Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Gaja Kochaniewicz

Smart lock for bike sharing in corporate environments

Master's Thesis
Espoo, May 27, 2015

Supervisor: Professor Jukka K. Nurminen, Aalto University
Instructor: Niko Päivärinta M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Gaja Kochaniewicz

Title:
Smart lock for bike sharing in corporate environments

Date:	May 27, 2015	**Pages:**	vii + 74
Professorship:	Data Communication Software	**Code:**	T-110
Supervisor:	Professor Jukka K. Nurminen		
Instructor:	Niko Päivärinta M.Sc. (Tech.)		

Bike sharing systems facilitate and promote biking as a public transportation alternative, providing benefits to the health and the environment. Contemporary bike sharing systems are automated and robust, but still have drawbacks. These are, for example, dependence on immovable stations with limited capacity and availability of service, only present in selected cities. Moreover, currently there are no solutions for automated bike sharing for private and corporate use.

In this thesis, we introduce a bike sharing system designed as a service for company employees. This system allows using the bikes for short trips or commuting. The focus of the thesis is on creating a smart bike lock, which is a device that enables automated bike access management. Currently, there are no smart bike locks available commercially off-the-shelf. However with our bike management system, relying on a custom-made smart bike lock, the companies can offer their employees shared bikes with minimal management effort.

The system has been tested in a controlled environment, and the device has been evaluated. The results are positive: the device fulfils the core requirements of the system and can be used as part of the service. The current smart bike lock is still a prototype. Further work is necessary to improve the design and add features such as short range wireless.

Keywords: bike sharing, bike rental, smart bike lock, Arduino, embedded programming, time-based one-time pad, Bluetooth Low Energy, NFC, GPS, GSM, power consumption

Language: English

Acknowledgements

I wish to thank the Erasmus program which led me to Finland where I stayed to do my master studies. I thank Aalto University for teaching me not only theoretical knowledge but also life skills. I am grateful to my advisor, Jukka K. Nurminen, for his help and guidance on this thesis as well as the courses he taught me during my studies at Aalto.

I wish to thank my employer and co-workers for the opportunity to do my thesis about work. I really enjoyed the tasks in this project. I am also indebted to Helsinki Hacklab for invaluable advice in project development and access to hardware that I could do my measurements with.

In my thesis writing I am especially grateful to Julia Casado, Eero af Heurlin, Otso Jousimaa, Robert Obryk, Ivan Raul, Jyry Suvilehto and Kliment Yanev for suggestions and feedback on my work. You were great help.

I am thankful for support and kindness to my family and all my friends, on- and offline.

Lastly I would like to thank Jere for his patience and encouragement.

Espoo, May 27, 2015

Gaja Kochaniewicz

Abbreviations and Acronyms

2/3/4 G	second, third, fourth generation of cellular telecommunications technology
BLE	Bluetooth Low Energy, unless otherwise specified refers to Bluetooth versions 4.0 to 4.2
IDE	Integrated Development Environment
EEPROM	Electrically Erasable Programmable Read-Only Memory; a type of non-volatile memory
FTDI	Future Technology Devices International company; here refers to devices providing USB to serial connectivity using a chip made by the company
GPS	Global Positioning System; a satellite-based navigation system
GSM	Global System for Mobile Communications; a 2G standard
M2M	Machine to Machine communication
microSD	miniaturised Secure Digital card; a small memory card
NiMH	Nickel Metal Hydride batteries
NFC	Near Field Communication
PWM	Pulse-Width Modulation; a technique of sending an oscillating signal between devices
RTC	Real-time Clock; an embedded chip that keeps accurate track of time
SMS	Short Message Service
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TOTP	Time-based One-Time Password

Contents

Chapter 1

Introduction

People who commute on a daily basis have a choice of different transport options that include cycling, driving a car, or taking public transport and walking. For many the first option is too exhausting and the last option is too slow. The third option of public transport for many requires also walking. As a consequence, most people decide to go by car the whole way. Nevertheless many people would like to have transport options that let them use their car less and conveniently exercise.

Some companies want to help their employees with this situation. Usually they provide bikes that can be used for short trips to and from work or during work hours. There is no system in place to control access to these shared bikes or to allow reservation in advance. We want to change that by creating a bike sharing system for companies.

The contribution of this thesis is a system that manages bikes automatically. Users of the system can locate bikes in the city and use them if they have the right to do so. The users may pick up and return the bikes whenever they want to from the place the bike is at and leave it at their destination. We expect that people will use them for short trips, such as getting from the workplace to nearest train station.

1.1 Problem statement

The work in the thesis is a bike sharing system with focus on the smart bike lock, a device that provides access control to the shared bikes. There are currently no commercial off-the-shelf devices that would fit exactly with the bike sharing system we have in mind. There are some projects in development which goal is to create such devices, but they are not available yet, so we are working on our own solution.

In addition to the smart bike lock, the system relies on a smartphone application to authenticate the user to the service. The service is deployed on a server holding synchronized data of all users and bikes. The server dispenses access codes to the smart bike locks to authenticated users to let them use the bikes. The smart bike lock is an important element because it provides security and access control for the shared bikes. It holds the reserved bike for user who made the reservation and prevents theft when a bike is left in a public place.

Although smart bike locks are a new idea, the concept of bike sharing is not and bike sharing programs already exist. However they rely on bike stations embedded in the pavements and they are usually only present in larger cities. These bike sharing systems have limitations and they would not work for this kind of project.

Setting

The work described in this thesis is done at the company Inline Market Evolutions Oy, based in Helsinki. The company offers business to business solutions in various fields. Nevertheless, they have not worked previously with electrical devices manufacturing, so this kind of task is unfamiliar for the company.

The challenge of the project is therefore to develop a new service and design a hardware device to implement it. The team working on it is composed of three people and the work is divided among them. The server and smartphone application are developed by Lulit Woldemeskel and Jacopo Chiapparino. Their work is not detailed in this thesis.

Problem definition

The thesis context is a bike sharing project, but the implementation and evaluation parts of the thesis focus mostly on building one part of the project, which is the development of a functional and adequate smart bike lock.

The base requirements for the smart bike lock in this project are twofold: to provide access control to the bike with reasonable level of security, and the ability to operate for extended periods of time with minimal to no maintenance. The lock should operate through spring, summer and autumn without issues. It needs to be reliable, so that the users can, at any given time, start using the bike and return it safely. Therefore it needs to be robust, easy to attach to a bike, power-efficient and straightforward to use.

The project has been planned to iterate rapidly over prototypes to make a simple device composed from ready-made elements. The electronics and

the software use the Arduino platform with some modifications. The current result is a plain functional device that supports the system and fulfills all the key requirements. The device is a box inside of which is a key that opens a regular bike lock that the bike is locked with.

1.2 Thesis structure

The rest of the thesis is organized as follows. Chapter 2 describes the background of the project from the history of bike rental, to current bike rental systems and their efficiency. Car sharing is then mentioned as a similar type of service. It is followed by descriptions of upcoming smart bike locks and wireless technologies used in them. Chapter 3 presents the complete system design, which includes use cases, service architecture and explanations for some planning decisions. Chapter 4 elaborates on the design of the smart bike lock. It presents the design logic, the constituting elements and the implementation. This description also includes the issues that were encountered during the development and their solutions. Chapter 5 evaluates the system by reviewing the collected user feedback, as well as by analysing the technical issues and power consumption of the prototypes. Chapter 6 discusses the applicability of the system in a real environment, its weaknesses and possible improvements. It also elaborates on these improvements ranging from small incremental changes to complete redesign . Chapter 7 concludes by providing a general overview of the project and by discussing the lessons that were learned from it.

Chapter 2

Background

This chapter covers the history of bike and car sharing systems. Next is description of projects, which are currently in progress, with similar goals to this thesis work. Last part is about wireless technologies.

2.1 Bike rental and bike sharing

A previous study[30] defines what bike rental and bike sharing and how to distinguish these two kinds of service:

Bike rental is a service provided in a few locations and clients can rent a bike for periods of time from days to weeks.

Bike sharing is a concept with a slightly different format. The bikes are distributed around an area in many stations. Users are encouraged to take bikes from one place and return them to another one. The length of the trips made with the bikes is supposed to be very short, from few minutes to few hours. The system uses automation and money, cards or phones to simplify pickup and return procedures at the stations.

2.2 Bike sharing systems

Biking is a healthy transport alternative to cars or public vehicles[11]. Most people know how to bike[3], but there are still segments of population that do not do it frequently. This can be caused by many reasons, such as not owning a bike or having trouble with including biking in the daily workflow. Some people find themselves in situations where they wish they had a bike they could use immediately.

