Delft University of Technology
Master's Thesis in Computer Science

Energy Saving in Bluetooth 4.0 Wireless Locks Connected via Mobile Phone to the Internet.

Tristan Timmermans

embedded software

TU Delft Delft University of Technology

Energy Saving in Bluetooth 4.0 Wireless Locks Connected via Mobile Phone to the Internet.

Master's Thesis in Embedded Systems

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Tristan Timmermans
t.t.m.timmermans@student.tudelft.nl

9th of July, 2014

Author
Tristan Timmermans (t.t.m.timmermans@student.tudelft.nl)
Title
Energy Saving in Bluetooth 4.0 Wireless Locks Connected via Mobile Phone to the Internet.
MSc presentation
23rd of July, 2014

Graduation Committee

prof. dr. K.G. Langendoen (chair)	Delft University of Technology
M.A. Zuñiga Zamalloa, PhD.	Delft University of Technology
dr.ir. J.A. Pouwelse	Delft University of Technology
ir. B. Goranov	Ubiqu Access B.V.

Abstract

In a modern environment users are more dependant on their ability to use their mobile phone for nearly everything. They communicate with almost any device and those devices provide nearly limitless access to for instance a watch or in new applications, a lock. Issues encountered with these devices are the almost always lack the ability to keep functioning for extended periods of time while having high transfer speeds and not a lot of space to store batteries. We present a means to keep a device operating for years with high transfer speeds in Bluetooth low energy connections, while functioning for year on end while connecting to a smartphone for access to the internet. To keep the solution working, high transfer speeds and minimal delays are needed as people are not willing to wait for extensive periods for a lock to open. This has to be done while operating for years with minimal space requirements.

Our design uses the the Bluetooth low energy (LE) specification to keep the device connected to the mobile phone. The phone is connected to both the device (a lock) via Bluetooth LE and the internet. While the Bluetooth LE specification is low power, it is still considerably wasteful if high transfer speeds are used. As the internet connections of mobile phones are still considerably slow (even HSDPA) in response time, we can use this in our advantage to save power. The delays in these connections can be used to put the device to sleep. We also maintain full compatibility with the Bluetooth LE standard and keep high transfer speed when needed. Our solution provides the best transfer speed combined with the lower power consumption of low transfer rates. This is done by dynamically throttling the delays between each radio wakeup event.

The method is tested for different delays expected in connection types like GPRS, UMTS and HSDPA. As a result device is able to operate for over 35% longer and save 144% power per message exchange without significant additional delays and without compromising the Bluetooth specification. It is also possible to use the technique without any low level access to the link layer which is preferable since it is not always possible to get access to the Bluetooth stack.

Preface

This thesis is a step towards longer lifetimes for embedded devices and my Masters Degree. This is especially important to companies building products which have to survive in the field with limited power supplies. The work presented here is part of, and created for Ubiqu B.V. in collaboration with the Embedded Software (ES) Group of the Delft University of Technology. I hope that this work provides an adequate improvement in usable devices so they can operate within a longer time frame than ever before.

First I would like to thank Boris from Ubiqu Access B.V. for his patience and support during the course of the development of the thesis. I would also like to thank Marco for his guidance and help. Of course I would also like to thank Koen from the Embedded Software Group of the Delft University of Technology and Guus, Rogier, Bart and Léon for their advice and help. Last, but not least, I thank my parents for their patience and unconditional support during the thesis and my Masters Degree.

Tristan Timmermans

Delft, The Netherlands
14th July 2014

Contents

Chapter 1

Introduction

Embedded devices can be used for multiple things: sensors, actuators, controller and combinations of these. The idea of wireless embedded devices is to connect them together either for communication amongst each other or to connect to a bigger or even global network of (wireless) devices in the Internet of things[1]. A key problem with embedded devices is power consumption: they use too much too fast to be able to sustain themselves for more than a few months. The goal of this master thesis is to show an application for wireless embedded devices and let them be able to operate in an enclosed environment which has restrictions on space and access to power. While in product development this also includes quality control of software and hardware these properties are not the focus of this thesis.

The thesis focusses on power usage in an embedded device in a specific application: A user (person) wants to open a door. To do this he uses his mobile phone. This works by letting the mobile phone communicate with a device, the qBox, via (regular) bluetooth. The qBox then communicates via the mobile phone to the internet and sends an open request to a server on the internet. The 'real' lock is connected to the qBox and is not actually directly controlled. This way a lock (or the qBox) can actually poll a server to receive access for user. One of the problems is the lengthy communication needed to verify the phone and user and give access to the qBox. This application is not limited to a lock though, as the device can control either a lock or anything else like an actuator.
As locks need to operate for extended periods of time without maintenance they need to be self sustaining in both power usage and software and hardware reliability. This is currently done by connecting a third party lock to the qBox which is controlled by the phone. This is however a third party solution and not preferable. Even these devices need to have their batteries renewed every 6 months. Power consumption is thus a restraint even when the qBox is connected to a wall socket. Figure 1.1 gives a short overview of the situation.

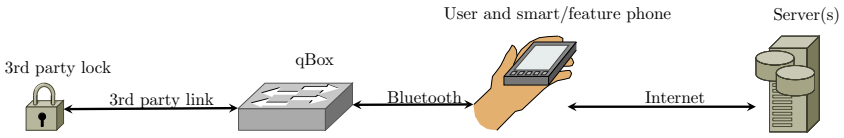

Figure 1.1: Devices and connections in a lock, qBox and phone setup.

The choice for this subject is related to my interests in embedded systems in general and finding a practical applications for wireless devices disconnected from the mains power[1]. The research in the field of power saving has been going on for some time over the past decades but the technology has only recently reached proportions for mass producing embedded devices with lifespans over a few months in the recent years. This can be seen as a challenge for me to device a way to apply most of the theoretical knowledge into a practical application.

The purpose of this thesis is to provide the reader with insight into power usage and restrictions in embedded wireless systems within a specific domain: A wireless, battery power lock. The device needs long operation (in years) and might be severely restricted in lifetime from frequent use.

1.1 Project background

Ubiqu Access B.V. (hence forth called Ubiqu) is a company specializing in secure (wireless) communication. One of the solutions Ubiqu provides is a link between your mobile phone and a digital lock inside a door or other device which needs access control. This has already been showed in Figure 1.1. A user with a device, in our case a phone, wants to enter a 'secure' area or have access to a secured device. In the current setup the user connects to a qBox (the device providing the access logic) and requests access. The qBox then requests access from a remote server. While it is possible to save access restrictions on the phone we want to be able to allow and/or deny the user access on the fly. The qBox communicates via the pohe to the server to request whether the user has access or not. After communicating the qBox either rejects or grants access by signalling the lock or an other device to open or operate.

A problem with this setup is that the qBox needs mains power. This is because it uses a bluetooth connection and has a secondary device to open a third party lock. This bluetooth connection uses a lot of power. Typically it would use around 100 to 250 mA continuous drain. This is due to the inefficient use of power by the Bluetooth 2.0[2] chipsets and the need for the

[1]Mains power is the 110/220/230v AC power out of regular outlets

use of third party transmitters. A solution would be to use an embedded device inside the lock as a stand alone device and get rid of the third party device. This way there would be no need for the separate qBox and we can then design the device to be sufficiently power efficient to have it operating for long periods of time without mains power. A simplified overview is given in Figure1.2.

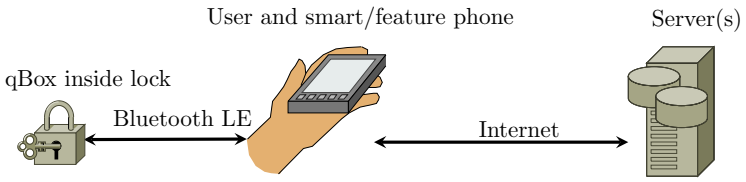

Figure 1.2: The device is inside a door and does not need any third party devices.

Since we want to use a low power wireless link, this also would mitigate the need for modifying the door to have a power supply. As it is usually not possible to modify the space inside the door or it is very expensive to do so. This is especially true if it would needed to be done in large numbers. Using a (very) small battery would be an option but even with good quality batteries life expectancy might be an issue as space is very limited. The device should be able to communicate with phones as those are the main source of identification and provide a link to the outside world for the device via the internet connection of the phone. This connection can thus also be used by an embedded device if the phone can communicate with the device directly. Another problem with low powered wireless connections however is that they either have very short windows of communication which gives quite high response times, or they lack sufficient throughput to enable the data to be sent quickly (in order of tenths of milliseconds). Normal, RFID based, digital in place replacements for modern doors/locks operate about 50000 times on a single battery. We should mimic this requirement with our own device.

1.2 Problem statement

The goal of this thesis is to develop a basis for a platform which can have both a sufficient throughput in data communication for human interaction and sufficient lifetime for low maintenance. The main aspects of this research will focus on the following properties of this Embedded System:

- **Communication throughput** The transmission ratios should be sufficient to communicate with the internet via a relay device (e.g. mobile phone or specifically designed device). Communication should occur

within several seconds (preferably 4, maximum 6) to let the user, the human factor, have the idea of instant updates/acception of his or her request. This has been shown to be the worst a person will accept in recent studies[3] and around the same time a person needs to get his/her keys. As this is dependant on radio communication it can be a major factor in wakeup cycles and radio activity and can increase power usage if not properly timed and controlled. A solution would be to start the connection ahead of the user selecting the lock. This however creates a problem when multiple devices are near: How to be sure which device the user want to use? Connecting to all devices in range comes at a great penalty as most protocols and protocol stacks do not support fast connections switching.

- **Peripheral power usage** MCUs usually have components connected to them and these components use power while active and might even have leak currents which draw a continues stream of power from the power source. These peripherals are usually not optional and thus need to be taken into account while designing the system. One should check for (hardware) related problems and enable/disable the peripherals as needed. It might be that the power usage of peripherals is outside of our control.

- **Peak current** Most battery powered systems and energy aggregators have a limited maximum current which can be drawn at any time. Due to usage of peripherals and MCU components the maximum peak current might exceed the capacity of the energy supply. This had to be taken into account by either hardware design or by systematic sequences of enabling devices to limit the drawn current. The maximum power capacity of the batteries should also be considered as well as the maximum current which can be drawn.

- **Overall system response** As the system will have to be used by people it should not behave in any way which would annoy the user. In this case it would require the lock to be opened in at least the same time as a user would open the lock by hand with a key or token. While it is related to device throughput it can be affected by the internal workings of the device as calculating lengthy adds to the response time.

- **Minimal space requirements** The device needs to be able to operate within an enclosed environment and should thus be limited in space needed for the electronics and power storage. While we do not know the exact requirements, smaller is better.

[4] These properties result in the following research questions:

- Is it technically (and possibly on industrial scale) and physically feasible to extend the lifetime of embedded wireless devices with limited battery power to years of operation?

  It can be seen that in theoretical terms as shown by Gomez et.al.[4] a lifetime of multiple years can be achieved for a C2032 button cell. This is however done with unusable response times of 32 seconds for a maximum 23 bytes of data. If devices requires several hundreds of bytes of communication this would take too long. On the other hand, the 1 MBit/s throughput of BTLE is only good for some days of operation is continuously used. Using a larger battery is going to provide a higher usage time but it also uses a lot more space which should be as small as possible. It should also be usable on industrial scale so overly complex hardware is not advisable and expensive. This would require simplicity, large margins of error and take into account producibility and operating conditions and not just theoretical application.
  There are also other communication methods which might enable long lifespans like ANT(+) and ZigBee and those can also be considered. The choice of communication methods is discussed in Section 2.1.

  Therefore we present these additional questions:

- Can a humanly acceptable throughput and response time be created while maintaining battery life for a device? This means a time from start to end in less than 6 seconds and sufficient throughput to and from the wireless device to send all data within that period. It is preferred to finish within 4 seconds.

- Can dynamic adjustments to the wireless device specifications increase battery life while maintaining response times in compliance with the specification of the wireless link?

- Are new or existing techniques needed for wireless communication within the frameworks provided and can they communicate with phones without losing compatibility, while limiting power consumption during communication?

- What is the influence of the 'hostile' environment (steel, wooden door, other devices on the same frequency band, etc.) on the power output of the transceiver and the signal strength to the receiver? Or is the influence of those parts negligible and can it be ignored?

In summary and in cooperation with the previous requirements the following functional requirements should be met:

- A response time within at most 4 to 6 seconds.

- No modification the the environment in which the device needs to operate (e.g. no power connections can be made) and the space needed should be a low as possible.
- The device needs to operate for at least two years. This way maintenance schedules are in line with or better than modern RFID locks which have advertised lifetime of 3 year **or** 30000 to 40000 readings.[]
- The device needs to be able to do 50000 open/close movements.
- The device needs to be able to communicate with a consumer phone directly.

1.3 Approach

First, the correct hardware should be chosen. There are several options which could fulfil the requirements of being able to communicate direct with peripherals and be able to measure voltages. Thus a specific hardware version has to be chosen. To get an idea of power consumption, and which parts of the code draw what kinds of current, I need to measure some baseline values and determine the most costly steps. A good indication is wireless communication[4]. Another part is the availability of power sources. If larger power sources are available the device has the ability to draw more peak current and have a longer operational lifetime as more power available at the same current draw means a longer lifetime.

The next steps needed are the evaluation of the most costly step and a solution to reduce power consumption in this step. If a solution can be found, detailed measurements have to be made to compare the solution with a given baseline measurement and compare the results. If sufficient, or at least indicating sufficient savings, a conclusion can be drawn about the power consumption, response time and the global lifetime.

1.4 Organisation

In Chapter 2 I will describe the available low power communication protocols available on mobile phones in Section 2.1 and in will show the available chipsets in Section 2.2. In Section 2.3 I will show additional power saving techniques. Section 2.4 will show what response times are acceptable to users.

Chapter 3 will give a more detailed description of the available hardware and Section 3.1.1 shows the available microcontrollers in more detail. Section 3.2 shows the final hardware selection.

In the following Chapter 4 I will do some exploratory measurements which shows power consumption and setup in Section 4.2 and 4.3.

