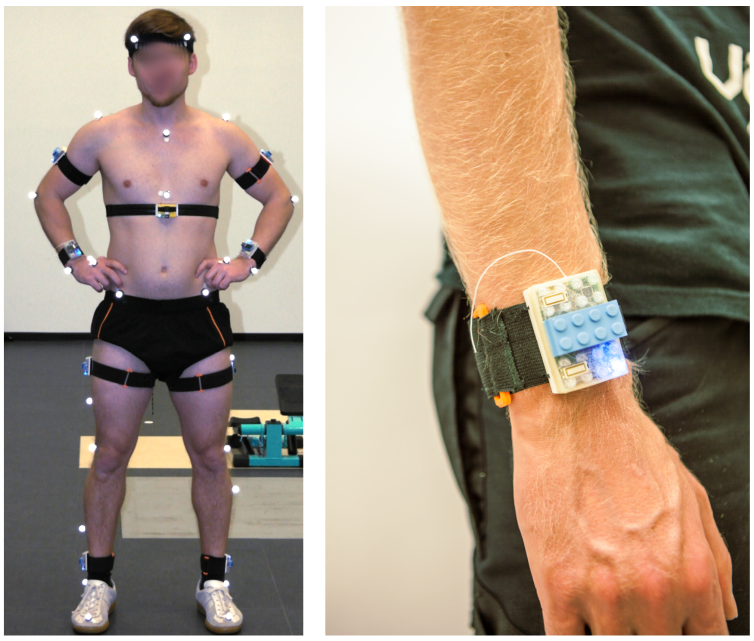
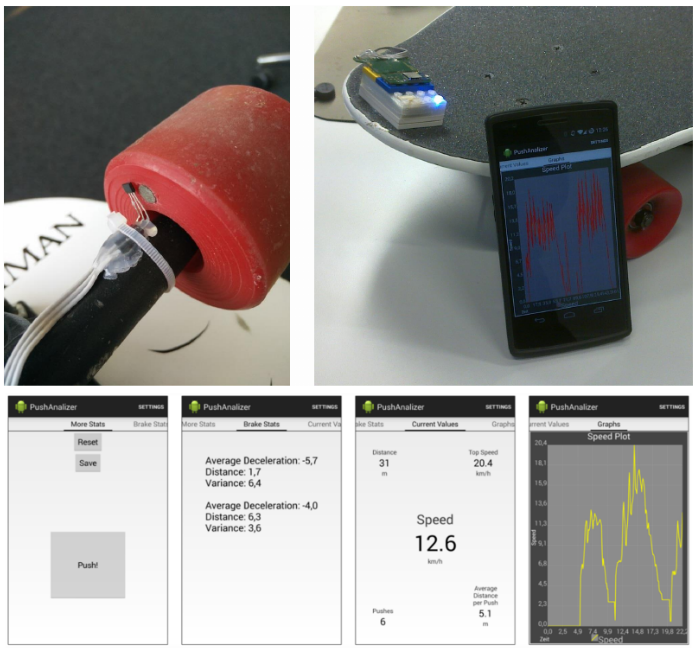


Figure 6.13: In motion tracking scenarios, BRIX$_2$ modules can be attached to the wearer's limbs using custom straps.

orientation data of each device. For this purpose, the student developed a firmware that allowed him to record data with a rate of 100 Hz to the SD card. The rate is mainly limited by the speed of the write-operations when storing data. While recording different motion patterns with the $BRIX_2$ system, he recorded motion data with a Vicon camera system at the same time as a ground truth for his later analysis. In postprocessing using different MatLab scripts, the student calculated the joint angles between the wearer's limbs from the orientation data recordings. With its compact design, a battery lifetime of around 5 hours and several GB of data storage on the SD card, $BRIX_2$ modules proved to be well suited for that kind of task. Using the RF interface and a modified version of the Matlab software, the system could also work wireless in real time. Mechanically, the modules were mounted on Lego plates that are sewn to the straps. The friction based connection was sufficient for regular body movements so no module fell off during the experiments. This project shows that the $BRIX_2$ system can well be used for IMU-based motion capturing.

Physical Computing: "Skate Analyzer"

Basic skateboarding skills include accelerating the board using one foot, referred to as *"pushing"* and stopping the board, again using one foot with a maneuver called *"footbrake"*. Especially the latter is crucial for a safe skateboard ride but often beginners tend not to practice the footbrake enough, which results in a lack of board control. As a project for the lecture *"Informationstechnik im Sport"*, the student implemented an application that uses a $BRIX_2$ module to measure the movement and speed of a skateboard and processes the data on a smartphone. Both components are connected via BTLE. The app on the phone rates and displays the efficiency of the user's push and footbrake maneuvers. The hardware consists of a $BRIX_2$ module attached to the top of a skateboard deck. A hall-effect sensor is placed next to one of the front wheels which contains a tiny magnet. The sensor is connected to the $BRIX_2$ module via a Breakout extension. A sketch on the user controller measures the rotation frequency of the wheel and the acceleration of the board. The BTLE extension allows to stream the data to a smartphone in the user's pocket, see Figure 6.14 (top). The app displays plots of the acceleration and speed of the board over time as well as current, average and top speed, traveled distance, average distance per push (a measure of the efficiency of a push) and details on the footbrake maneuvers, see Figure 6.14 (bottom).
The project was developed over several weeks and the student did not have any prior knowledge in microcontroller programming. An interesting side note is that the student needed precise timing for the hall sensor measurements which are beyond the standard Arduino programming language. In order to configure interrupts that trigger a counter and timers that measure the precise time of a wheel rotation, he consulted the datasheet of the Atmega32U4 and managed to implement those low level functions far below the Arduino abstraction layer. This example shows that



Average Deceleration: -5,7
Distance: 1,7
Variance: 6,4

Average Deceleration: -4,0
Distance: 6,3
Variance: 3,6

Figure 6.14: Skate Analyzer: Hall sensor and BRIX$_2$ attached to a skateboard (top) and the corresponding smartphone app (bottom).

users do not necessarily rely on the abstract Arduino language but can also even go down to the hardware level in order to implement timing-sensitive applications and gain absolute control over the controller.

Wireless Sensor Networks: Protocol Evaluations

This ongoing (as of 2016) project is dedicated to the evaluation of different wireless protocols as well as different wireless technologies for WSNs in order to develop an adaptive protocol that adjusts itself to changing networking conditions. In the first iteration of the project, the internal RF transceiver of BRIX$_2$ was used and multiple algorithms were implemented on the user controller. Different variants of the Time Division Multiple Access (TDMA) and Carrier Sense Multiple Access (CSMA) strategies were tested in a star network topology in different environments. The goal was to optimize the packet timing while maintaining an acceptable level of packet loss, thus maximizing the total throughput of the network. Further testing of those algorithms is planned for other wireless technologies such as BTLE and Ultra Wide Band (UWB). The internal CC1101 wireless transceiver of BRIX$_2$ facilitated the initial tests because of its simplicity. With almost direct access to the hardware layer, developers are able to precisely control every aspect of the wireless network without any limitation, obfuscation or abstraction caused by existing protocol layers such as Bluetooth or ZigBee.

Due to its modular design, BRIX$_2$ could be equipped with different sensor extension modules in order to test the networking protocols with real sensor data. The integrated battery and compact design of our platform allowed easy transportation and rapid deployment of the sensor network in different environments indoor and outdoor. Tests were performed for several hours without the necessity to recharge the batteries. Since BRIX$_2$ is relatively inexpensive compared to other WSN evaluation platforms such as the Waspmote (see Section 2.3.2), we were able to test networks with a higher number of nodes on the same budget.

Smart Objects: "Assisted Juggling"

Juggling requires two fundamental skills: hand-eye coordination and precise timing. One student addressed the latter in her BRIX$_2$ application that was implemented during a hands-on session of the "Ambient Interfaces" lecture. Her scenario was practicing the 3-ball-cascade [154], the most basic pattern for juggling three balls. It involves throwing the next ball into the air before the first one is caught again, thus keeping two balls in the air all the time. The goal was to let the balls tell the juggler when they need to be thrown by haptic feedback. The student used three BRIX$_2$ modules equipped with VibroSound extensions as juggling balls.

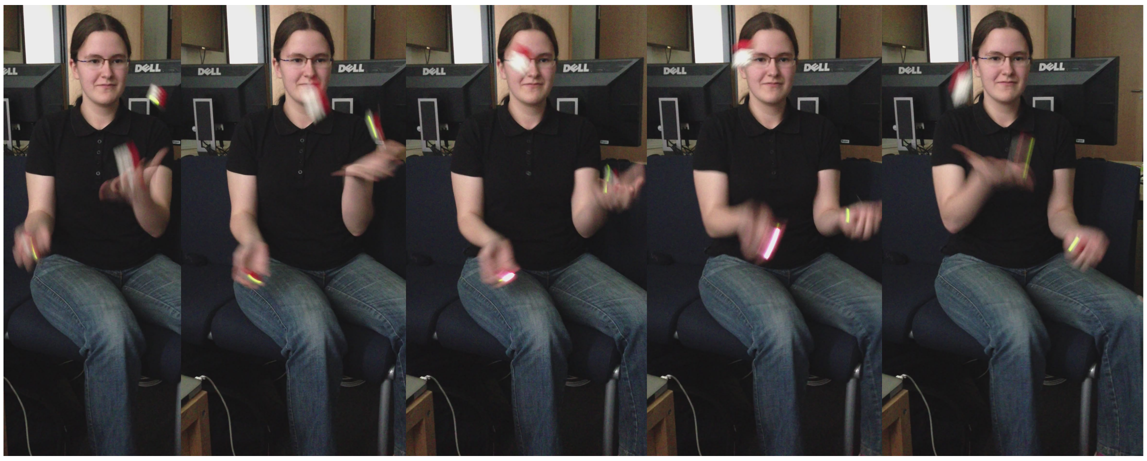


Figure 6.15: Juggling sequence: The second module lights up red and vibrates when it is supposed to be thrown.

