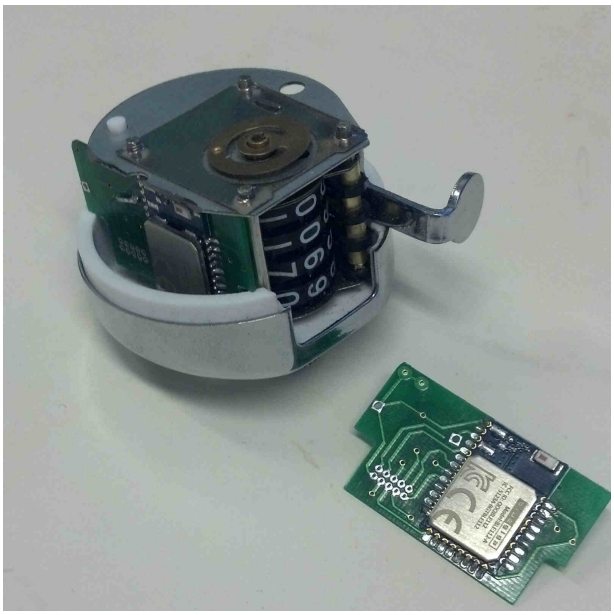
CHALMERS



Digitising statistics handling through hardware and mobile web technology

Master of Science Thesis in Programme Computer Systems and Networks

Tomas Ohlson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

Digitising statistics handling through hardware and mobile web technology

Tomas Ohlson

Examiner: Olaf Landsiedel

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

Abstract

Today mechanical tally counters are used to collect statistics. However, data stored in the mechanical counter is hard to overview and draw conclusions from. This thesis presents an approach which utilises a wireless low energy protocol to transfer the data of a mechanical counter to a smart phone and then to a server. The data stored in the server is distributed and displayed to the users of the service in a lucid and comprehensible manner. The goal is real-time communication and distribution of the data while optimising energy consumption, to the point where the battery of the tally counter can last several years. The focus of this thesis is evaluating the balance between energy conserving approaches and real-time communication.

The thesis concludes that Bluetooth low energy is the best suited technology for the task. Among the approaches tried, its star topology gives good battery life, real-time communication and easy configuration.

Keywords: Bluetooth low energy, energy efficiency, cross platform, smart phone

Acknowledgements

I would like to thank my examiner and supervisor of this master thesis, Olaf Landsiedel for his expertise and advice during the project. I would also like to thank Per Hallgren, Jonathan Ström and Alexander Widar for their invaluable support in the many technical issues arising software wise. Creating the hardware many issues was set right by the many engaged students and alumni of ETA helping with all kinds of hardware issues.

Tomas Ohlson, Göteborg 06/06/13

Contents

1

Introduction

This master thesis project was originated from its author. The idea comes from a project built during the autumn of 2013 but grew into this thesis work. The older project utilised the now 15 years old Bluetooth classic (Bluetooth 2) inside of a mechanical tally counter. It has many problems, unsolvable with the implemented technology. Due to advancements in energy conserving technologies and mobile platforms, this thesis takes the lessons learnt from the previous project and aims to develop the platform with more efficient technologies.

1.1 Purpose

New low energy protocols like Ant or Bluetooth low energy, opens up new possibilities for applications that has been technologically latent for some time. This project studies the potential benefits and drawbacks of these wireless technologies in context of sending the state of a mechanical tally counter. The application on the handset, and back-end on the server, utilise real time communication with the mechanical tally counter. For each event a long chain of events is triggered which is also studied. There are challenges in terms of battery life, having a responsive system and the related security implications of utilising a device connected to the Internet of things.

1.2 The current technology

Statistics are commonly gathered manually with mechanical tally counters and handled with pen and paper. In this project the process is digitised in order to to evaluate what gains such an approach would have. A system requires a chain of events across several mediums to take place enabling to get and save the data in a digitised way.

It is essential to get the data in a simple format to be able to observe trends not previously hidden for the average user with the statistics in paper form. The data is either written down or manually handled. The human factor leads to loss of information and is costly as it takes manual labour. As many users might interact with each other the data needs to be stored fast and displayed back in real time.

1.3 Problem formulation

The thesis aims to investigate how a digitalised counter system compares to a conventional mechanical system. To be able to compare the two, some problems needs to be investigated further. The existing mechanical system has no battery or bandwidth constraints, which one could argue makes it simple. The digitised approach should meet the performance demands set for real time communication and its set battery restrictions in order to be evaluated.

These questions are evaluated and answered.

- What constraints will a low energy technology set on the system?
- Will the new system come with any drawbacks or limitations for statistics collection?
- How will the system perform in terms of connectivity, response time and reliability?

1.4 Goal

Counting with a modified conventional tally counter makes the data available digitally. All data should, from the users hand, to the digital graphs be confidential, have integrity and high availability. The user experiences an easy and intuitive user

interface, ensuring simplicity throughout the system. Another important part of the overall properties is that the system is inexpensive, power efficient and durable.

1.5 Contributions

The goals above state motivations for the project and the need for evaluating the technology in its context. Results from this work reveals that it is possible to collect data that was typically lost or in comparison inefficiently collected. It implements a low energy technology giving the mechanical counter potential battery life of years. The mechanical counter is coupled to the cloud via a handset where the data is stored. As a system it can distribute data from one mechanical counter to any user who wants that data. The data could for instance automatically adjust an entrance fee for a night club which counts all visitors entering and setting a price dependent of how full it is.

1.6 Outline of thesis

The digitising statistics handling through hardware and mobile web technology thesis focuses on the design of the system. The goal is to implement a working product, which makes the design very important.

- Chapter 2 The chapter explains the concepts surrounding Bluetooth low energy and the Internet of things.
- Chapter 3 Includes both an analysis of Bluetooth low energy power consumption and several market oriented implementations made with it.
- Chapter 4 The chapter handles overall approach, the results and later detected problems of previously made decisions.
- Chapter 5 concludes the thesis by considering what it accomplishes and presents possible future applications.
- Chapter 6 contains the final conclusions

The thesis starts off with background, leading up to the design and later on the evaluation.

2

Background

A trend in technology is that more and more things are connected to the Internet. Many areas such as the tally counter or the waterbottle etc. still lag behind and have much to gain from using Internet connected technologies to automate processes such as the connected waterbottle [1]. Tally counters are used in varying occasions and are just another example of something that greatly benefits from automation. In this chapter the core concept of the thesis are explained, to later be set into context in the design and evaluation phase.

2.1 The Internet of things (IOT)

The IOT expression was coined in 1999 by Kevin Ashton, which created the global standard for radio frequency identification (RFID)[2]. The vision has been expanded since then and includes most Internet connected devices today rather than just RFID which was the reason for its inception. Many other examples of IOT exist like smart grids and most advanced modern sensor systems. In laymans terms, the internet of things is simply put that all things from the dishwasher to the mouse-mat could benefit from being connected to the internet and may for example talking to eachother without human interaction. The three different concepts are explained later in this section. Another widely adopted name for the IOT trend is the machine to machine (M2M) paradigm. M2M means that different machines share information with each other over standardised protocols drawing conclusions themselves and possibly making decisions. The decision pro-

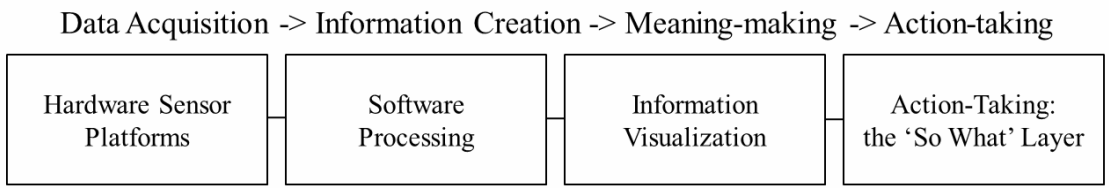


Figure 2.1: The process of machines talking to machines and taking action

cess is described in figure 2.1. First data is acquired and information is made from that data, the information is used for figuring out a meaning of the data collected which is at last used for action-taking.

Machines can now handle data without a human hand open the possibility of utilising large amounts of data in a way not previously possible. Since 2008 the number of devices connected to the Internet has overtaken that of connected people. With more devices connected follows greater challenges in the data analysis and security areas. The IOT vision has three primary areas.

- *Things-oriented:* Is based on unique product identification and tracking. RFID is often used as a synonym for the trend, even though the subject is far wider.
- *Internet-oriented:* Corresponds to construction of the IP protocols for enabling smart objects, which are connected to the Internet. This trend supports the ”web of things”, where web protocols and interfaces can control physical objects.
- *Semantic-oriented:* The core concept is the separation between the meanings of data from the actual data itself. This means that standardised resource descriptions of physical ”things” aggregated to a useful virtual object make it easier for users to actually utilise the available information.

Some of the areas that might be fundamentally changed according to Aggarwal et al. [2] are

Product inventory tracking with the use of RFID as all physical objects will have a unique identifier in the digital realm making it easy and inexpensive to track as it travels throughout logistics chains.

Smarter Environment Embedded sensors can monitor environment data like temperature or humidity etc and for example enable other machines in the area to regulate heating and ventilation.

Social Sensing Wearable sensors can monitor data about individuals. Examples of data might be location, speed, video etc.

The main issue comes with the amount of data suddenly available. Some data must to be confidential, other data should be widely available. Aggregating data and setting permission from the many different sources is hard, especially considering the possibility of information leakage in aggregated data. Confidential data might be possible to extrapolate from aggregated data through removing known data etc. Some services might be coupled to larger systems thus easily circumventing taken security measures. If the services are hacked the larger system may be compromised if precise care is not taken. A example is a mobile phone with all its services and applications that might be overtaken and can then channel malicious intent to other system.

One of the larger problems according to Aggarwal and Charu C. is that of noisy data. Data from the many different sources might be noisy and incomplete as well as redundant. The bad data has to be treated. If not treated, a large chunk of the data might be unusable and/or redundant. A lot of database space will be used for garbage data. Important solutions come with the emerging field of big data analytics. Sensitive information can be obscured in seemingly insensitive data, only to later be extracted by malicious forces. With modern algorithms the sensitive and bad information can be removed from the data, making the data trustworthy and more useful.

2.2 Bluetooth low energy (BLE)

BLE is becoming popular and ever wider spread since its adoption. Since the majority of smart phones have started supporting it [3] a great many new appliance areas have come. In this section the technological features are be described. According to Bluetooth.com [4] 100% of smart phones and 77% of PCs shipped in 2010 had Bluetooth. 100% of the smartphone market ,that will reach 700 million in 2015, will support BLE.

A breif walkthrough of the BLE standard is made in this section. Further information can be found in the cited paper by Gomez et al. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. [5].

2.2.1 What is BLE

History and brand

BLE is an open standard that started out as the protocol Wibree and was released by Nokia, a finish telecommunications company. Wibree was merged into the main Bluetooth standard in 2010 when the Bluetooth core specification Version 4 was adopted [6]. BLE is also called Bluetooth smart which is a branding from the Bluetooth special interest group (BSIG) to make consumers aware of the power efficient nature of the technology. The difference between Smart and Smart ready is that the latter supports both Bluetooth classic and low energy whilst the first just supports BLE.

Technology

The main focus of BLE is to make sensor data available over a long period of time and with a small physical device. It does this by implementing very low power consumption, fast connection times, reliable data transfer and RSA encryption of transactions, ensuring security. To setup a connection requires about 50 times less time than Bluetooth classic setups, while only consuming 5-10 % of the power. Frequencies used are all inside of the license free 2.4 GHz band making sure that inexpensive development is possible in all countries.

Reliable and robust connections are achieved through Gaussian adaptive frequency hopping[7]. Hopping allows the device to quickly hop within a wide frequency band. Advantages of this kind of hopping are that the interference is reduced and that crowded frequencies are identified and avoided. Broadcasting of BLE provides transmitting data in a reliable, connection oriented way. Other important areas are data privacy and integrity as wireless applications with mission critical requirements often have concerns within these areas. BLE design incorporates a high level of security as features enable possible authentication, authorisation, encryption and man in the middle protection.