Chapter 5 provides power estimations and a guess of the lifetime of the device. Section 5.1 and Subsections 5.1.1 and 5.1.2 give an overview of the power estimations an possible savings in hardware and software. Section 5.1.3 shows the cost of communication.

A more detailed analysis of the communication is given in Chapter 6. Section 6.2 shows the analysis and timing of the Bluetooth and all other network connections and 6.3 provides additional information to the use of connection intervals in Bluetooth low energy connections. Section 6.4 shows how to predict the delay in the entire network and use the connection interval in Bluetooth low energy connections to save power if we know the delay.

Results of all power and communication delays will be provided in Chapter 7 and the conclusion is given in Chapter 8.

Chapter 2

Related work

The related work chapter describe the use of wireless communication by mobile phones for uses besides calling in Section 2.1. This is extended into a short description of different communication protocols available for mobile phones in Section 2.1.1. In Section 2.2 a short description of Bluetooth Low Energy chipsets is given.

2.1 Wireless communication on mobile phones

Since we use a mobile phone as a (semi)transparent communication medium we need a method for connecting to the device. While mobile phones mainly use their GSM or newer equivalent for communication, these systems are not useful for inter device communication as they are both relatively high power[5] and do not support point-to-point links to peripherals. These forms of communication also use a vast amount of energy mostly due to the use of powerful amplifiers. While this might be manageable in the future[6, 7], it creates a power hungry device with the technology available on the market. There are more communication systems included in mobile phones in a wide variety of price ranges: most phones, even cheaper versions, support some version of the Bluetooth [2, 8, 9] specifications which gives any form of Bluetooth communication a broad adoption advantage. Most phones, especially the more expensive models, have most of the IEEE 802.11 standard[10] in one form or another (a/b/g/n/h) 802.11 can achieve good transfer rates and a greater range compared to Bluetooth at the expense of power usage and complexity.

Other options for low power communication are ZigBee[11, 12], ANT(+)[13] and Near Field Communication (NFC)[14, 15]. The ANT(+) platform was designed for fitness devices and displays, and is supported by some phones[13] but this is mainly by phones from Sony and Apple and only in high end devices, lacking the support for mass market which Bluetooth (LE) offers. Bluetooth LE support is present in most mobile operating systems

or will be added soon[16, 17] ZigBee is used in lots of wireless sensor networks but has little or no support for mobile phones and is thus limited to specificly designed devices to support (inter)net communication.

2.1.1 Communication protocols

The existing communication protocols have several up and downsides which limit their use in our application. Near field communication has a limited range and throughput[18] with a 20cm idealised range but an effective range of around 4 cm and a maximum of 424 KBit/s raw throughput rate. Another downside of NFC is the limited support on phones[19] of which most are expensive devices, but the adoption rate is increasing to lower end phones as well. Since it is likely the device would be covered and not have an open line of sight with the phone, short range is a problem when using NFC.
While ANT(+) has a range, power and speed comparable to Bluetooth LE (4.0), the adoption rate of ANT(+) is limited[13] even more than NFC. Since it has to be a commercially viable product, high adoption rates are needed and this limits the usefulness of ANT(+).
While the ZigBee protocol is very useful in low power communication, it is mostly used for device to device communication without user input. A major disadvantage of using ZigBee is the need for a commercial license. The throughput and range are comparable to Bluetooth LE (4.0) and ANT(+).

The only real candidate for low wireless communication in our application is Bluetooth LE (4.0). The transfer rates and power usage are comparable or at least as good as other communication methods and adoption is expected to grow[20] since dual mode devices are drop in replacement for older Bluetooth 2.0/3.0 modules in phones. Due to this 'drop-in' replacement possibility offered by some of the manufacturers of Bluetooth chipsets, higher adoption numbers are expected compared to when new chips have to be included in a design. These reasons leave it as the only viable option, as well as the experience of Ubiqu with the regular Bluetooth 1.2/2.0 specification.

2.2 Low power Bluetooth 4.0 Chipsets

Most low power chipsets have various peripherals, support a wide variety of interconnects, and have multiple on board logic and analogue devices. These devices can be seen as System on a Chip (SoC) devices or as Microcontrollers. Since the SoCs have lesser need for additional chips and interconnects both power and production space can be saved by using a single SoC instead of multiple chips. There are a wide variety of *relatively* low power chipsets in the range of micro ampères of power usage[21].There are simple 8bit architectures like the Inter 8051 architecture from the 1970s with micro ampère power consumption. There are also chipsets well into the ampère

range with multi-core mobile phone SoCs like the OMAP 5430[22] which can be used as Bluetooth cipsets. While the later have significantly more processing power and advanced digital signal processors (DSP), analogue to digital converters (ADC) and advanced transceivers, these are not required for most of the low power applications like sensors (networks) and remote control.

More details about possible Bluetooth chipsets is available in Chapter 3.

2.3 Power saving techniques for lower power wireless links

While it is possible to save power by using specialized hardware, it is possible to save power with software implementations as well. This can be limited to radio communication schemes but sometimes it is not possible to change the protocol or radio communication method.

In wireless sensor networks there are several techniques which can limit the power consumption of the entire network. Some of these specify specialized Media Access Controls (MAC) protocols which are limited either to Ad-hoc[23] or look to work on a greater scale[24]. They can look to minimize collisions, create efficient data processing and transportation or use data compression techniques to save radio on time[25].

Most of these schemes for power saving are not applicable in our case because they either require MAC layer access and modification of the radio standard. This is the case in Bluetooth communication when special transmission schemes are needed to limit the radio slots used. Another method proposed is the use of energy harvesting[26] which can be advantageous if sufficient power sources like mechanical-, solar- and thermal-sources are available. They do not actually save power but they reduce the consumption from a battery source by providing (limited) power themselves.

Power saving in devices which can communicate with a phone is problematic since communication standards need to be adhered. They mostly rely on Bluetooth to communicate with each other and due to the fast frequency hopping scheme of Bluetooth with windows of only 625μs it is difficult to achieve compatibility[27] when limiting the channels used. To achieve a better sleep duration the method proposed in [27] requires the use of the sniff-mode for listening slaves which is not always fully supported by all operating systems on phones. Others[28] try to modify the Link try to stay fully compatible with the Bluetooth standard by varying link layer properties of the specification with some success but note that these implementations and methods result different improvements on different chipsets[29] which limits

predictability.

Other methods include the analysis of the systems compiler[30, 31] and generation of instructions to asses locality of data and minimise the use of power hungry processor features. Also analysis can be done in memory access schemes[31] as they can consume power if done improperly.

2.4 Human response to devices activation

One of the aspect which requires some explanation is the need for a maximum communication time of 4 seconds. Studies[3] have shown that users waiting for a device opening a door have limited interest in waiting a long time for the device to respond. The typical time before a user switches, or longs for switching, back from an electronic activated phone/lock system to a normal key/lock is around 5 seconds maximum[3]. The selected four seconds response time is lower than the maximum set by the given studies, but the time measured was assuming the device gave no response during communication. If we do provide some feedback we assume the time a user is willing to wait can be stretched to six seconds.

Studies[32] have shown that the time in which a human perceives an operation as continuous, has no delays between feedback moments longer than 1 second. An example of this is a moving progress bar which updates to a higher number at least once a second. The same study also shows that a human begins to alter his or her perception if the retrieval of a web page takes longer than 2-4 seconds. While we are not retrieving a web page but controlling a physical object, and the given time for those is longer[3]. It might be good to keep the 4-6 seconds range as maximum communication time, assuming the user only gets feedback at the start ('opening') and end ('opening the door') of the sequence. Providing additional feedback, like some indication the device has heard something from the server, can stretch the positive perception longer.

Chapter 3

Hardware selection

This chapter introduces the related connectivity needed to perform all functions needed for the device to operate as a Ubiqu qBox or endpoint. In section 3.1 the needed peripherals and their requirements will be mentioned and explained. In Section 3.1.1 the available microcontrollers for the already mentioned communication methods will be given and in Section 3.2 a short conclusion will be given for the most effective solution.

3.1 Connected peripherals

The hardware requirements are based on support of connected peripherals as sufficient Input/Output ports need to be available for control and adc's need to be available for measurements.

- A servo for controlling the lock.
- One ADC for measuring the voltage drop of the controlled servo.
- Three LEDs for debugging purposes. These could be omitted in production environments.
- A buzzer for user feedback (piezo electric).
- Two I/O ports for activating the servo (not actively directly powered but switched).
- (optional) A UART (SPP/I^2C) for communication to a additional security chip.

The servo has an average power rating of 40 mA at 3.3 Volt and a peak start up power of 150 mA. This is beyond anything which can be powered by most microcontrollers on (gp)IO-ports and they need to be switched. This can be done by a FET and the voltage of the servos is measured by one of

the ADCs on the microcontroller. This is done to keep track of the health of the servo and check for the need of replacement. This data has to be communicated back to a central device or the mobile phone used to control the device.

3.1.1 Microcontrollers

There are multiple bluetooth microcontrollers available who support bluetooth low energy like the TI254x series and the nRF51822. We will discuss two in more detail in Section 3.1.1. They mainly come in two types: A multi protocol version with full (regular) bluetooth compatibility and stand alone bluetooth 4.0 modules who only support the low energy standard. Examples of the dual protocol modules are the TI256x[33] or even a multi tranceiver chipset as the Broadcom BCM43341[34] but they generally use an orders of magnitude more current and thus power at standard operations and sleep mode at battery voltage (around 3V) and are therefore not a good candidate for low power solutions which do not specificly need any other than Bluetooth LE support.

There are two main chipsets available for low power Bluetooth LE: the CC254x series[35] from Texas Instruments and the nRF51822[36] from Nordic Semiconductor. The CC254x and nRF51822 will be discussed in Section 3.1.1.

The 8051 microcontroller based packages

The 8051 microcontroller is an old beast from the 80's[37] an has seen widespread implementation in almost anything requiring a microcontroller and its predecessor and architecture can even be found in every PC's keyboardcontroller. For Bluetooth LE applications this microcontroller has the advantage of being implemented in many different modules which do not require additional shielding and regulatory checks. There are multiple suppliers for these chipsets and this has the advantage that one does not have to rely on one supplier.

As supply we selected three chips based on the Texas Instruments CC254x:

- The BlueRadios BR-LE4.0-S3A with a TI CC2541[38]
- The Bluegiga BLE112 with a TI CC2540[39]
- The Bluegiga BLE113 with a TI CC2541[40]

The difference between the devices are subtle. They can be replaced by each other with minimal adaptation to the linker and memory layout scripts and have the same Bluetooth LE stack available from Texas Instruments. Both chips and modules have sufficient GPIO ports available for driving LEDs and activating the servos and have at least two ADCs available next

to the internal battery monitoring ADC. The compiler provided by IAR[1] is capable of handling 8bit, 16bit and 32bit operations and can emulate those in 8bit instructions if necessary which creates easier development. Both chips have SPP busses available and can handle I2C either via hardware or by a bitbanging (e.g. software implementation) driver as can be seen in Table 3.1.

A major advantage of the TI254x chips is the size of the consumer base and it being based on other previous 8051 chipsets which limits the amount hardware errata needed and possible errors in production. The modules provided by BlueRadios and Bluegiga are both certified which significantly speeds up production time and lowers production cost as no FCC/ECC certification is needed for distribution. Some countries (e.g. France) do need testing and verification or cryptography included in the software but this is independent of the chipset used.

Type	LE4.0-S3A	BLE112	BLE113
PIO (4 mA)	17	17	17
PIO (20 mA)	2	2	2
Serial communication[1]	I^2C, UART, SPI	UART, SPI	I^2C, UART, SPI
ADC[1]	Max. 7	Max 7.	Max. 7
Timer output[1]	3	3	3
V_{dd}	1	4 (including USB)	2
Ground	5	4	10
No-connect	2	0	2
Other	Reset, RF Ground and Antenna	Reset, RF Ground and Antenna	Reset, RF Ground and Antenna

Table 3.1: S3A and BLE122 I/O overview.
[1]Shared with PIO ports.

The Cortex-M0 microcontroller based packages

Another package is available with Bluetooth Low Energy (LE) support from Nordic Semiconductor, namely the nRF51822[36]. This package has a ARM Cortex-M0 cpu and is this is a true 32 bit architecture. This device has some disadvantages which limit its use in a production environment:

- No large consumer base for the Bluetooth LE stack.
- No certification (FCC/ECC) available for available modules.
- Only half of the memory is persistent in low power mode.
- A slightly higher power consumption in cpu deep sleep mode as the CC254x, in which the device will mostly reside.

[1]IAR is a supplier of compilers and IDE for embedded development.

On the other hand, the advantages are:

- Numerous compiler support: The ARM architecture is supported by a vast array of compilers and platforms. This would enable development from reliable compilers like the GNU Compiler Collection[41] and LLVM[42], and no vendor lock-in.
- Faster processor as it is a true 32bit architecture and twice the memory compared to the CC254x chipsets.

The lack of certification is a no-go for this chipset. Certification is expensive and the higher power consumption on paper might limit the lifetime of the device but this is only marginally.

3.2 Hardware Selection

Due to lack of certification and the availability of the Cortex-M0 package and large usage base of the CC254x we selected this chipset. As there is little difference in the given 8051 chipsets, the software developed for the chips should be exchangeable between the two CC254x variants. The device has sufficient external communication methods and the power consumption is on paper sufficiently low for our usage profile.

Chapter 4

Exploratory Measurements

In this Chapter, I describe how I measured power consumption. A short introduction to the chipset and its characteristics is given in Section 4.1. In Section 4.2, I will give a more detailed description of the hardware involved and the methods used to measure the current drawn by the device. In Section 4.3, I will discuss the connections used. Section 4.4 provides possible pitfalls concerning the measurements and I will show how to deal with those.

4.1 Introduction

To be able to measure power consumption reliably there are several variables involved. First we need to know the electrical characteristics of the device. As we already know, we use the CC2540 [35]. We know that the power consumption will be between 0.9μA and 34mA. This is however a range of 5 orders of magnitude and might give problems measuring either the lower end of the scale or the higher end. Furthermore we need to be able to measure on sub microsecond scales.