She used the accelerometer on each module to estimate the apex of its trajectory when thrown. As soon as the acceleration drops to zero, one can assume that the juggling ball has left the hand of the thrower. Given a constant throwing heigth and a constant reaction time of the juggler, the correct moment to throw the next ball can be determined with sufficient precision. If that moment is reached, the module sends an RF message to the module in the ball that is supposed to be thrown next.

That particular juggling ball vibrates, telling the juggler to throw it in that exact moment. In Figure 6.15 we can see the module in her right hand lighting up red when the module in the air reaches its apex (third image). The red LED indicates vibration. As the module in the air descends, it stops sending data to the module that was now thrown, the red LED turns off (last image).
The BRIX$_2$ platform suits this application well, because it is lightweight and robust. Without any modifications, the base modules with two attached extensions could withstand the mechanical stress of juggling and occasionally dropping them on the floor. The compact design of BRIX$_2$ would even allow an integration into real juggling balls without any modification.
In contrast to other platforms that implement wireless communication through Bluetooth or WLAN, BRIX$_2$ modules can easily exchange data and information between multiple nodes. The basic and simple implementation of the wireless feature does not require a complex protocol stack or abstraction layers and is transparent to the user. This way the student was able to make three independent BRIX$_2$ base modules interact, despite having no previous knowledge regarding wireless communication or microcontroller programming.

Interactive Wearables: "BioSense"

When working in an office, the correct body posture is widely regarded as one of the keys to avoid back problems while sitting all day. In her project, which was part of the "Sensor Systems" lecture, one student developed an application that monitors and corrects the body posture of an office worker [155]. The correct body posture is in this case defined as maintaining legs at a 90 degree angle and an upright upper body. The student attached a BRIX$_2$ module including an SD card and VibroSound extension to the user's neck and lower leg using straps, see Figure 6.16.

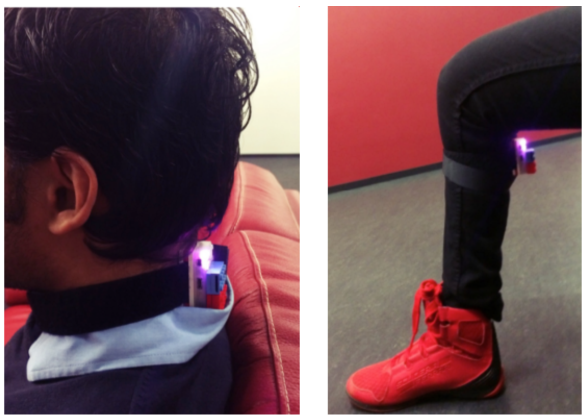


Figure 6.16: BioSense module placement: Sensors and actuators are mounted on the neck and the lower leg of the user.

In a first experiment, the student recorded the motion patterns of two participants working in an office during a period of 60 minutes. The data from both modules was written onto an SD card and showed when and for how long the user sat with an incorrect posture. In the second experiment, she applied a vibration feedback to either neck or leg if the user's posture was not correct. At the same time, the student recorded when and for how long the posture remained incorrect despite the feedback. Comparing the results of both runs, she concluded that feedback actually helped the participants to identify and correct a wrong posture.
In this application, the compact size and low weight of BRIX$_2$ allowed to attach them to a human body without restricting or hindering the wearers movements. Wireless operation is possible because of the internal battery. Using the SD card extension, the student was able to easily record motion data and transfer it to a computer for further evaluation. The VibroSound module allowed her to experiment with different types of feedback in this interactive application.

Smart Environments: "Weather to Go"

"Weather to Go" [156] is an auditory ambient display that informs users about the upcoming weather situation when leaving the room. Weather forecast data is presented as a soundscape which is played back on speakers close to the door as long as it is opened. In order to measure the opening angle of the door, a BRIX$_2$ module was attached to the door and connected via USB to the application, see Figure 6.17.

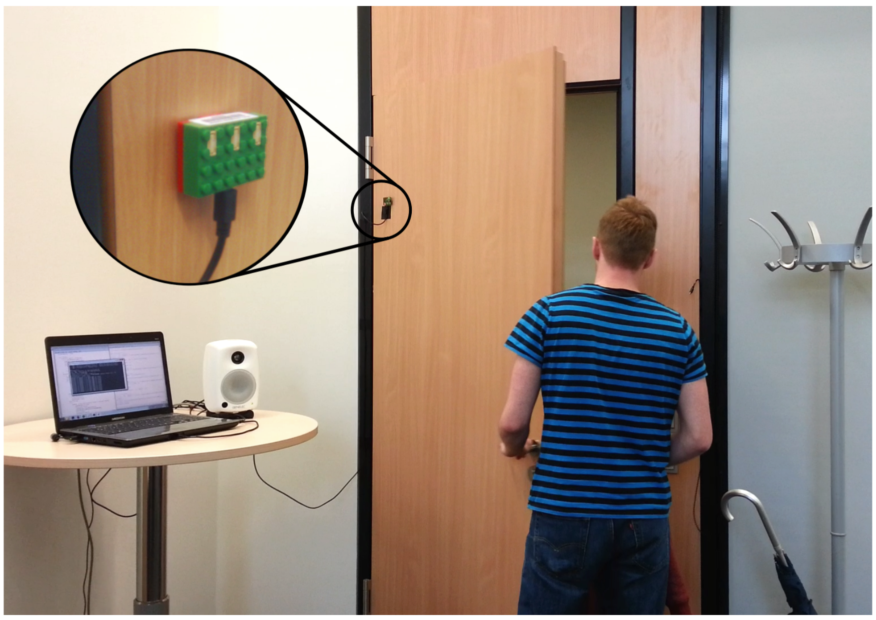


Figure 6.17: Weather to Go: A BRIX$_2$ modules is fixed to a door in order to measure the opening angle.

In the first version of *"Weather to Go"*, only gyroscope data was used to determine if the door is open or closed. Due to the gyroscope drift, this only worked for a limited number of opening and closing cycles. The stability of the system can be increased by also using the magnetometer or only the magnetometer. BRIX$_2$ was chosen for this application because it contains all necessary sensors, can easily installed on a door and only requires a single USB connection for data and power. It could also run off the battery with a wireless data connection to the host system. The quick installation process makes it possible to use the system outside the lab environment, for example in private homes during a long-term user study or during demonstrations at other research facilities.

Closed Feedback Body Area Network: The Haptic Belt

The goal of this project was to assist people with balance problems, for example stroke patients, to regain their natural body balance. The haptic belt, see Figure 6.18 (a), is equipped with haptic actuators (haptuators), placed at the four cardinal directions, close to the center-of-gravity. Sensors determine the posture of the upper body and the feedback provided by the haptuators supports the natural signals from the patient's vestibular system. In a training scenario, patients practice a challenging balance task and are supported by the haptic belt until their body learns to maintain balance on itself again.

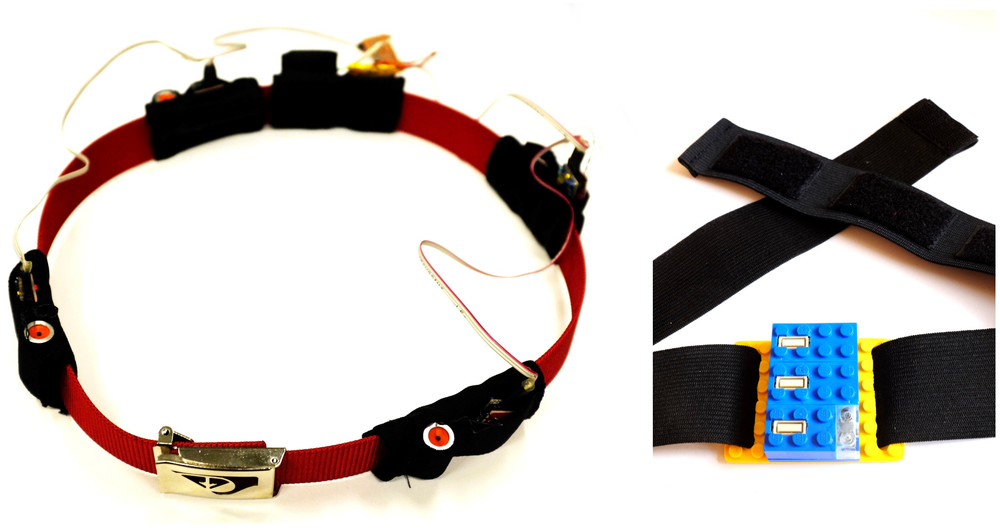

(a) (b)

Figure 6.18: The haptic belt (a) and the chest strap with a BRIX$_2$ module used as an orientation sensor (b).

The system consists of two parts: The haptic belt serves as an actuator and provides the haptic feedback to the wearer. A chest strap with a BRIX$_2$ module, see

Figure 6.18 (b), measures the posture of the upper body using the IMU and communicates with a second BRIX$_2$ module inside the belt wirelessly. The control signals for the haptuator are generated by the module inside the belt based on the orientation data and fed into the signal generator and amplifier circuits of the haptuators via UART. The haptic belt was designed and built by an experienced hardware developer and is a good example for BRIX$_2$ as a capable prototyping toolkit. Our platform combines all key aspects that are required in this application. The integrated motion sensor with DMP, a wireless interface to communicate within the BAN and a battery that allows standalone operation. Inside the belt, BRIX$_2$ can communicate through different interfaces, including I2C, UART and GPIOs. In addition to that, our platform is compact and light weight, so it can easily be integrated into body-worn devices which do not restrict the wearer's movements.

6.4 BRIX$_2$ User Survey

In order to assess the user acceptance of the BRIX$_2$ system and collect ideas for future improvements, we have conducted a survey among 20 users of the system. For this we designed a questionnaire (see CD-ROM) that we handed out to users after they had become familiar with BRIX$_2$, for example in the middle of a student project, after some lectures or at the end of a workshop. The survey is not intended to be a usability study or the like but a way to collect detailed feedback from our users. Since the group of participants is relatively small and homogeneous, their statements might not reflect a general opinion. In this section, we present a qualitative evaluation of the survey results and show selected comments of users.