Standard

As with other widely adopted open standard technologies BLE [3] has been thoroughly tested through the method of peers reviewing, making sure that no matter the manufacturer, BLE devices will still be able to communicate. A comparative table is available in specifications table 2.1. The theoretically unlimited number

Technical specification	Classic Bluetooth technology	Bluetooth low energy technology
Distance/Range	10-100 meters	10-100 meters
Symbol rate	1-3 Mbps	1 Mbps
Application throughput	0,9 - 2,1 Mbps	0,250 kbps
Nodes/Active slaves	7	Theoretically unlimited
Security	56 to 128 bit	128 bit AES
Robustness	Frequency-hopping spread spectrum	Frequency-hopping spread spectrum
Latency (from not connected state to send data)	100+ ms	as low as 6ms
Government regulation	Worldwide	Worldwide
Certification body	Bluetooth SIG	Bluetooth SIG
Voice capable	Yes	No
network topology	point to point, Scatternet	point to point, star
Power consumption	1 (reference value	0,01 to 0,5 (use case dependent)
Service discover	Yes	Yes
Profile concept	Yes	Yes

Table 2.1: A comparable table of Bluetooth classic and BLE

of nodes a master can have as slaves is more specifically defined as 5917 in the paper Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology by Gomez et al. (2012) [5]. The number of slaves are limited by the persistence of their connections defined in the standard as the connection interval which can be set as high as 4000 ms. The high number of nodes can only be achieved by directed connections, using a specific mac address to connect which has an average connection latency of 3 ms at 0,5 meters. The alternative is to scan for nodes in the vicinity and connect, which takes significantly longer.

2.2.2 Bluetooth classic, dual mode and low energy

Devices, such as headsets or speakers etc, already use Bluetooth classic which makes it is necessary for most handsets to have backwards compatibility to this technology. The Bluetooth 4 specification allows devices to implement either, or both BLE and Classic systems. Devices with a capability to implement both are called dual-mode devices. The dual-mode enables compatibility with Bluetooth 1.0 and newer.

Even though BLE utilises the same frequency (2,4 GHZ) as Classic it does not have compatibility due to the modulation system. Dual-mode devices use the fact that the same frequency is used and can use the same antenna even though the modulation is different.

2.2.3 Packet format

The packet format is a generic format defined by the standard. The standard packet can be seen in figure 2.2. The packet layout is explained with the following list.

- *Preamble:* is either 010101010 or 101010101
- *Access address:* advertisement packets make use of a fixed access address of 0x8E89BED6. Data packets use a random access address depending on the packet type.
- *PDU:* protocol data unit, depends on the packet type
- *CRC:* a 24-bit CRC checksum used to protect PDU

The packets sizes are used to estimate the performance of the whole system and to understand the custom advertisements made in chapter 4. The standard advertisement packet is depicted in figure 2.3, while the standard unencrypted data packet in figure 2.4. Smartphones advertisement packet can contain 0 to 31 bytes of data. The encrypted packet layout is left out as its not used in this project, but is a part of the standard.

2.2.4 BLE & classic state machine

The state machine is useful to understand when going through the logic of the app and hardware, and while reviewing the design and evaluation chapters of this

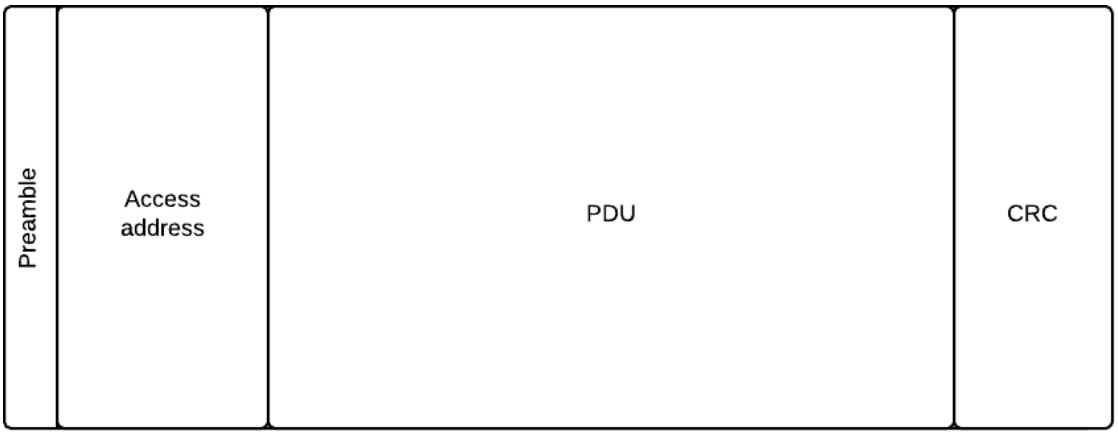


Figure 2.2: The generic packet layout of BLE

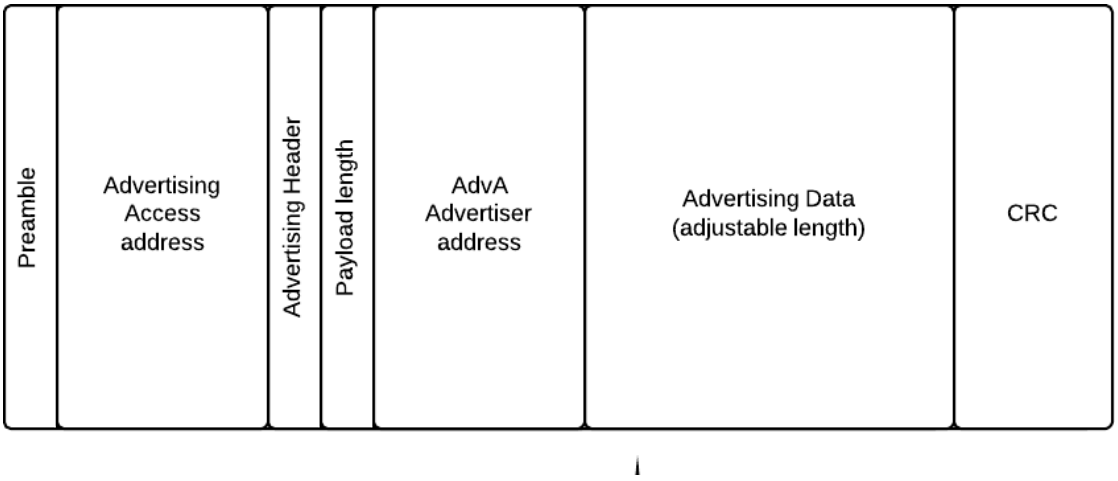


Figure 2.3: The advertisement packet

report. The state machine in Bluetooth classic seen in figure 2.6 is telling for some of the optimisations of the technology when comparing it to the state machine of BLE seen in figure 2.5. The main optimisation is the decreased number of states which makes the machine much simpler, and that in BLE all states can go directly to standby to save energy.

2.2.5 Generic Attribute Profile (GATT)

The GATT profile/database is similar to the much older Attribute protocol but encapsulates attributes into services and the data is exposed as characteristics. The main difference is that the attributes are customisable for the user while the older

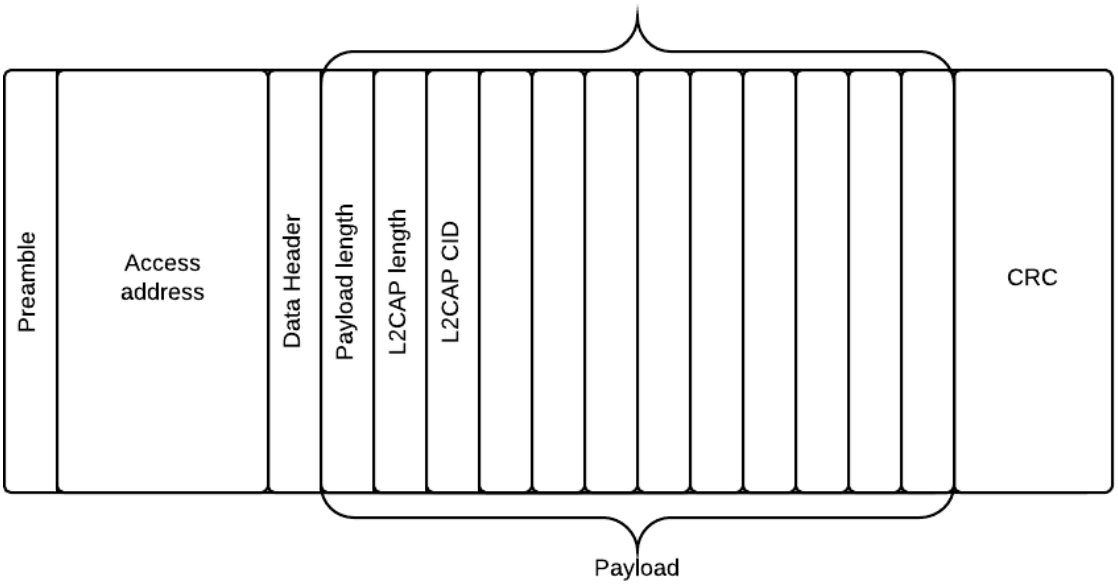


Figure 2.4: An unencrypted data packet can have 0 to 27 bytes of payload

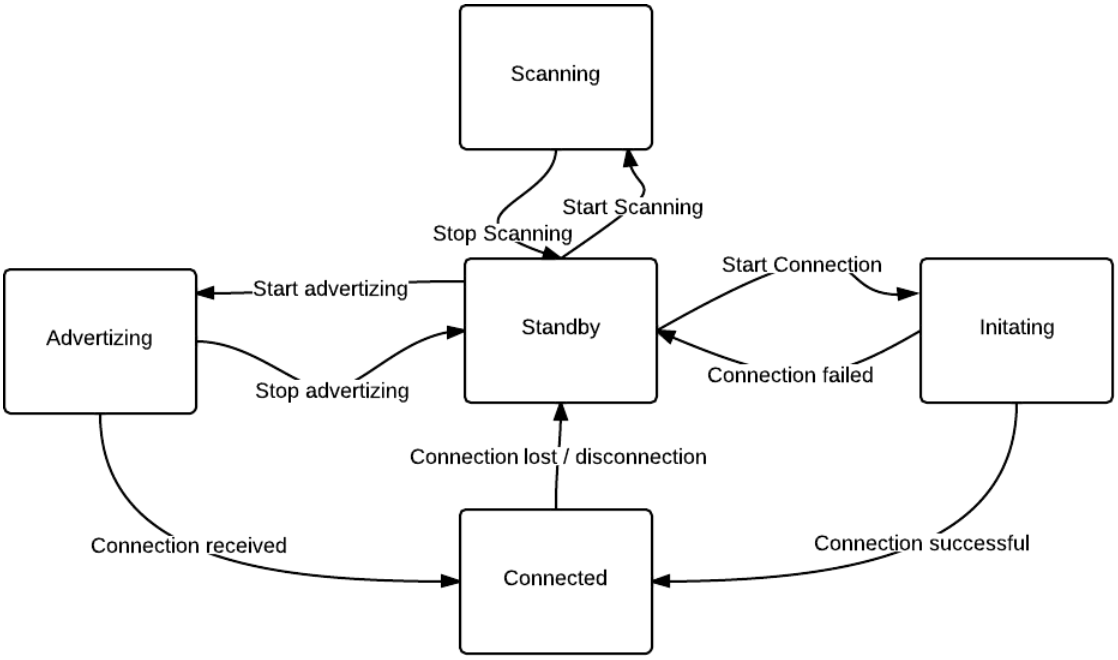


Figure 2.5: The link layer state machine of BLE

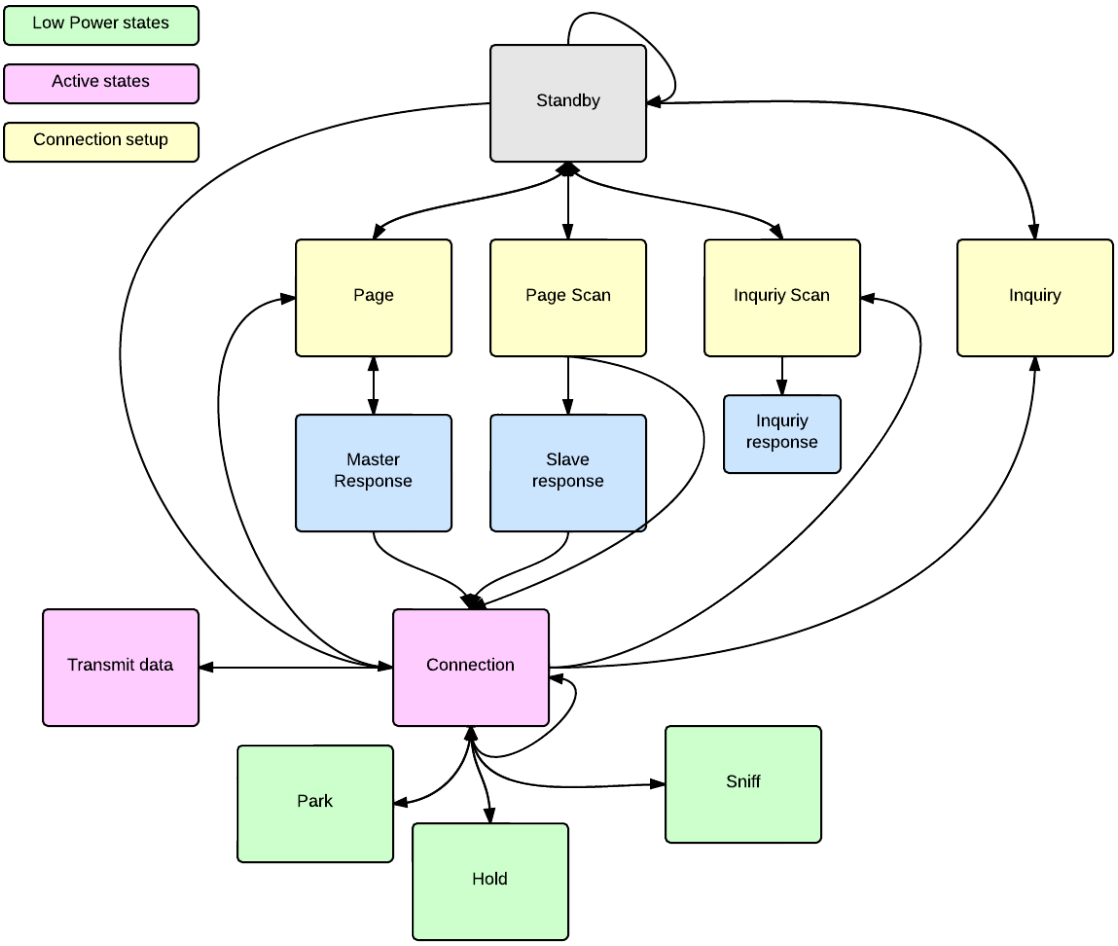


Figure 2.6: The link layer state machine of Bluetooth Classic, the blue boxes represent messages that can be read by any listening process

attribute protocol (ATT) had a preset number of attributes. The customisability makes it possible to create unique services visible only for clients who read from it.