The CC2540 is able to run on 40MHz but effectively this is lower due to most instructions needing more than one cycle to complete[35]. Most calculations, which cost a considerable amount of power, are in the order of several 10ths of microseconds to tenths of milliseconds at most. The cpu itself consumes significantly less power than the radio and most calculations can be done while the radio is active. Since the cpu has to be active while the radio is transmitting we can use this time to do our calculations. The wireless transmissions last from 80 microseconds to around 330 microseconds [43] and can be repeated several times in a row with a 80 μs gap in between. This means we need to have a resolution greater than 25kHz.

To measure the power consumption reliably there are several conditions which need to be fulfilled:

- A stable power source needs to be connected to the device

- Sufficient amplification, preferably without additional artefacts, needs to be available to measure both sub micro-ampere scales and tenths of micro-ampere scales.
- Sufficient time resolution should be achieved to detect rapid changes in power consumption and thus a greater bandwidth than 25kHz.
- To keep track of the progress of each connection, a method is needed to display this progress.

4.2 Hardware

One of the companies providing the CC2540 chip also delivers a test, develop and debug board (BLE112 evaluation board) which has several additional features like GPIO headers and a 3.3v differential signal which delivers a voltage collected over a shunt and amplifies this to give a measurement of the current used by the processor with the following formula:

$$I_0 = \frac{3.3 - V_0}{30} \tag{4.1}$$

The current through the processor can thus be easily measured and the device only measures the CC2540 processor and not the additional peripherals.
To achieve sufficient temporal resolution I need to measure above 25kHz as mentioned before. While we can use a memory scope to do this, an easier configuration is possible with a measurement device from National Instruments. This USB-6211[44] has a 250kHz bandwidth. This is however only possible with one channel in most configurations and I possibly want to measure additional signals. With up to four signals a resolution of 100kHz is achievable but buffer under runs might occur at this bandwidth. This does not happen at 50kHz. This is still sufficient and can thus be used to measure the current (differential voltage) and some additional signals. The accuracy of the USB-6211 is 41μV per level in the range of -5V to 5V and under 100kHz and therefore the 0.9μA can not be directly detected as it would require a sensitivity of at least 2.7μV.
For stable voltage measurements I need a stable voltage source. The default voltage source for the evaluation board is a micro-USB plug which is normally connected to a USB phone charger. These phone chargers, however, are less than perfect and have severe ripple and switching spike in the voltage source. This makes measurements difficult next to impossible. A better source is a laptop USB port or an independent voltage source. The independent voltage source from Delta Electronics is an analogue (E030-3) 0-30V 0-3A power source. For reference I measured the difference between the independent voltage source and the device connected to a laptop and

they both produce considerable noise signal. To be sure all noise is Gaussian and can thus be ignored (e.g. it will cancel itself out) the data is plotted for a period of 1.6 seconds over the active PM2 mode. This mode should use 0.9μA of power and would report either 0A, assuming 0V is a valid level, or anything in the range of +/-41μA from the zero. The resulting data set of the mentioned measurement is plot into Figure 4.1 as a histogram to show the clustering of the data. The mean of the data is 556.59μA. A Gaussian is fit onto the data to show the validity of the claim that the noise is Gaussian around the mean of the data set. I will discuss the problems with this data a bit more in detail in Section 4.4. As a result of this data, the device is calibrated to subtract the mean of the PM2 mode of the result minus the 0.9μA leaving a noisy signal around 0.9μA instead of 556.59μA.

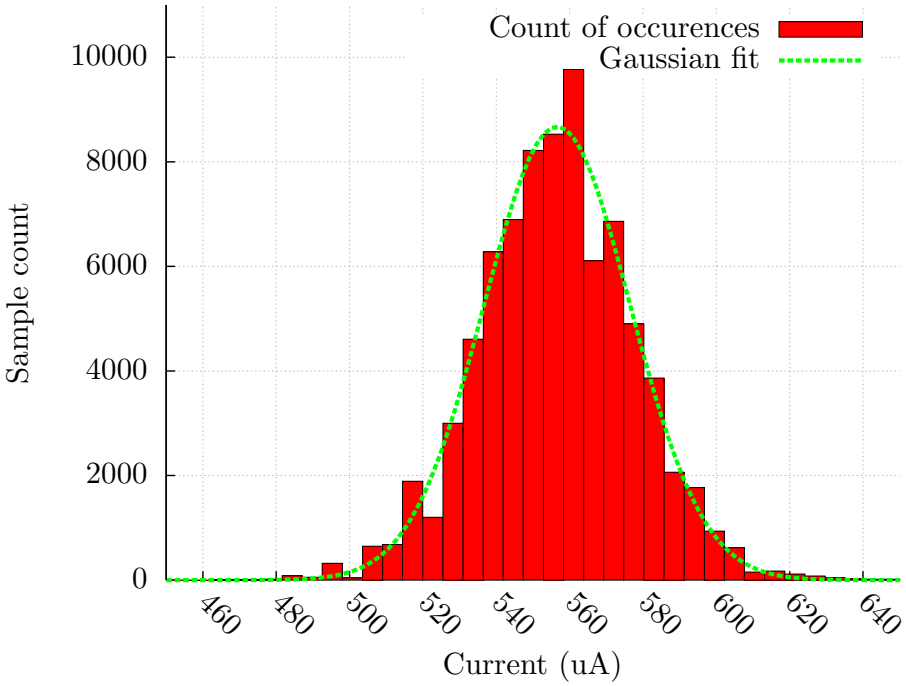

Figure 4.1: Histogram and Gaussian curve fit over 1.6 seconds of idle operation in PM2 mode with a mean of 556.59μA

4.3 Setup

To measure the current, the evaluation board provides a differential voltage with which the current through the CC2540 processor can be measured with Equation 4.1 . As can be seen in Figure 4.2 and 4.3 the evaluation board is connected to the National Instruments module by a coaxial cable. The

ground and three GPIO ports are connected to voltage in ports 1 to 3 and can be triggered to show steps in the process.

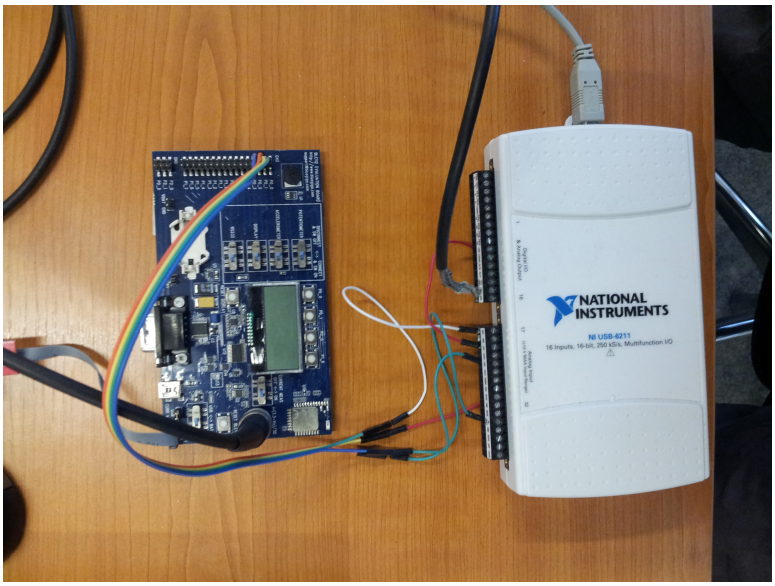

Figure 4.2: Setup with NI6211 and BLE112 Evaluation board.

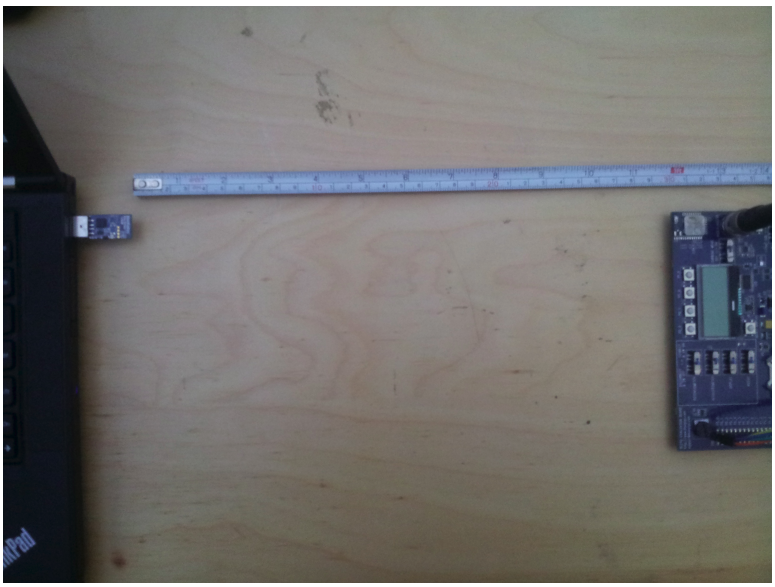

Figure 4.3: Overview of distance between radios of 30cm end to end. The radios are about 4mm further apart.

The additional connection shown in Figure 4.2 is the debugger. A power

source is not connected in this image.

4.4 Setup problems and solutions

There are two main possible problems with this setup:

- A lower accuracy than the specified power consumption in PM2 mode.
- Noise on top of signals.
- A relatively low temporal bandwidth.

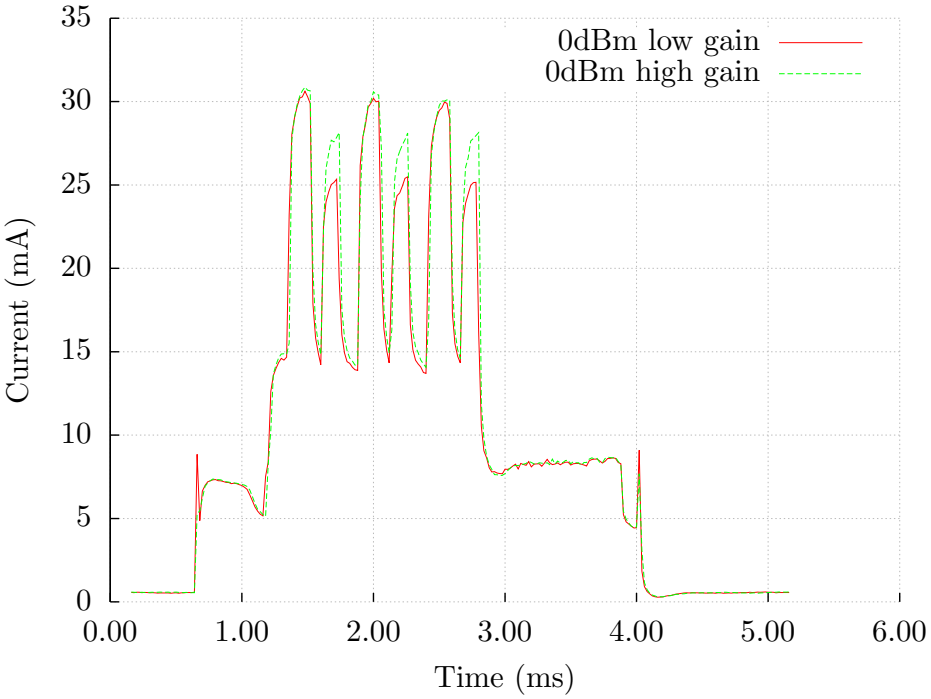

Figure 4.4: A display of two advertisement messages with 0dBm transmission power and low and high gain receivers at 50KS/s.

Since noise signal is Gaussian, as shown in Figure 4.1, it is not problematic as long as it remains this way over all measured data. In all measurements there has been no indication that it would be otherwise and is probably white or thermal noise. Since most of the data is repetitions of the same waveform over and over again (for instance the advertisement packets of the signal remain the same for its entire lifetime), the assumption can be made that the resulting average is near to the true power consumption.

It is very difficulty to detect the 0.9μA sleep current we have to make an assumption: all data around 0V, after calibration as mentioned in Section

4.2, is assumed to be from the PM2[1] mode. PM3[2] mode is never used, which is in the same order of magnitude as the PM2 mode (0.9μA vs 0.4μA) and should be considered if used. The PM1[3] mode is also not used in our experiment but this should be detectable as it is three order of magnitude bigger and well within the detection sensitivity of the measurement devices with a specified power consumption of 235μA. All transmission and active modes are in the order of milliampères and can be easily detected as separate from the PM2 mode.
The temporal bandwidth could be problematic if aliasing occurs during the switching of power states. As can be seen in Figure 4.4 which is recorded at 50kHz there is no clear evidence of aliasing and there is sufficient resolution to detect the changes and incorporate them into the results given the assumed speed changes in the Bluetooth LE radio model.

[1]PM2 is the lowest power sleep mode in which the CPU can be without losing clock and memory coherence.

[2]PM3 is the lowest power mode the CPU can be in but is only wakeable from an outside source or GPIO interrupt.

[3]PM1 mode is used when the device goes to sleep for less than 3μs. This mode is not used by the bluetooth stack.

Chapter 5

Power consumption analysis

This chapter will show the power consumption of the device during different states and show the reader the areas in which there is room for improvement. The chapter will give several options of saving power compared to normal operation and provide the reader with sufficient understanding of the power consumption of the device to understand the choices made. Section 5.1 will provide some methods for conserving energy in hardware and software and in the Subsections there will be a more detailed overview of the different methods and their results.

5.1 Power estimations and saving

To achieve a long lasting device you can either reduce power consumption or increase the available power. Reduction can be achieved in various ways like reducing the power of wireless transmitters, limiting MCU awake time and limiting the number of enabled peripherals. Power consumption can also be limited by reducing the dissipated power by a DC/DC converter. If consumption can not be limited, a larger power supply can be used. This has however limitations as there might be limited space available for large batteries or power supplies.