6.4.1 The BRIX$_2$ User Survey Questionnaire

The questionnaire was handed out to the users in either German or English language, depending on their preference. In the following, we only refer to the English version. The document contains 59 questions, around half of them are to be answered in a Likert response format with five items, two positive, two negative, “don’t know”. We decided not to add a neutral element in order to force a decision if participants chose to share their opinion. The “don’t know” item was on the far right, separated by a line from the other items. The rest of the questions were either free text response format, an interval scale rating or a binary response format.

6.4.2 About the Survey Participants

The study was conducted among 20 people, 13 of them Master students, the rest Bachelor students, all of them in computer science. The average age of the participants was 24.6 years. 16 participants reported German as their first language, one English and three Hindi or Tamil. The majority of the participants identified as male (16), two as female and two did not specify.

Prior Knowledge

In order to evaluate the prior knowledge of the participants, we asked them to assess their level of experience with computers, electronics and microcontrollers. In addition, we asked for the programming languages they were experienced in as well as the operating systems they use. Most participants listed at least two programming languages. Only two did not list C or C++ as programming languages they are experienced in, but did list Java, which is closely related to C++ regarding the syntax. This means that all participants were familiar with the Arduino programming language, based on C and C++. Most participants named multiple operating systems they use: 18 listed Linux, 9 Windows and three OSX.

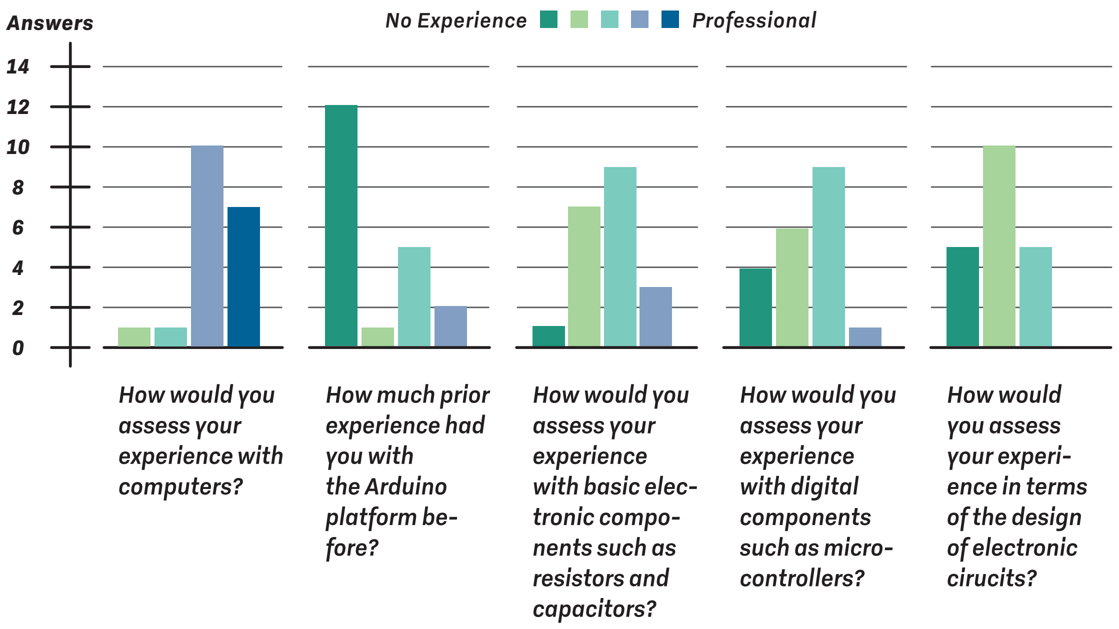


Figure 6.19: Self assessment of computer and electronic skills on a five-stage scale from professional to little or no experience.

Skill Levels

While all participants rated themselves as very experienced with computers in general, only 35 % of them reported to had hands-on experiences with the Arduino platform, see Figure 6.19. In the free text questions, 13 out of 20 participants stated that they had never used Arduino before. Knowledge on basic electronics like resistors and capacitors was a little more common than knowledge on for example microcontrollers. Most participants had no or only little experience with designing electronic circuits. In general we can conclude that while the software knowledge is quite profound among the participants, the knowledge about computer hardware and electronics is only basic.

6.4.3 Getting Started With Programming BRIX$_2$

After being asked for data on their person and their background, participants were asked to share their experiences with the setup of the Arduino IDE and the LiBRIX$_2$ as well as their thoughts on starting programming by trying and modifying the examples provided with LiBRIX$_2$. The first questions had to be answered in a Likert response format.

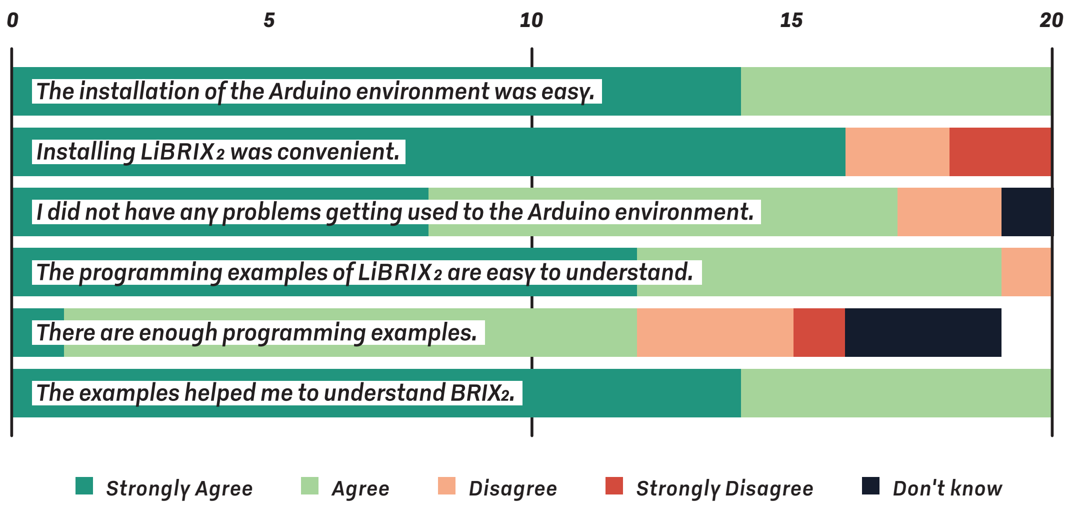


Figure 6.20: Survey results for questions regarding getting started with BRIX$_2$.

Setup

The results show that most participants rated the installation of the Arduino IDE as easy and without any problems, see Figure 6.20. Adding the LiBRIX$_2$ to the IDE was also simple for most users. The four negative ratings may be the result of confusion since the question was inverted in the survey ("Installing LiBRIX$_2$ was inconvenient"). In the free text questions on the LiBRIX$_2$, none of the users that rated negatively on this question provided comments on installation issues. According to our subjective experience, the majority of the survey participants got used to the Arduino IDE quite easily.

Programming Examples

Most participants reported that they were able to understand the programming examples without problems. At this point it should be noted again that all users in the survey had programming experiences and most of them listed C/C++ as programming languages they have worked with. The opinion on the number of programming examples is a bit more diverse. The original question was “I would have liked more programming examples.”. Along with this question, there was a free text question on topics that were missing dedicated programming examples. Users listed “The serial port”, “The Mozzi framework” and “The servo extension”. In response, a dedicated example and more details on communication via the serial port were added to the documentation. In general, the programming examples were rated as helpful. We can therefore conclude that no participant experienced any fundamental issues with LiBRIX$_2$.

6.4.4 BRIX$_2$ Hardware

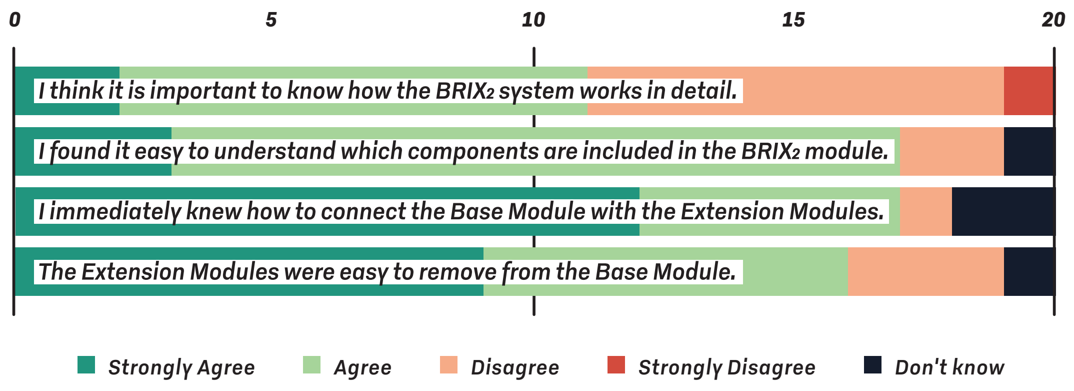


Figure 6.21: Survey results for questions regarding understanding and handling BRIX$_2$.