2.2.6 Generic access profile

The database relations are depicted in figure 2.7. A service in GATT is a similar to a array with characteristics which can be said to be values in the array. Services are defined in the service specification at Bluetooth.org as a collection of characteristics and references to other services. There are primary services and secondary services, where the primary service is the service storing the data for the main function of

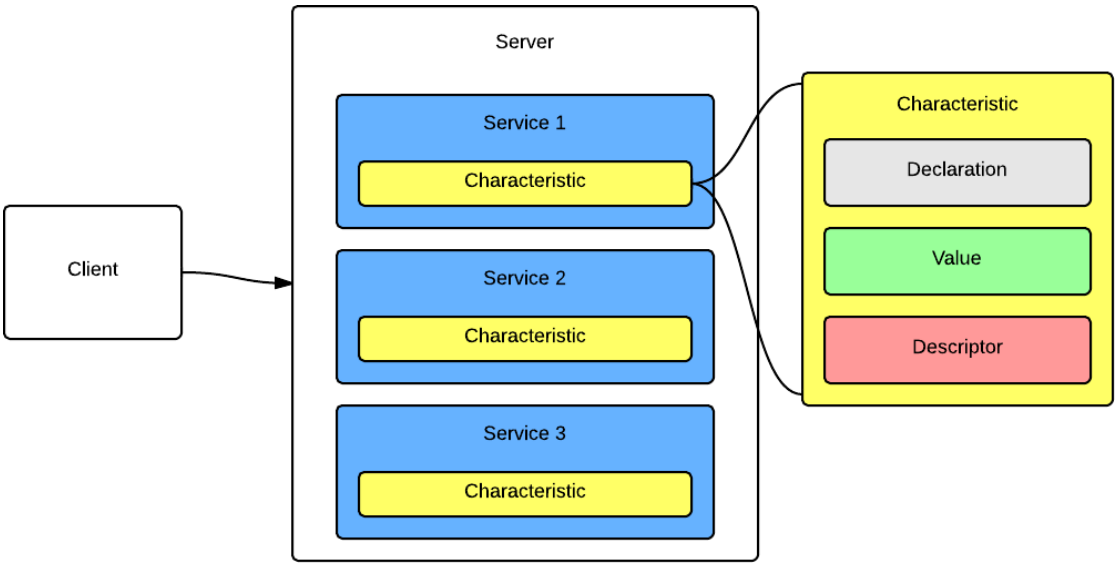


Figure 2.7: Gatt defines, services groups, characteristic groups, declarations, descriptors

the device. The profile defines the device roles. A broadcaster sends advertising events which includes characteristics and service data. The next role is observer. An Observer receives advertising events and listens for characteristics as well as service data. Neither the broadcaster nor the observers receive any data. The Peripheral role is always slave and does connect to some master which in GATT has the role Central. Both Peripheral and Central receive and send data, but the Peripheral advertises, in contrast to the Central which never advertises.

2.2.7 Bluegiga Script

Bluegiga, like many of the BLE chip manufacturers use their own scripts to simplify prototyping for their users. The language created by Bluegiga is abstracted ANSI C code which compiles to machine code. It supports all of the BLE specific functions and does have some restricted support for the rest of the C libraries, see figure 2.8. There are several layers of abstraction:

- First the Generic Access Profile, GAP, allows the management of discoverability and connectability modes and open connections.
- The Security manager provides access the Bluetooth low energy security functions.

- Attribute database a class to access the local attribute database.
- Attribute client provides an interface to discover, read and write remote attributes.
- Connection, provides an interface to manage Bluetooth low energy connections.
- Hardware, an interface to access the various hardware layers such as timers, ADC, and other hardware interfaces.
- Persistent Store, use to access the parameters of the radio hardware and read/write data to non-volatile memory.
- System, various system functions, such as querying the hardware status or reset it [8].

The architecture is designed in such a way that all BLE functions are event oriented. Each function generates callbacks after some event which generated data from the BLE hardware logic.

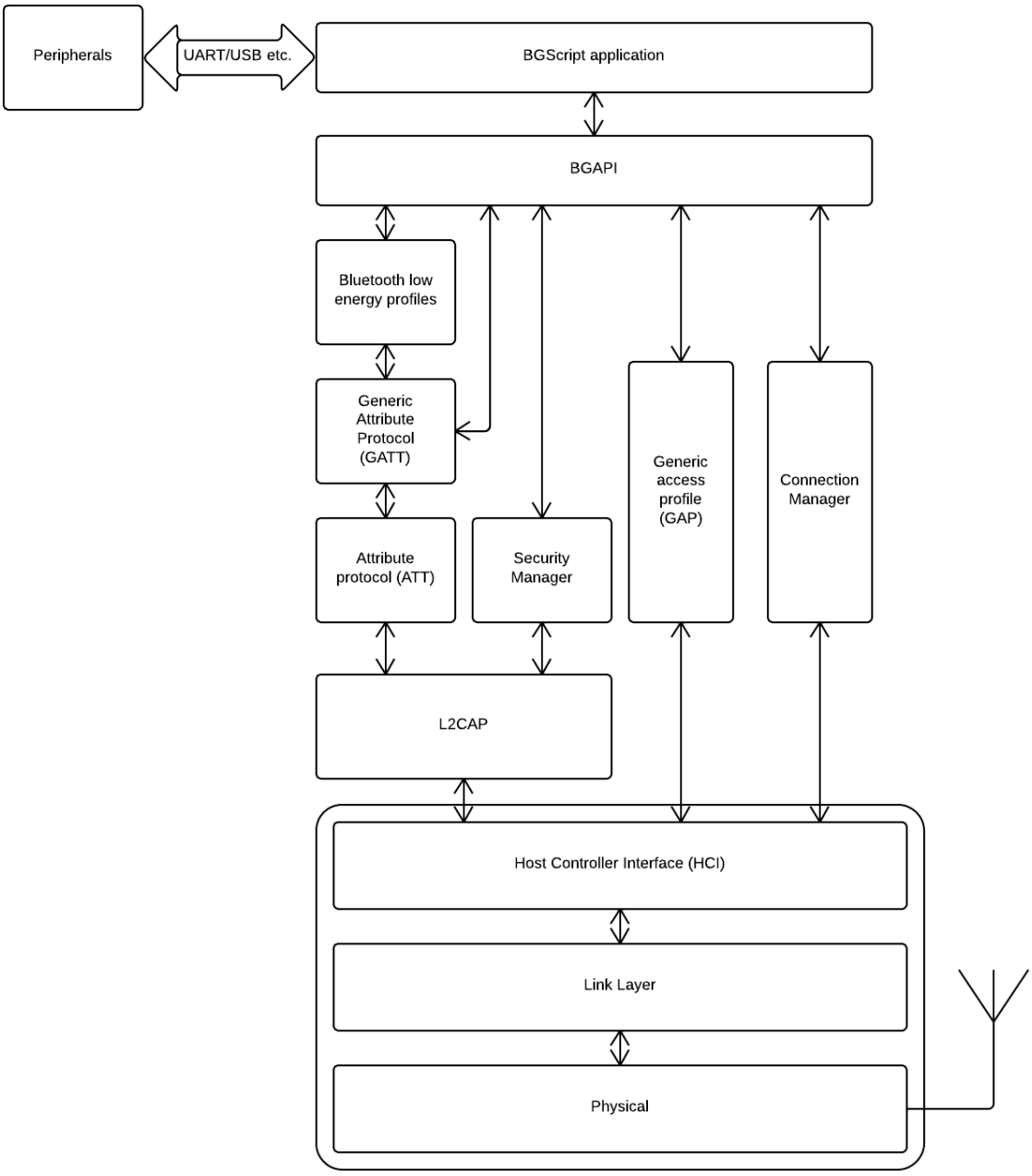


Figure 2.8: The Bluegiga script in single stack mode

3

Related work

Previous similar projects has been conducted in this area, for instance Stick N Find and Nike Fuelband. Beside the practical projects a few relevant studies are introduced in this section for understanding the capabilities of the technologies mentioned in the background. The work helped in finding the challenges, understanding the possibilities of the technology and evaluating the project.

3.1 Bluetooth low energy analysis

Previous studies in the area of low energy protocols has evaluated the performance and usage of the Texas instruments CC2540 BLE system on a chip (used in Bluegigas BLE112 and many other Bluetooth chips available). BLE technology is gaining traction in the area as wireless sensing applications are ever more useful with a connection to a mobile phone.

Compared to Bluetooth classic with 13 states in its state machine, as seen in figure 2.6, BLE is simplified giving less complexity and a easier development platform. BLE uses five states in the link layer state machine 2.5 making the complexity much lower and thereby making designs more power efficient with less code. When looking at all the features of BLE as stated in the cited paper by Elke Mackensen, Matthias Lai, Thomas M. Wendt about Performance Analysis of a Bluetooth Low Energy Sensor System, it can be concluded that the BLE technology is a big leap forward in the area. [9]

	Application	Battery life (days)
Stick N Find	Localise tag	365
Pebble	Watch	14
Fitbit Flex	Fitness	4
Nike+ Fuelband	Fitness	2
Heartbeat	-	-
Zaplox	Hotel door locks	-

Table 3.1: Table showing the products available on the market application area and battery life

3.2 Similar use cases

The focus of this project is the implementation and many of the already implemented projects gives design advice and helped with architectural decisions. Since the number of smartphones in the world has been greatly increased over the past several years, more and more applications and peripherals are appearing. As the computing power and connection bandwidth to the Internet is increasing every year, the applications become more advanced. Many of the solutions in this chapter have similarities with this thesis, with similar requirements and design goals. A summary of the projects described in this chapter can be seen in figure 3.1.

All the above projects have in common that they would not be possible without the use Bluetooth low energy or similar technologies for several reasons. Partly because of insufficient battery life with ordinary Bluetooth, the insufficient real-time feature or use-ability too complex for the user.

In the past it has been possible to make low energy protocols, but never before has the protocol had such a widespread adoption. Today Bluetooth Classic is supported by most phones, many cars etc. Bluetooth 4 and Bluetooth low energy are having an ever larger part of the market as the handset manufacturers' phase out Bluetooth classic as primary Bluetooth protocol. [10]. The projects below makes use of several of the technologies described in this thesis, the projects are described to understand the capabilities of the technologies.

3.2.1 Stick N Find

The device uses Bluetooth low energy in a power efficient mannor as is done in the theses project. With Stick N Find a user with an app on a smartphone the user can find keys or something else which has a Stick and find badge attached to it. Stick N Find is basically a BLE module strapped to a battery, and the app tracks the signal strength.

No coupling is required Stick N Find is able to couple any of their devices and leaves the authentication to the app. By sending advertisements as seldom as possible the battery lasts a full year. The application can't really tell if it's farther away or not but rather tell its user that the radio signal is stronger or weaker. Improved battery life is achieved by implementing efficient sleep patterns.

3.2.2 Pebble

Pebble is a smart watch that lets the users look at notifications on the watch or use its buttons to change song for example.