5.1.1 Power saving techniques

The most common technique in saving power on embedded (wireless) devices is reducing the MCU awake for a too long time time by either limiting the wakeup time for radio communication on a low level[45] or by limiting the power usage of the MCU by lowering its operating voltage[46]. In wireless devices this can be combined with higher level network protocols to make sure the device is not too long awake to listen to other devices as the receiver usually consumes an equal or greater amount of power than a transmitter. Other methods are limiting transmission power or receiver gain, but those

only limit the power consumed during transmission by a small margin and are still dependant on the amount of time the device is awake as will be shown in Section 5.1.2.

5.1.2 Hardware and software power saving

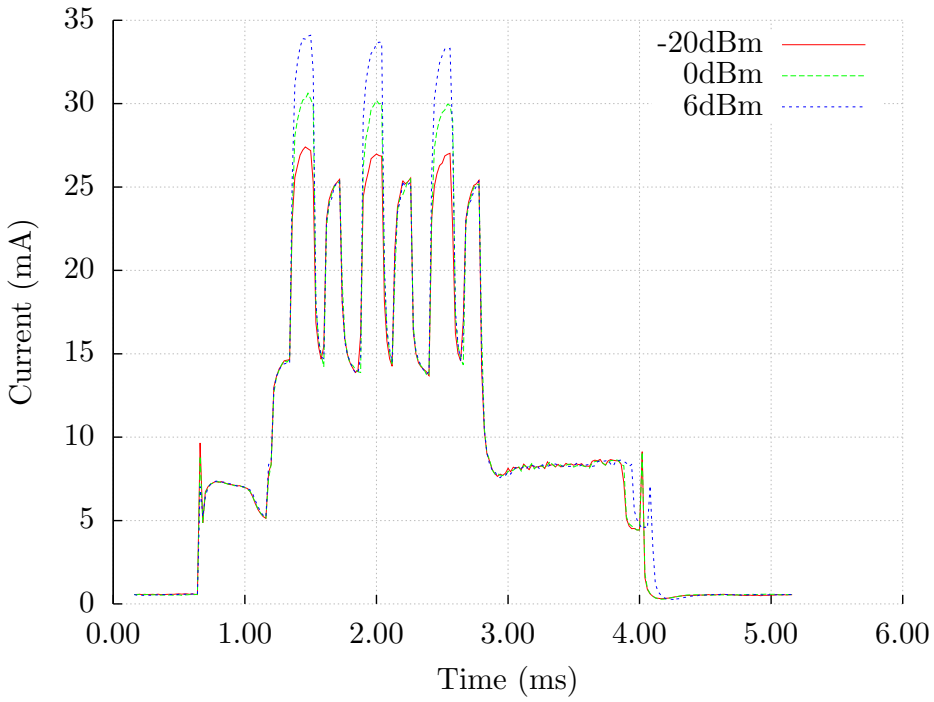

Figure 5.1: Power profile during an advertisement message with -20, 0 and 6dBm transmission operating modes.

Figure 5.1 shows the power consumption during a message on the CC2540 chipset with three different transmission power states and during a change in receiver gain. Table 5.2 shows the total power (in J) consumed by the device during one of those states and during sleep mode. It can easily be seen that there is some gain in lowering the transmission power but, as can be expected, the MCU uses significantly more power than in sleep mode. Another disadvantage of lowering the transmission power is the limited range in a noisy environment: During RSSI tests, the range drops significantly when the transmission power and amplifier gain is limited. This affects both the wireless communication active time and the amount of time the device spends in send/receive mode due to retransmits or failed connections.

Mode	Current	Power
Active RX (min)	19.6 mA	58.8 mJ/s
Active TX (-20dBm)	24 mA	72 mJ/s
PM1	235 μA	0.705 mJ/s
PM2	0.9 μA	0.0027 mJ/s
PM3 (external interrupt)	0.4 μA	0.0012 mJ/s

Table 5.1: Differences in power used as advertised by the TI2540 specifications[35] at a voltage level of 3.0V.

5.1.3 Communication

Communication is the most power consuming task the device needs to perform. Between each communication step it needs to perform some form of HMAC (RFC2104)[47] calculation and while this consumes some power, it is considerably less than while actually communicating. One disadvantage of using Bluetooth LE is the lack of deep sleep modes during communication, which forces the device to keep waking up while maintaining a connection. This will be described in more detail in Section 6.3 where I will discuss the Bluetooth LE connection interval which decides the time between wakeup moments for communication and computation. Looking at the raw power consumption data in Table 5.1 we can see the big difference between the given power usage in deep sleep (PM2) and normal operation. Taking into account the periodic wakeup to advertise the device and process data, we can calculate the expected lifetime which is given in Section 5.1.4. Table 5.2 shows the differences in transmission power and gain compared to the power consumption of sending one advertisement message over the specified three different channels in Bluetooth LE. As can be seen, the difference in total consumption is only marginal (at most 11.5%) when we change the transmission power and receiver gain during any form of transmission. For our measurements we take the lower end as basic point. If in the field the communication would suffer because of communication drops, it is advisable to increase to transmission power and receiver gain accordingly until satisfactory communication can be achieved, at the cost of power consumption. Please note that the values in Table 5.2 can be a bit higher than the operational consumption due to two LEDs being enabled during operation.

5.1.4 Estimates on power consumption and supplies

To give an estimate of power consumption and needed battery power, I use the measurement methods explained in detail in Chapter 4 to measure average power consumption during specific steps of the process. For lifetime expectancies I used the estimate of 10 usages a day, and normal operation in between these events. An advertisement interval is 1000ms, therefore the

Transmission Rating	Gain	Power	Difference
-20dBm	low	156 μJ	-
0dBm	low	163 μJ	4.4%
6dBm	low	172 μJ	10.2%
-20dBm	high	164 μJ	5.1%
0dBm	high	167 μJ	7.0%
6dBm	high	174 μJ	11.5%

Table 5.2: Differences in power used for one advertisement message.

device always wakes up at least once a second to advertise itself to the world and show it is alive. For transmission power and amplifier gain I used the least power efficient settings. This is done to make sure the connection is not lost during communication and provides worst case consumption measurements. Section 5.1.4 shows the available power sources and 5.1.4 provides the estimates for the device lifetime.

The power consumption of a servo is fixed and needed for operation of the device. While we take it into account, it is not subjected to change as the selection is out of our control. The servo is activated for 1 Second and thus uses in that period 120mJ of power per operation. The main power saving techniques used in these examples are only relative to all other modes of operation.

Power supplies

The device has to be stand alone and can not be connected to a power source like an AC/DC converter connected to the mains power. This results in several options of power supplies but I will limit it to industry standard batteries with a nominal voltage rating of at least 2.4 and at most 3.5 Volts. These batteries can supply the power to the TI CC254x devices without complicated circuitry. One could argue that energy harvesting would benefit a low powered device but those usually require modifications to the environment or additional circuitry which would require more board space and thus limit the operational use of the device.
Energy harvesting by means of motion is possible, for instance by using a dynamo connected to a door crank, but these systems are mostly mechanical and could limit the lifetime of the devices by having the lowest mean time between failure of all parts involved. To support this claim, the USA Army Electronics Command considered a hand cranked 30V generator which had a MTBF of only 2903 hours of operation, which is not impressive.[48]. While this is continued use, the device was considerably larger than anything we had in mind which in general means less failures and a longer lifetime. Therefore we expect a smaller device to operate for an even shorter period

of time.
As an example crank could supply sufficient power as there are numerous versions available on the internet (e.g. by checking Amazon.com for hand cranked torches) which deliver around 12V max. at 200mA output. This is more than sufficient to provide around 2.4J even cranked for one second. The main problem remains that it takes up a lot of space and has not been tested in the field. Another problem is that the on-time of the device is probably longer than the cranking time which requires the energy to be stored. This would require some form of capacitive energy storage, which can be charged fast enough, but this requires again additional circuitry and space, which is already limited.

Another addition could be to use a DC/DC converter bypass module[49], which allows the device to run in the most efficient mode at 2.4v at all times during sleep and low power modes, but this is not used due to circuit complexity. The manufacturer claims a decrease of power used by 30% at best. While this is probably overestimated even a little gain of 10% would merit its use in a production environment.

Table 5.3 shows the Joules of power provided by some industry standard batteries. To keep the calculations simple, I ignored typical characteristics of batteries like a voltage drop over time when the battery is drained and difference in internal resistance while under load. These factors do matter when operating for a longer period but require extensive testing beyond the scope of this thesis.

Type	Size (cm^3)	V	mAh	total J	J/cm^3
Alkaline AA (2x)	30.7	1.225	1150	10143	165
Lithium (CR2)	20.6	3.0	750	8100	405
Li-Ion (CR2)	20.6	3.6	750	9720	471
CR2320 (large button)	4.06	3.0	175	1890	465

Table 5.3: Different power supplies and their maximum power rating. Note that some versions require two batteries to get to the specified minimum voltage of 2.4V.

Conclusion CR2320 batteries have an advantage of being of high capacity per cm^3 but lack the ability to deliver substantial currents. This limits their usefulless to us as the device should be able to drive a servo. CR2 batteries show the biggest promise as they are of small size, are able to deliver substantial currents and have a very good power to size ratio. The biggest disadvantage of the CR2 batteries is their price: 3 to 4 Euro a piece for normal lithium cells and even more for li-ion variants. A 'double sized'

CR2, which is about as big as a AA size battery only a little wider, could also be used if more power is needed over time, but for space considerations we assume a single CR2 lithium cell as our default battery. If a single cell is insufficient, a double cell could be used which would under perfect conditions double the total power available.

Consumption

Power mode	mJ/s	days on 8100 J	Modifiable
Sleep	0.027	3472	No
Advertisement	0.134	699	Yes
Communication (full speed)	17.28	5.42	Yes
Servo control	120	0.78	No

Table 5.4: Power consumption (in J) over a one second period in different modes of operation on a 3V battery.

As stated before in Section 5.1.4 I used the average of 10 usages per day as a baseline. To achieve 50000 operations in two years, an average of over 68 operations per day would need to be achieved. We take an average daily as our guide as the numbers are easier to understand from a human perspective. The 68 operations per day would for instance be a side entrance of a building and considering the constraints set to a minimum lifetime and considered operations total (50000) the device would simply live longer if less operations would be performed. This is due to it being able to sleep longer and communicate less. As a more busy scheme we assume 100 entries per day. Table 5.4 shows the averages in J over one second of operation in a specific mode.

Changing power consumption in the specified modes Table 5.4 gives an overview of the difference in power consumption between the defined modes. This shows that it is easiest to change the communication and advertisement states. The sleep mode is fixed at the power consumption of the chipset for the PM2 sleep mode at 0.9μA. We should however try to operate the device as much in this mode as it is the most efficient mode. Servo activation can not be changed and is fixed. We can change the advertisement mode by increasing the window in between advertisement messages[9]. Increasing this period creates a higher delay but might save power in the long run. Increasing this delay also has the disadvantage that a user will not be able to detect the device quickly. The detection is dependant on the interval between advertisement messages and a higher interval simply means a longer detection interval. As this would adversely affect user perception if

over more than one and at most two second[32] thus we leave the advertisement interval at 1 second maximum to avoid negative feedback from users as they can only connect to something they can detect. The 1-second advertisement interval should also be added to any communication time as it is the time needed to establish a connection and thus create an even bigger constraint on the maximum time a device could communicate. The only real change would be in the time a device is communicating. If possible we should thus reduce the time needed in this mode or limit the amount of time in maximum connection speed.

To calculate the lifetime of the device, a single 8100 J CR2 cell is used as stated in Paragraph 5.1.4.
The power used by the servo is fixed at 40mA over 1 second at 3.0V. This value would lead to 120 mJ and thus a total usage of $50000 \cdot 0.120J = 6000J$ assuming a perfect power conversion and no mechanical and electrical losses. This would only leave 2100 J on the single CR2. Therefore I used the double CR2 option which would give a total of 10200 J of energy for a lifetime operation. This assures a limited space requirement of a bit larger than a single AA sized battery but only by a very small margin, which should fit in most, if not all environments we foresee. A double CR2 is also used in many industrial applications like RFID locks[].

The values are calculated using the following formula:

$$P_{total} = \frac{P_{battery}}{(86400 - (t_{comm} \cdot t_{ctime})) * 0.134mJ + ((t_{comm} \cdot t_{ctime}) * 17.28mJ)} \tag{5.1}$$

Assuming $P_{battery}$ as the battery power available to us in Joules, t_{comm} as the number of communications needed per day which we assume to be the same the number of operations and with t_{ctime} as the time needed to finish one operation. This results in a number of days the device can operate without completely draining its power supply.

Conclusion concerning power usage

Since the device uses most of its power during communication, we need to seek a method of limiting the use of communication time or find a way to let the device sleep during communication. A drop of power during communication to about 2 J/day would enable the device to achieve nearly 760 days of operation from a bit over 500 and assuming the device is actually asleep during most of its communication steps we should limit it to this as much as possible.

Table 5.5 shows that the power consumed while sleeping is almost constant in relation to the power used while communicating. In the table I chose to show the communication time of 4 and 6 seconds, which are near to the maximum and over the maximum we set out as the maximum a person

Op./day	Comm. time (s)	Idle (J/day)	Comm. (J/day)	Tot. Days
0	0	11.58	0	881.0
10	4	11.57	0.71	830.3
10	6	11.57	1.06	807.0
68	4	11.54	4.84	622.3
68	6	11.52	7.27	542.7
100	4	11.52	7.10	546.9
100	6	11.50	10.69	459.68

Table 5.5: Lifetime calculations for varying communication time and operations/day.

would wait and their relation to the number of operations per day. For a low amount of operations we took 10, the 68 represents the average for the total of 50000 operations in two years and 100 for busy doors to compare with.