Only half of the participants considered it important to know how the system works in detail, see Figure 6.21. This indicates that a certain level of abstraction in the hardware and software is feasible, but all information should still be accessible and present. Ninety percent of the participants stated they found it easy to understand which components are included in the platform, information that is clearly stated on the BRIX$_2$ website. Regarding the handling of the BRIX$_2$ system, a fundamental action is connecting extension modules to and disconnecting them from the base module. Almost all participants stated they immediately knew how to connect extension modules to the base module, which proves that this aspect of the design is rather intuitive. In order to evaluate the quality of the mechanical connection, we asked whether the extension modules were easy to remove from the base module. 85 % of the participants agreed and 10 % disagreed. The question was formulated in reverse in the original questionnaire, so errors might have occurred while filling in the surveys. To find out which extension modules are most popular, we asked the participants which modules they used for their project. It is important to note that

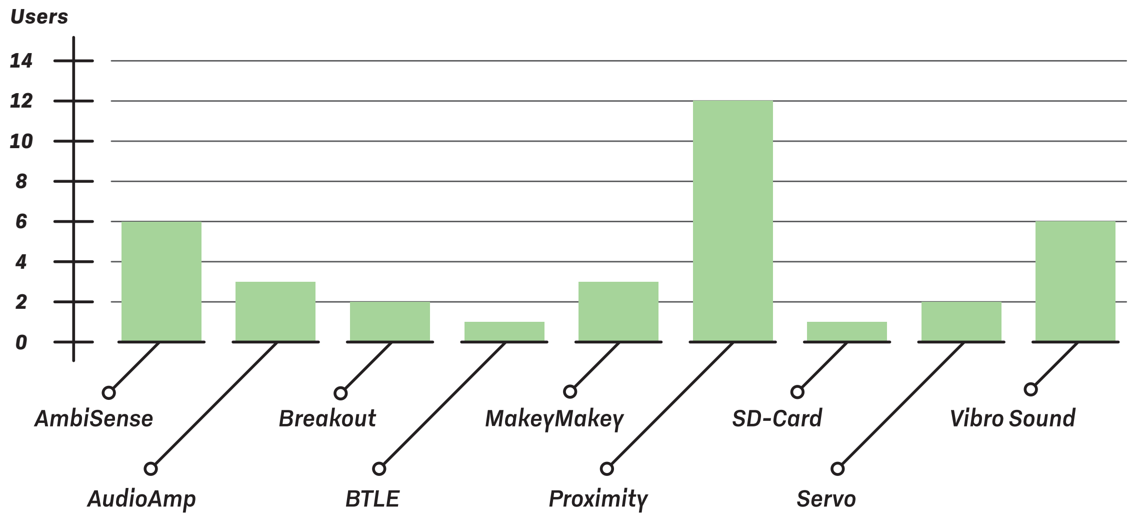


Figure 6.22: Which extension modules were used by the participants?

some participants filled out the questionnaire after doing a workshop or a lecture in which they could just experiment with the system and freely choose the components, while others used the platform for a certain purpose from the start on. This means they only used the extensions that were required in their application or instructed to use. However, in Figure 6.22, we can see that the AmbiSense and proximity modules were the most frequently used sensors whereas the VibroSound module was the most frequently used output extension. All three of those extensions are easy and intuitively to use. In contrast for example the BTLE module is rarely used, possibly because it involves a level of complexity, requiring a BTLE end device and a custom application on that device. When asked about technical difficulties with the base module or the extension modules, two participants reported issues with the virtual serial port, which sometimes disappears when the hardware is reset or reconnected. Regarding the extension modules, two users had difficulties maintaining the electrical connection to the VibroSound extension modules. These participants were using BRIX$_2$ modules with case design II (see Section: 4.3.3), which had a top plate that was too thick and thus did not allow proper electrical connections in some cases. This issue was solved with case design III, see Section 4.3.4. Please note that the button, potentiometer and infrared extension modules were not part of the survey because they were developed later, in response to demands from users.

6.4.5 Documentation

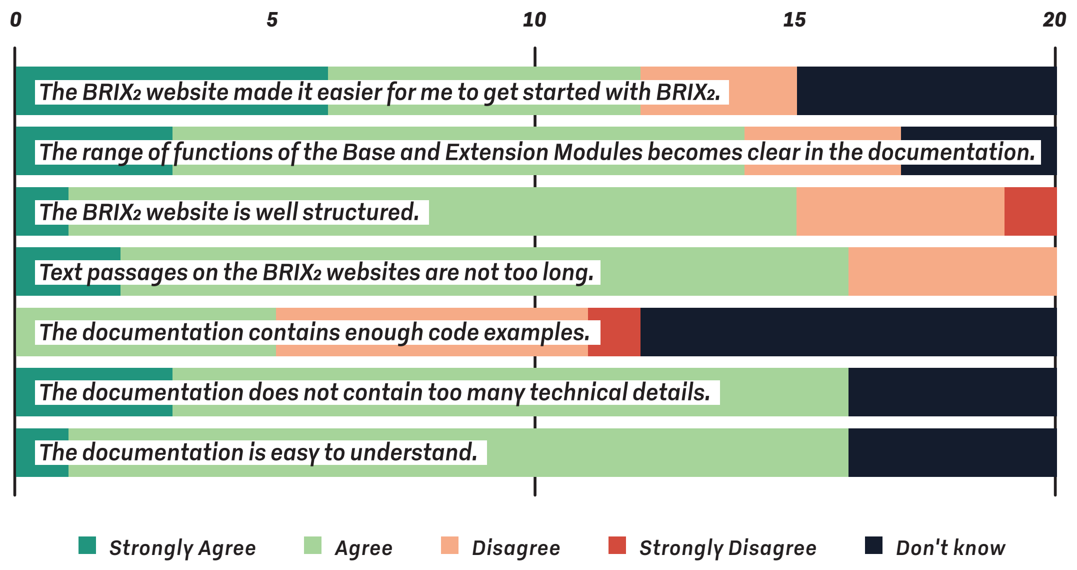


Figure 6.23: Survey results for questions regarding the BRIX$_2$ documentation.

To assess the quality of our documentation we asked the participants some general questions regarding this aspect, see Figure 6.23. The BRIX$_2$ website received mostly positive feedback from the participants. The majority of them agreed that the website had helped them to get started with the system and that the range of functions of BRIX$_2$ was clearly stated. In order to help more users getting started with BRIX$_2$, the *"Getting Started"* section of the website might have to be revised. 75 % of the participants stated that the website was well structured and the texts were not too long. The same number of participants agreed that the documentation did not contain too many details and was in general easy to understand.

6.4.6 Motivation and General Opinions

A crucial aspect of BRIX$_2$ as an educational tool is to motivate students to learn more about the field on their own. This is why we asked the participants about their personal experience with BRIX$_2$, see Figure 6.24. 95 % of them stated that working with BRIX$_2$ motivated them to learn more about physical computing. 75 % agreed that the experience with BRIX$_2$ got them interested in the functional principles of electronic devices in general. All participants stated that they find BRIX$_2$ easy to get started with and would also agree that the platform would improve lectures on physical computing, electronics and sensor systems. Please note that self-reports of survey participants only reflect a subjective opinion which might not be generalized.

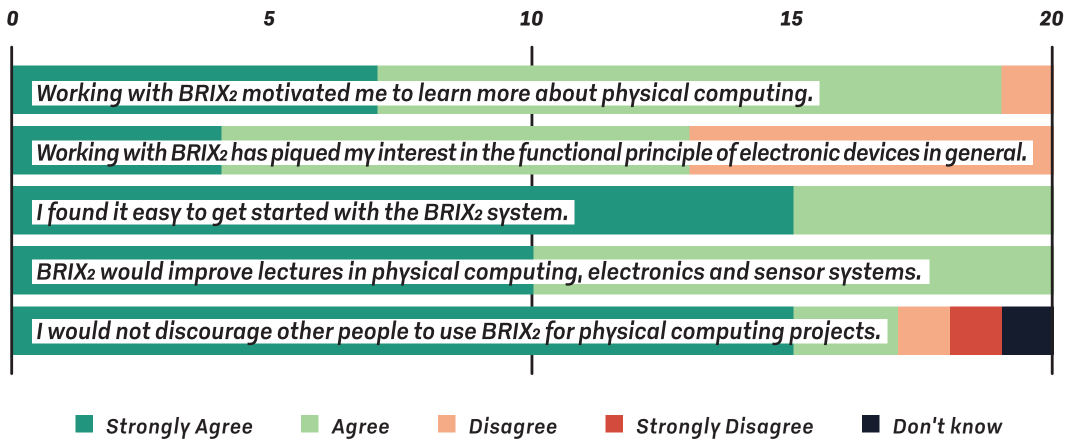


Figure 6.24: Personal experiences of the survey participants with BRIX$_2$.

Next we assessed how BRIX$_2$ inspired the participants. For that we asked them what they would do with the platform if they could use it for a longer period of time, see Figure 6.25. More than half of the participants stated that BRIX$_2$ could be a solution for an application they had had in mind before, but never implemented. Since the hardware knowledge of most participants is not profound (see Section 6.4.2), it seems plausible to conduct that the BRIX$_2$ system can empower users to attempt implementing applications that they did not have a solution for earlier. Working with BRIX$_2$ gave eighty percent of the participants ideas for new applications that they might implement in the future. Most participants stated they could imagine to apply BRIX$_2$ for their personal use as well as to base a student project, Bachelor's or Master's thesis on the platform. Please note that some of the participants were already working on such projects with BRIX$_2$ when they filled out the questionnaire. The free text answers to the question on projects that the participants would like to implement with BRIX$_2$ range from home automation to robots and quadcopters.

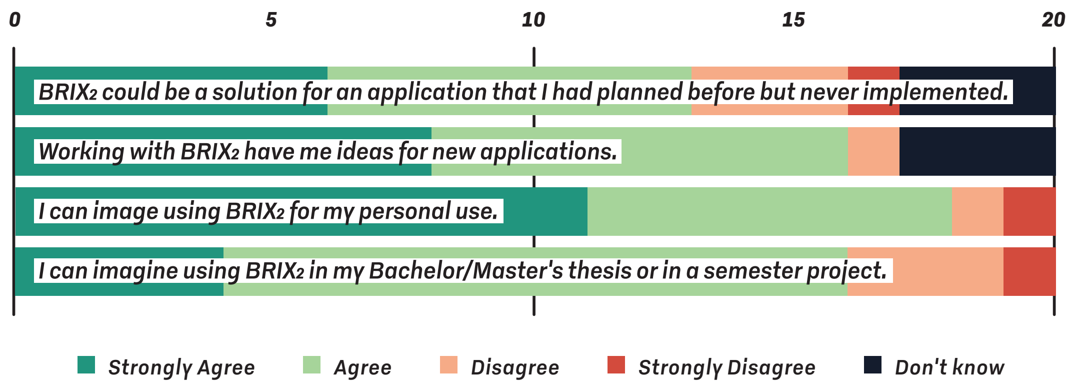


Figure 6.25: In how far did working with BRIX$_2$ inspire the participants?