2.2.1 Impact of bike sharing systems

At least some of the issues that prevent people from biking can be solved by bike sharing services. A study conducted in Washington D.C.[10] concludes that access to shared bikes can promote cycling among people from new segments of the society compared to bike owners. This confirms that good bike sharing services contribute to an increase in cycling in the society.

Studies[40] looking at the frequency and patterns in use of bike sharing systems show that many people manage to incorporate the use of rented bikes into their daily routines. Between the most and the least frequent users of the system, researchers noticed a large group of people who used the system as a transport alternative equal to other public transport options. This shows that there are people who would use a bike in their daily routine on a semi-frequent basis if they had the option to do so. This segment is a potential target group of this thesis project.

2.2.2 Bike sharing history

Bikes can be rented in many cities across the world from public, local self-service sharing systems. The bikes are most commonly distributed across the city in parking stations that the user can use. Number of stations, their positions and size varies between cities. DeMaio[13] calls these systems the 3rd generation of bike sharing, meaning there is a history of significant changes in how the systems are ran.

The first bike sharing service was started already in 1965, in Amsterdam[12]. The project was called "Witte Fietsen" (Eng. White Bikes) and it was basically a service offering white-painted bikes to the public without a fee or strict control. DeMaio reports that due to the essentially non-existent security systems in place, this bike sharing fell apart only after a few days.

Although the first system failed, bike sharing was not abandoned. Instead, it has evolved and that experience has paved the road for new systems. The first service of the next generation in bike sharing started in 1991 in Denmark[12], where main improvements were more robust bikes and pickup and return stations located around the city with coin deposit machines. The system still allowed the users to stay anonymous, which facilitated theft. Consequently, in the following 3rd generation, this was changed. In 1996 at Portsmouth University in England the bike sharing program Bikeabout[12] introduced user accountability for the bikes.

Figure 2.1: A bike station of the Velib bike sharing system[42]. This photo shows bikes attached to the bike stands on the street, in Paris.

2.2.3 Contemporary bike sharing

Currently public bike sharing systems follow the idea which was introduced in 2004 by OYBike in London[31]. It was the first system to deliver self-service automated renting from stations distributed around the city. This kind of automated stations are now distributed in many large cities around the globe operated by dedicated companies. Users can rent a bike with mobile applications, text messages, magnetic cards and credit cards.

Rental systems also vary on the payment model[30], such as hourly, monthly fees, annual subscription or free of charge. There are programs that penalise the users for using the bikes for too long; others give free rides for complying with rules. Some bike sharing systems are ran by cities themselves, others by advertising companies like JCDecaux or Clear Channel. The variety of programs is considerable and the ways they approach the business model vary.

It is obvious that the bike sharing programs are succeeding. One example is the Spanish Bicing system[30] which increased the number of bikes and stations eight times after the first year and a half of operation, from May 2007 to the end of 2008. It is not an exception, as many others have had similar success. Moreover, there is an increasing number of cities joining with their own systems. The majority of the programs are in Europe, but there are also a few in Asia and the Americas[44].

Furthermore, rental systems are still struggling with some issues, such as the distribution of the bikes across the city as they are used which results in

the overcrowding of some stations while others are left empty. For example a Parisian program Velib[12] (see Figure 2.1) is trying to remedy this by incentivising the users to pick up and return bikes in specific patterns. Modelling a system that can manage the users' actions in a busy city is a non-trivial task.

The necessity to use the parking stations of limited capacity is a drawback of the system. On the other hand, the advantages of this kind of system is the robustness and good security offered by the parking stations. Unused bikes have strict locations and are connected to the common network. Bikes do not need individual tracking systems to be located and users can easily spot the stations.

Helsinki bike sharing

Currently, Helsinki has no station-based bike sharing service. There are a few small bike rental services but they have few locations and are not automated.

The city of Helsinki attempted to install and run a bike sharing system called CityBikes but the program was discontinued in 2010[45] due to vandalism and subsequent increasing costs of operation. Since then there were various news announcing plans to create a new bike sharing program[46], but at the moment, in 2015 there is no running public program with easy to rent bikes in the area of Helsinki. People who want to use a bike are left to use smaller private bike rentals or own bikes.

2.3 Car sharing services

Bike sharing can be compared to car sharing. Both of these services are revolving around sharing of vehicles and have been gaining popularity in recent years. This section goes over three examples of car sharing demonstrating the evolution and technological enhancements to the service.

Singaporean car sharing

Singapore has strict car ownership laws and the prices of vehicles grew quite high for average household. As a result a car sharing co-operative scheme[36] has been implemented in 1997 to provide an alternative to car ownership. The members of the system could reserve a car 24 hours in advance, and then pick up the key from a key box. The service had yearly fees as well as a small payment for travelled distances.

This car sharing scheme was not automated and cars had stations. The reservations were made through phone calls to an officer who would prepare

the car keys on demand. The ability to place a reservation on a short notice and payments being based on distance rather than on time were completely opposite to arrangements of car rental services. That arrangement made the car sharing system popular. The number of members of the co-operative more than doubled in two years from the initial 120.

The service is comparable to the station-based bike sharing systems because it also has stations. It is however ran by a co-operative rather than a company or the city.

car2go

In 2009 a new type of car sharing service called car2go was started in Ulm[15]. It was not based on stations like the Singaporean car sharing, but rather the cars could be picked up and left anywhere. This system is known as "free-floating". The cars are tracked with GPS and users can user them anywhere within the perimeter of the city. The service knows not only the position but also the status of the fuel, cleanliness and overall shape.

Cars can be reserved as well as picked up ad-hoc. Pick up of the car is fully automated. Users need to be registered and poses an RFID card that opens the car door. Every user is by the system about the state of the car including dirt and possible damages. If any issue arises, customer support is called automatically. If there are no problems, user can start using the car. Service fee is only one and depends on how long the car is used, counted with increments of 1 minute.

The car2go service is operated by a company. There are no stations in the service and because of that it is very flexible. The system of modified cars could be compared to mounting a permanent smart bike lock on bikes and using them to run a service.

Green Move

In a paper from 2012[4] a plan for "peer-to-peer", free-floating car sharing called Green Move is presented with a plan for spreading the use of electrical cars. Each car gets equipped with a special Green e-box, which much like the bike locks has wireless capabilities, as well as GPS and GSM. To open the car, user has to use a smartphone with a short-range wireless protocol like NFC or BLE to communicate with the Green e-box. Both the smartphone and the Green e-box communicate with the server which is said to not only log the location and usage statuses, but also predict routes, speed and time of travel and use it for the reservation logic to help users find available cars near them.

Green Move is trying to convince car owners to share their electrical, emission-free cars with others for the sake of the environment. Similarly, they go for a technical solution with the Green e-box packed with features. The paper mentions also their complex reservation system which actively collects information and predicts vehicle availability for the users, which would also be useful for a bike sharing system.

The Green Move system is a combination of the previous two car sharing systems. It is planned to rely on the users renting out their own cars like a co-operative. Additionally, the cars would be equipped with a special e-box comparable to a smart bike lock with GPS, GSM, short range wireless and other features.

2.4 Smart bike locks

Despite the rising popularity of the bike sharing systems, there are limitations. The main issue is that they are not everywhere. Many towns and cities still have no good automated bike sharing service. However, the shape of bike sharing may change in the near future with new technologies bringing new possibilities to create and run smaller bike sharing enterprises. At the forefront of the technology are smart bike locks that may be able to democratise bike renting and sharing. It would mean that mean that not only large companies but also small entrepreneurs can run bike sharing services. This thesis project is working to create such a system.

Most of the devices described in this chapter have used Kickstarter which is a crowd-funding platform. Various small companies and individuals use it to showcase their prototypes and gather monetary support from the public. In turn they promise to develop services and produce devices. The people who support the project can order some of the first models but the projects are not always finished or shipped.

In 2013 two Kickstarter projects on personal smart bike lock were funded. Since then, these projects have been in production to deliver in each case a working hardware product as well as a software system. A few months later other similar products appeared on Kickstarter and are in various stages of progress. All of these projects have similar functionality of access control to using a privately owned bike. As no project has delivered a working product yet, the following overview is mostly based on the information provided by the designers and manufacturers.

2.4.1 The Lock Box

First, The Lock Box[16] should be mentioned even thought it is not a smart bike lock. It was founded on Kickstarter in June 2013 and manufactured and sent to the backers in September 2014. The Kickstarter page describes the product as a very simple padlock-shaped box for a car key (see Figure 2.2). The creator's idea was to keep safe keys to cars borrowed from rental services. It was later extended to accommodate other valuables like credit cards. Various items can be stored in the padlock attached for example to underside of a car. Users would open it by entering the correct unlock code set in the box.