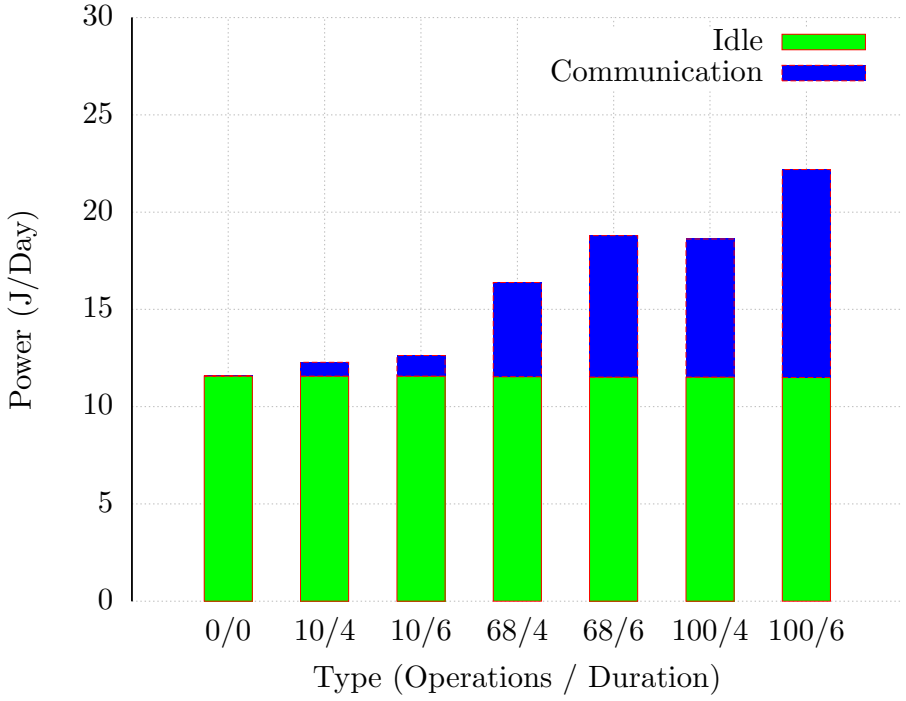

Figure 5.2: Power usage comparison for idle and communication power consumption.

For comparison Figure 5.2 shows a clear picture of the similarities in the base power usage and communication power usage.
The results show that two years of operation is difficult to achieve but we

can get close if we lower the power usage while communicating. There are however other requirements like response time to consider. These values are best case: There is no reconnection needed and the device and connection never fail. Any additional failures would require the device to be in a connection for longer and thus require more power and less time available.

Chapter 6

Network Analysis

This chapter will show the usage of the network by the UQ protocol for communication between the device, a phone and the server. An introduction is given in Section 6.1 and 6.2 about the usage of the network. Section 6.2.1 provides an overview of the usage of Bluetooth Low Energy in the network and Section 6.2.2 provides an overview of possible delays encountered in the network using different connection methods. In Section 6.3 I will give a more detailed overview of the difficulties using rate switching with Bluetooth Low Energy, and in Section 6.4 it is shown how we need to predict the delay if we do not know the expected delay within the network.

6.1 Introduction

The protocol used in the communication between the device, phone and server is the UQ protocol. This protocol is build from a a header with data payload in which the header takes care of the security of the message via a (H)MAC[47]. The header is most of the data in a message as it contains the target device UUID[1], a sender UUID and 16 bytes of (H)MAC with some control and anti-tamper data. Through the data, the phone, server and device can send and receive messages from each other like the initiation of the opening of a door, request of additional data from each other as in a status update and verify each others. The exact details of the messages are not needed for the research but the global scheme is mostly the same in each message and has an initiation (open), handshake and finishing part. While the protocol is good to use security wise and it has lots of redundancies to check for tampering, it is bulky which can be problematic on slow connections. Another problem is the delays introduced in the network which can keep the device awake and waiting for data for a long period of time (seconds). These delays are caused by the communication of a mobile

[1] A UUID, or universally unique identifier, is a way to represent unique devices and consists of 16 bytes

phone over slow networks like GPRS. This chapter will analyse the delays in the network and provide options to deal with these delays in a way the Bluetooth Low Energy protocol can keep operating without modifying the lower levels of the stack.

6.2 Analysis

Figure 6.1 shows the chain of events in a UQ message. This chain has at least three communication steps between the devices over entire network. This is the device, the phone and the server and back. These communication steps require the message to pass through the endpoint, our device, and to and from the phone and server to achieve verification of the phone (user), device and validity of the request (open). These communication steps are shown in Figure 6.1 are all required and can be extended with requests from the server for battery and device status. As we need to communicate with the server, an internet connection is required on the phone though which the device communicates. How this connection is made is not an issue as long as it is available. This availability is communicated through the UQ protocol, and if not available the response will be invalid and the connection dropped. For now we assume that the connection is always available.

The data in the first step seen from the endpoint is seen in Equation 6.1 where t describes communication delays and p describes processing delays.

$$t_{tot} = t_{bt} + p_{HMAC} + t_{bt} + p_{phone} + t_{net} + p_{server} + t_{net} + p_{phone} + t_{bt} \quad (6.1)$$

In this equation there are two main points of processing (p_{phone}, p_{server}), besides the data processed in the device (p_{HMAC}), and two main causes of delay in the connection as mentioned before (t_{bt}, t_{net}). The processing time in the server and phone are not a real point of concern as they are orders of magnitude faster (sub milliseconds) than the delays within the network and processing time on the device itself.
The processing time of the HMAC, which is a process by which the validity of the message can be later confirmed, can be done in several ways and is not of interest to us at this point. It can be done by an additional security processor or smartcard which does this process faster and with less power consumption than can be done within the processor of the device. For our purpose we ignore this time and power consumption as it is small and requires a lot of additional hardware which is both expensive and difficult to operate with little profit in lowering power consumption.
Two main points of delay which remain are t_{bt} and t_{net} representing the bluetooth and network (phone to server) delays. A complete communication cycle in which the device has to wait for a response is shown in Equation 6.2

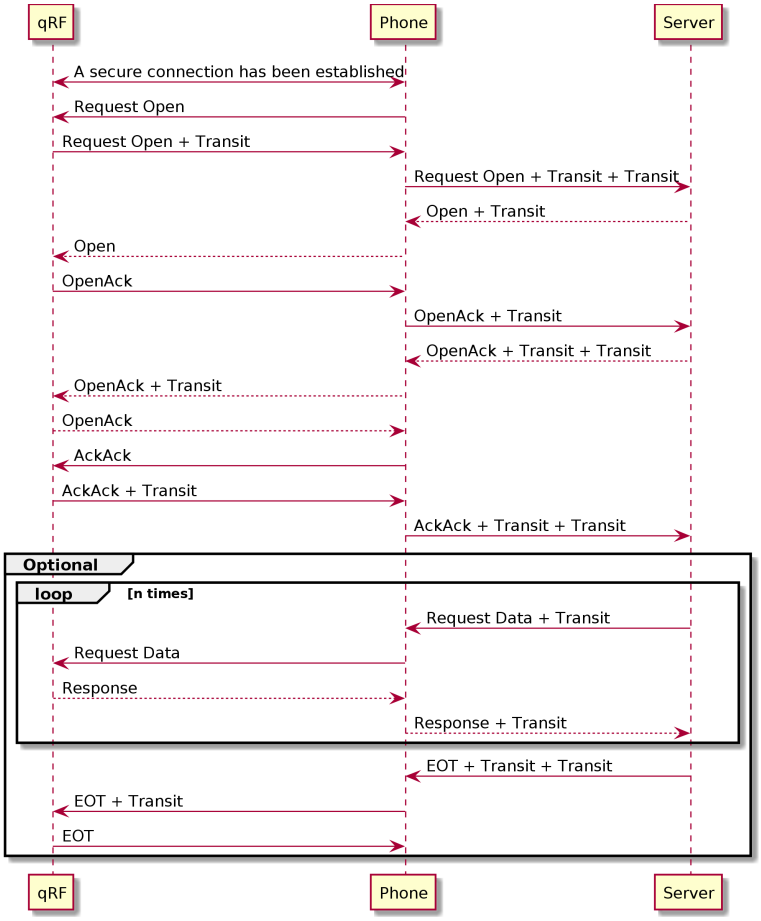

Figure 6.1: Sequence diagram of complete UQ protocol communication with an endpoint.

$$t_{delay} = 2 \cdot t_{bt} + 2 \cdot t_{net} \tag{6.2}$$

6.2.1 Bluetooth connection

The bluetooth connection is a BT4.0 (or Low energy, BTLE) connection which has the advantage of a relatively low power consumption compared to original bluetooth but still has a decent maximum throughput rate of 1 Megabit/s of raw data. In our case, the TI CC2540 can achieve a payload throughput rate of 5.2KByte/s. For further comparisons we assume a message size of 160 bytes which is an average message size for the data transmitted.

Bluetooth LE messages

The bluetooth connection on the device used can send four messages of max 20 bytes per connection interval. A connection interval has a minimum length of 7.5ms. For data validation and confirmation the message contains two more bytes of non-payload data. These bytes are used to know which byte arrived and in case one or multiple are missing to know the message length. Since it is possible to receive only one block of data we have to send the complete message length with every block. Code 6.3 shows the composition of the data block where `BTLE` is the bluetooth low energy header, aes and crc data and the numbers represent the bytes used.

```
[BTLE] [1] [1] [1-18] [BTLE]
```
(6.3)

This block can thus send 18 bytes of data per 20 bytes of payload. A normal message of 160 bytes would thus take at least 9 blocks to send resulting in using at least three 7.5ms intervals. This is however an optimum and the throughput rate generally is a lot lower. If the link quality is low, throughput is lowered. This can have influence on the number of intervals needed to send the entire message and thus on the amount of time needed before a connection can go into another interval delay. This is discussed in more detail in section 6.3.

6.2.2 Network connection

The second cause of delays, in which the device has to wait and thus sleep, is the network connection of the mobile phone which was represented by t_{net} in Equation 6.2. This connection can be either WiFi (abgn/ac), a mobile connection like GPRS or UMTS, or a wired connection in case a device other than a phone is used as a substitute; e.g. a laptop. To know the delays which might occur I measured the round trip time of a new connection on a mobile phone. The phone sends 256 bytes of information to a server which returns a new set of 256 bytes back to the phone. Figure 6.2 and Table 6.1 show the common round trip times of this message to a remote server. The remote server was located 9 hops away (via wifi) in another building connected to a 100MBit ethernet connection via a 4Gbit uplink. The phone used for measurements was located inside a building with a steel frame, limiting connectivity.

The results of the HSDPA measurement are a bit too high compared to standard HSPA due to the connection trying to get maximum bandwidth but not being able to achieve this and throttling back to UMTS in more cases. This is something which is present in real situations and thus the measurement is relevant.
The wifi results are higher than one would expect due to the presence and interference in the 2.4GHz band. At the time of measurement over 50 wireless networks were engaged and several hundreds of devices connecting to

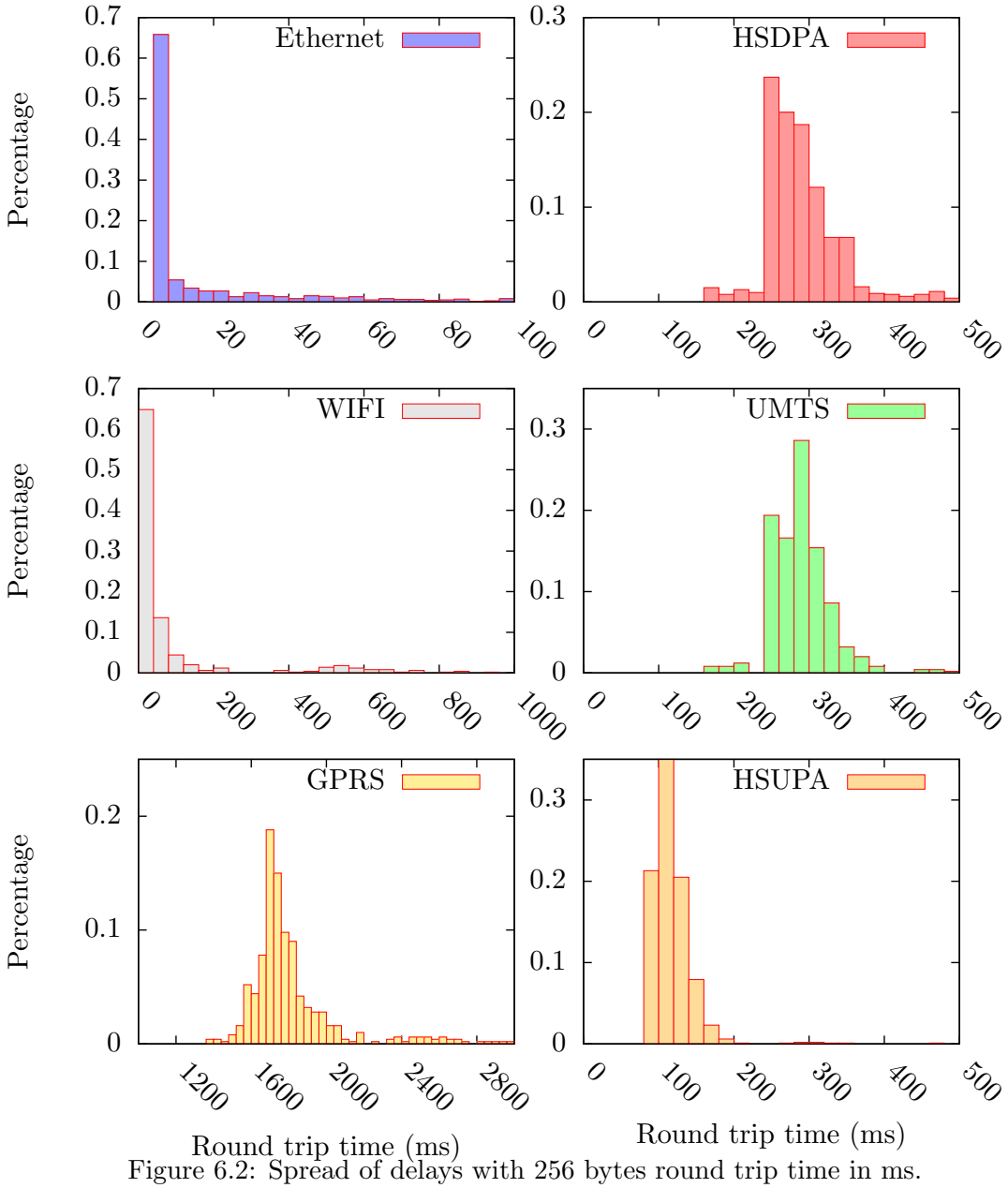

Figure 6.2: Spread of delays with 256 bytes round trip time in ms.

them resulting in some congestion. Peaks of over 2 seconds of delay were common and some exceeded 5 seconds.