6.4.7 Suggestions for Future Revisions

Since we are eager to include users into the design process for future BRIX$_2$ revisions, we asked the participants to propose additional features or changes. Potential extension modules that participants asked for were a (video) camera, a display extension with several LEDs, a loudspeaker module, a WLAN- and a button module. As a reaction to this, we created the button module and included a piezo speaker into the former vibration module, which then became the VibroSound module. The rest of the suggestions are part of our future work. When asked what changes might improve BRIX$_2$ in the future, the participants named more connectors, an option to connect extensions via cables, improvements on the IDE, more extension modules in general and a reset button on the base module. These are also issues we address in Section 7.2.

6.5 Conclusion

This chapter was separated into three main sections. In the first section, we summarized the technical specifications of all BRIX$_2$ hardware components including the extension modules. The data was partially taken from the datasheets of the components we used and partially resulted from our own measurements. In addition, we discussed how selected components of our system performed in real-life scenarios. These illustrative applications are supposed to provide the reader with a better sense of the capabilities of BRIX$_2$. We have presented for example a qualitative analysis of data recorded with different sensors as well as calculations of the performance of different components under varying conditions.
In the second section, we have shown examples of how we used BRIX$_2$ in teaching and prototyping scenarios. We reported our experiences with lectures in which we used our toolkit to teach ubiquitous and physical computing as well as sensor systems. BRIX$_2$ was in general received well by the students and in our opinion significantly increased their efficiency and motivation in the practical exercises of the lectures. Not only had students quick initial success, but also a long-term motivation to work with the system. Even students who never programmed before were able to develop applications beyond the scope of example programs we provided. Having discussed the capabilities of BRIX$_2$ as an educational tool, we subsequently regarded examples in which our system is used as a prototyping toolkit. We introduced different projects and applications of researchers and students that are based on our platform. The selection of these examples aims to represent the five different fields of applications that we introduced in Chapter 2 and thereby demonstrate that BRIX$_2$ can serve as a prototyping tool in all these fields, according to our hypothesis stated in Chapter 1.
The third section was dedicated to a qualitative analysis of a survey we conducted among users of our system by custom questionnaires. We asked 20 persons who had just used BRIX$_2$ about their experiences with our system and on their opinions regarding different aspects of our platform as well as about ideas for future revisions and extension modules. The majority of the participants reported that they found it easy to get started with BRIX$_2$ and the Arduino IDE and that they were able to use the hardware intuitively. Almost all participants stated that BRIX$_2$ would improve lectures in physical computing, electronics and sensor system and that working with our platform motivated them to learn more about physical computing. As a response to request from users that we extracted from the survey, we developed additional extension modules such as the button module, integrated further example sketches into the LiBRIX$_2$ and added details on different topics to the documentation.

7 Conclusion and Future Work

In this thesis we have described the design and implementation of the BRIX$_2$ system, a toolkit which on the one hand allows rapid and easy prototyping of ubiquitous computing applications and on the other hand is applicable for learning and teaching in the fields of physical computing, sensor systems and electrical engineering. We derived the design of our platform from a detailed survey of 28 different devices from the fields of microcontroller development boards, physical computing platforms, wireless sensor nodes, inertial measurement platforms and wearable electronics platforms. We expected that if a platform combined the principal capabilities of the devices we analyzed, it could be used as a prototyping and teaching tool in all five fields, which as a whole resemble the demands of ubiquitous computing as a field.

In order to integrate all required features into a single system, we decided for a modular approach. The functionalities that were most likely to be used in the majority of future applications were integrated into the base module. It represents the fundamental component of any BRIX$_2$ application and contains two microcontrollers, an inertial motion sensor, a wireless transceiver, a battery and three extension ports. The compact and lightweight design especially facilitates mobile and wearable applications. Further functionalities and features can be added by stacking extension modules onto the base module. This allows users to precisely match the hardware to the requirements of their application and thus reduces a potential overhead. As an initial kit, we developed 12 different extension modules containing sensors, actuators or communication interfaces. To keep our system easy to use, especially for beginners in programming or electronics, we designed BRIX$_2$ to be compatible to the Arduino IDE, a programming environment for Atmel AVR microcontrollers that is widely used among developers, students and hobbyists and therefore well documented.

In a first production run, we built around 40 base modules and more than 100 extension modules, which were used as a teaching platform in lectures on ambient interfaces and sensor systems. Our BRIX$_2$ modules also served as a prototyping tool in research and collaboration projects, student projects and Master's as well as Bachelor's theses. By presenting successful BRIX$_2$ projects from the five different fields of application we introduced in Chapter 2, we have demonstrated that our system can serve as a prototyping tool in all of them. Some of these projects were implemented by beginners with little experience in electronics and programming, others by experienced developers, which indicates that our platform can adapt to the demands of different kinds of users and thereby fulfills our expectation stated in Chapter 1. A survey we conducted among students and researchers who had used BRIX$_2$ in lectures and projects showed that our platform was well received. The participants considered it a useful tool for teaching and reported that working with our system piqued their interest in physical computing. Learning about microcontrollers and sensors with BRIX$_2$ empowered many of them to approach new projects they were not able to implement before.

In the following we conclude the contributions of our project to research and teaching before we present a brief outlook on potential future developments and applications.

7.1 Results and Contributions

In this section, we would like to summarize how our system supported and facilitated teaching and applied research especially at CITEC. We have separated the projects into two categories, teaching and prototyping, according to the two main fields of application we targeted with our platform.

7.1.1 BRIX$_2$ as a Teaching Platform

Up until now (2016) the BRIX$_2$ system was used in five lectures, two Master's theses and one Bachelor's thesis. On the one hand, we used the system in the lectures to introduce students to microcontrollers and physical computing. This initial knowledge and experience represents a significant empowerment, because even though the students only briefly worked with microcontrollers, sensors and actuators, they soon discovered the potential of this technology and the possibilities it offers them.
On the other hand, especially in the "Sensor Systems" lecture, our platform enabled the students to explore the characteristics and behavior of sensors in practice. Here, they quickly discovered discrepancies between idealized theoretical models of sensors and real world scenarios, for instance when they tried to calculate the position of an object based on the double integration of acceleration data. As a result they are now able to more precisely judge the capabilities of different sensors they plan to use in future projects.
The lectures were typically followed by projects in which students could pick an individual topic and use one of the platforms we introduced them to in order to implement an application or perform a study. BRIX$_2$ was preferably picked for mobile and wearable projects such as those we introduced in Section 6.3.2. We as lecturers were often surprised by the complexity of applications students implemented after working with our platform for only a few hours as their first experience with microcontrollers and sensors.

In the Master's thesis "Measuring Body Postures with low-cost Inertial Sensors", see Section 6.3.2, a student explored the potential of low cost inertial sensors for full body motion tracking. His experiments and implementation were based the BRIX$_2$ system as a wearable, programmable IMU that allowed him to pre-process and record motion data. In another Master's thesis, our platform was used to evaluate the BLE113 Bluetooth transceiver that is built into the BRIX$_2$ BTLE extension module as well as the BLE112 which was implemented as a wireless module for the B2DK. In the Bachelor's thesis "Weather to Go", see Section 6.3.2, a student used our platform as a sensor to measure the opening angle of a door.
As we could see in the results of our user survey, see Section 6.4, which was conducted mostly among students who had worked with BRIX$_2$ in lectures or projects, the vast majority described their experience with our platform as positive in general. All participants agreed to the question whether they found that BRIX$_2$ would improve lectures on physical computing and electronics. Judging from the personal feedback that we received from students after lectures, working with our platform was a valuable and exciting experience for them.

7.1.2 BRIX$_2$ as a Prototyping Toolkit

In CITEC large scale projects like the Cognitive Service Robotics Apartment (CSRA) [1], smart environments blend with smart objects and even robots, involving sensors, actuators and wireless communication. Humans can interact with this environment for instance through sensors and interactive devices worn on their body. BRIX$_2$ is well suited for prototyping in those scenarios by providing all the functionalities necessary for initial implementations. In the following we present a list of all projects at CITEC and in other institutes that our platform was used in.

Research and Student Projects

- **WSN Protocol Evaluation:** Multiple BRIX$_2$ modules were deployed as a WSN to test different routing strategies and protocols, see Section 6.3.2.
- **Haptic Belt:** A belt was equipped with haptic actuators ("haptuators") to help stroke patients maintain body balance. A BRIX$_2$ base module measured the pose of the upper body and controls the haptuators, see Section 6.3.2.
- **Haptic Shoe:** The follow-up approach of the haptic belt. Here, BRIX$_2$ was used during the prototyping phase.