Pebble has the ability of scaling from a push notification mode, running on Bluetooth low energy, where little data is needed, to Bluetooth 3.0 High Speed where a lot of data could be sent. Pebble may also use the possibility of switching to Bluetooth classic if that's where two devices have compatibility. [11]. The dual mode chip works great with many devices but requires a much larger battery than what is available as the hardware solution used in this thesis project.

3.2.3 Nike fuelband & Fitbit Flex

Nike fuelband and Fitbit Flex are devices that are strapped to the wrist of a user where it monitors pulse, movement etc. The data is transmitted to a handset and analysed. The main product focus is fitness and health applications. Nike Fuelband and Fitbit flex work for a few days on one charge and continuously send data to the connected phone. The batteries are a rechargeable lithium polymer type with 75-105 mAh depending on the bracelet band. This makes for a nominal four days of standard usage, which is mentioned in the Nikes manual [12]. The rechargeable battery is a useful feature but requires action from the user in the form of charging it and keeping track of battery levels, making it harder to operate and makes the device more bulky.

3.2.4 Evaluation of a BLE heartbeat sensor

The heartbeat sensor sends data from a commercial grade heartbeat sensor and is supposed to replace an ordinary wired heartbeat sensor. An electrocardiogram monitoring system built with BLE is one of the first implementation utilising the iOS implemented Bluetooth stack. The research idea was to test the technology in a field dominated by other technologies to evaluate if it would improve the usability of the monitoring system. The results of the paper by Bin Yu and Lisheng Xu and Yongxu Li [13] in which the heartbeat sensor is evaluated, it concluded that the technology was ready for usage and proved it by testing several configurations with other existing technologies. The integration with the phone made it possible to send data in real time to the doctor of the patient, greatly improving care of the same.

The conclusions clearly state that superior battery life and usability of the device is fundamentally superior to other previous solutions making the technology the most suitable choice for a similar heartbeat sensor.

3.2.5 Zaplox

The implementation makes it possible to unlock any door with any Internet connected phone through a browser or app. Access is received through a main terminal with Internet connection, which makes it possible for the user to obtain an access key to its door in a app and open it via the main terminal which in turn communicates the command to the door lock which actives a engine for turning the lock. It makes it possible to save money. Administration with the mobile system requires no keys or other hardware. The founder Stefan Gripwall goes through the technology in the tech report in the cited paper [14].

4

Design

Design for the purpose of this particular project is defined as architecture, design and implementation. The chapter will go through the overall architechture, prestudy, design and implementation of the system.

4.1 Architecture

Designing the architecture for the system requires the study of its requirements and its general characteristics, as well as its individual parts. The pre-study of technologies as well as the scope of the architecture is included in this chapter.

4.1.1 Overview

To get a full overview of the system, its building blocks have to be seen as a complete chain. Each building block integrates with the others, but can then also be seen as standalone modules. The information flow starts with the user utilising the mechanical trigger to moving the inside mechanics triggering a digital event. The digital event is encoded and propagated to the handset. The handset sends the information as a authenticated request to the server back-end, which in turn decodes the data and stores it. The data is then pushed to or requested by the services users. The users use the data to construct graphs of the information. The process is depicted in figure 4.1.

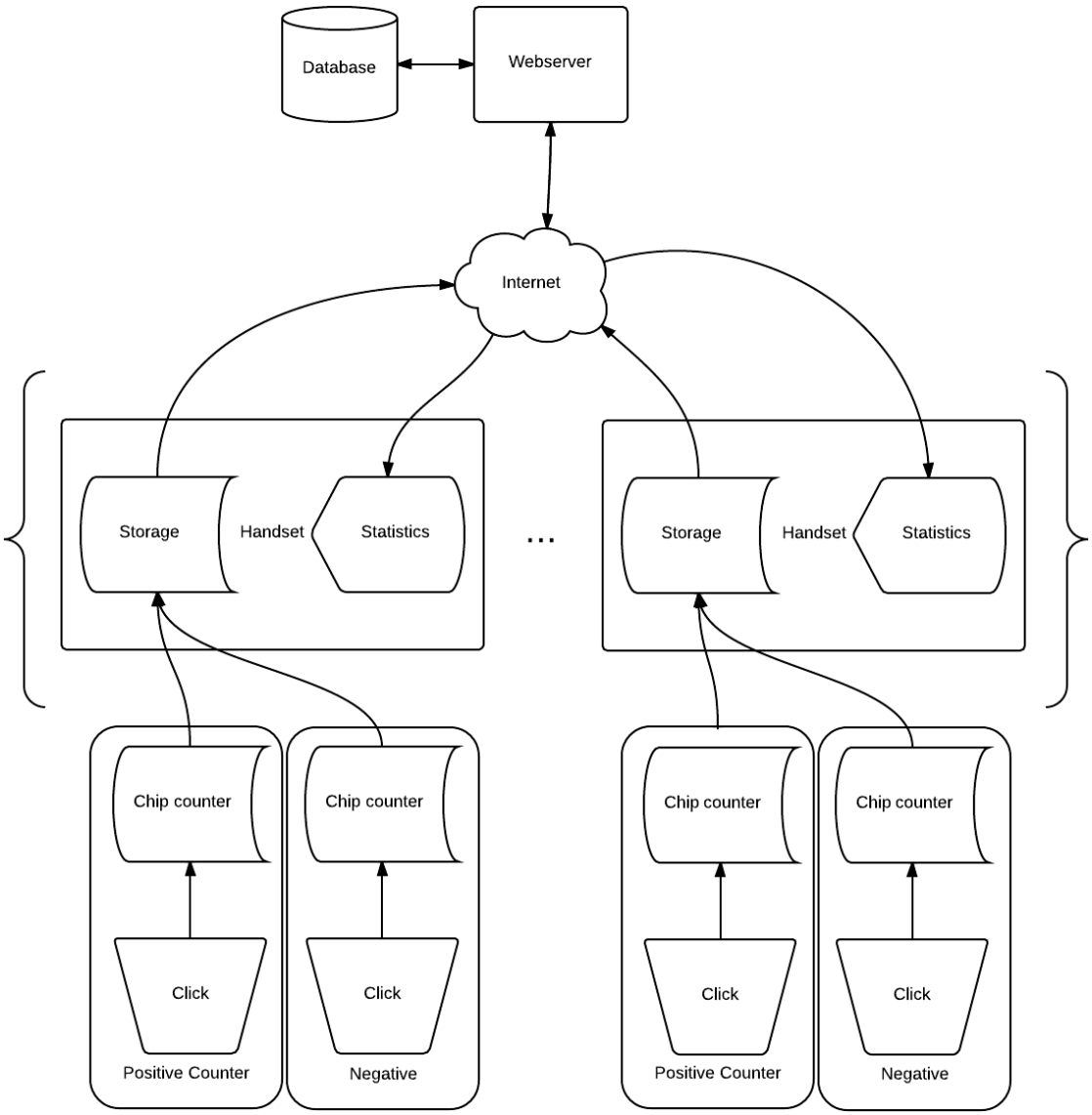


Figure 4.1: System dataflow of the architecture in detail.

Figure 4.2: A mechanical counter with a BLE chip

4.1.2 Technology pre-study

Before implementing the findings in this study, a great deal of research was put into the requirements of the technologies of the total system. Implementing a coupling to the mechanical clicker involves multiple design choices. There are decisions to be made considering connectivity, battery time, handset platform, back-end language as well as usability. The areas to be studied are:

- Wireless platform
- Handset platforms
- Platform specific language or cross platform implementation.
- Back-end platform

The pre-study is distributed over this chapter and discussed in each of the building blocks that are analysed. In the wireless platform study a previous platform existed and is considered in as a part of the study.

4.1.3 Mechanical counter and BLE chip

The sensor is the most significant part of the entire system. The foremost requirement being sending the user input to some storage. Sending the user input requires a correlation between events in the mechanical counter and the digital one. A good correlation makes sure that the data is accurate and usable for any kind of statistics. Error frequency of the data collection does not have to be zero, a 95% hit rate would still fulfil the purpose. The error frequency is a heuristic from actual usage which stems from the buffer of people that Swedish fire plan rules requires all buildings to have. [15] Because of fire rules it is important that the

counter display never runs out of battery, ensuring that its user always knows the state of the counter. The fire regulations set a limitation on using a mechanical as opposed to a digital counter in this project.

Keeping the digital counter synchronised with the mechanical counter is quite challenging as all available solutions require a battery or a lot of modifications to the mechanical counter. A clear limitation is the use of an unmodified mechanical counter, as the cost of re-engineering the mechanics were too great for this project.

One of the greater obstacles is the connectability, as the battery upkeep costs for a continuous wireless connection are great. The requirements of the connection is such that all events can be counted and eventually be stored. The connection has a certain bandwidth capacity for updating the data in the data store. A good heuristic for data synchronisation is that syncing occurs at least once a minute even though humanly experienced real-time is even better. The system has a good enough reaction time so that statistics mined from the data can be useful and granular enough to base decisions on an analysis of that data in its context.That is why the bar is set to at least once a minute and with a bandwidth great enough to send at least a basic integer with all the overhead required to do so.

As the counter is seen as a things oriented IOT item as described in the background 2.1, the architecture allows it to change its functionality by using some abstracted interface enabling software services to be enabled on the chip. With many counters may be used by a single handset each device needs a unique reference which enables the user to keep the data safe from other users. The mac address of the wireless chip will satisfy the uniqueness.

pre-study of wireless protocol

The first version of the system used Bluetooth classic and required a lot of space for the battery and printed circuit board. Those requirements was hard to meet as the result still required a lot of charging. The end product of that implementation takes much of time to build for the requirements to be meet. The system is economically infeasible. As the analysis of the old system was done, a lot of effort went into investigating alternatives. Several alternatives stood against each other. The alternatives were

- ZigBee
- Bluetooth Classic
- ANT

- Wifi
- Bluetooth Low Energy

All the alternatives meet demands on throughput and an ability to have more than one device coupled which is why that won't be discussed [16]. Listed technologies are evaluated below:

Zigbees primary focus is distributed low power radio networks, well suited for a task like this one. The technology does not take up a lot of space and can be run on a coin cell battery as the power consumption is measured in hundreds of nano amperes. The main problem is the handset connectivity, ZigBee is not a widely adopted as standard on smart phones making it quite unusable for the task. A option for utilising ZigBee would be building a dongle for smart phones, transferring the data in a efficient way. The problem with building the dongle is that it would decrease usability as different hardware interfaces would be needed for different phones. Another problem would be the cost associated with designing and building a dongle, it rules out ZigBee as a alternative.

Bluetooth classic was as stated before, the protocol on which the previous implementation was made. The advantages is that the standard is widely adopted and available in most handsets. With wide adoption follows good documentation and good examples, making it easy to build the software associated. Users knows what it is and how it works. The primary issues became clear after a simple prototype was built. The battery life was about 12 hours and then needed charging, Bluetooth classic was simply not very efficient compared to many of the alternatives. Another big disadvantage with high energy consumption is that a bigger battery which will be needed takes up lots of space and adds complexity to the construction.

ANT and ANT+ is proprietary protocol from Dynastream that is energy efficient and easy to implement as it can basically be treated as a black box in the network when it is fully modularised. ANT has in likeness to ZigBee not been widely adopted by handset manufacturers, maybe because of the fees associated with being proprietary. ZigBee seems to fulfill most of ANTs advantages, and would be preferred in most cases.

WiFi has a wide adoption and is mainly built for high speed networking and would require a large battery and was ruled out early in the comparison.

BLE turns out to have many advantages with the main one being connectivity. Since October 2011, Apples iPhone 4S has supported the BLE stack fully but since than a few other manufacturers has followed Apples example. Google an-

nounced that support for BLE will be integrated in Android before the end of the year 2013 [17]. The platform mostly has a good documentation among hardware for the machine coding. The reviewed BLE manufacturers does still have some documentation child sickness for some functions of the standard.

The cost of the chips is still a problem as the platform still does not have the matureness of Bluetooth classic. As with Bluetooth classic and other technologies prices comes down after some time and will later be standard for many devices. As stated in the background 100% of phones with Bluetooth connectivity will support BLE by 2015.

Choice of protocol

As the protocol was built with low energy consumption in mind primarily for the purpose of gathering sensor data, Bluetooth low energy fits well with the purpose of the system and is the clear choice of this comparison. The handset part of protocol choice is explained in section 4.1.4.

After choosing wireless protocol the only chip considered in this report is Bluegigas BLE112 running Bluegiga script, BGScript. The hardware is not going to be considered other than by its reference in discussions about battery consumption and choosing of language. It is assumed that there are errors in the hardware which occurs with some probability. Authentication of the wireless messages is handled by the protocol itself.