As seen in Table 6.1 the delays with a mobile network connection can exceed several hundreds of milliseconds. This time can be used to throttle down the bluetooth connection and thus save power by not waking the processor. Details about how this is done can be seen in Section 6.3.

Since it is not known if there is already an active connection or the con-

Type	Min.	1st Qu.	Median	3rd Qu.	Max.	Mean
GPRS	1375	1694	1764	1857	5359	1860
UMTS	167	265	285	306	1281	295
HSPA	89	100	106	126	707	116
HSDPA	162	255	284	314	821	292
WiFi (56Mbit,g)	16	28	33	62	10147	247
Wired	6	6	7	17	137	18

Table 6.1: Delays using different networks on a mobile phone or laptop over 1000 samples of 256 bytes round trip time.

nection needs to be setup I did not measure the difference between an active connection or one which also includes setup time. The values in Table 6.1 are thus worst case but give a clear indication in the differences in the possible connection setups.

Results

The delays in the network connections show that there is sufficient room for optimization. Even de smaller delays of around 300ms are usable to go into sleep mode, as the device is able to do this properly if the delay is greater than 3ms. While this might not be useful in case a Wifi or good HS(D/U)PA connection is used but if the connection is flaky (slow) or switches to UMTS or GPRS a bigger delay is common and it is useful to consider using this delay to sleep.

6.3 Bluetooth connection interval

When a device connects with a Bluetooth low energy connection, it sets a specified time between each moment the sender sends and the receiver listens. This interval dictates the throughput and the amount of time the device is awake. This is thus a trade-off between speed, throughput and lifetime as the device will always use the send and receive radio at least once even when no actual data is transmitted. This means the connection interval should be adapted to the specifications of the device. In our case though, we need both battery life and throughput. Quick switching between connection intervals could create an advantage as we can delay if we wait for data or otherwise speed up if the throughput needs to be at its highest.

Introduction: A Bluetooth low energy connection has a minimum for starting a connection interval of 7.5ms. In this interval data can be sent and received, either by notification or by indication. Notifications are send and not confirmed in the next interval, but indications are confirmed and

resends in case of failures can be handled by the link layer. This method can only send one packet every 7.5ms due to hardware and software library limitations while the notifications can be send four in one interval. To increase the throughput I use notifications and handle send failures in a higher level protocol as mentioned in Section 6.2.1.

To change the amount of time the device wakes up, a *change interval request* is send by either the Master or the Slave in the established connection. The sender first checks if the requested interval is possible and sends a request. For our test we assume all legal values in the Bluetooth 4.0 specification are possible. The requests send contains both the minimal, maximal interval and the instant. The minimal and maximal interval indicate the range in which the other device has to negotiate the next interval. The instant is a value which tells after which amount of intervals in the *current* interval the new value will commence. This value is set to 6 in all known implementations though it can be changed at the link layer. This will most likely break compatibility with numerous devices and we want compatibility. We thus have to wait at least 6 intervals resulting in seven additional intervals between each change. Additional data like the transmission window is sent but not relevant to our situation.

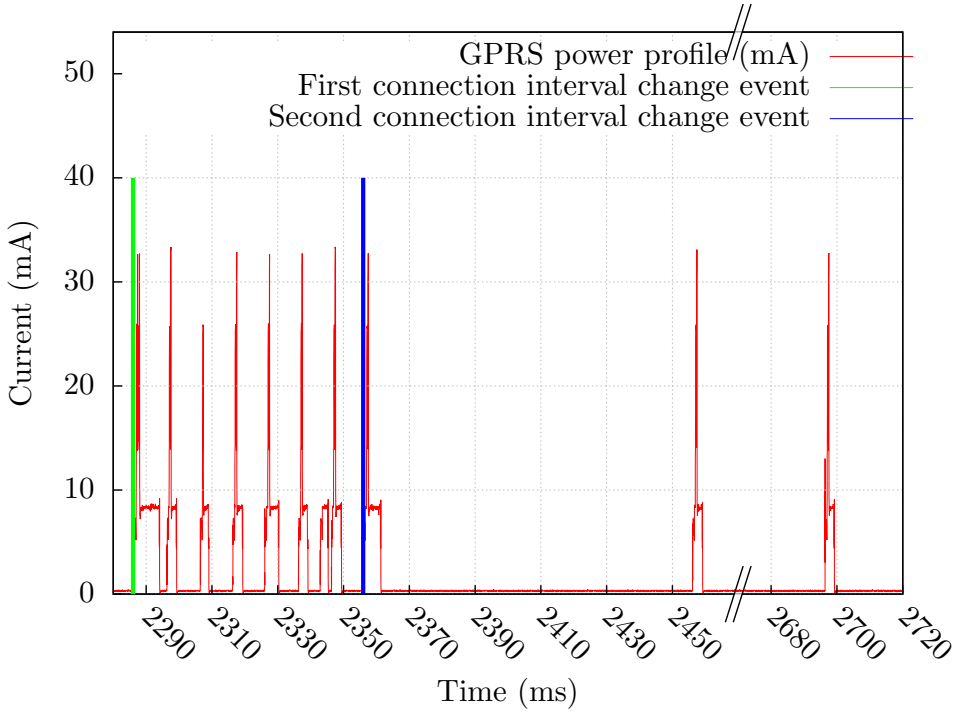

Figure 6.3: Connection intervals (10ms start) switching to 240 ms.

As stated, the instant is the amount of *current* intervals after which the

new interval will start. This results in a minimal time needed before a change can be applied assuming the minimal interval of 7.5ms:

$$t_{minchange} = 7.5ms \cdot 7 = 52.5ms \tag{6.4}$$

This results in 1 interval for the data to be acknowledged and 6 more for the change to apply which can be seen in Figure 6.3. If one considers the minimum of changing back to a higher (7.5ms) interval this is a minimum of twice this value. Therefore any update lower than 105ms is not needed: There is no gain in sleep time over a normal fast connection interval.

The intervals needed to change are normal intervals and can be used to transmit data. If the connection is sufficiently stable and has a decent RSSI one can consider to send the data in the first six intervals by first calculating the probable delay, then sending the data in those intervals and switching to a lower interval. If one has to resend data in the new interval the penalty is significant, especially if a large delay is expected as the retransmission has to wait until the next interval.
Dropping the connection and reconnecting might look like a good option, but the this results in waiting for at least one the next advertisement interval and these are generally set at second(s) intervals. If the connection is secured with an AES encryption this also has to be reconnected which takes at least 4 hence and forth communication steps at best, not taking into account the time it takes to generate a temporary AES key.

To maximize the sleep time in the new interval, the expected delay is either completely covered by seven intervals or has a small margin. Figure 6.3 shows a switch between 10ms and 240ms intervals. This results in an expected delay of at least 1750ms and this can be changed to support additional delays. It can be seen in Figure 6.3 that this would save 161 connection intervals of 10ms in wake up time and leave the processor in deep sleep mode. The Figure shows this even with the split time axis. If one is able to transmit all data in the first seven intervals, even a short 100ms delay could save half of the intervals with the processor in deep sleep mode if one changed to a 15ms connection interval.

6.4 Delay prediction

To know the delay the processor has to wait one could ask the phone for information about the connection. This is however not always possible as different phones and operating systems do not always share information about their connectivity with the software. The Ubiqu protocol and software supports iOS, Android and Nokia (J2ME) devices which provides a limitation to the information reliably being available. To know the true delay we have to keep track of the delay in the data sent en received. This

can be easily done on the CC2540 as it contains a 32KHz crystal and time similar to those used in real-time clocks. The clock keeps running during the deep sleep mode to enable a complete measurement.

Storing the data only provides a history. To predict the next delay I use the size of the data sent and the delay measured as a function and apply linear regression. More advanced systems are possible but the device only has a very limited processor and no floating point unit. This is mitigated by using simple fixed point calculations but this requires large integers which are slow on a 8bit processor. Linear regression provides a way to get a prediction over a changing set of data and it is quick enough to be done in several milliseconds. The size of the sample set has to be determined experimentally and might change according to the situation the device is located in.

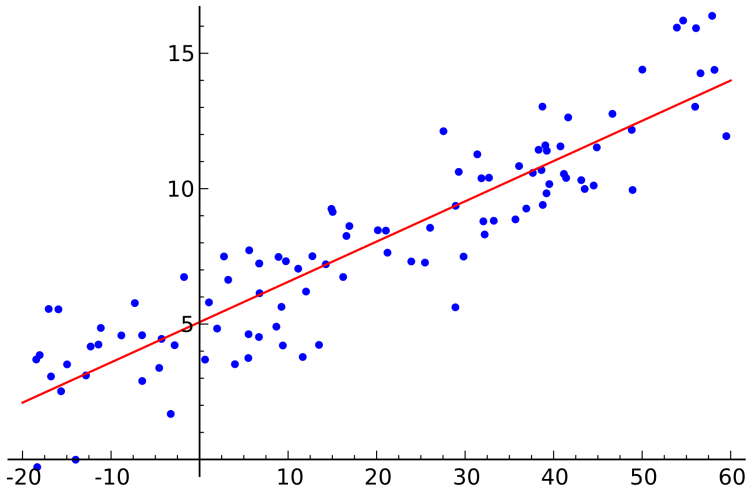

Figure 6.4: An example of linear regression. In our case the x-axis would represent the size of the data to send and the y-axis the time consumed.

Figure 6.4[1] shows an example of how linear regression would work. Due to the unpredictability of the data set, several delays for the same data size would be possible. This is averaged out by a simple linear regression algorithm. If the data points are very close together, like in a very fast connection, the result of the regression formula would be very close to the network delay.

$$time = a + b \cdot n \tag{6.5}$$

[1]Picture by 'Sewaqu' from `http://commons.wikimedia.org/wiki/File:Linear_regression.svg`

Equation 6.5 shows the composition of the expected delay. The a value is the common delay in the network. This can be due to several reasons like distance, hop count or transmission delays in network equipment. If the connection is slow, b will be large and every byte will send will add additional time before the server or the device receives the data. If b is small, a will dominate the delay prediction and the linear regression will behave like an average. If b is large, more accurate delays can be calculated for longer messages than with simple averages.

Chapter 7

Results

In this chapter I will provide the results of applying the delay prediction, rate switching and provide power measurements of those as described in Chapters 4 and 6. Section 7.1 shows the method of testing the system and in Section 7.1.1 the results are given for the device without rate switching and delay prediction. Section 7.1.2 shows the results by adding sleep cycles and Section 7.2 gives an estimation of the devices achieved lifetime with the modifications enabled.

7.1 Power consumption during transmission

The power consumption during transmission can be measured by taking a fixed amount of time, measure the entire cycle and take into consideration the remaining time. To do this we take a 12 second measurement, of which 2 seconds are ahead of the start of the connection and ten seconds after start. This way both lead in and lead out of a long communication cycle are represented and by fixing the window we can make a more appropriate estimate of power saving during the transmission stage.

To model the network we use a computer with a Bluetooth LE stick attached to it. This stick mimics the responses from a server and phone and simulates a network by creating delays to a response or request from the device. In Section 6.2.2 we discussed the possible delays within the network. For the measurements we drop the HSDPA delays as they are quite similar to UMTS and fix the lower bound to the results obtained from measuring delays in a WiFi link, as well as a 700-1000ms round trip time for reference whose values are randomly generated. As a reference we have made 500 measurements of round trip times for each and use those times as a pool to select random delays from for our simulation as can be seen in Table 6.1. The measurement is recorded in 20us windows or 50KHz. The device gain is set to high and the device transmission power range to +6dBm. For comparison I also did the measurements with the UMTS delays and

the device set to low gain receiver and -20dBm transmission power. This measurement (UMTS low power) is included to show the limited difference in power consumption. In the test setup there is no change in communication rate but in real applications choosing a lower transmission power settings and receiver gain might influence the quality of the Bluetooth LE link.

7.1.1 Baseline measurements

The first measurements were made without any sleeping during communication waiting times. Figure 7.1 shows a complete cycle with added indicators. As can be seen in Table 7.1.

Type	Avg. communication (s)	Avg. power (J)
GPRS	6.02	0.063
700-1000ms	3.25	0.039
UMTS	1.52	0.024
UMTS low power	1.53	0.023
HSUPA	1.00	0.019
WiFi	1.17	0.020

Table 7.1: Average communication time and power consumption over 10 measurements of different connection simulations over a 12 seconds range.

While there is a big difference in both GPRS and UMTS values compared to the other two, the total communication time and power consumption quickly level out to around 0.020 Joules for a complete run. This indicates that, while the connection might be fast on the phone or network end, the bluetooth connection and limitations of the device are the main cause of delays. The small difference in WiFi and HSUPA can be explained as the WiFi measurements have a single long connection stage in one of the measurements. This is to be expected in the real world due to (non)-interoperability of WiFi devices with other 2.4 GHz devices as we also saw into. Any improvement towards the 0.020 Joule range for the other values would be a significant improvement.

Figure 7.1 also shows the limit of this range as can be seen that the delays are under or near the minimal (56ms and up) for which a sleep cycle can be begun.