[1]https://cit-ec.de/en/content/cognitive-service-robotics-apartment-ambient-host

- **Infrared Person Tracking:** A custom extension module equipped with an infrared blob tracking camera was used to track persons in a room. A BRIX$_2$ module streamed the blob positions as an HID to the application PC for increased performance.
- **Interaction Experiments:** In order to silently send signals between the control room and an experimenter without disturbing the participants, a pair of BRIX$_2$ modules turned hand motions on one device into light signals on the other, thus providing simple, bi-directional communication.
- **Sonified Aerobics:** A number of BRIX$_2$ modules were attached to a human body and streamed motion data wirelessly to a PC, which sonified the motions in real time, thus providing an auditory representation of the body posture and movements. [63]
- **UWB Evaluation:** To evaluate a novel UWB module, a custom extension module was developed and equipped with the device. Multiple BRIX$_2$ nodes allowed rapid deployment of different network configurations.
- **BTLE Evaluation:** In order to find out if the BTLE technology is feasible for wireless streaming of sensor data recorded on the bodys of athletes, our BTLE module (see Section 5.3.4) was used for experimentation and measurements. [82]
- **IMU Evaluation:** To compare the DMP performance of the MPU9150 integrated into BRIX$_2$ and the more modern Bosch BNO055 IMU, a custom extension module for the BNO055 was developed and allowed to record data with both sensors at the same time.
- **Weather to Go:** Our platform was used to measure the opening angle of a door in order to detect if somebody opens it. In an improved version, a proximity sensor underneath the handle helps to determine of the door is opened from the inside or the outside.
- **Analysis of Throwing Movements:** BRIX$_2$ enabled sport students to record and analyze characteristic acceleration patterns of a human when throwing a ball.
- **InfoPlant:** Multiple BRIX$_2$ modules were used in the first prototyping phase to test different sensing modalities and actuators for an augmented plant that can be used as an information display. [157]
- **BioSense:** BRIX$_2$ was used to measure the body posture of an office worker and provide haptic and auditory feedback in oder to correct the angle of the upper body towards a more healthy position, see Section 6.3.2.

- **Assisted Juggling:** Three BRIX$_2$ modules were used as smart juggling balls that provide beginners with a haptic signal that indicates the correct timing for throwing the next ball, see Section 6.3.2.
- **Measuring Body Postures with Low-Cost Inertial Sensors:** By distributing 10 BRIX$_2$ modules on a human body, it was possible to record full body movements of a person and compare it against a ground truth recorded with a camera tracking system. [153]
- **Skate Analyzer:** A skateboard was equipped with a BRIX$_2$ module which measured accelerations as well as wheel rotation and sent the data to a smartphone via BTLE for further processing and visualization, see Section 6.3.2.
- **Performance Evaluation of Wireless Sensor Systems:** Multiple BRIX$_2$ modules were used to evaluate data throughput rates in different wireless network topologies. [158]

Industry Collaborations

- **Innovative Bedienung für Smart-Home Komponenten (IBSKOM):** In this "it's OWL" project, BRIX$_2$ modules were used to build the first functional prototype of a novel, cube-shaped motion based control device for smart environments.
- **KogniDoor:** BRIX$_2$ was used during the prototyping phase for an intelligent door, equipped with sensors and actuators.

7.2 Visions and Future Work

In this section, we discuss different potential improvements and future developments of the BRIX$_2$ platform including novel extension modules and an updated base module. Furthermore we provide some long-term perspectives for our toolkit.

7.2.1 Suggestions for BRIX$_2$ Future Revisions

In Chapter 6, we already pointed out potential improvements for future revisions of our platform. In this section we briefly touch on the latest developments in technology and discuss how the next version of BRIX could profit from recent electronic components and manufacturing techniques.

Electronic Components

The rapid development in the sectors SoCs, MEMS devices and microcontrollers leads to the fact that an electronic device, which was up to date during design time might be outdated by the time it is implemented. Event though we chose the components for BRIX$_2$ carefully and selected modern devices like for example the MPU9150 shortly after their release, there are more powerful alternatives available on the market now, only three years later. For us as developers, this represents an ongoing challenge to stay up to date, but also means a constantly increasing number of potential applications that we can accomplish only through improvements of the technology that becomes available to us.
Since our platform is modular, improvements in technology do not force us to redesign the whole system but allows us to swap only parts of it. A novel base module would still be compatible to the existing and growing set of extension modules and vice versa. In the following we discuss some suggestions for components that could be integrated into future versions of the BRIX$_2$ base module.

Microcontroller Had we not listed Arduino compatibility as one of our key requirements while selecting a microcontroller for BRIX$_2$, we would probably have used a modern ARM processor, for example from the Atmel SMART series [2]. With much higher clock speeds, bigger memory and features like Direct Memory Access (DMA) or FPUs and a lower power consumption, these devices are in any regard superior to the 8 bit microcontrollers we used in our design. The latest Arduino products incorporate 32 bit microcontrollers like the ATSAMD21G18 [3], which would allow us to base a future BRIX$_2$ revision on a more powerful device while still maintaining the full Arduino compatibility.

[2]http://www.atmel.com/products/microcontrollers/arm/default.aspx
[3]https://www.arduino.cc/en/Main/ArduinoBoardZero

IMU Today Invensense is no longer the only manufacturer that offers integrated data fusion in IMU sensors. We already experimented with the Bosch BNO055, which is a more open device with regards to the fusion algorithm than the MPU9150 and easier to handle from the software side. Maxim Integrated released the MAX21100 in 2015, which also supports 9-DOF sensing and integrated motion data fusion. It is to be expected that MEMS IMUs will become increasingly capable in the next years.

Wireless Transceiver In the last years, wireless transceivers became less expensive and more compact, especially WLAN and BTLE devices. The ESP8266 WLAN SoC for example is available for less than 3 USD and not only includes the wireless transceiver, but also a low-power 32 bit application processor, which is supported by the Arduino IDE [4]. Still, if we re-designed the BRIX_2 base module today, we would have to re-evaluate which wireless protocol or technology is optimal for a general purpose platform and would have to accept a compromise. However, this problem can most likely be overcome in the future through Software Defined Radio (SDR) devices that can operate on an almost arbitrary carrier frequency and are therefore compatible to almost any wireless standard.

Enclosure Design

In Section 4.3.5 we already mentioned that recent advances in 3D printing and other rapid prototyping techniques allow for ever more precise, less expensive and quicker manufacturing of objects made from a growing variety of different materials. This will allow us to print friction based connections with almost the same precision than injection molding, so we would no longer need to incorporate Lego bricks into our designs. We could also move to different enclosure designs or accessories that for example integrate the BRIX_2 base module into a wristband while maintaining the concept of stackable extension modules at the same time.

Potential Extension Modules

Inspired by the feedback of users and by our own work on the BRIX_2 project, we had several ideas for additional extension modules. Some of these originated from users' requests, others are technology driven, meaning they incorporate novel components that were released after design time of BRIX_2. In the following we list some of these ideas for future extension modules.

[4]https://github.com/esp8266/arduino

Barometric Pressure Sensor These sensors are available as MEMS devices in compact packages and allow to measure the barometric pressure, which can then be translated to an altitude above sea level. This way the extension module would be possible to measure the absolute location of a BRIX$_2$ application in the z-axis within a centimeter range, given stable barometric pressure conditions.

GPS Extension Module This module would allow a satellite based localization in outdoor scenarios and allow for location based applications. Unfortunately we could so far not identify a GPS sensor that is small enough to be integrated in a standard BRIX$_2$ extension module.

Improved Proximity Sensor As an improvement for our popular proximity extension module, see Section 5.3.2, we could incorporate the Avago APDS-9960 proximity, ambient light, RGB and gesture sensor [159]. The device can not only measure proximity but also detect different touch-less gestures performed within range of the sensing element. The photo diodes inside the device can also be used to measure the intensity as well as the color of ambient light.

Infrared Camera Sensor Using the Panasonic AMG88 infrared array sensor [160], BRIX$_2$ can be equipped with a low-resolution infrared camera. The extension module would allow applications like spatially confined temperature measurements or person detection.

Multiplug and Cable Connectors In some cases the three extension module slots are not sufficient to attach all functionalities required by a specific application. Since our extension headers on the base module are parallel, we could build an extension that multiplies the number of available slots. Some users requested cable connectors in order to use extension modules with a certain distance from the base module. Both components could be part of a future BRIX$_2$ connector kit.

7.2.2 Long-Term Perspective

Besides technological advances for future BRIX$_2$ components, we do also consider different future perspectives for BRIX$_2$ as a project. In the following we briefly touch upon making our platform a product and our ideas for future applications in teaching with BRIX$_2$.

BRIX$_2$ as a Product

Although our platform is open source and other developers are able to build their own versions of our system, this still involves a certain effort, facilities and costs. In order to make BRIX$_2$ more available to other researchers and the public, a promising approach is making the platform an actual product that can be purchased as a fully functional unit. Other projects like littleBits (see Section 2.2.5) or Phidgets (see Section 2.2.4) are good examples for projects that started off in universities, became a product and are now well known in their target areas of application around the world. Since BRIX$_2$ is Arduino compatible, it could also be part of the Arduino AtHeart program [5] which would provide a higher visibility of our product and better integration into the Arduino community.

Further Potential for BRIX$_2$ as a Teaching Platform

While our platform was successfully used as a teaching and learning tool in some lectures, it might be interesting to center a whole lecture around the system. Students would not only be able to use BRIX$_2$ as a toolkit but could also actively take part in its development. They could for instance design, build and test further extension modules or software components. Not only would the students learn about system and hardware design but also BRIX$_2$ as a project would profit from novel ideas and additional hardware as well as software components. Especially in combination with the release of our toolkit as a product, students are likely to be motivated by the perspective of potentially designing a piece of technology that becomes available to the public.