The description on the Kickstarter page is not long but from pictures it seems the lock has no electronics. However, the Lock Box has several things in common with the prototypes of this project, which is why it is worth noting. Both of them are built as an attachment that holds a key. In case of the Lock Box, it is a car key, while in case of this thesis it is a key for a regular bicycle lock.

Another interesting fact about the Lock Box is that while it was planned to start shipping in 3 months, the project took altogether over a year to complete and ship. Now it can be bought online from the manufacturer's shop.

2.4.2 BitLock

BitLock[29] is a smart bike lock that completed its Kickstarter campaign in November 2013. It is made by a company based in the US with manufacture taking place in China. The device, as described on the Kickstarter page is a modified bike U-lock (see Figure 2.5) with Bluetooth Low Energy (BLE), which communicates with the user's smartphone. On the outside, the lock has only one button to both close and open it. Inside it has a very small embedded circuit and a high-power, 3.7V non-rechargeable AA battery. The lock is expected to work up to 5 years with that battery (2400mAh) even with the wireless functionality which needs to be on at all times to detect the approaching user.

Communication between the user and the lock is supported with smartphone applications. Manufacturer promises to provide versions for iPhone and Android. They assure they have tested the application on popular phone models to ensure it performs well. Moreover the application is to be released to Apple's and Google's marketplaces for easy install.

The smartphone application and a smartphone with BLE are necessary for this system. Without them the user cannot open the bike lock. The goal

Figure 2.2: The Lock Box[16].

Figure 2.3: The Noke U-lock and the Noke[18].

Figure 2.4: The Skylock[39].

Figure 2.5: The BitLock and its electronics[29].

of the product is to replace keys with a smartphone for bike security. This is meant to make the lock easy, more reliable and minimal in maintenance. Furthermore, the application is supposed to support sharing the bike with other users. This means the BLE of the lock will be capable of pairing with more than one smartphone making it possible to share the bike in groups or communities.

The lock appears complex and seems to include state-of-the-art ideas. The electronics are tiny and use only an AA battery fitted inside a U-lock, which is a small space. Moreover, it is equipped with BLE which is in use probably almost all the time. This can mean high power use, but creators

assure that the power source which is a 2400mAh battery will last 5 years. These features make it an interesting study of a product similar to the goal of this project. However, there is one thing in which BitLock misses out with this project, which is the form factor of the lock. The U-lock is not permanently attached to the bike and any user that can open it can also steal the lock. The goal of this project is a more permanent fixture.

2.4.3 Skylock

Skylock[39] is similar product to BitLock. It is crowd-funded outside of Kickstarted and set to ship in the summer of 2015. Manufacturer, Velo Labs Inc is also stationed in the US. Their website describes the product as BLE-enabled U-lock equipped with sensors, a touch interface and a solid, rubber shell (see Figure 2.4). It has a solar panel and can be additionally charged via USB. It has been designed for easy use, snapping to lock when in position and touching on the side to unlock. To open the lock, a smartphone with BLE 4.2 and an application will be needed. The smartphone will have to run a recent OS, like iOS 6.1 or Android 4.0.3 to have the software work with it.

This product could be a candidate for a closed and power self-sufficient solution. This would allow the complete locking of the enclosure preventing tampering and breaking. It is possible however that the low efficiency of the solar panel and long-term storage in the dark would make use of the USB charger a necessity. It is still an interesting solution to consider.

2.4.4 Noke U-lock

Noke[18] is another U-lock based product (see Figure 2.3) on Kickstarter funded in April 2015 set to complete on September 2015 by FUZ Designs stationed in the US. Creators say their product uses BLE 4.0 and is designed to be opened with a smartphone via BLE as well as a BLE keyfob or, in case of emergency, a series of clicks on the side of the lock. It is supposed to come equipped with an anti-theft loud alarm coupled with motion sensors. Aside from that, creators promise it will be robust and water resistant. Its battery is estimated to last up to several years if the alarm is off. The product can then be charged with a micro-USB connection. Again, it would use a smartphone application for using as well as sharing of the bike. The detailed requirements for running the application are not supplied.

The company working on Noke U-lock also has in progress a similar project of a Bluetooth padlock, called Noke (see Figure 2.3), which is delayed as its shipping deadline passed in February 2015. There is a review[8] of a trial version of the product online describing its use. According to the

article, the padlock needs to be paired with a BLE phone with the Noke application. Lock can be opened and closed freely by hand when the phone is within the vicinity of the phone. The presence of the phone allows unlocking. As a result if a paired device is close enough, the lock can be opened. Manufacturer states it will be possible to pair it with different phones so more than one person can use it. It can also be opened by a custom combination of presses on the lock handle in case a phone is not available. The usability was rated as good by the reviewer who also mentioned the final product will ship in June 2015.

Noke seems to have experience with building embedded devices, but they too are delayed with deliveries which does not bode well for the end result. It seems though as their electronics are close to completed if they were able to send a unit to be reviewed.

2.4.5 Lock8

Last in the lineup is Lock8[24], which is the only bike lock that does not have a shape of a U-lock and is produced by a German company. Still, it is also late with delivering the product despite local manufacture. The device is however much different. It is an attachment to the bike's rear wheel's frame with a pluggable cable (see Figure 2.6).

The product's Kickstarter page says it is equipped with BLE, GPS and GSM. It is also said to have a robust security system with temperature, motion and gyro-accelerometer sensors detecting theft attempts, sounding a loud alarm and sending a notification to the user's smartphone. Designers say it will detect attempts to move the bike or to break the lock in ways such as freezing, cutting, drilling and burning. The insides of the lock will not be accessible and the only way to charge it will be via a USB port or by biking as it will have a dynamo.

With these scenarios predicted, the lock's design is to clearly stay in place secure and attached to the bike. However, it can also cause problems as the user cannot replace the battery and must connect the device via USB to a power source. It seems that creators are not worried about this and expect the dynamo to keep the battery full, so the capacity is likely sufficient to work for a year or so with all the wireless features in use.

When the battery is charged, the user should be able to use the provided mobile application and BLE to open the lock. Application is to use shared BLE keys so access to the bike can be shared. As the lock is a permanent fixture, this makes for a good bike sharing device. However the price of the product is steep so even the Kickstarter page suggest sharing it to earn the money back. There are few details about the application and its features

Figure 2.6: The Lock8 mounted on a bike[24]. It is attached to the rear wheel. The lock has a cable connected to it from both sides. It is locked to a pole on the street.

aside from that it will allow sharing the bike.

Although the project does not focus on the software side, there is plenty of detailed technical specification, including operation and storage temperatures, sensors and loudness of the alarm system. The lock is programmed not only to lock and unlock, but also to detect theft and notify the owner. It is clear that the design has been carefully analysed and planned to be almost as robust as the station-based bike sharing systems are.

The form and technology of this lock make it closest to the ideal form of this thesis project. It is an enclosed device that can be permanently attached to a bike. It is tamper-proof and can be used in public and as a main element in a bike sharing system. Lock8 is therefore a reference point in both form and technology design. In its concept it is very sophisticated and seems to be complete as a product.

2.5 Wireless technologies

Among the many wireless standards only two are common enough to be viable for this project. Many other technologies like WiFi are high power, or like ZigBee not present in current smartphones. The technologies that are used in the smart bike locks are BLE, NFC and cellular networks.

Cellular network

There are currently three generations of cellular networks: 2G, 3G and 4G[7]. While 2G is still in use in many parts of the world, it is getting replaced slowly by its successors, 3G and 4G. These networks are used most commonly by mobile phones to send messages. The coverage for 2G and 3G is good in cities and towns and the system is maintained by the cellular network companies. This makes 2G and 3G good channels for long distance communication.

Bike sharing systems benefit from using the cellular networks. For example Lock8 uses GSM, a 2G network, with a Machine To Machine (M2M)[43] SIM card to connect to the network and send the owner an SMS text message in case of theft or other issues. If the lock were part of a bike sharing system, the text messages could be sent to the server as part of an automated notification system.

Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a recent communication standard made with the goal of lowered power consumption. As a result the standard has short range, but still is able to communicate over the distance of maximum 50 meters[37]. This means that walking past a BLE device like a bike lock with a BLE-enabled smartphone may be enough to pair them and exchange some information[8]. An important note is that BLE is incompatible with the previous versions of Bluetooth, so while many old phones were equipped with an older Bluetooth version, they cannot communicate with the new chips. BLE is used by all smart bike locks mentioned in this chapter.

Near-Field Communication

The Near-Field Communication (NFC) standard is based on the RFID technology and allows communication over very short range of maximum 20cm[17]. In practice this means that to get devices to pair and exchange the information, they have to be very close to each other. The NFC is considered secure due to that short distance.