4.1.4 Handset platform

The handset has several base requirements, such as being able to communicate with the mechanical counter in an energy efficient way. Another important feature is that the communication is standardised and work with many devices in the market, enhancing the usability and cost efficiency of the device. Pebble utilises a dual mode chip for communicating with all devices with bluetooth making ideal use of the bluetooth technology. Pebble utilises the broad availability of bluetooth classic while also accessing the power efficiency of BLE. The solution is quite complex compared to other projects studied in this project but makes sure most handsets works with the solution.

Syncing logic of merging new data could be done by the server to save the burden of syncing the many clients with local data stores. Instead of flushing data stores and keep track of the current state of the mechanical counter by receiving new data from the cloud is more efficient implementation wise.

The handset is also communicate with a database back-end where authenticated requests will be able to store data from the mechanical counter.

A more far-fetched goal is to be able to shut the clicker down depending on subscription keys, providing a software as a service platform (SAAS). As many APIs are far from mature a huge challenge is to make it work on the different handset platforms.

Rights of setting up an actual connection to the mechanical counter, such as keeping track of what access which user has to what counter, is handled server side with the user getting access keys from the server to be utilised by functions from the BLE standard such as directed connect.

Pre-study of OS platform

Currently two platform dominate the market, Android and iOS. Together they control over 90% of the world smartphone market as of 2013 [18].

iOS is chosen as the primary platform since it is the only platform with widespread BLE support. As secondary platform Android Motorola Razr with a Motorola BLE stack implemented does have some problems but is investigated for implementation. The reason for pursuing a Android implementation is because of its simple development environment in contrast to iOS platform dependent environment.

With many platforms of implementation and a non-intricate user interface a

cross platform interface is a good solution for saving time and making the interface generic, independent of platform. The decision will be analysed in chapter 4.1.4.

Pre-study of application platform

For making it easier to handle multiple platforms a ”cross-platform” is used during the project as it works with any platform less code is needed in the total code base, less code has a tendency to produce fewer bugs. A cross-platform can use the same app codebase over several OS platforms. The best established platform for cross-platform building is the open source framework PhoneGap which builds on the Apache foundations Cordova. The best known alternative is Appcelerators Titanium which makes better use of native user interface components. The primary advantage of PhoneGap is its reliance of web technology which opens up for use of Javascript libraries which can simplify development as tooling becomes easier in terms of libraries. Withe that background the choice falls on PhoneGap.

Javascript is known for its ability to ensnare its developers into complex file systems if not handled correctly. This project requires a great amount of complex data syncing with Javascript. To make the complex solution work with as few bugs as possible a modern framework is needed. Structuring the application views and code is simplified with a Javascript model view controller (MVC) framework. There are many available frameworks available for making the job, the consideration fell on several different frameworks, and no framework at all (HTML+CSS+Javascript).

- HTML + CSS + Javascript
- MarriotJS
- AngularJS
- BackboneJS
- SpineJS
- KnockoutJS

The features considered makes for easier developing, antagonistic with the problems that are associated with Javascript. Utilising RequireJS made it easy to import and handle the many libraries and their dependencies as well as the many views and models created with Javascript to modularise the code.

For plotting the data in appropriate charts there are several libraries available as well. The most commonly used open source chart tool is JQPlot. JQPlot is a flexible tool that can manage most of the ordinary graphs and does fit well with the requirements of the project.

Choice of OS & application platform

The pre-study showed that only one platform had a functional BLE stack at the time of the study, namely iOS. The operating system has several examples of implementations already seen, all of the implementations in the background has iOS applications. The cross platform Phonegap in combination with BackboneJS is a well proved solution. [19]

4.1.5 Server

The server mainly needs to handle storage of information and access privileges for the information as well as for the physical counters as mentioned with things oriented IOT in section 2.1. A service will have as many use cases as there are applications in the market, more it is important that permission handling is adequate. Service needs to be highly scalable as data from all mechanical counters will be stored in its database. Any service must have a high reliability for making sure that no requests are missed. The system might be used in any part of the world at any one time, meaning it must be functional at all the hours of the day.

Data noise problems are solved by keeping the state of the each counter at each event stored in the database to make sensor resets or other problems apparent.

A server will distribute the data to all nodes in need of updates on the data. There might be several other nodes requiring updates when new data is inserted into the database which makes real time communication close to a need. Servers needs to be capable of real time synchronisation as the users will depend on its real time data.

Back-end technology choice

As back-end there are many options to consider as framework maturity, security record, stability, documentation etc. What is needed for this study is a secure back-end with a good stability necessary for a storage back-end with simple data access from an API. With many peers accessing the APIs, any database changes

may be requested and the need to update data for the users will be equally pressing. To satisfy the need for updates of the data, real time communication is needed. A server with a stable easily implemented push data ability is a requirement in a server for this project.

- Scala with its advanced typing system and compile time checks running on the JVM makes it easier to write as few bugs as possible.
- Python is a efficient language which is easy to learn and has many useful libraries. It satisfies all prerequisites needed for this project.
- NodeJS is Javascript based web server. The main advantages with using NodeJS is that almost all code can be written in one language, removing some of the complexity gained from the many steps and layers in the topology of the app.

As to choosing among the proposed languages each one has opportunities and pitfalls. While both Python and Node are well known for their fast development rate Scala is dragged down in the short run. Scalas has advantages in the long run, larger projects with many developers can make good use of a well typed language to easier understand each others code. As the API requires good database scalability and secure code Scala is an appropriate language even though both of the other languages does fulfil the necessary prerequisites as well. In the choice of database the main contenders was first of all No-SQL and SQL. Many argues that one of the big advantages of No-SQL databases in general is that their unassociative nature gives better scalability than with SQL databases. A good example of this Google big Table which scales better than any SQL service has performed so far. There are advantages and disadvantages with both solutions [20]. The complexities for the implementation could potentially be higher for with No-SQL as relations are harder and more often leads to consistency issues. In this project the data structure is quite plain and has a maximum of a few levels. Among the No-SQL databases MongoDB is the simple to implement and is chosen for the project.

4.2 Setup

The setup chapter is the full overview of the implemented system. The compromises are made clear in the architectural overview of the system , figure 4.3. The system handles a event in the mechanical counter by sending a digital signal over Bluetooth with the latest counter value to the handset. The data is then relayed

through the Bluetooth's logic to the server where the data is aggregated and returned to the app to form statistics. The data flow is described in detail in this

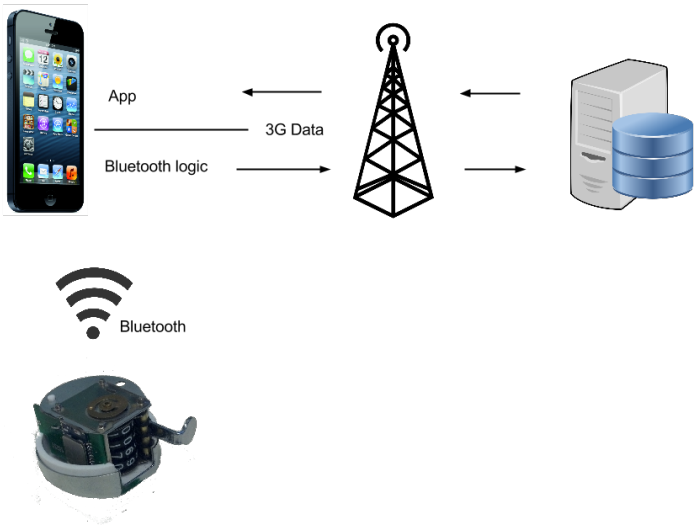


Figure 4.3: System dataflow

section as all the different parts are gone through.

4.2.1 Hardware

The pre-study showed that Bluetooth low energy is the only technology power efficient enough that could be coupled to any phone making it the choice. An implementation with Bluetooth classic is infeasible as it does not meet the requirements of the system. One of the main problems is availability as the battery life is about 12 hours. This id mainly due to the hardware needing recharging because of the protocol high battery drain. The charging adds complexity for the user and developer.

Confidentiality is also harder with authentication, being done with user password in Bluetooth classic. The implementation would not fit the requirements set in the architecture.

Little material is available for making a choice of chip. After looking at datasheets, blogs and forums the best suited chip for this study is Bluegigas BLE112 as its well proved by being one of the oldest BLE breakout boards. The

documentation is good and has been used in many older projects. The BLE112 contains a Texas Instruments CC2540 chip with a BLE radio module and a 8051 microprocessor on the same die, the same chip which most implementations of BLE is built. As BLE112 was released in 2011, implementations and help is available. Bluegiga provided several good code examples for getting started with the chip. Complementary to the chip coding examples is several applications available for different platforms making it easy to test hardware configurations.

The hardware's code flow goes from the wake up from a click to the sleep at the end of a wake up cycle, the link layer machine clearly show how it moves between the states 2.5. Figure 4.4 shows the specific flow of the hardware code in a simple fashion. The initialisation part which is made at power on and basically sets up the variables including the GATT database as well as the value buffers which are used for the counters and temporary storage of lengths etc.

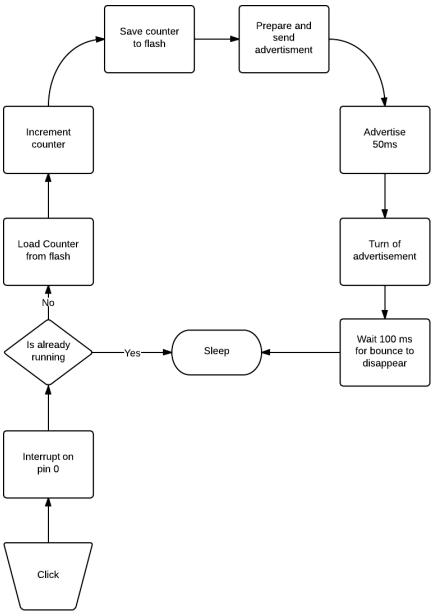


Figure 4.4: Flow of the hardware code

Coding language

The hardware and its code supports both C through a IAR (the Swedish company) compiling software, and Bluegiga own custom script, BGScript. The main problem with IAR is that it is very costly as the development environment as well as the hardware programmer hardware has to be bought. The C code offers good customizability as much libraries are available. The libraries enables different of

services and decoding of data making it easier to vary functionality depending on need, which would simplify power consumption optimisation in the future work, when the application is more complex.

The implementation in this project is made in BluegigaScript (BGScript) as it offers a simple syntax similar to bash or JavaScript. The language is event oriented, making it easy to find structure the language without the code getting long via the functions. The main problem of BGScript is the lack of advanced mathematical function which makes development tricky when developing more advanced functions, which won't be a problem during this project.

Bluetooth protocol

Standard BLE showed to have some disadvantages in the implemented context. As the mechanical counter is working in real time so should the communication, stated in the architecture setup. When setting up a connection the mean time for establishing a connection is 1.28s when returning from a sleep mode and setting up a GATT server. When in connection, this is not a problem and the data will be sent in real time. The problem could be solved through long polling and other optimisations but remained one of the contributing reasons for trying to make a custom protocol for solving the problems. Setting up long polling or similar proves to be hard as no implementation examples or use cases is available in the documentation. The custom protocol have a clear and working example, while proving a short connection time of a few milliseconds.

Customised protocol With a customised protocol the data is piggybacking (included inside of the message window) with each advertisement broadcasted which makes the messages visable without a connection. The piggybacking is widely known to be utilised when sending SMSes. A sent SMS message is piggybacked on a advertisement message sent to the base station, not costing any data to send as it fits inside the specified packets overhead.

Piggybacking the data in an advertisement message gives a very short response timing but requires specifying each byte specifically instead of relying on the standard protocol. Authentication is made by strapping on some unique id in the data which offers no security except for the obfuscation. As no connection is setup there can be no guarantees that the sent message actually arrives, the best way to make it more reliable is to resend. If resent enough times, the sender can with some certainty know that it will arrive, in this case heuristically taken number of 95% will make the system act reliable enough. Problems arise if many messages in a row don't arrive, meaning no real time data transfer can be set or promised.

The customised advertisement works good for many platforms but after testing iOS stood out as a platform which did not fit the requirements set in the architecture. More on this in sub section 4.4.3. With a harder implementation in which each counter is setup with some address to connect to, a direct connection can be made as a slave to the handset being set up as master. The connection time will be as fast as or faster than custom advertisements and a connection can be kept promising secure data transfer.

Energy management

The Energy management leaves many considerations to be made including everything from battery size to code run time. With BLE it is possible to use small amounts of energy as the primary purpose is collecting sensor data computing power is kept at a low. The state requiring the most energy is the transmitting/receiving state, keeping the time in that state at a minimum is key to keeping the consumption low.