[5]https://www.arduino.cc/en/ArduinoAtHeart/Products

Appendix

List of Abbreviations

ACK	ACKnowledge
ADC	Analog to Digital Converter
API	Application Programming Interface
B2DK	$BRIX_2$ Development Kit
BAN	Body Area Network
BGA	Ball Grid Array
BTLE	Bluetooth Low Energy
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CIT	Continuous Integrated Testing
CITEC	Center of Excellence in Cognitive Interaction Technology
CLK	Clock Signal
CNC	Computer Numerical Controlled
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSRA	Cognitive Service Robotics Apartment
CTS	Clear To Send
DC	Direct Current
DIP	Dual InLine Package
DMA	Direct Memory Access
DMP	Digital Motion Processor
DOF	Degrees-of-Freedom
DSP	Digital Signal Processing
DTR	Data Terminal Ready
EEPROM	Electrically Erasable Programmable Read Only Memory
FAQ	Frequently Asked Questions
FFC	Flat Flex Connector
FPGA	Field-Programmable Gate Array
FPU	Floating Point Unit
FSR	Force-Sensing Resistor
GATT	Generic Attribute Profile
GPIO	General Purpose I/O
GPS	Global Positioning System
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications
HCI	Human Computer Interfaces
HDMI	High Definition Multimedia Interface
HID	Human Interface Device
HSV	Hue, Saturation, Value

I2C	Inter-Integrated Circuit
I/O	Input/Output
IC	Integrated Circuit
IDC	Insulation-Displacement Connector
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
ISP	In System Programming
JTAG	Joint Test Action Group
LAN	Local Area Network
LED	Light Emitting Diode
LCC	Leadless Chip Carrier
LCD	Liquid Crystal Display
LGA	Land Grid Array
LiPoly	Lithium Polymer
LSB	Least Significant Bit
MAC	Media Access Control
MCU	Microcontroller Unit
MEMS	Micro-ElectroMechanical System
MIDI	Musical Instrument Digital Interface
MISO	Master In, Slave Out
MLF	MicroLeadFrame
MoCap	Motion Capturing
MOSI	Master Out, Slave In
NAK	Negative ACKnowledge
OSC	Open Sound Control
P2P	Peer to Peer
PAN	Personal Area Network
PC	Personal Computer
PCB	Printed Circuit Board
PIFA	Planar Inverted Folded Antenna
PLC	Product Life Cycle
PWM	Pulse Width Modulation
QFN	Quad Flat No leads
RAM	Random Access Memory
RF	Radio Frequency
RFID	Radio Frequency Identification
RGB	Red, Green and Blue
RSSI	Received Signal Strength Indicator
RTC	Real Time Clock
RX	Receive

SCL	Serial Clock Line
SDA	Serial Data Line
SDK	Software Development Kit
SDR	Software Defined Radio
SMT	Surface Mount Technology
SoC	System on Chip
SPI	Serial Peripheral Interface
SRAM	Static RAM
TDMA	Time Division Multiple Access
TQFP	Thin Quad Flat Package
TWI	Two Wire Interface
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UWB	Ultra Wide Band
VPU	Video Processing Unit
VQFN	Very Thin Quad Flat No leads
WLAN	Wireless LAN
WSN	Wireless Sensor Network

Bibliography

[1] I. Bernard Cohen. Howard Aiken: Portrait of a Computer Pioneer. page 329, 2000.

[2] Berkeley EECS Department, University of California. Mark D. Weiser Biography Page. 1999.

[3] Jan Anlauff, Tobias Großhauser, and Thomas Hermann. tacTiles. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*, page 591, New York, New York, USA, 2010. ACM Press.

[4] EIA. EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange. 1969.

[5] NXP Semiconductor. I2C Bus Specification and User Manual. page 64, 2014.

[6] Atmel. AVR910: In-System Programming. page 12, 2008.

[7] IEEE. IEEE Std 1149.7-2009. *IEEE Std 1149.7-2009*, pages 1–985, 2010.

[8] Intel, Compaq, Hewlett-packard, Microsoft, Lucent, Philips, and N E C. Universal Serial Bus Specification. *Group*, page 650, 2000.

[9] Atmel. Atmel AVR 8-bit and 32-bit Microcontrollers. 2016.

[10] Microchip. PIC Microcontrollers, Microchip Technology Inc. 2016.

[11] Parallax. BASIC Stamp, Parallax Inc. 2016.

[12] Mike Szczys. TI makes a big bid for the hobby market | Hackaday. 2010.

[13] Texas Instruments. MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller (Rev. J). page 76, 2011.

[14] Texas Instruments. TI LaunchPad - BoosterPacks. 2016.

[15] Texas Instruments. Build Your Own BoosterPack for TI LaunchPad.

[16] Hernando Barragan. Wiring: Prototyping Physical Interaction Design of the Academic Programme. 2004.

[17] Casey Reas and Benjamin Fry. Processing: A Learning Environment for Creating Interactive Web Graphics. *Proceedings of SIGGRAPH '03*, 2003.

[18] Hernando Barragan. The Untold History of Arduino, 2016.

[19] Motorola. SPI Block Guide. 2003.

[20] FTDI. FT232R USB UART IC. 2015.

[21] Arduino. Arduino Website. 2016.

[22] Eclipse. Eclipse - The Eclipse Foundation open source community website. 2016.

[23] Atmel. Atmel Studio. 2016.

[24] Arduino.com. Arduino Forum. 2015.

[25] Lauren Orsini. Arduino's Massimo Banzi: How We Helped Make The Maker Movement - ReadWrite. 2014.

[26] Wikipedia. List of Arduino boards and compatible systems. 2016.

[27] Alibaba. Arduino Pro Mini-Arduino Pro Mini Manufacturers, Suppliers and Exporters on Alibaba.com.

[28] Peter Jamieson. Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?

[29] Noelle Swan. The 'maker movement' creates D.I.Y. revolution - CSMonitor.com. 2014.

[30] Deloitte Center for the Edge Media and Maker. Impact of the Maker Movement. 2013.

[31] Wikipedia. Breadboard. 2016.

[32] Seeedstudio. Buy Grove - Starter Kit for Arduino [110060024] | Seeedstudio.

[33] Seeedstudio. Seeedstudio Wiki: Grove.

[34] Google Inc. Google Trends. 2016.

[35] S. Greenberg and C. Fitchett. Phidgets: easy development of physical interfaces through physical widgets. *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209 – 218, 2001.

[36] Ayah Bdeir. Electronics as Material : littleBits. *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 3–6, 2011.

[37] Littlebits. What is littleBits? 2016.

[38] LittleBitsbitLab. Github: LittleBits. 2015.

[39] Andy Schmeder Adrian Freed. Features and future of open sound control version 1.1.

[40] The MIDI Manufacturers Association. MIDI 1.0 Detailed Specification. page 88, 1995.

[41] Hermann Kopetz. Real-Time Systems. 2011.

[42] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 364–369, 2005.

[43] Business Wire. Crossbow Technology Releases TelosB Mote Platform; Crossbow Furthers Commitment to Provide Leading Edge Technology to the Research and University Community to Advance Wireless Sensor Network Development | Business Wire. 2005.

[44] Wikipedia Org. IEEE Std 802.15.4-2006. *IEEE Std 8021542006 Revision of IEEE Std 8021542003*, pages 0_1–305, 2006.

[45] Libelium. Waspmote Overview - Sensors, Wireless Protocols, Specifications | Libelium. 2016.

[46] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, mar 2002.

[47] Wikipedia. ISM Band on Wikipedia. 2016.

[48] Wikipedia. OSI model - Wikipedia. 2016.

[49] Texas Instruments. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. page 94, 2014.

[50] Invensense. MPU-9150 Product Specification. page 24, 2015.

[51] Invensense. ISZ-2510 Product Specification. page 52, 2012.

[52] Bosch Sensortec. BNO055 Intelligent 9-axis absolute orientation sensor. page 105, 2015.

[53] Xsens. MTi 10-series and MTi 100-series User Manual. page 83, 2014.

[54] YEI Technology. 3-Space Sensor Miniature Attitude & Heading Reference System User's Manual. 2014.

[55] Femtoduino. Kickstarter: IMUduino Wireless 3D motion, BLE, 10 DoF IMU, HTML5, Arduino by Femtoduino.com. 2014.

[56] Hillcrest Labs. BNO070: High Accuracy 9-axis System in Package for mobile, wearable, robotics, and IoT devices. 2015.

[57] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. The LilyPad Arduino. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 423, 2008.

[58] Eva-Sophie Katterfeldt, Nadine Dittert, and Heidi Schelhowe. EduWear. *Proceedings of the 8th International Conference on Interaction Design and Children - IDC '09*, page 9, jun 2009.

[59] M O'Mahony and SE Braddock-Clarke. Techno Textiles 2: Revolutionary Fabrics for Fashion and Design. jan 2005.

[60] Sebastian Zehe, Tobias Grosshauser, and Thomas Hermann. BRIX - An Easy-to-Use Modular Sensor and Actuator Prototyping Toolkit. *Tenth Annual IEEE International Conference on Pervasive Computing and Communications, Workshop Proceedings*, 2012.

[61] Tobias Grosshauser and Thomas Hermann. Sensor Fusion and Multi-Modal Feedback for Musical Instrument Learning and Teaching. pages 19–21, oct 2010.

[62] Tobias Grosshauser, Bettina Blaesing, Corinna Spieth, and Thomas Hermann. Wearable Sensor-Based Real-Time Sonification of Motion and Foot Pressure in Dance Teaching and Training. *Journal of the Audio Engineering Society*, 60(7/8), 2012.

[63] Thomas Hermann and Sebastian Zehe. Sonified Aerobics - Interactive Sonification of coordinated body movements. *The 17th Annual Conference on Auditory Display, Budapest, Hungary 20-24 June, 2011, Proceedings*, 2011.

[64] Hirose. DF-17 0.5mm Pitch Board to Board Connector. 2009.

[65] E.T. Jaynes and F.W. Cummings. Embedded Interaction - Interacting with the Internet of Things. *Proceedings of the IEEE*, 51(1):89 – 109, 2009.

[66] Phillip Torrone. Why the Arduino Won and Why It's Here to Stay - Make: DIY Projects and Ideas for Makers. 2011.

[67] Linus Akesson. Craft - An ATmega88 Video and Sound Demo. 2008.

[68] Sheueling Chang Shantz Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle. Cryptographic Hardware and Embedded Systems - CHES 2004. 3156, 2004.

[69] Wikipedia. List of Arduino Compatible Boards, 2015.

[70] Arduino. Arduino Products Website. 2016.

[71] Atmel. ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet. page 50, 2015.

[72] Arduino. Arduino 9 Axes Motion Shield, 2016.

[73] Memsic. MTS/MDA SENSOR, DATA ACQUISITION BOARDS, 2011.

[74] Adafruit Industries. FLORA 9-DOF Accelerometer/Gyroscope/Magnetometer, 2016.