The range of NFC holds advantage over BLE in scenarios where there are many bikes with similar locks in the same area. With an NFC lock, a user would have to just approach their bike and unlock it by tapping the lock. In the case of BLE, the user would have to select it from a list and confirm. It might not be a major issue if the mobile application is programmed well to handle this contingency.

A problem that NFC has but BLE does not is that it is sensitive to metal. We have found that an NFC antenna used in embedded programming could

not read an NFC tag when propped against a metal surface. To make it work, NFC-enabled element requires shielding so putting it in a metal container or attaching it to a metal frame might significantly impair its performance.

NFC market penetration

NFC has a problem with market penetration. While it has made it to many smartphones before BLE and many older phones from the last 2 years are equipped with it, there is the notable exception of Apple products. Only the most recent model 6 of the iPhone has been equipped with NFC and currently it is still exclusively used for the ApplePay service. While last year there were news of Apple giving access to use NFC to application developers[28], there have been no signs of it happening yet.

In the United Stated 41.3% of smartphones are Apple models[23] which is a significant fraction. Moreover the company is widely recognized, which might be why pictures of iPhone devices were used to display the mobile phone functionalities in the Kickstarter campaigns for the smart locks. The attention to Apple might explain why the smart locks prefer to use BLE. NFC might seem less like a viable option to them in comparison. On the other hand, statistics[19] on the whole world show decrease in popularity of Apple phones to 20.4%, which means there are markets where Apple products are much less significant.

2.6 The smartphone as a key

The importance of including Apple devices in such projects is not only because it is a large share of the market. It is important because most of users of smartphone devices like iPhones and Android smartphones have their device with them at all times. These smartphones have now increasingly more processing power and many hardware features like wireless chips for WiFi, BLE, NFC and GSM. These wireless features are already used by a variety of applications.

This is why the wireless bike locks are trying to replace traditional keys with smartphones. That replacement is expected to bring better security and usability, as some users might be more prone to forgetting their bike lock key than their smartphone. Additionally it simplifies sharing the bike with other smartphone users without borrowing a physical key.

Chapter 3

System design

This chapter describes two use cases and resulting design of the project. Because the project is still in development, first part of the chapter lists the expected functionality with explanations followed by the current status of the project.

3.1 Project setting

The project has been started with the goal of providing service in a business to business model. The company at which the project is developed intends to sell the solution to client companies in various forms like complete setup, partial setup up or improvement of an existing bike sharing program. There are already companies in the Greater Helsinki area that offer their employees the perk of using company bikes. They are however often not managed properly and bikes are used only in the first come, first served model.

First come, first served is not desirable because there is no of information about bike availability which makes relying on them a gamble. This is an important factor for the usability of the system and it relates directly to the main use case for the bikes. Many companies, that have this kind of bike sharing, are interested in implementing access control and reservation system.

The last mile problem

The bike sharing benefit is supposed to reduce the frequency with which employees use their cars to come to work. For many the reason to use a car is the lack of end to end public transport connection between their home and the company office. If an employee decides to use the public transport, they will need to walk a few kilometers from the nearest bus or train stop to

the office. This is called the "last mile" problem. Many people faced with it often choose to use their car to get from home to the office. However, if at the last step of the journey with public transport there was a bike waiting for that employee, they would be more inclined to use that option. This way they would contribute not only to a cleaner environment, but also to their own health.

Use case 1: Travelling to and from work

Because there is an element of trust in the bike waiting at the last stop, this use case must start when the employee is planning the journey, and decides to use the company bike. The employee needs to be able to to reserve a bike in advance for a specific time window using a bike sharing system. The system should ensure that the bike will be waiting at the expected destination at the time window reservation has been made for. The employee will then go to the location bike is stationed at and use it to reach other offices. System will update the bike's user and their location. Once the trip is done, the bike will be again accessible to other users informed of the new location. After work the employees will be able to do the same trip in the other direction, from company office to a bus or train station and then use public transport.

Use case 2: Trips during work hours

Another use case for the bikes are short trips from the office to nearby locations and back during office hours. Those could be for example visits to customers, business meetings and lunch trips to restaurants. Some of these might be planned in advance and so bike could be reserved earlier, but the employee could also take any free bike when needed. Ad-hoc renting is simple since the system delivers information about nearby accessible bikes.

3.2 The service

The service's requirements are based on use cases 1 and 2. The results are described in the following sections starting with a description of how the system works from a user's point of view.

3.2.1 User guide

In order to use our bike sharing system, a user must have a smartphone with installed application for bike sharing. After logging into the application, the user is presented with a list of bikes that are available. Clicking on the bike

brings up the map with the bike's current location. The user can then chose to use the bike by clicking a button. The user is then forwarded to the next screen with instructions on how to unlock the bike and the code to open the bike lock.

The smart bike lock is mounted on the frame of the bike. It is a box with a button, a led light and a number pad. To start using it, the user has to click the button. The lock signals with a small led light that it is ready for input. The user can enter the numeric code in and once it is recognized as correct, the box opens. The user has to open the lid of the box by hand and retrieve the key for the bike from inside. After that, the lid can be closed. To close shut the lock of the lid, user must click the asterisk button.

The key retrieved from the box opens the regular bicycle lock that keeps the bike locked. The user keeps the bicycle lock and the key with them when they travel and once the trip is over, the bike can be locked again and the key can be returned the same way into the box.

The smartphone application stays in the background during the trip keeping track of the changing location and counting the travelled distance. The kilometers travelled will be listed after the trip in the user's profile as a history record.

3.2.2 Service architecture

The structure of the system was designed to suit company-related use cases. The system is composed of 3 elements:

- server - keeps track of statuses of all bikes, holds the reservations, grants the access to bikes
- smartphone application - provides users access to the system, makes reservations, assists during the journey
- smart bike lock - access control to the bike, secures the bikes in place (development described in Chapter 4)

In the most robust configuration of such a system all elements should be able to directly communicate with each other:

Smartphone and the server
: This connection is necessary in the system to authenticate the user to the system, fetch fresh information about the bikes and communicate that the user is renting a bike. The connection is made over the Internet, which can be accessed via WiFi or a mobile phone connection like 3G and 4G.

Smart bike lock and the server
: This connection is optional and its main use would be to report the status of the bike as well as its location. It would be mostly a security measure to locate the bike when it is not used by anyone. This feature would require GSM and GPS.

Smart bike lock and the smartphone
: This connection is also optional and its main use would be to improve the usability for the users of the system. The users would be able to lock and unlock the bike with the application on the smartphone directly instead of typing in the code. For this connection a short range wireless like BLE or NFC would be suitable.

Other, less robust configurations are possible. For example we can give up the direct connection between the smartphone and the smart lock or the smart lock and the server. This greatly simplifies the project on technical side. In the simplest form only one direct connection is necessary, and that is between the smartphone and the server to receive live updates.

As a result the bike lock can be built without any wireless abilities as those are expensive to implement and maintain. Instead the user with a mobile phone can be used as a bridge between the lock and the server. This means the user would have to transfer a password provided by the mobile phone to the lock correctly for the latter to open.

3.2.3 Requirements

The functional requirements described below are only relevant to the current iteration as the possible future iterations are not yet all decided by the project team. Their description is expanded later.

User requirements

The user should be able to do the following actions with a smartphone:

- log in and log out of the system
- get a list of available bikes and their locations
- unlock a bike and use it
- return the bike and lock it
- see the journey in the system as a route and as a distance and time summary

In the pilot phase functionalities such as users enrolling themselves via the mobile are not available as adding new users needs to be controlled. Similarly adding the bikes, especially since new bikes can only be added in the system when there is a hardware lock mounted on it.

Additionally, the reservation system for bikes has not been finished yet. In simple form, the reservation system can be just the server making the bikes unavailable some time before the reservation starts. There are, however still unsolved issues like what should the system do if a previous user is late in returning the bike. There are examples of systems[4] which track shared vehicles in motion and estimate time of arrival at expected destination. It would require quite a bit of work to implement for this example, but it can be seen as the upper bound to complexity of such a system.

Technical requirements

The technical requirements can be divided by the part of the system whose functionality they describe. The currently implemented requirements are as follows.

Server

The server's requirements are to be a database of users, bikes and companies which use the system. The server needs to keep up to date information about the state of the bikes and based on that generate and send a valid access code when it receives a valid request. There needs to be an API for the mobile application to use to communicate with the server. As an alternative to the application the server can also host and offer a mobile website.

Mobile application

The application has access to the server via the mobile API and sends updates about its state. It also fetches the access code when necessary. In the case of the mobile application location can be tracked using the GPS to calculate the travelled distance and to inform the server about the final location of the bike.