The main motivations for choosing the chip over others is its power consumption of 400 nA in deep sleep and 900 nA in sleep providing a good ground for a long lasting battery life of the chip. To get a perspective, transmitting requires 24 mA, receiving 19.6 mA and idling 0.245 mA *CC2540 states:*

- *Transmitting* 24 mA
- *Receiving* 19.6 mA
- *Idle/Sleep state PM1* 0.245 mA
- *Sleep state PM2* 900 nA
- *Sleep state PM3* 400 nA

With the standard protocol as described in the previous section 4.2.1, much time can potentially be spent setting up a connection after a long sleep. As the long connection time will drain the battery a customised protocol could get the energy efficiency to another level. The customised protocol, with a 95% certainty of data being received on the other end, a mean transmitting time of 50ms is possible granting 25 times better energy efficiency in the transmitting/receiving state. The design space for different battery sizes is limited and the minimum solution uses a battery of 40 mA while the largest solution uses a battery of 225 mA. No matter the battery size the coding of modules is what keeps the energy consumption at bay.

One of the large problems detected during the implementation phase was how hard it is to measure the very small currents used in the chip. The currents are large enough to drain the battery but too small to measure with ordinary tools. At the later stages of implementation the battery status was saved with an ADC and sent to the handset making it easier to evaluate the battery drain of different solutions.

4.2.2 Handset

The handset part of the solution will be running a combination of native and web code only compatible with Apple iOS operating system. As the Bluetooth low energy stack is only implemented fully on iOS the solution will not work on any other platform until the stack has been implemented. An implementation was investigated for android but proved to be too hard, and was not further pursued.

iOS app

As Apple implemented the BLE stack in 2011 a lot of earlier work has already been done with it. This project implementation took its base from a heart rate measure app example supplied by Bluegiga. The workflow used is collecting the data and forwarding it to JavaScript in the web interface of the app which in turn forwards the data to the server via an ordinary HTTP request.

There are several problems concerning the iOS library. Even though the library fully supports the BLE standard, making the BLE process work in the background is still not possible if utilising a custom protocol described in 4.2.1. The library supports reconnecting to BLE services in the background and not listening to broadcasted messages. The problem became evident late in development as the documentation did not mention how ”backgrounding” handles Bluetooth running in the background. Even reading up on Apples design guidelines for BLE did not give any notice of this. As of the WWDC conference in June an extended backgrounding service seem to be coming with iOS 7 which might solve the problem. [21]

The implementation of the customised advertisement malfunctioned with the backgrounding mode. A rework of the customised protocol and energy consumption strategy needed to include the original protocol which iOS backgrounding features accepts. The finished protocol does a 1.8s transmit to set up the connection with the handset, enabling the iPhone to catch the transmitting service through the service id. Each time new data is put into the GATT database a indication is thrown to the handset making sure that the data is forwarded.

Android app

The Android app required lots of work as the Motorola APIs for BLE is not well documented. There are several examples available to get started with. The first problem to come with the examples was that the BLE devices was possible to find with the scanner but it took a long time, sometimes up to thirty seconds which caused the battery to be drained fast. As described in the iOS implementation the connection should take less than a second, which it did with iOS. When a device was found problems arises when functions from the APIs meant to connect to services simply return null. After much searching for a solution, a lot of answers seems to say that the API is broken.

None of the examples seem to work with the latest version of android to Motorola Razr (4.0.4) as the stack was originally implemented for Android 2.3.3 which seem not to have been fully upgraded. This conclusion came from searching Motorola forums for answers. The conclusion came after some time to abandon the project altogether and stick with an iOS only implementation.

Cross platform app

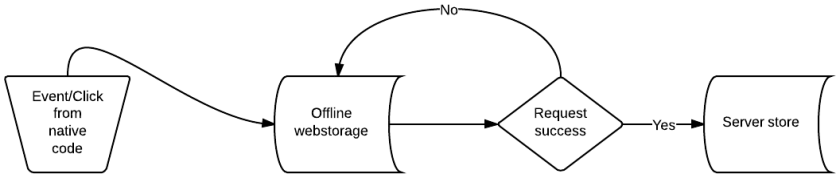


Figure 4.5: The flow of the web based app

A solution for a cross platform app started out as with building a common app for all platforms displaying some temporal modularisation of the events, and a tool for displaying statistics in graphics. Building the main platform was done with the setup tool Yeoman for setting up the boilerplate with BackboneJS using best practice standards.

The BackboneJS platform requires several plug-ins to be able to reach the demands of the architecture. Plug-ins used includes view manager and offline synchronisation enabling easy correlation between the remote database as well as the local database. Offline storage makes sure that any failed requests or pushes to the server causes no inconsistency problems. It utilises hashes as keys when storing data models of the data into the web storage of the mobile browser, this can be seen in a overview of the data flow seen in figure 4.5.

Click	
	Event id
	Time stamp
	Latitude
	Longitude
	Amount
	Machine
	Machine value

Table 4.1: The table shows the data saved from an event in the mechanical counter digitised

The data itself is pushed via platform native language to the JavaScript functions as PhoneGap has implemented none of the native BLE libraries for any platform yet. Once a function is activated the requests to the server is done accordingly.

When new data is sent to the back-end, the data comes back to all nodes who is set to be notified and have their data updated. Data is pushed to the clients to lower the latency and increase propagation speed of data. Data will via Backbones data models be aggregated and inserted by JQPlot into a standard HTML canvas. Plots are built in such a way that depending on the data the graphs will adapt to the structure of it. A user is able to modulate the looks of the graph.

An alternative UI implemented later just displays a graph in a web view inside of any app. It makes integration with other platforms simple but still utilises all benefits mentioned above.

Statistics

Statistics shown is calculated by the data received via the APIs. Many fields stored in the database makes many graph types possible most of which will not be used in this project. Data available for statistics gathering is showin in table 4.1. Data used is primarily the time stamp, the amount and the event id. A machine value is for keeping track of what values the machine is currently storing at each click. At each data instance the location is saved as well. Location data could be used in the future which will be mentioned in the evaluation chapter5.5. Keys and identification of the devices is kept in the standard user tables of the database, it

became the implementation in accordance with the things oriented IOT.

4.2.3 Back-end

A minimum viable API will implemented in the back-end, enabling peers to authenticate themselves to get the requested data needed to make simple graphs of the data. It is important that the back-end database is built in a quite general way as with semantic orientation explained in the background, the final representation to each user event might vary depending on implementation. Data aggregation might use other relationships than originally thought.

If some other device is coupled with the database or with other forms of data and other needs of rights management. A general implementation makes sure that it does not destroy the consistency of the database.

Implementation of the back-end is done with Scala and the other with Python, this was done as a unplanned design revision needed as the back-end was not built generally enough, this will be explained further in this section. The respective web servers used for the implementations are.

- Play for Scala
- Tornado for Python

4.2.4 Redesign of the handlers

An implementation was made using a secure log in module which made it simple to add any kind of OAuth authentication. Secure log in solutions has problems with strict APIs from the authentication provider, making it harder to authenticate automatically with back-end machines. Problems of the module did not show until trying to authenticate with other OAuth than Facebooks.

There is much competence in Python available at the laboration site which is a big advantage for Python. With Python as the second choice in the pre-study it is choosen for a fast reimplementation of a back-end. In addition, the documentations and community is far more developed than for Scala beginners.

Database

The setup of the database with MongoDB enables storing data easily through with the Python integration. As the data has a small depth and small amounts of data in each field, there will be many fields in the collections as each mechanical increment is saved. The retrieval of data is made with MapReduce queries in the document tree.

Scalability is inherent with MongoDB as it has built in support for replication with other servers. The easiest implementation of replication would be with a single master node relaying load to slave nodes handling consistency.

MongoDB integration

MongoDB integration chosen for this project has been elected on the grounds of popularity. Integration is basically an Object relational model (ORM) which converts an object to BSON in a structured way. When the object is stored it can thanks to the ORM be extracted and remade into the unique object it was before insertion. The main benefit of this is that development is greatly simplified as the database model made by their implementation can be abstracted.

In the original implementation Scala implemented towards the database with Cashbah and using the ORM tool Salat. The implementation was complex in relation to Python as all data needed to be type checked, slowing development significantly but working in a more secure way with the data. With Python the Pymongo plug-in did everything needed by the architecture and made storing data in a safe enough way.

Authentication

Authentication is crucial for the API as multiple different services might access the API in the future. With the ”Secure log in” module for the Play web server many different authentication services are available, but in this project the only implemented service is Facebook authentication. During the implementation it became clear that one of the main services are integration towards other platforms. As most other platforms in the area can answer towards an OAuth server if properly configured that became a priority within the authentication part of the project.

In the Python implementation of the back-end, a new authentication service was used, the service gives the server the ability to give out rights and not keep

records of the rights. The rights are instead saved by hashing making hashes of the rights with some unique identifiers, making sure that the user. The Python implementations authentication service will not be deeper discussed in the thesis.

Hosting

Hosting of the services is important as it needs to have high availability as well as a good scalability of performance and space. There are many different hosting platform options.

- Heruko
- App engine
- Compute Cloud
- Amazon EC2

For the purposes of this project the most suited platform is Amazon EC2 as it is well tested and can host the database on the same server. The main advantage to Google Compute Cloud is that it is a proved platform that has existed for many years running services in the range from NetFlix which is one of the most demanding services of the Internet to the location based social tracking service Foursquare. Running a micro instance Linux server makes it possible to make a flexible platform which can be duplicated and possible to load balance with higher loads. Even though both Heruko and Google app engine scales automatically it is better to keep the data store closer to the actual server as its done on a EC2 instance. [22]

API functions

The API covers most of the simple features needed even though the can be extended to cover many more aspects and correlations of the data. The simple list API calls available is.

Get User For getting the user data

Get Clicks Getting Clicks of some event

Post Click Posting the clicks

Get Event For getting the events

Post Event Post events with event id etc

Get Events Gets all the events of the user.

The implemented handlers are quite simple and the queries to the database are performance tested so that no single query takes a large amount of the available performance.

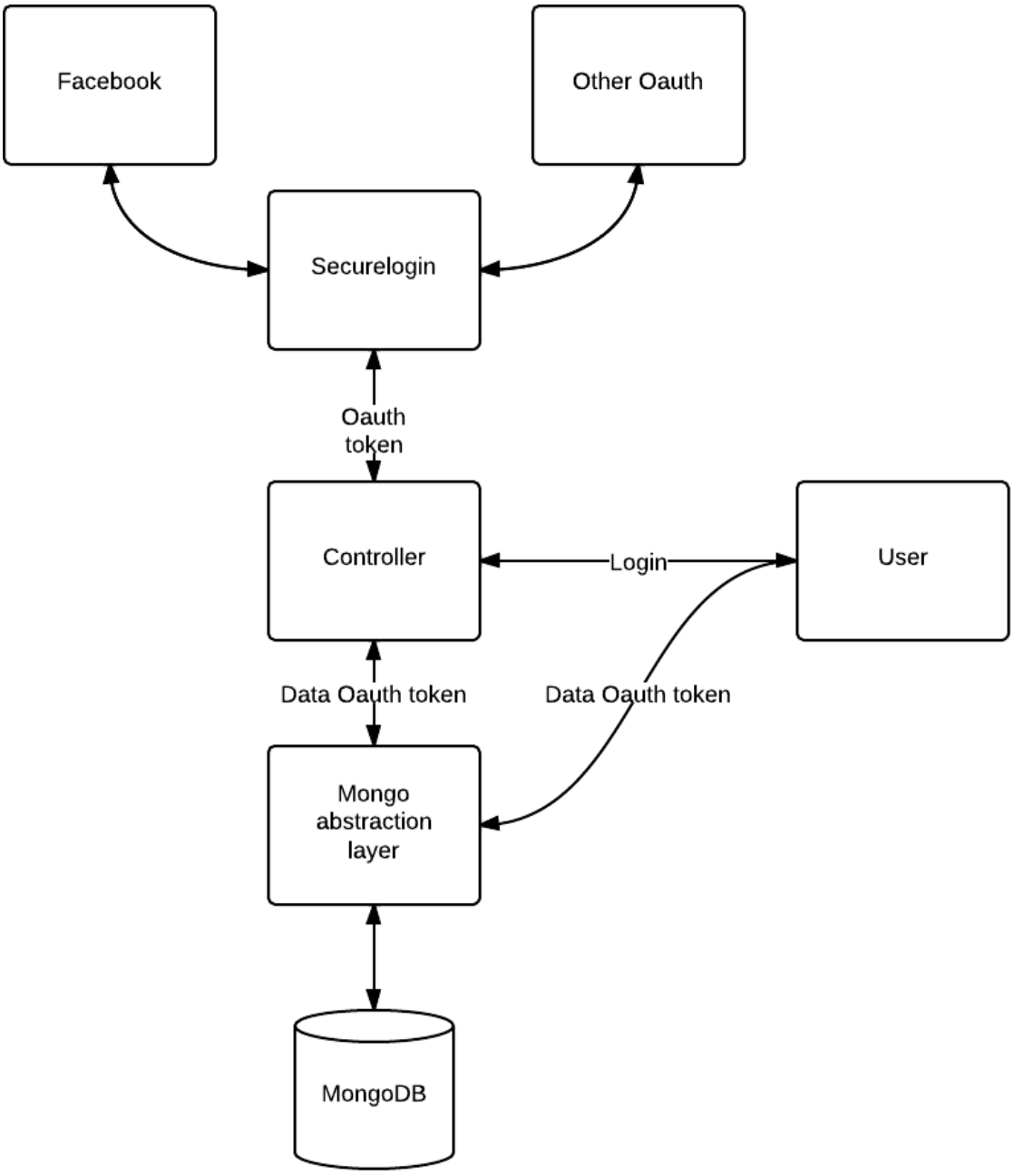


Figure 4.6: The flow of requests in the client server logic. The client logs in and gets its token which can later be used for accessing the data in the datastore and keep a persistent connection available

5

Evaluation

The chapter is divided into several parts, each symbolising an individual part of the system. The hardware performance is evaluated in terms of battery longevity and accuracy, while the app is evaluated in terms of platform fit. The main focus is on the total system, with the message overhead in mind and latency aspects of sending it.