7.1.2 Adding sleep cycles

To be sure there is sufficient time to sleep while waiting for data from an external source we need to either know the delay in the network as discussed in Section 6.4 or know when to wake up. Since we have no exact knowledge we have to guess the delay and make some assumptions. First we assume the

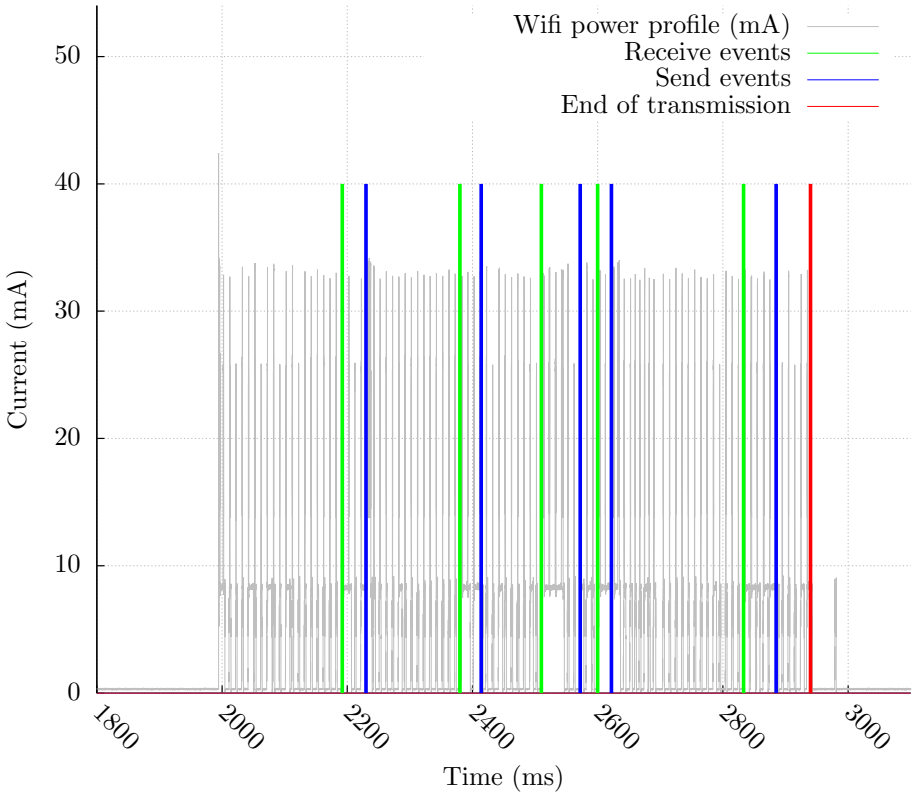

Figure 7.1: Cut out of the connection profile for a WiFi connected back-end. No sleep moments are noticeable due to the small delay in the image.

connection is not going to change a lot in a short time. This is needed since very big changes in connection delay will result in unreliable predictions. Secondly we assume that for (very) slow connections the delay corresponds to the message length. With these assumptions we can try to guess the next delay based on the previous delays. To do this we use a simple linear regression over the size of the message and the previous delays as with slow connections (e.g. GPRS connections typical rate of only 33Kbit/s[50]) the data rate influences the total time needed. A 256 Byte package would consume at least 62ms at 33Kbit/s and possibly more as data rates are usually slower and accumulate with message size added to the delay already present in the network. Due to the way the UQ protocol works it is possible to piggyback messages as it was shown in Figure 6.1 (The pluses indicate piggybacking). It can thus create messages of considerable length (in order of several hundreds of bytes). This is not tested however and the largest message size is 278 bytes in our test setup. As a possible delay is know with the data available, we can alter the connection interval as shown in Section 6.3 we need to take care of the specific functions of the radio and

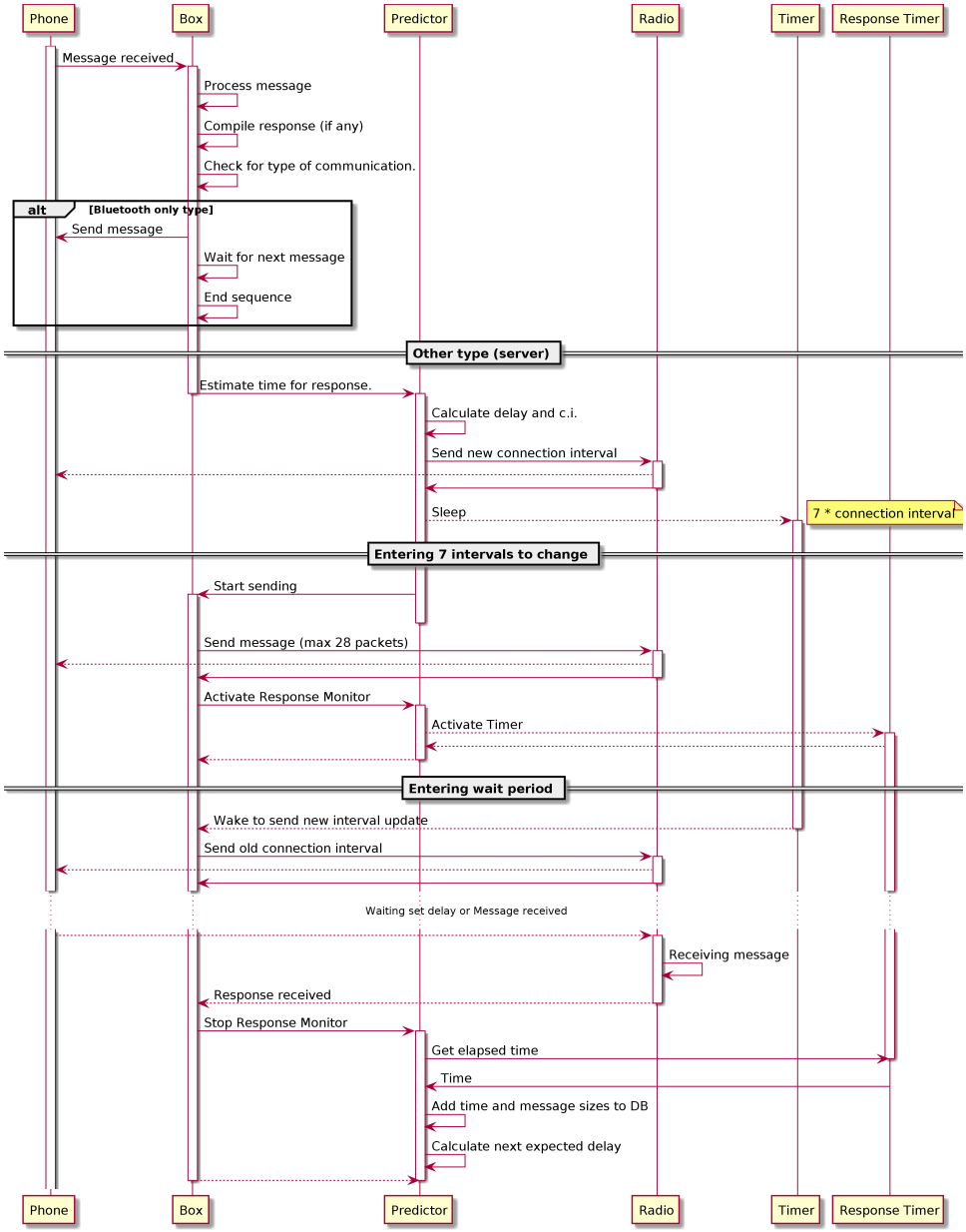

Figure 7.2: Sequence diagram of the start of receiving one message, sending a response and waiting again to receive the next response to that message.

the communication scheme in the Bluetooth LE stack. Figure 7.2 shows the sequence of events needed to achieve the sleep cycle, taking into account the specifics of the Bluetooth Low energy communication scheme. We should take into account that during the change of connection intervals we can still send data. We use this to send the response to the phone while already

switching to the next interval. Caution is needed as missing the 6 interval windows available will result in a significant delay.
Not included in the scheme in Figure 7.2 is the possibility of errors in communication. A bad link in either the Bluetooth link or rest of the network would create considerable more delay but for showing the principle involved we left this out of our schematic overview. The program does check for Bluetooth link quality and considers this in predicting delays. Bad connections do occur and can be mitigated by keeping the data rate higher. Predicting this can be done by checking the RSSI[1] values of the connection and adjusting the connection interval to cope with limited throughput.

By adding sleep cycles in the waiting period of every connection we tried to limit the time awake. The setup is the same (12 seconds) as in the baseline method and results in a connection/power profile as can be seen in Figure 7.3.

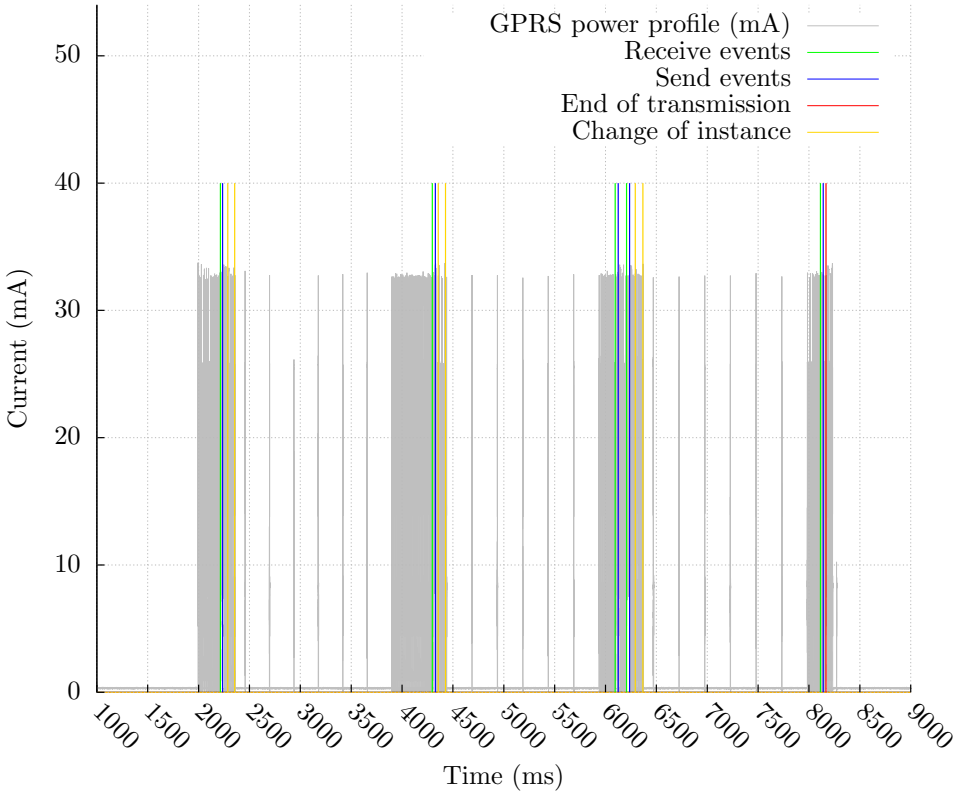

Figure 7.3: Complete 9 seconds connection profile of a GPRS connected back-end with sleep cycles enabled.

[1]RSSI or received signal strength indicator, is a measurement of the power of a received radio signal.

The connection back-ends are the same as with the baselines and provide the results as in Table 7.2 where the baseline is 100%.

Type	Avg. comm (s)	% vs base	Avg. power (J)	% vs base
GPRS	6.31	+4.6%	0.026	-144%
700-1000ms	3.36	+3.2%	0.022	-74.8%
UMTS	1.62	+6.2%	0.021	-17.2%
UMTS low power	1.57	+2.6%	0.020	-13.3%
HSUPA	1.09	-7.4%	0.020	+1.8%
WiFi	1.17	-	0.020	-

Table 7.2: Average communication time, power consumption and improvement compared to the baseline over 10 measurements of different connection simulations over a 12 seconds range with sleep cycles enabled.

The data shows that it is possible to get near to the 0.020 Joules provided the device sleeps as needed. Even six second long connection cycles using the GPRS connection are able to get down to 0.025 Joules which is a significant improvement as can be seen in Table 7.2 over 7.1. The WiFi connection as a back-end did provide some cycles in which it slept but this resulted in a longer over all connection time which did not significantly alter the power consumed during the cycle.

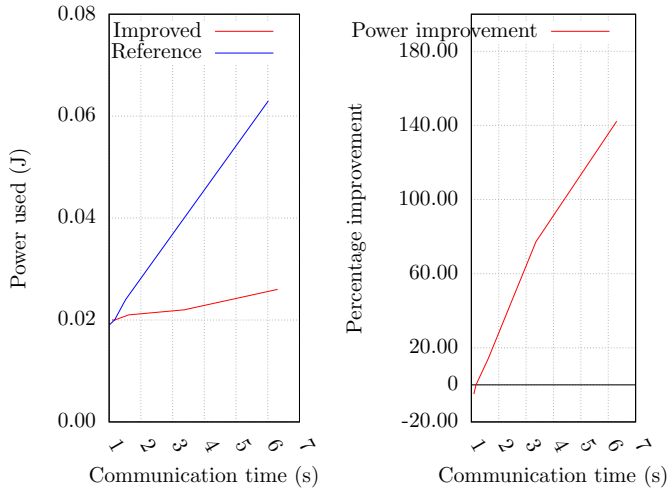

Figure 7.4: Power savings with sleep enabled compared to normal, set out over connection time.

The use of lower power settings for receiver gain and transmission power did give some advantage and a bit less gain in power savings. The overall

reduction in power savings versus the connection time is shown in Figure 7.4.

7.2 Lifetime improvements

From the data given in Section 7.1.2 and Table 7.2 it is possible to construct new lifetime estimates which are shown in Table 7.3. The table shows that with the sleep cycles enabled it is possible to extend the lifetime of the device to the requested 2 year operation. In the case of slow communication this is an improvement of 222 days. Another option would be to lower the connection advertisement to 2 seconds, but this limits the response time severely as it would guarantee a minimum connection time of 3 seconds and possibly longer and would thus limit our 6 second requirement in which the GPRS is already over the set limit.