Smart bike lock

The bike lock needs to receive and compare input from the user to the correct numeric code and open the lock when it gets a match. It must always have enough battery to allow the user to use the lock.

3.2.4 Stakeholders

In the first two use cases, the company employees are the users of the system, renting and riding bikes. However they are not paying for the service, which is a benefit provided for free by their employer. The employer has a contract to use the system and pays the company running the service making it a business to business model. The company that runs the service provides the servers, mobile applications and the hardware for access control. The bikes can be owned by either company. They would however require upkeep on the hardware side so the project would have another stakeholder to do maintenance.

There is also one more stakeholder that could be part of the project, if the design of the lock also included a GSM connection. That connection would require a mobile phone operator to lease the connection from, possibly at slightly different prices and with different features than consumer phones as only the SMS feature would be needed.

In the planning phase an ad-driven revenue model was considered as one of the ways to pay for the maintenance of the system and keep the service subscription cost low. This is an idea similar to how JCDecaux and other companies earn money from their bike sharing system.

3.2.5 Long term use

The project is currently still in development and needs more work in development. However it is already being tested and the maintenance requires little work: server administration, user support, battery replacements and bike maintenance. Since bikes are used and left outside, there is a possibility of vandalism, improper use of the devices and theft which are hard to predict. Despite the risks we have found the system to work well during internal trials.

3.3 Access control

To access the bike, the user has to be authenticated to the server to gain the authorisation to access the bike. There are many ways that this can be implemented. In the project we rely on the smartphone to communicate with the server to authenticate the user and get the authorisation.

3.3.1 Numeric code

The authorisation can be done via password. In this case the password is a short numeric code that the smartphone fetches from the server and the user has to input by hand into the smart bike lock.

Securing numeric input is difficult because it has to be simple to use at the same time, which limits the length of the code. To make it worse, anyone passing the device on the street can have a go at it. Comparing it to wireless security where the attacker needs to have at least a set of tools, it is very easy to look at a legitimate user input the code and then repeat it or just try any code.

As using one fixed password is too dangerous, security measures like usage of one time passwords have to be implemented. There are standards suggesting use of various algorithms to generate hard to predict and guess numeric codes. One of them is the Hmac-based one-time password algorithm made by M'Raihi et al[25], where numeric codes are generated with a secret key, an incremental counter and a hash function. This ensures that every time a code is generated it is different.

The downside of a counter-based algorithm is however uncertainty of whether both the server and the bike lock have a correct count. The server might get multiple requests for a code if the user makes mistakes while using the mobile application. On the other hand if the bike lock only increments the counter once a correct value has been put in, it can be susceptible to a brute-force attack.

There is however another variation of that algorithm TOTP created by M'Raihi and another team of researchers[26]. This algorithm relies on a clock to generate timestamp which can be used instead of the counter. The result is very hard to guess, secure numeric code. Without an easy way to guess the code, attacker will have to default to using a brute force attack that will have a large number of possibilities: $digits\ used^{code\ length}$. The number of possibilities is then tied to the code length and to the number of digits user can enter. For example, a code made of five numbers, each between 0 and 9, has $10^5 = 100000$ possibilities.

Length of the numeric code is related to the usability as shorter code will be easier to enter. Number of digits used in the numeric code is related to implementation and size of the number pad the device uses. Using fewer digits allows using a smaller lock enclosure.

3.3.2 Short range wireless

Alternative to the numeric code is using a short range wireless like NFC or BLE. This means the smartphone would have to have that wireless functionality enabled so the bike sharing application can pair with the smart lock and send it the confirmation message. Therefore the communication protocol would have to be built so the bike could be unlocked on first time the smartphone would be paired with the smart bike lock. Moreover the confirmation message has to be secure and generated specifically for the bike lock to confirm the user is indeed using the right bike.

From user's point of view, the usage of the wireless authentication would mean picking a bike in the system and letting the phone connect to the smart bike lock. For NFC the communication would mean tapping the phone on the smart lock of the bike. For BLE it would mean being in a distance of a few meters from the bike.

3.3.3 GSM

While BLE and NFC deliver communication link between the lock and the mobile, there's also a possibility to implement a direct link between the lock and the server. This is possible with GSM and a m2m SIM card. The lock could exchange text messages with the server on the status of the bike, reservations and users wanting to unlock it. The idea to implement a GSM system for access control has however a which is that the user communicating with the server can fake being next to the bike, which a short range wireless could confirm. That means the system would have a security hole that could be abused by pranksters. However the m2m communication could be used to track the bikes in the city if combined with GPS.

3.4 Security and location tracking

Currently, the project requirements are based on an assumption that users are not malicious. Therefore the security of the current implementation concentrates mostly on access control, but there is another security issue which is bike localization.

Using smartphone's GPS

In the current implementation of the system, the smartphone is used as a proxy to the server to communicate the state of the bike. The information is based on the action of the user as the bike lock has no connection to the

smartphone. If the bike lock was capable only of short range communication, the smartphone would become a gateway to communicate with the server. The lock could communicate its state, logs, recent access and tampering attempts. It would however still be limited to only when the bike is paired with a smartphone. It does not protect the bike when it is left alone on the street and thief could break the lock and take the bike away. Another security measure should be implemented to remedy this.

Long range wireless

The bike itself should be able to communicate periodically to the server its location and maybe notification of theft attempts. The best tool for that is GPS paired with GSM. While it is possible to use GSM and distance from base stations to determinate the location, it would require a contract and interaction with the network provider to get the locations.

Another alternative could be using WiFi if the places bike was parked at had its own wireless network. It would mean the sharing system would only allow moving between a limited number of stations with safe wireless networks. Moreover the network would be accessible for potentially malicious attackers in a large area around the station and the communication between the bike and the server would be in danger. In case of poorly implemented security protocols the attackers could generate fake bike reports and take the bike away unnoticed.

GPS and GSM

Location based only on the WiFi could be inaccurate compared to GPS, which could result in the users walking a around the location to find the bike. Based on those reasons GPS and GSM are the technologies of choice to locate the bike and report its state to the server.

Chapter 4

Implementation

This chapter concentrates on the implementation of the smart bike lock hardware and software. The subchapters go in detail through design choices and issues for each hardware part. Last part of the chapter describes in detail the software of the lock.

4.1 Tools used to develop the bike lock

Developing a smart bike lock with a variety of features including wireless functionality requires working with embedded devices. There are many architectures with various features that can be used to develop such a project, from 8bit to 64bit chips. Aside from architecture, there is difference also in the popularity, ease of use, vendor and community support. Unfortunately for some platforms it is harder to find materials to learn from. That limits what could be used as a prototyping platform. Moreover since the project is still in early stages when the platform is selected, it is hard to estimate what architecture properties would be necessary to make a good prototype. In this project the exact architecture limitations were not known and so the choice of a platform went towards finding the platform with the best support, which turned out to be Arduino.

4.2 Bike lock prototyping

While working and testing the prototypes security has lower priority as tests are done in a controlled environment. Still, functionalities that can be used to collect information about the users' and system's behaviour are important. This information can be used to diagnose the issues, usage patterns and flaws in the system. It is worth noting that while the smartphone and server

communicate and log user behaviour and errors, the bike lock lacks a direct connection which makes it harder to monitor.

The functionality of the completed lock should include a connection to the server independent of the smartphone, a fine-grained positioning system and theft-detection system. These are features similar to the Lock8 bike lock, however they are quite advanced and require much more work to implement. This is why they were not included in prototypes even though these features would be handy in a smart bike lock.

Therefore, despite the high requirements for functionalities for the final bike lock, in the early iterations it is sufficient for the lock to have a form factor similar to the Lock Box. It is a simple box that fits a key for a traditional bike lock accessible only with a correct numeric code. In order to restrict the access to bikes to users who have used them before. With that as a goal, the bike lock in early prototypes includes electronics to generate new codes in sync with the server to ensure better access control.

4.2.1 Arduino

Arduino[5] is an open-source hardware platform developed in Italy in 2005 based mostly on ATmega chips. In the 10 years on the market it has established as a family of products and gained many clones and extensions made by its developers, users and other companies. These devices of the Arduino ecosystem are also well supported with libraries and samples of code. It's not unusual to find ports of code for this platform. The variety and readily available amount of material make it a common hobby platform as the entry barrier is low and it allows for faster creation of working prototypes. There are many projects shared under an open-source license on websites such as Instructables (`http://instructables.com`). These projects vary in quality and complexity but as most of them are described in detail, they make great teaching material for beginners. Furthermore there are courses and tutorial resources available.

The ease of use for this platform comes from simplifications made on two sides. On one side, Arduino hardware is based on ATmega microcontrollers which were well-known even before Arduino gained popularity. They have good support and have been well tested for variety of simple uses. Additionally, in Arduino the ATmega chip is built into a board that has all the necessary electronics that deal with power, connecting to a computer and making the design safe and robust. These components have been designed to serve multiple applications and it is great for beginners[21].