5.1 Hardware platform

In this section the resulting hardware issues are discussed and analysed as well as the integration towards the handset platform from a hardware point of view. The most significant findings incorporates connectivity, accuracy and battery life for the hardware platform with the implemented technology. Many advances has been done since the older platform that was available when the project started. The goal is to find out if the decisions made during the project are the right ones and what could be made better in future work.

5.1.1 Bluegiga BLE112

The hardware base throughout the project is the BLE112 since choosing the BLE technology. The challenges are many with the platform. Brown outages (sudden voltage drop) resetting the whole chip, the main cause of the problem is bad design

of the hardware. It is hard to foresee such a problem when measuring small currents when only having imprecise tools available for use. The hardware has too few and small capacitors to catch the relatively big variation in voltage, but is improved during the project. The documentation has in many circumstances been limited making it hard to assess the best practice implementation for the platform. The evaluation of the different approaches was made harder by improper testing tools. The testbed DKBLE112 was the foremost solution but was not available until the very end of the project. The testbed could have helped with appreciating power drain and code efficiency on the chip.

The above argumentation shows that the platform still has some immaturity to be taken into consideration.

5.1.2 Power consumption

One of the greatest challenges of the project is to control the power consumption of the device. Some of the drainage is caused by the hardware, other by software. The different approaches are explained in the design section 4.2.1. The battery used for the hardware platform is a 40 mAh, but a second solution with 240 mAh (CR2032 battery) is proposed. The energy consumption is estimated with 10 transmissions per hour which is an estimated usage at a club two days a week with two thousand clicks per day.

- *Approach 1, always connected and advertising:* The battery drain is 15 mAh during scanning and 27 mAh during advertisement, in between the battery drain is 0,2 mAh which gives the solution a battery time of about two hours depending on presence of other devices. The main advantage is that the connection is set which makes sure that packets aren't missed and the connection is in real time.

- *Approach 2, sleeping when idle:* The platform wakes up on interrupt from an event/click. The hardware stays awake 1.5 seconds for setting up a connection and then goes to PM3 sleep enabling a drain of 0.0004 mAh, the battery life in theory would then be 13 days.

- *Approach 3, custom advertisements with data piggybacked:* The platform wakes up on interrupt from an event/click. The hardware transmits for 50 ms without connecting and waits for signal bounces to die out for 100ms in an idling state. The platform then goes to sleep. Then the battery life is in theory about a year.

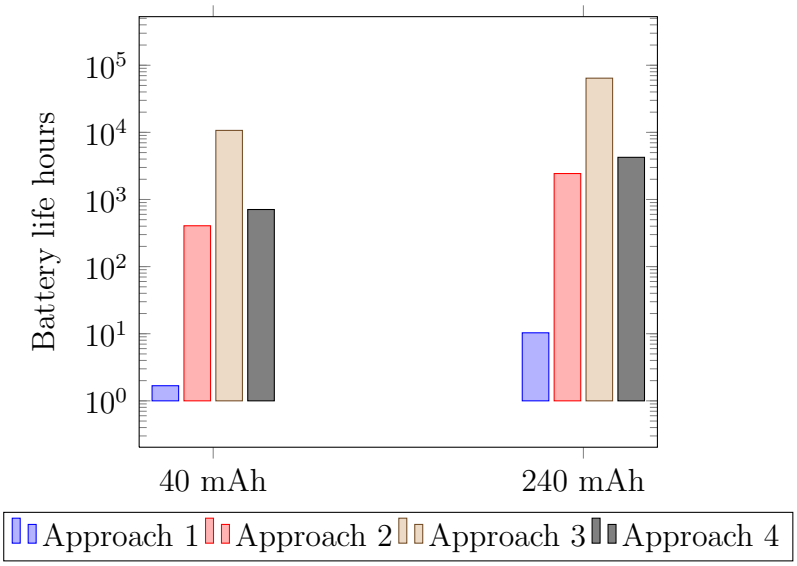


Figure 5.1: The different approaches theoretical battery life on 40 mAh and 240 mAh batteries

- *Approach 4, keep microprocessor on:* A fourth approach was just keeping the microprocessor on to count making it able to go to PM3 sleep most of the time and while counting mostly staying in PM2 sleep, sending the data once a minute during use. Each time the event/click is detected the timer is set to go to transmit and go to sleep a minute after the event was detected.

In retrospect the different approaches all had their charm, the preferred and most functional solution is approach four. It loses much of its real time appeal but makes it up in form of reliability. The approach proves itself in graph 5.1 with having a reasonable battery life while still keeping a good enough real time communication. Approach three does have a fast connectivity as explained in section 5.1.1, the problem mainly is its bad reliability. The theoretical battery life

is calculated with equation 5.1 which results in graph 5.1.

$$\begin{aligned} Capacity_{real,mAh} &= Capacity_{rated,mAh} * 0.85 \\ Awake_{tot,ms} &= Wakeups_{perhour} * Awake_{duration,ms} \\ Sleep_{ms/h} &= 3600000 - Awake_{tot,ms} \\ Energy_{Awake} &= Awake_{tot,ms} * Awake_{consump,mAh} \\ Energy_{Sleep} &= Sleep_{consump,mAh} * Sleep_{ms/h} \\ Drain_{real,mAh} &= \frac{Energy_{Awake} + Energy_{Sleep}}{3600000} \\ Life_{days} &= \frac{\frac{Capacity_{real,mAh}}{Drain_{real,mAh}}}{24} \end{aligned} \tag{5.1}$$

The equation takes 85% of the rated capacity to account for some discharge. The total time awake is calculated by estimating the number of wake-ups per hour and the duration of each wake-up. The values combined gives the total consumption per hour in the woken mode. The actual drain per hour is calculated with the total consumption per hour in woken mode plus the total drain in sleeping mode divided with the number of milliseconds per hour. Lastly this is converted into days by dividing with 24 hours.

The theoretical results shows that optimisations can make a big difference. The real results can only be estimated, as most of the theoretical drains lasts over hundreds of days. A formula for calculating the battery drain was attempted as the hardware contains a reference voltage inside of the microprocessor. The longest running hardware available has been working for three weeks. Lack of precise tools for debugging the drain of energy made it hard to verify that the code had configured to the hardware correctly.

5.1.3 Connectivity

The aspect of the connectivity is discussed in section 5.1.2. Graph 5.2 shows two separate tests with the custom advertisement approach, the goal was reaching at least the 95% heuristic accuracy, as explained in 4.2.1. The test is simply two users connecting to one phone and sending clicks/events by clicking on the mechanical counter. The distance to the phone was about 1 meter which would equal to the distance to a pocket where the phone would usually be at in a real situation. The battery wasn't noticeably affected during the test which was measured with a voltage meter.

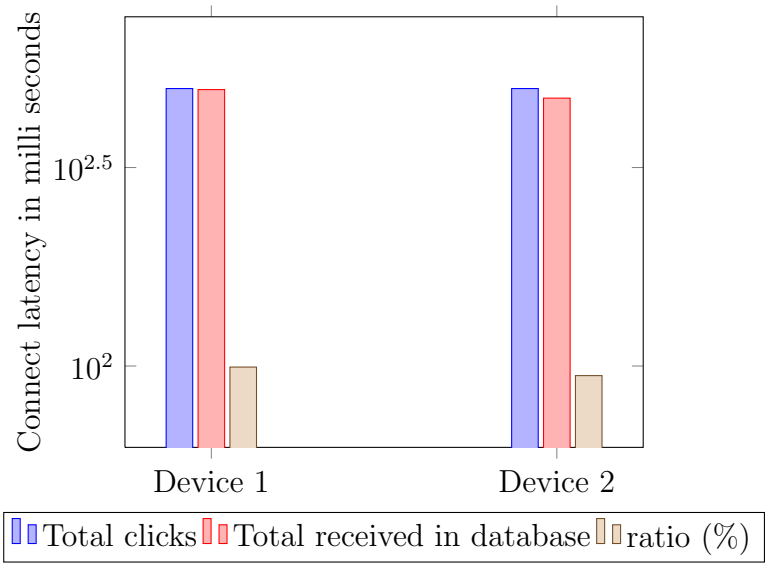


Figure 5.2: A trail was done with approach 3

This test shows that the devices was meeting the requirements from the beginning, integrity is kept and availability is good enough, while confidentiality mainly is achieved by obfuscation with this approach. What could be done to better the results and what further testing could be done is discussed in the future works section 5.5.

5.1.4 BLE protocol

The BLE protocol meets the expectations to fit the specification in accordance with the goals set in the original thesis proposal. A firm connection to the handset does have good real time properties sending data each 0,634μs, explained in chapter 2, even better than with a customised protocol. With the right configuration and hardware the protocol can reach that fast response time and still keep the battery drain to a level where it is possible to keep the system going for several years. With a directed connection, something hard to implement with BGScript, will make it possible to lower the connection time to a few milliseconds. The lower connection time will still the drain on the battery and make this protocol the one best fitted to meet the requirements of prerequisites defined by the hardware.

5.2 Handset platform

Three handset platforms are evaluated for the project. Different platforms each had their individual treats but iOS was the only one to meet all requirements. Much of the API worked as expected except for iOS the backgrounding mode. Backgrounding modes has troublesome battery optimisations properties, explained in section 4.2.2, making some of the protocols standard features unusable. Battery optimisations made the approach with custom advertisements useless in backgrounding mode as the state requires a set connection for listening to any services.

An implementation on Android would have been possible with Samsungs S3. Many of the projects described in related works already have an implementation for the S3. The problem with this was that the testbed did not work. The Motorola Razr implementation did not fully work. What could be seen in that implementation is that it was more of a port from Bluetooth classic than a native implementation of BLE. This could be seen with long scanning times and many steps for setting up an actual connection. The iOS implementation of the API had superior connection times, which gave results much in accordance of studies mention in chapter 2 which all used implementations of BLE in full compliance with the standard. The handset native code on all platform still has maturity issues.

5.2.1 Cross platform app

The cross platforms applications made fulfillment of the specification easy. At the end of the implementation period it was clear that most of the usage of the hardware would be integrated with other platforms instead of a standalone app. The back-end will mostly communicate with other data platforms, the implemented web view can display any data in the form of graphs etc. One of the problems with Phonegap is the performance. As Phonegap runs a web-view where HTML and CSS makes up the interface, the performance may suffer in comparison to an implementation made in native code. In the project it did not interfere with the final implementation.

5.3 Server platform

The implemented server is feed with data from the hardware solution via a long chain of events on the way (explained in section 5.4). The server implementation

proves good adaptability and real time communication, as discussed in the design chapter, and further evaluation is not needed for confirmation of conformance with the system requirements. The server evaluation is an important part of the platforms scalability and that discussion has already been handled in chapter 4 referring to papers on the issue.

5.4 Overall performance

The total performance of the integrated system meets the initial specification set out in the architecture but does not come without problems.

5.4.1 Overhead of sending an event

To understand how much data is handled by the ”IOT”-concept, the amount of bytes that is needed to send a single click across the infrastructure to store and display should it be explained. Further explanation of the BLE packet is explained with figure 2.3, 2.2 and 2.4. The way from counter to storage can be seen in figure 5.3. A single advertisement in which the data travels in the customised protocol approach is 47 bytes, the data is then capsulised many times on the way and is sent from the phone to the storage in a request which is 742 bytes. Data is finally stored in 138 bytes of data in the database. All data has been received from the system by sniffing packets and reviewing actual storage.