Op./day	Comm. time (s)	Idle (J/day)	Comm. (J/day)	Tot. Days	% red.
68	4	11.54	4.85	622.3	-
68	6	11.52	7.27	542.7	-
68	UMTS (1.62)	11.56	1.36	789.5	7.5%
68	700-1000ms (3.36)	11.54	1.49	782.7	20.4%
68	GPRS (6.31)	11.52	1.77	767.3	30.5%
100	UMTS (1.62)	11.56	2.00	752.4	10.4%
100	700-1000ms (3.36)	11.53	2.19	743.2	26.4%
100	GPRS (6.31)	11.49	2.60	723.6	36.5%

Table 7.3: Lifetime results for varying communication time and operations/day. The reduction value (last column, in %) is 1 - (old days/new days). The first 2 values have been added for comparison and are from the baseline measurements.

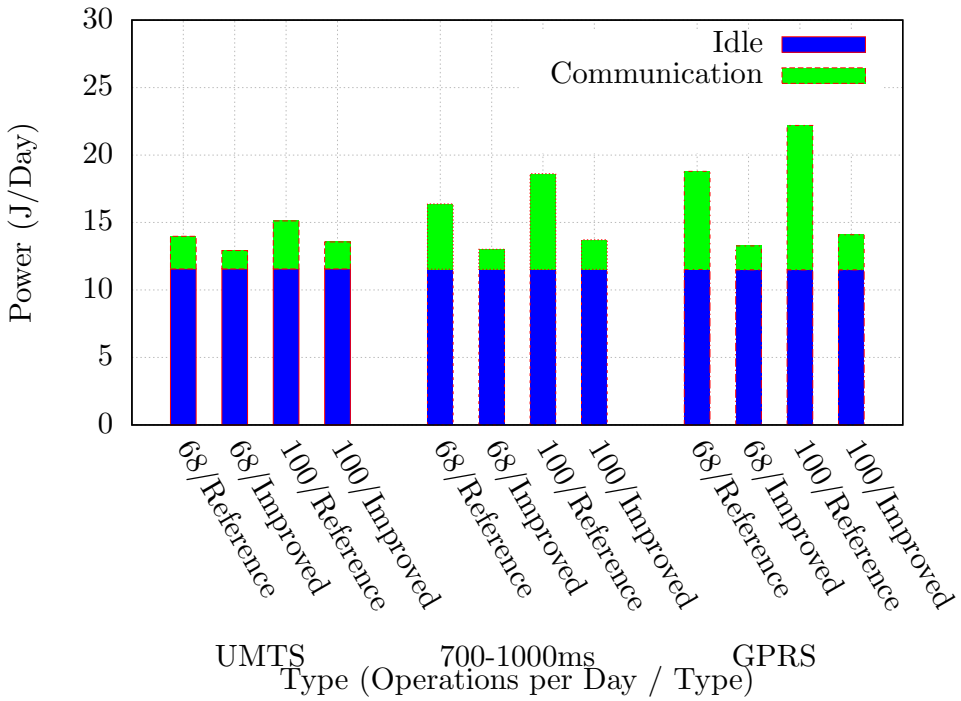

Figure 7.5: Power savings compared to estimations (2/4/6 seconds vs UMTS/700-1000ms/GPRS measurements).

Chapter 8

Conclusion, Discussion and Future Work

8.1 Conclusion

The sleep cycle proposed in 5.1.3 results in an improvement of the lifetime of over 35% and lower power consumption of 144% when the delays in the network are large. I chose to limit the size of the battery to a single AA size, but only with two CR2 lithium batteries instead of a regular AA Alkaline. This way the device can keep the functional requirements of 50000 operations, maintain a lifespan of over two years and stay mostly within the given reaction time. This fulfils the goald set in Section 1.2. Section 5.1.4 showed that with the sleep cycles enabled, the device only performed marginally worse (between 2.6% and 4.6% slower) in overall communication time. The power consumption reduced significantly during communication and in Table 5.5 it was shown that the two year minimum was achievable even with slow communication over GPRS connections. This was not the case without rate switching introduced in Section 6.3. The improvements also indicate that longer communication time would save even more power and thus increase the lifetime of the device. Also, if more cycles are needed before the end of the transmission as indicated in Figure 6.1 then the saving in terms of power consumption would increase significantly.

8.2 Discussion

There are some problems with the approach as indicated. One of the items which the systems has difficulties with is switching between different backend methods of communication. While there is a smaller difference between the power consumed during communication by switching the update frequency of the Bluetooth connection, the reaction time and global communication time would become similar to the slowest connection settings for all con-

nections. This is the result of the system guessing the linear regression as proposed in Section 6.4 over all connections made, regardless of discrepancies in speed and delay. A smarter system to do this would be advisable but the possibilities on the device are limited. A smarter guessing algorithms usually mean a greater complexity and possibly the need for floating point operations which are not available on the simple processor. If the phone in question would switch between networks, guessing would become more difficult and a smarter predictor would be needed.
A way to improve the system would be to select a short array of information so the history of connections and delays would not limit new connections to a faster historical speed or newer faster connections to an older slower speed. The trade-off in this is that slower communication time, and thus larger delays, generally mean lower power consumption. Since the users experience is affected by the speed of the connection, I do not advice increasing the connection delay to save power as it still uses more than a faster connection and the user might sense a slow device as six seconds is already quite a long time to open a door.

8.3 Future Work

Future work can focus on several different perspectives: the bluetooth stack, delay prediction and optimization in the used network protocol:

- **Bluetooth stack**: If one could allow the devices to update without using a large ‘instance‘, which is 6 on almost all devices, the devices can switch more quickly between intervals. This would however result in possible conflicts if the device switches too fast as an interval could be missed and the device disconnected. This can only be done at a very low level in the Bluetooth Low Energy stack and one would need access to a stack or build a stack.

- **Delay prediction**: If delays can be predicted more accurately it could save more power as there are less moments in which the device is already in a higher update interval and not receiving data. This might still be limited by the speed of the processor on the device.

- **Network protocol**: Using a network protocol with less transmission delays and less data overhead can increase speed and lower delays, specificly in the device itself and the phone. There is already a project under way at Ubiqu to reduce the overhead in the protocol.

Bibliography

[1] Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010.

[2] Bluetooth SIG. *Specifications of the Bluetooth system*, version 2.1 + edr edition, July 2007.

[3] Lujo Bauer, Lorrie Faith Cranor, Michael K Reiter, and Kami Vaniea. Lessons learned from the deployment of a smartphone-based access-control system. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, pages 64–75. ACM, 2007.

[4] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[5] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, pages 21–21, 2010.

[6] Efficiency breakthrough promises smartphones that use half the power. `http://www.technologyreview.com/news/506491/efficiency-breakthrough-promises-smartphones-that-use-half-the-power`, Oktober 2012. [Online; accessed 2013-06-03].

[7] SungWon Chung, Philip A Godoy, Taylor W Barton, Everest W Huang, David J Perreault, and Joel L Dawson. Asymmetric multilevel outphasing architecture for multi-standard transmitters. In *Radio Frequency Integrated Circuits Symposium, 2009. RFIC 2009. IEEE*, pages 237–240. IEEE, 2009.

[8] Bluetooth SIG. *Specifications of the Bluetooth system*, version 3.0 + hs edition, August 2009.

[9] Bluetooth SIG. *Specifications of the Bluetooth system*, version 4.0 edition, June 2010.

[10] IEEE Computer Society, 3 Park Avenue New York, NY 10016-5997. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, ieee std 802.11-2012 edition, March 2012.

[11] ZigBee Alliance. *ZigBee Home Automation Public Application Profile*, revision 26, version 1.12 edition, February 2010.

[12] ZigBee Alliance. *ZigBee IP Specification*, 13-002r00 edition, February 2013.

[13] Ant+ mobile environment. `http://www.thisisant.com/business/opportunities/mobile//?android-api/`, May 2013. [Online; accessed 2013-05-27].

[14] Roy Want. Near field communication. *Pervasive Computing, IEEE*, 10(3):4–7, 2011.

[15] Diogo Remedios, Luís Sousa, Manuel Barata, and Luís Osório. Nfc technologies in mobile phones and emerging applications. In *Information Technology For Balanced Manufacturing Systems*, pages 425–434. Springer, 2006.

[16] Bluetooth 4.0 support comes to the nexus 4, might be headed to stock android too. `http://www.engadget.com/2013/05/15/lg-nexus-4-bluetooth-4-0/`, May 2013. [Online; accessed 2013-05-27].

[17] Bluetooth low energy overview. `http://msdn.microsoft.com/en-us/library/windows/hardware/jj159880(v=vs.85).aspx`, May 2013. [Online; accessed 2013-05-27].

[18] Information technology Telecommunications and information exchange between systems Near Field Communication Interface and Protocol (NFCIP-1), 2013.

[19] Various authors. List of nfc-enabled mobile devices. `http://en.wikipedia.org/wiki/List_of_NFC-enabled_mobile_devices`. [Online; accessed 2014-04-13].

[20] Bluetooth SIG. Bluetooth smart and smart ready products now available. `http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices-List.aspx`. [Online; accessed 2014-04-13].

[21] Alain J Martin, Mika Nystrom, Karl Papadantonakis, Paul I Pénzes, Piyush Prakash, Catherine G Wong, Jonathan Chang, Kevin S Ko, Benjamin Lee, Elaine Ou, et al. The lutonium: A sub-nanojoule asynchronous 8051 microcontroller. In *Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on*, pages 14–23. IEEE, 2003.

[22] Texas Instruments. *OMAP5430 Multimedia Device Engineering Samples 2.0*, swps052f edition, May 2013.

[23] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[24] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[25] MA Razzaque, Chris Bleakley, and Simon Dobson. Compression in wireless sensor networks: a survey and comparative evaluation. *ACM Transactions on Sensor Networks (TOSN)*, 10(1):5, 2013.

[26] Sujesha Sudevalayam and Purushottam Kulkarni. Energy harvesting sensor nodes: Survey and implications. *Communications Surveys & Tutorials, IEEE*, 13(3):443–461, 2011.

[27] Sumit Garg, Manish Kalia, and Rajeev Shorey. Mac scheduling policies for power optimization in bluetooth: a master driven tdd wireless system. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 1, pages 196–200. IEEE, 2000.

[28] Luca Negri. The power consumption of bluetooth scatternets. In *In Proc. IEEE Consumer Communications and Networking Conference (CCNC 2006*, pages 519–523, 2006.

[29] Luca Negri, Mariagiovanna Sami, Que Dung Tran, and Davide Zanetti. Flexible power modeling for wireless systems: Power modeling and optimization of two bluetooth implementations. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pages 408–416. IEEE, 2005.

[30] Lakshmi N Chakrapani, Pinar Korkmaz, Vincent John Mooney III, Krishna V Palem, Kiran Puttaswamy, and Weng-Fai Wong. The emerging power crisis in embedded processors: what can a poor compiler do? In *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 176–180. ACM, 2001.

[31] Massoud Pedram. Power optimization and management in embedded systems. In *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, pages 239–244. ACM, 2001.

[32] Fiona Fui-Hoon Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.

[33] Texas Instruments. *CC256x Bluetooth and Dual Mode Controller*, swrs121b edition, May 2013.

[34] Single chip, dual-band (2.4 ghz / 5 ghz) 802.11 g/n mac/baseband/radio with integrated bluetooth 4.0, nfc + fm receiver. `http://www.broadcom.com/products/Bluetooth/Bluetooth-RF-Silicon-and-Software-Solutions/BCM43341`, June 2013. [Online; accessed 2013-06-05].

[35] Texas Instruments, Post Office Box 655303, Dallas, Texas 75265. *CC2540/41 System-on-Chip Solution for 2.4-GHz Bluetooth low energy Applications*, swru191d edition, March 2013.

[36] Nordic Semiconductor, P.O. Box 2336, 7004 Trondheim, Norway. *nRF51822 Multiprotocol Bluetooth 4.0 low energy/2.4 GHz RF SoC*, product specification v1.2 edition, May 2013.

[37] Jan Waclawek. The unofficial history of 8051. `http://www.efton.sk/t0t1/history8051.pdf`. [Online; accessed 2013-06-11].

[38] Blue Radios, 7173 S. Havana Street, Suite 600, Englewood, CO 80112, USA. *Bluetooth 4.0 Low Energy Single Mode Power-Optimized Class 1 SoC Module*, 2012.

[39] Blue Giga, P.O. Box 120, 02631 Espoo, Finland. *BLE112 DATA SHEET*, 1.3 edition, May 2013.

[40] Blue Giga, P.O. Box 120, 02631 Espoo, Finland. *BLE113 PRELIMINARY DATA SHEET*, 0.55 edition, May 2013.

[41] Inc Free Software Foundation. Gcc, the gnu compiler collection. `http://gcc.gnu.org/`. [Online; accessed 2014-04-29].

[42] Chris Lattner. The llvm compiler infrastructure. `http://llvm.org/`. [Online; accessed 2014-04-29].

[43] Robin Heydon. *Bluetooth Low Energy The Developer's Handbook*. Prentice Hall, first printing edition, 2012.

[44] National Instruments, 11500 North Mopac Expressway Austin, Texas 78759-3504 USA. *DAQ M Series Manual*, April 2009.

[45] Koen Langendoen and Andreas Meier. Analyzing mac protocols for low data-rate applications. *ACM Trans. Sen. Netw.*, 7(2):19:1–19:40, Sept. 2010.

[46] Johan Pouwelse, Koen Langendoen, and Henk Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, pages 251–259, New York, NY, USA, 2001. ACM.

[47] Hugo Krawczyk, Ran Canetti, and Mihir Bellare. Hmac: Keyed-hashing for message authentication. 1997.

[48] K. J. Widiner. *TECHNICAL REPORT ECOM 01605-F DEVELOPMENT OF GENERATOR DIRECT CURRENT G-63 ()/G (HAND CRANKED)*. Varo Inc Electrokinetics Division, Santa Barbara, California, USA, March 1967.

[49] Texas Instruments, Post Office Box 655303, Dallas, Texas 75265. *Step Down Converter with Bypass Mode for Ultra Low Power Wireless Applications*, slvsac3c edition, May 2011.

[50] Peter Benko, Gabor Malicsko, and Andras Veres. A large-scale, passive analysis of end-to-end tcp performance over gprs. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1882–1892. IEEE, 2004.