On the other side, to make the programming simpler, Arduino delivers its own programming environment and language. It is equipped with a rich

set of libraries and code examples and the programming language is similar to C, which is wide-spread in embedded programming. Together with this there's an IDE, adapted from another open-source project. Its intention is to make programming for Arduino as easy as plug-and-play. This programming environment is available online complete with drivers, libraries, compiler and firmware uploader, which is all that it needs. Its popularity shows that it does succeed in making usage of Arduino very simple compared to other embedded devices.

Disadvantages of Arduino

While Arduino is great as a starting hardware platform, it is said to be a bad choice for designing the final product for sale. In communities of ATmega users and other embedded programmers there is criticism of Arduino. This criticism concerns mostly the power inefficiency of the Arduino models and the low quality of code in the base libraries[38]. While these accusations are not unbased, the amount of libraries available for the boards and their extensions simplify development.

Another issue Arduino has is that the regular Arduino models such as Uno are equipped with on-board elements that use too much power. The increased power consumption to much higher levels than necessary means it is very inefficient. This makes Arduino basic models a bad choice for projects running on battery. However there are also many articles online on how to keep them in sleep mode to save more power[14]. There are also many tips about hardware improvements to the board like removal of onboard components[33] or even building Arduino-compatible circuits with a set of minimal, low-power components[35].

4.3 Design of the solution

When the project was first planned a large pool of probable solutions and implementations were considered. The first iterations were chosen to be small and manageable in a short period of time. The initial goal was to make a working device and test it during autumn before the beginning of winter. The second iteration was created during autumn but the testing was minimal until the spring of 2015.

(a) Lock lid closed. (b) Lock lid open.

Figure 4.1: The second iteration smart bike lock, Viola, mounted on a bike. It is attached to the bike frame with screws.

4.3.1 Iterations

There have been so far two major hardware iterations. First iteration, codename Marco, is not in use anymore. The second iteration produced 3 devices called Ilpo, Viola and Satu. All of them are operational, and while Ilpo and Viola are stable products mounted on bikes, Satu is used for testing of experimental features.

The created smart bike locks have been equipped with the following hardware:

Marco Arduino Uno R3, DS1307 Real Time Clock, Batam B2122 servomechanism, keypad, red led, green led

Ilpo Arduino Pro Mini, DS1307 Real Time Clock, MicroSD board, Batam B2122 servomechanism, keypad, bi-color led, button, power switch

Viola Arduino Pro, DS1307 Real Time Clock, MicroSD board, Tower Pro SG92R servomechanism, keypad, bi-color led, button, power switch

Satu Arduino Pro Mini, DS1307 Real Time Clock, MicroSD board, Batam B2122 servomechanism, keypad, red led, green led, button, power switch

4.4 Hardware

As mentioned in Section 4.2.1, the project is based on Arduino boards. The main board does not have all the necessary hardware features, so the project uses a few extensions.

4.4.1 Main circuit board

The main board of the lock is based on the ATmega328p[6] which is an 8bit microcontroller with 16MHz at 5V, 2KB SRAM and 32KB flash memory. The Arduino Uno, the basic model made by Arduino LLC, was chosen at the beginning for the first implementation and was named Marco. Since there were no major issues with running the first versions of the firmware, we decided to continue with the same architecture and use Arduino Pro and Arduino Pro Mini for the second iteration. These two boards produced by SparkFun are using a very similar design to Arduino Uno, with ATmega328p. Locks Ilpo and Satu were based on Arduino Pro Mini and Viola was based on Arduino Pro.

The difference is that the Arduino Uno is less energy-efficient than Pro and Pro Mini, as the SparkFun models have no USB controller and their voltage regulators are more energy efficient compared to the one Uno has. Because the USB controller is gone, the Pro and Pro Mini both have to be programmed with a separate board providing the serial connection to the computer. In this project we use an FTDI chip that plugs into USB port.

Moreover Arduino Pro Mini has been designed to fit all the electronics in 1/8th of the area Arduino Uno takes. The smaller size is both an advantage and a disadvantage. It takes less space but also it is harder to connect the other elements. Due to that there was a lot of work soldering and re-soldering

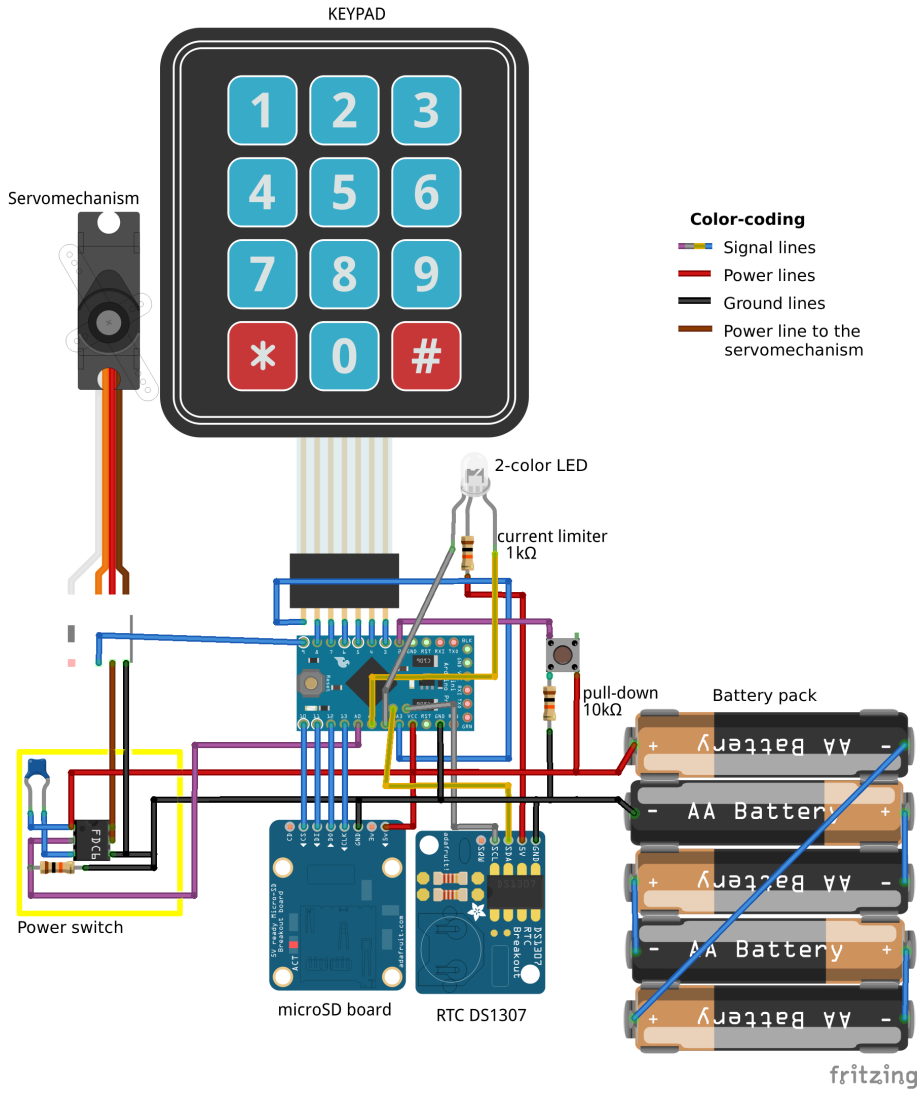

Figure 4.2: Electrical design of the hardware. The diagram displays the ready-made boards looking very similar to the physical elements. The lines have been coloured to make it easier to distinguish the lines that cross in the picture but are not actually connected. Black and red lines are power and ground respectively. All grounds are common and all power lines have the same voltage of 5V and draw power from the same source: the battery pack. Brown color is used for the power line going to the servo which supplies 5V but is not connected all the time. All other colors are just regular signal lines.

the Arduino Pro Mini sets to find the best fit for all the electronics in the housing box.

Lastly, the Arduino Pro Mini comes with 2 more pins made available to the user. While the number of pins is defined by the microcontroller,

Arduino Uno does not have all of them available to the user. Arduino Pro tries to imitate the shape of the board and also hides those two pins. This is the only incompatibility between all the three boards. Otherwise, they can all run the same firmware and perform the same way except for the Uno consuming more power.

The compatibility was important as the development of the firmware did not require rewriting to accommodate for change of the board. Also because all used versions are compatible with the Arduino flagship model, the chances that Arduino libraries will work correctly are highest. However the development is now coming to a point where a change to the microcontroller might be required either to reduce the power of this particular implementation or to use a microcontroller with more static RAM (SRAM).