5.4.2 User testing

Partners

Due to unforeseen circumstances, the idea of testing in a realistic controlled environment was not used. Instead each part was tested in different environments. This may be a component that needs to be taken into consideration while evaluating results.

5.4.3 Comparison to conventional analog counting

While comparing conventional mechanical counters verses the digitalised versions showed that the opportunity cost of automating the process which is done in this

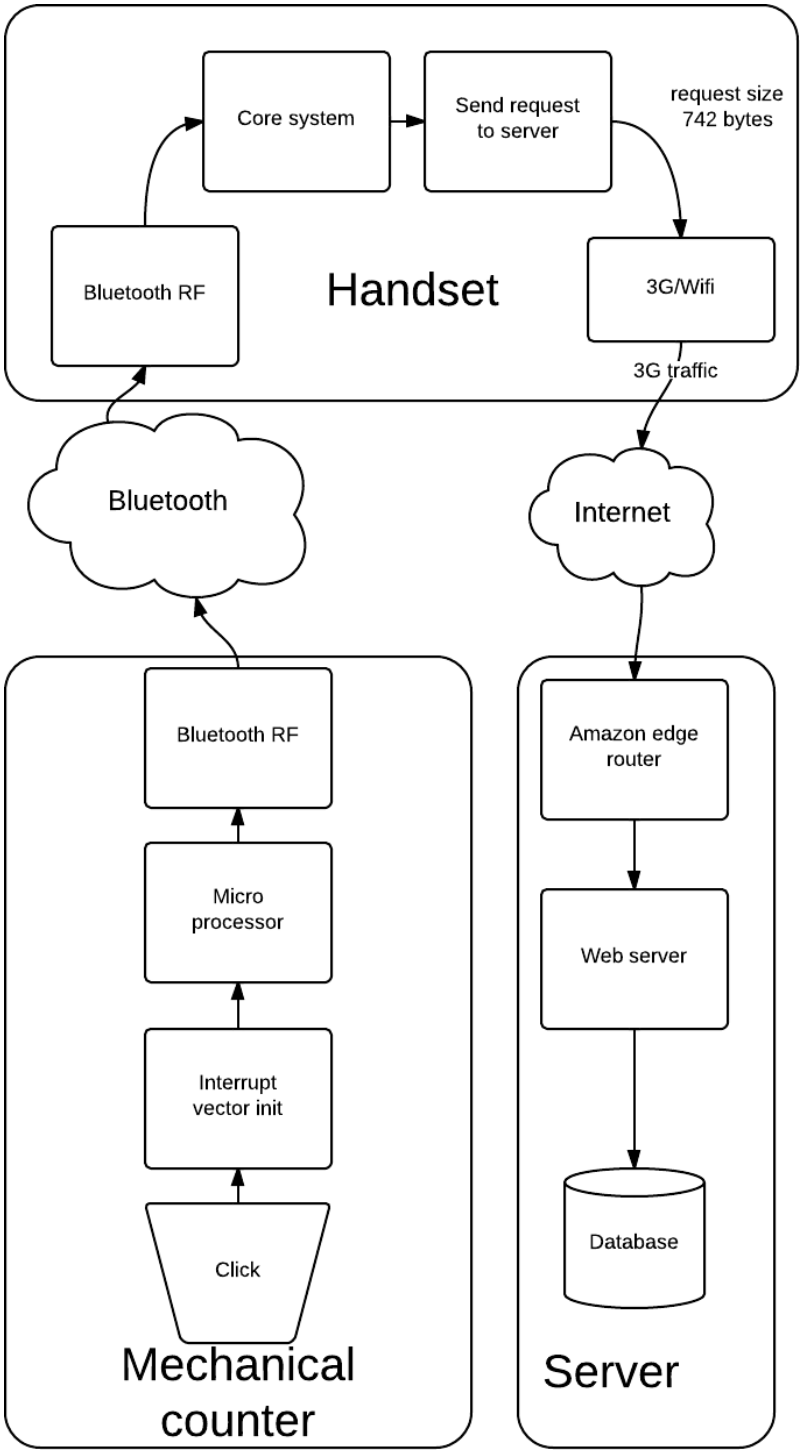


Figure 5.3: The path of a click/event from mechanical counter to storage in the cloud

study, is low. It can be evaluated in several ways, in this project the part that will be evaluated is if the implemented solution can match the accuracy good enough for its purpose. In section 4.1.3 of the report the heuristic of the accuracy is explained.

In the report many drawbacks and advantages for different approaches has been reviewed. The most challenging problem that required many revisions was the underlying hardware which did not detect every click. The problem resulted in many approaches for circumventing or solving the problem software-wise explained in section 5.1.1. The solution did meet the specified requirement set and would prove useful in usage.

The conventional counter does through its custom engineering with a high degree of certainty detect the clicks of the user. The usefulness of the data in terms of safety would prove greater with a higher accuracy. The problems faced during the prestudy and during the implementation will in the end require a better built custom made hardware with a higher certainty of detecting the clicks.

5.5 Future Work

The findings from this study shows that it may be useful to further investigate other possibilities with the BLE on this, and other platforms.

5.5.1 Usage of platform

In this project the implemented solution has a wide range of use cases and is adaptable for many different kinds of data. The potential use case of software built in this is extensible as collecting any event with some sensor and forwarding it to the back end is a quite generic approach. The industry is using the technology of the project to make more products for more and more sectors as explained in chapter 2. With the technology becoming more standard the users will get used to the abilities of the BLE technology simplifying integration in other sectors as well.

5.5.2 Alternative hardware platforms

As the logic of the back-end has been tested in this project a alternative hardware approach storing other data would be interesting for the technology as well as in-

tegrating other data into the platform. Other data might be an automatic counter where the mechanical counter is replaced by a laser interrupter.

5.5.3 Usage of the specific implementation

The remade mechanical counter platform used for the project has its use cases in many different industries. The different industries such as warehouses for counting inventories or amusement parks for counting visitors from alternative entrances, might need different features for collecting special kinds of data or their implementation specific modifications.

5.5.4 Further work on hardware software

An implementation in C instead of JavaScript would open limitations that the BGScript has. The main limitations did not affect the project as the more advanced functionality that C gives access to was not needed during the project. With C an advanced authentication could be made on the hardware side enabling users to acquire keys from the server which could be authenticated. The C implementation would make the code more manageable and open up for importing more libraries and custom open source solutions. A good set of features would be to enable software as a service (SAAS).

With SAAS it would be possible to turn on and off and on real time communication depending on subscription model. Another important feature would be the possibility would of getting metrics on usage to better serve customers and use the full capability of BLE. It would work offline and with different handsets as it would all be handled on the counter hardware side.

If the counter could keep track of each other's states the solution would work offline and without a handset. The coupling would allow other devices to give input on data in the surrounds. It would be interesting to investigate if the BLE chips could auto-switch between slave and server in a orderly way and thereby enabling the use of mesh networks that might couple to a common handset.

5.5.5 Back-end future work

As the data that comes from the counters will provide much more data than what is used in this project there is a great potential for some big data aggregations. As

described in detail at page 36 in chapter 4

5.5.6 Mobile platform compatibility

What has become plausible during this work is that the technology is moving fast with many new inventions based on BLE is coming out just this year. Sometime in 2013 Googles Android team announced on the conference Goolge IO that Google will build a common BLE stack implementation for all androids [17]. Since Broadcom released its BLE stack for Android to open source, much has happened, and some custom-built open source solutions for BLE has been available since the beginning of 2013.

Right now about 60% of iPhones and about 20% of Androids (Samsungs Galaxy III and IV) does support BLE, if the prognosis is correct most of all sold smart phones in 2014 will support BLE as well as Bluetooth all the way to version 1.0 via the dual mode compatibility.

6

Conclusions

The manual statistics gathered will remain an important tool for many industries. With the technology evaluated in this thesis, many steps of the traditional usage is automatised. With further development of the low energy technology protocols, integration with similar traditional mechanical technologies will be increasingly easy. The increasing ease of integration has a downside in complexity as many things become connected. The different orientations of Internet of things where machines can make the decisions and act upon them without interaction with any human hand in the process will be an important solution for making the complexity of many data sources manageable as the flow of information grows.

Bluetooth low energy (BLE) is a technology of that future. Its energy requirements are among the lowest of the low energy protocols and as can connect to most smartphones, in the author's humble opinion it will be one of the dominant technologies for the Internet of things in the coming years. For the focus of this thesis, BLE satisfied the requirements and offered much flexibility for further improvements to the platform. Examples can be extended to more services, time-stamping data etc which would not require much development. The low energy nature of the technology makes the jobs of the hardware developers easier as less space is required for battery or charging logic.

The main drawback of BLE in this context is its inability to connect with all smartphones as the BLE stack isn't implemented with the major mobile operating systems. The documentation leaves room for improvement and is still not well-built but it is improving over time. The problem will according to the Bluetooth special intrest group [4] be solved by 2015 when all smartphones shipped will include a

BLE chip and a fully implemented stack, making the technology mature. The web technology available for mobile applications make customisation fast and easy. It does not offer all the features that a native UI has but does handle most tasks with the big advantage of using the same code base for all platforms. For the project the cross platform makes it possible to make graphs and handle server authentication for getting and sending data with a single code base. Future possibilities for BLE could be further development in integration as the system still needs native code.

Bibliography

[1] BluFit, The smart waterbottle (2013).
URL http://www.blufitbottle.com/

[2] C. Aggarwal, N. Ashish, A. Sheth, The internet of things: A survey from the data-centric perspective, in: C. C. Aggarwal (Ed.), Managing and Mining Sensor Data, Springer US, 2013, pp. 383–428.
URL http://dx.doi.org/10.1007/978-1-4614-6309-2_12

[3] B. special intert group, Specification of the bluetooth system (February 2013).
URL https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=269452

[4] A. West, Smartphone, the key for bluetooth low energy technology.
URL http://www.bluetooth.com/Pages/Smartphones.aspx(June2013)

[5] C. Gomez, J. Oller, J. Paradells, Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology, Sensors 12 (9) (2012) 11734–11753.
URL http://www.mdpi.com/1424-8220/12/9/11734

[6] B. special intrest group, Bluetooth sig extends bluetooth brand, introduces bluetooth smart marks (Oct. 2011).
URL http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=138

[7] Bluetooth.org, A look at the basics of bluetooth® wireless technology (10 2013).
URL http://www.bluetooth.com/Pages/Basics.aspx

[8] Bluegiga, Bluetooth smart (Oct. 2012).

[9] T. M. W. Elke Mackensen, Matthias Lai, Performance analysis of an bluetooth low energy sensor system, Ph.D. thesis, University of Applied Sciences Offenburg (2012).

[10] M. Swan, Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0, Journal of Sensor and Actuator Networks 1 (3) (2012) 217–253.

[11] Pebble, Pebble (May 2013).
URL http://getpebble.com/#how-it-works

[12] Nike, Nike Fuelband+ Manual, http://support-de-de.nikeplus.com/ci/fattach/get/299224/1358178888/redirect/1d/session/L2F2LzEvdGltZS8xM

[13] B. Yu, L. Xu, Y. Li, Bluetooth low energy (ble) based mobile electrocardiogram monitoring system, in: Information and Automation (ICIA), 2012 International Conference on, 2012, pp. 763–767.

[14] S. Gripwall, Mobile keys open new opportunities for access control, Tech. rep. (2012).
URL http://www.zaplox.com/about-us/press/press-release-may-2012/#.UgO2JJIde-M

[15] M. Millbourn, Regler för samlingslokaler, Räddningstjänsten (10 2013).
URL http://raddning.com/foretag-och-organisationer/samlingslokaler/

[16] P. Smith, Comparing low-power wireless technologies, Tech. rep., CSR PLC (2013 (accessed)).
URL http://www.digikey.com/us/en/techzone/wireless/resources/articles/comparing-low-power-wireless.html

[17] L. Page, Google io, Google, Moscone center, CA, USA, 2013.

[18] IDC, http://www.idc.com/getdoc.jsp?containerid=prus24108913, Tech. rep., IDC (2013).
URL http://www.idc.com/getdoc.jsp?containerId=prUS24108913

[19] E. Marriner, Mvc jacascript applications, Tech. rep., - (2013).
URL http://edwardmarriner.co.uk/JavaScriptMVC.pdf

[20] R. Cattell, Scalable sql and nosql data stores, SIGMOD Rec. 39 (4) (2011) 12–27.
URL http://doi.acm.org/10.1145/1978915.1978919

[21] S. Jawanda, Apple raises the bluetooth smart ready bar... again, Bluetooth.org, 2013.
URL `http://blog.bluetooth.com/apple-raises-the-bluetooth-smart-ready-bar-again/`

[22] T. Matei, Mongodb performance in the cloud (2013).
URL `http://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1300&context=etd_projects`