[75] Seeedstudio. Xadow - IMU 9DOF | Seeedstudio, 2015.

[76] Honeywell. 3-Axis Digital Compass IC - HMC5883L. page 20, 2010.

[77] Wikipedia. Nickel–cadmium battery, 2016.

[78] Wikipedia. Energy Density, 2016.

[79] Wikipedia. AA battery, 2016.

[80] Wikipedia. AAA battery, 2016.

[81] Ltd Guangzhou Markyn Battery Co. Lithium-ion Rechargeable Cell Battery Datasheet, 2007.

[82] Michael Adams. Entwicklung und Evaluierung eines BLE Multitransmitterszenarios für ein Vitalparametermonitoring. Master's thesis, Universität Bielefeld, Bielefeld, Germany, 2015.

[83] Wang. LI-POLYMER BATTERY PACKS Specification. page 6, 2006.

[84] Peter Carpenter. Balancing Li-Po Battery Packs, 2002.

[85] IQ Technologies. How to rebuild a Li-Ion battery pack, 2008.

[86] Microchip. Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers. 2014.

[87] Sparkfun. LiPoly Charge Circuit, 2012.

[88] Microchip. 2.0 MHz, 500 mA Synchronous Buck Regulator. 2012.

[89] Invensense. MPU-6000 and MPU-6050 Product Specification, 2011.

[90] AKM. AK8975/AK8975C 3-axis Electronic Compass. 2010.

[91] Atmel. USB DFU Bootloader Datasheet. 2008.

[92] Gil Reiter. Wireless connectivity for the Internet of Things. Technical report, Texas Instruments, 2014.

[93] Michael Bailey. General Layout Guidelines for RF and Mixed-Signal PCBs. Technical report, 2011.

[94] Wikipedia. RF Module, 2016.

[95] Digi. RF Modules - Wireless Radio Transceivers, Transmitters & Receivers - Digi International, 2016.

[96] Atmel. ZigBit 2.4GHz Module with Dual Chip Antenna, 2016.

[97] Bruce A. Fette. RF Basics: Radio Propagation | EE Times, 2007.

[98] Mobileinfo. Frequency Band and Licensing Requirements For Broadband, 2001.

[99] Lou Frenzel. What's The Difference Between Bluetooth Low Energy And ANT?, 2012.

[100] Thomas Aasebo. Wireless Technologies, 2012.

[101] Thisisant.com. ANT Message Protocol and Usage. 2014.

[102] Dynastream Innovations Inc. AP2 RF Transceiver Module. 2012.

[103] Nordic Semiconductor. nRF24AP2 Single-chip ANT ultra-low power wireless network solution. 2010.

[104] Atmel. ZIGBIT 2.4GHZ WIRELESS MODULES ATZB-24-A2/B0. 2013.

[105] Drew Gislason. *Zigbee Wireless Networking*. 2008.

[106] Digi International. XBee ® /XBee-PRO ® RF Modules. *Product Manual v1.xEx-802.15.4 Protocol*, pages 1–69, 2009.

[107] Digi. XBee/XBee-PRO ZigBee RF Module. page 235, 2015.

[108] MaxStream. XBee/XBee-PRO OEM RF Modules Product Manual. 2007.

[109] Anaren. A1101R08C Anaren Integrated Radio Datasheet. 2013.

[110] Texas Instruments. CC1101 Low-Power Sub-1 GHz RF Transceiver. 2015.

[111] Atmel. AVR2051: SerialNet User Guide. 2015.

[112] Analog Devices. ADXL330 - Small, Low Power, 3-Axis MEMS Accelerometer. 2007.

[113] Anaren. A1101R08x User's Manual. 2012.

[114] Sparkfun. SparkFun WiFi Shield, 2015.

[115] Sparkfun. SparkFun PWM Shield, 2015.

[116] Arduino. Arduino Uno Schematic. 2012.

[117] TXC. SMD Oscillators TD Series. 2011.

[118] Zack Whittaker. Micro-USB to be universal EU phone charger, 2009.

[119] USB Implementers Forum. Universal Serial Bus Cables and Connectors Class Document. 2007.

[120] Texas Instruments. TPS2111 Autoswitching Power Mux Datasheet. 2002.

[121] Sebastian Zehe. BRIX2 Extension Modules - BRIX 2 - Research for Cognitive Interaction, 2015.

[122] Arduino. Arduino Playground - Shield Pin Usage, 2016.

[123] APEM. MJTP Series 6mm Tactile Switches, 2011.

[124] ACP Technologies. Carbon Potentiometers CA6, 2011.

[125] TAOS. TSL2560, TSL2561 Light-to-Digital Converter, 2009.

[126] Honeywell. Honeywell HumidIcon Digital Humidity/Temperature Sensors HIH6100 Series, 2015.

[127] Mike Grusin. TSL2561 Luminosity Sensor Hookup Guide - learn.sparkfun.com, 2013.

[128] Limor Fried. Using the TSL2561 Sensor | TSL2561 Luminosity Sensor | Adafruit Learning System, 2012.

[129] Sagar Sapkota. Fritzing Project – Arduino Light Sensor (TSL2561), 2015.

[130] David H. Hagan. Arduino library for the Honeywell HIH6130 Relative Humidity and Temperature Sensor, 2015.

[131] Vishay Semiconductors. Fully Integrated Proximity and Ambient Light Sensor with Infrared Emitter, I 2 C Interface, and Interrupt Function.

[132] Eric Rosenbaum. MaKey MaKey - An Invention Kit for Everyone, 2012.

[133] Jin Long Machinery. C1026B002F Coin Vibration Motor, 2009.

[134] ON Semiconductor. LV8413GP H-Bridge 2-Channel Motor Driver, 2013.

[135] EKULIT. SMD-P12A03 Transducer, 2013.

[136] Texas Instruments. TPA2005D1 1.4-W MONO Filter-Free Class-D Audio Power Amplifier, 2015.

[137] Lutz Lisseck. SimpleSDAudio – Hackerspace Ffm, 2015.

[138] THRh20. Arduino library for asynchronous playback of PCM/WAV files direct from SD card., 2015.

[139] Vishay Semiconductors. Infrared Emitting Diode, 950 nm, GaAs, 2009.

[140] Limor Fried. IR Remote Signals | IR Sensor | Adafruit Learning System, 2012.

[141] Vishay Semiconductors. TSOP62xx, TSOP64xx - IR Receiver Modules for Remote Control Systems, 2015.

[142] BlueGiga. BLE113 Preliminary Datasheet, 2013.

[143] Charlotte Seem. Design Note DN013 Programming Output Power on CC1101. 2007.

[144] David Kotz, Calvin Newport, Robert S Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *Proceedings of the MSWiM 2004*, 2004.

[145] S.L. Cotton and W.G. Scanlon. Characterization and Modeling of the Indoor Radio Channel at 868 MHz for a Mobile Bodyworn Wireless Personal Area Network. *Antennas and Wireless Propagation Letters*, 6(11):51–55, 2007.

[146] ROHM. SMLP34 Series PICOLED-RGB Datasheet, 2014.

[147] F W Campbell and R W Gubisch. Optical quality of the human eye. *The Journal of physiology*, 186(3):558–78, oct 1966.

[148] Wikipedia. Wikipedia: Pulse-Width Modulation, 2016.

[149] Kingbright. 1.6X0.8mm SMD Chip LED Lamp Datasheet, 2012.

[150] Stefan Buchgeher. RC5 (Dekodierung mit PIC-Mikrocontroller), 2004.

[151] Rode. NT3 Studio and Location Multi-Powered 3/4" Condenser Microphone, 2009.

[152] D W Robinson and R S Dadson. A re-determination of the equal-loudness relations for pure tones. *British Journal of Applied Physics*, 7(5):166–181, 2002.

[153] Mario Heinz. *Ein inertiales Messsystem zur ganzkörperlichen Bewegungserfassung.* Masters thesis, Bielefeld Unversity, 2015.

[154] Wikipedia. Wikipedia: Cascade (Juggling), 2016.

[155] Viswa Subramanian Sekar. BioSense. Technical report, 2015.

[156] René Tünnermann, Sebastian Zehe, Jacqueline Hemminghaus, and Thomas Hermann. Weather to Go - A Blended Sonification Application. 2014.

[157] Jan Hammerschmidt, Thomas Hermann, Alex Walender, and Niels Krömker. InfoPlant: Multimodal augmentation of plants for enhanced human-computer interaction. *Proceedings of the 6th Conference on Cognitive Infocommunications*, 2015.

[158] Nils Kucza and Jan Tlatlik. Performance Evaluation of Wireless Sensor Systems on the BRIX 2 Platform. Technical report, 2015.

[159] Avago. APDS-9960 Digital Proximity, Ambient Light, RGB and Gesture Sensor, 2013.

[160] Panasonic. Infrared Array Sensor Grid-EYE (AMG88), 2016.

Acknowledgments

I foremost thank Prof. Dr.-Ing. Ulrich Rückert and Dr. Thomas Hermann for their continuous support, without which BRIX$_2$ would not exist today. Being part of both their groups was truly enjoyable and gave me access to a broad range of expertise of many skilled colleagues. Their advice often helped me to overcome technical and scientific difficulties and allowed me to make the BRIX$_2$ system what it is today.
I especially thank Alexander Neumann for his immense help with getting BRIX$_2$ out into the Internet and Johannes Nathow for his tremendous support with beautiful figures and photographs.

Staying focused on a project like this for several years and finally writing a dissertation about it was a major challenge for me. Throughout this whole endeavor, I was lucky enough to be supported and encouraged by my friends and my family. Thank you so much for pushing me through the rough times and for enjoying the fun times with me.

Here's to you (in no particular order),

Keywan	Jessica	Jan	Anita
Steffi	Mum	Tobias	Jan
Uwe	Lara	Christale	Janina
Matthias	Judith	Kendall	Dad
Johannes	Jodi	Sabine	Barbara
Jörg	Adina	Max	Matthias