4.4.2 Keypad

The keypad is the way a user communicates with the device to unlock it. In this implementation a small numeric keypad is used. If this keypad were implemented as single buttons, it would need at least 10 buttons for numbers from 0 to 9. To make the number smaller, it is created as a grid of rows and columns that intersect to create a button-like setup.

To read a key press from the keypad, the software needs to check each button in a loop, waiting for input. Reading is done by powering each line for short moments and waiting for a reading from the lines that would intersect with them. There is a total of 7 pins, which are 3 columns and 4 rows. Aside from the numbers from 0 to 9, the keypad has space for 2 more characters, which are a * and a #, suitable for a phone keypad. While these buttons aren't necessarily required in this project, they are in line with 0, which is important and that means all the rows and columns are in use.

This means that it takes 7 digital pins on the Arduino to connect the keypad. Their location is not restricted as long as they are correctly mapped. The simplest would be to connect them in a row next to each other. In this project that was not possible as they would overlap with special pins such as the hardware interrupt pins 2 and 3, the pins used by SPI bus and the PWM pins used by servos.

4.4.3 Real Time Clock

Arduino can be turned into a somewhat accurate clock by programming with the help of a software library. The downside is that aside from imperfect accuracy the clock will forget the time once it loses battery power[27]. That would mean every time the bike lock's batteries are replaced, the current time

is lost or replaced with a time set at the moment the program was compiled and uploaded to the board. Moreover to have Arduino keep counting the time, it would be impossible to put it into deepest sleep (see Figure 4.6) as it uses the timers to count the time.

The solution used in this project is use of an external board equipped with an RTC chip, a crystal to keep the clock in better synch and a small CR1220 battery that will keep the counter even when the Arduino battery dies. Since the first iterations the board used for this was the cheapest option offered by Adafruit DS1307 Real Time Clock breakout board kit[1] running on 5V. While there is a warning about DS1307 being slightly imprecise, we expected it would be enough to keep the code generation close enough to the server side's.

The use of a dedicated time-keeping circuit is a valuable element in the numeric code-based solution. However this particular chip has issues, which have caused problems in the project. The first main problem is that except for the servomechanism, it is the only element requiring 5V to work with the Arduino. Other chips, like ChronoDot offered by Adafruit require only 3V, but are twice as expensive and have a number of redundant features. It was deemed unnecessarily complex when the hardware was first planned. Yet, if an RTC like that was used instead, the Arduino could have a different power source and possibly also save more power.

Another important issue is that in original RTC board extension, the main chip is a non-industrial version. In this case it means DS1307 is incapable of working in a wide range of temperatures, including below 0 °C. Even storing it in too low temperature during winter could cause problems. The problem was noticed before winter and changes in that area became a priority. While it was not planned, we decided it is better to be ready in case the device stayed in low and freezing temperatures. Thankfully, the same chip is also produced in industrial variant DS1307N with a wide-range of temperature-resistance and better accuracy.

Another argument for replacing the chip with an industrial counterpart was an unexpected event when the RTC's time changed by 7 minutes compared to real time, which made the Arduino board generate a wrong code for the user. Since replacing the chip with an industrial one, the issue has not resurfaced.

4.4.4 Servomechanism

The bike lock needs a physical lock with a moving element controlled by the board. There are several types of motors, like stepping motors, solenoids, DC motors and servomechanisms[27]. Most of them were deemed unsuitable

for this project. For example stepping motors need connecting with at least 4 cables and with software counting their steps, that is not always accurate. Solenoids are connected with with only 2 cables, but advice from the Helsinki Hacklab members was that they are sensitive to movement and could unlock when shaken. As this box was planned to be mounted to a bike frame, use of solenoid was ruled out.

Out of two remaining choices, a DC motor requires a board with an H-bridge to be operated, which would be an extra cost for the project. The last choice, a servomechanism (also called servo) can be connected directly to the pins of Arduino, which is why it was chosen[27]. Servos found in shops at the beginning of the project required a minimum of 5V. This requirement for the servo was one of the reasons to keep using 5V during the course of the project.

There are two servomechanisms used in the project. The first one is a more expensive Analog Feedback Micro Servo with metal gears Batam B2122. The second one, bought later is a cheaper Micro servo Tower Pro SG92R. Both were ordered from Adafruit Industries shop. We could not find detailed specification for either of them. The known differences between them, aside from price is that Batam has a position feedback line in addition to the basic connectors. However that functionality has not been used in the project so later we decided to try Tower Pro servo which has only internal feedback. The two servos are set to use slightly different values for lock positions requiring code changes.

The servos and voltage

Later in the project we noticed a difference in power draw and operational voltage. The Batam servo, which was used with Arduino Pro Mini models, works by drawing power directly from the board. However the Tower Pro servo, paired with the Arduino Pro board, would not work when connected directly to the board. Instead it had to be connected to the battery pack. It might be either the difference in their current draw or difference in onboard regulators of the Arduino boards.

While this workaround makes the Tower Pro servo harder to use, this model is better than Batam because it can also work with voltage as low as 3V. This was confirmed during power tests done on the devices, described in Section 5.4. This means that the initial assumption that the system will require 5V to power a servo in false and the system can be also implemented with lower voltage.

Power draw of the servos

Another issue, originally unknown but noticed in power measurements, was that both of the servos used in the project draw power constantly, even when not moving. They do that to hold a set position. While for some projects that feature might be desired, here it is useless. The lock is designed to hold its position without exerting the servo. The power the servo is drawing is lost. Additional power loss happens also when the servo is creeping, which is a movement happening randomly when servo has trouble setting a position.

The power lost by the idling servo has been estimated and described in detail in Section 5.4. These numbers are not high but over time the power loss becomes a large issue. Currently the servo is separated from the battery with a power switch described in Section 4.4.6.

4.4.5 SD card board

It is quite valuable to know what is happening with the device during use, especially in early stages of development. However the Arduino series based on chip ATmega328 has only small EEPROM storage and extracting the logs from it would require extra work. This is why the project includes also an extension board giving access to a microSD card. The board chosen for the project was picked from AdaFruit store[1] due to convenience. It has a simple design, uses different pins than the RTC and it can be powered by both 3V and 5V. It is also compatible with existing libraries and a detailed tutorial explains how to format microSD cards for logging purposes.

Currently the easiest to buy are microSDHC, high speed and high capacity cards. They can provide more than enough capacity for logging and are easy to plug in and insert and remove from the locket. A few 4GB capacity cards were bought for the project, but they were not used much due to problems described in Section 5.2.2.

4.4.6 Power switch for the servo

The project's main concern was to keep the design free of self-designed electronics so there is less risk the bugs in the implementation will be caused by this. However there are still elements that had to be added externally. The main addition in this area is the power switch board made by Eero af Heurlin[2], a member of the Helsinki Hacklab. He made the design as well as produced the board. Its schematic can be seen in the Figure 4.2 in the yellow rectangular frame.

In software, the power switch is driven by one of the pins of the board sending a signal for the power switch to supply power to the servo on demand. Most of the time the servo has no connection to the battery and is not attached by the software.

The importance of this addition was noticed during detailed power measurements done at the Helsinki Hacklab. The main point of it is to reduce the passive power draw of the servo, which contributes to faster draining of the battery (see Sections 4.4.4 and 5.4).

The Helsinki Hacklab is a hackerspace located in Helsinki. It is a non-profit organization that maintains a space for people to use for working on various projects such as hardware design and electronics. During this project we have visited the Hacklab to use their tools. Power measurements described in Section 5.4 were done there.

4.4.7 Other peripherals

Last to be described are the most basic elements: a bi-color red and green led and a button to interact with the user. They are connected to the board with long cables and glued inside the enclosure with hot glue.

The button was used as a solution to the problem of waking up the Arduino from deep sleep which it would enter to conserve energy. It is connected to pin 2, which is a physical interrupt. There is also another physical interrupt pin, number 3, which was used for the test feature which would detect if the lid of the bike lock box was closed. For now it has only been implemented in Satu lock. Currently the feature is on hold as with implementation of the power switch, which is a high priority feature, the Arduino Pro Mini ran out of pins. In Arduino Pro, which has 2 pins less, the microSD board has been disconnected to use other high-priority features.

The lid detection still needs some improvements. Once an elegant version is implemented it could be used in the stable versions. It will be useful for checking the lid before locking the box as well as automatic closing of the lock once the lid is lowered.

The button and the lid detection all require a pull up or pull down resistor. Without it, the connection to the board will have a problem with floating signal. When that happens, the microcontroller might read incorrectly a change of value on the pin. The led requires a current limiting resistor or it will draw very large amounts of power.

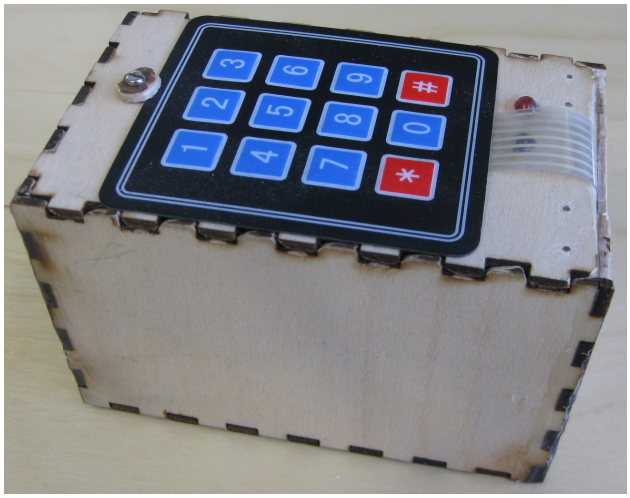

Figure 4.3: The enclosure of the first iteration bike lock, Marco. It was made out of plywood, hot glue and metal. Inside it has an Arduino Uno, a pack with 6 batteries, the RTC board and a bike lock key.

4.4.8 Enclosure

The bike lock in the first iteration had an enclosure made out of plywood cut with a laser-cutter. After cutting it was hand-modified to work as a box and while it was functional, it didn't look too advanced (see Figure 4.3). This size and shape went against the project's goal of making an appealing enclosure and for the future iterations a professional designer was employed to design and order prototypes made by rapid prototyping method (see Figure 4.1). The new enclosure is attached to the bike on the bike frame with screws going through the standard screw holes available in many bike models. The design however still lacks certain features and requires further remodelling. More on it in Section 5.3.

4.5 Power use of the device

The power source of the first prototype was a classic battery pack with 6 AA 1.5A batteries giving together an average of 9V and providing about 2000mAh. The Arduino Uno equipped with the peripheral boards and the servo would last about 2-3 days before the batteries would have to be changed. This is why sleep was pursued as a strategy to decrease power consumption. However the power consumption remained large and the size of both the board and the battery pack drove the decision to move to Arduino Pro and

Arduino Pro Mini.

In the second iteration with the new boards, deep sleep was implemented extending the life of the battery making it possible to have the lock work for 10-14 days on one set of batteries. After the measurements described in Section 5.4 were done and the power switch was applied, the time of working on the battery was extended to 11-13 months. Now, with knowledge of the recent improvements in battery life, we are looking into other ways of further improving it.

The change of boards was a minor step because all other elements remained the same. Among them, were notably a cheap RTC extension and a servo, both of which required 5V input voltage. As all the other elements in the system can either run on 3.3V or 5V, it was decided to keep using 5V. However, as the project progressed, it became apparent that it would have been better to move to lower voltages to reduce the power draw.

As mentioned in Section 4.4.1 Arduino Pro and Pro Mini both were created to use less power than Uno. Additionally, for each board SparkFun has a 3.3V and a 5V version. All of them are based on ATmega328p, but the 3.3V versions operate with 8MHz. It can be seen in the manual for this ATmega series[6], that the power draw of the microcontroller depends on its speed, so a chip working at 8MHz consume less than at 16MHz.

The difference between a 5V and a 3.3V Arduino in this case is the speed at which it is running. Moving from 5V to 3.3V would mean decreasing the speed. The current implementation of the hardware and software is actually very fast and deals only with user input, which in slow compared to the microcontroller's processing speed. The decrease from 16MHz to 8MHz would go unnoticed by the end user. The benefits from lowering the voltage are worth checking this alternative.

4.5.1 Batteries

There are multiple ways to power an Arduino. Picking the most suitable one requires planning and understanding of how the board operates as well as how the power source behaves. If the source delivers a stable 5V over time, it could be fed directly to the board. However, that is not the case for most sources, such as batteries or accumulators. Moreover, depending on the type of the battery, the voltage might drop steadily over time and require using a voltage regulator.

Arduino Uno is equipped with a voltage regulator, which is robust but inefficient. It lowers the voltage of the source to a stable 5V while losing some power. It also requires that the source stays a bit above 5V so it can transform it to a stable 5V. This means at least 6-7V should be used as input.

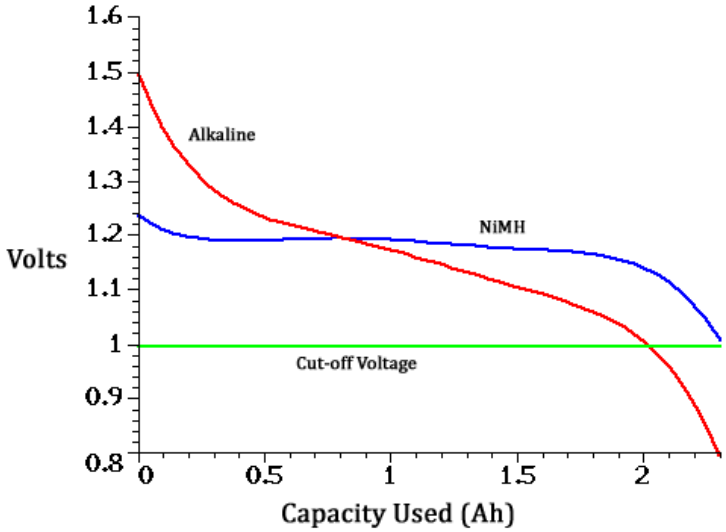

Figure 4.4: Diagram of voltage change in function of capacity of batteries[41]. It compares the characteristic of the alkaline and NiMH batteries in function of lowering voltage over time. It is clear that NiMH batteries mainting a stable value of 1.2V much longer than alkaline batteries.

Following that, the first iteration was equipped with a commonly used box holding 6 disposable, alkaline 1.5V AA batteries, giving together an average 9V.

In the second iteration, it was decided to lower the voltage a bit and move to a reusable source with a more stable voltage. The choice were NiMH batteries, because the voltage they output has a flatter, more stable curve while discharging. Regular NiMH aren't however recommended for low power applications as they have high self-discharge rate, meaning they lose power over time. Fortunately in recent years a new variant of NiMH, called Low Self Discharge (LSD) has been introduced and it was thought to be a type good battery to continue the project with[41]. Firstly, it would deliver a more stable voltage around 1.2V (see Figure 4.4). Secondly, the LSD variant delivers much longer shelf-life, which would be quite important for keeping the bike lock in use for over a year.

However, in both iterations where AA batteries were used, a downside to the choice became apparent quite fast, which was the size of the battery pack. The first iteration's battery enclosure fit 6 batteries. In an attempt to shrink it, the second iteration moved to 5 using batteries. At first they were connected by soldering on the cable connectors but this turned out to be too flimsy. To fix this, an uncommon enclosure for 5 batteries was found and

ordered from China via the Alibaba online shop. Unfortunately, out of the minimal order of 5 enclosures, 2 were broken and only 3 could be used. Now they are used for the three prototypes, and while AA are easy to replace, the packs are still quite large.

The devices use currently locally sourced Varta High Rechargeable ACCU Ready to Use accumulators with the capacity of 2400mAh. It was noted in first weeks of use that they did not hold for the expected 2 weeks, but would discharge completely before that time. The voltage of the batteries ended up uneven even though when connected they were all with the same voltage. That behaviour is unexpected and troubling. Plausible causes are poor quality of the battery packs or the accumulators themselves.

An alternative to the AA accumulators are Lithium Ion batteries, which are smaller. They are also commonly used in mobile devices nowadays. However a one cell Li-Po delivers only 3.7V which means the battery would either have to be made of 2 cells, requiring delicate charging, or it would need an additional board acting as a step-up, increasing the voltage to 5V. The latter case is more recommended, but stepping up the voltage also adds slightly to power consumption of the system. The amount of power lost by stepping up depends on the chip used but sellers like Pololu[32] estimate about 10-30% power lost in their devices.

It is worth noting the if the bike lock was built on 3.3V Arduino Pro Mini, it could be fine with a 3.7V battery without a step-up. It would require testing though.

4.6 Software of the device

As the project is in prototyping phase to reduce the amount of work, it has been started using the wealth of libraries and ready code examples offered by Arduino environment.

In development of the software good code quality guidelines are followed. The code has been split into files and well commented. Once connected to a computer, the device produces debugging messages over serial connection about its internal state and actions. Each area of the program's operation is described below in more detail.

Code logic

The code of the bike lock went through multiple major versions, each adding new functionality or being a refactored iteration. Figure 4.5 presents the

simplified logic of the device. A more detailed diagram of the program is in Figures A.1 and A.2.

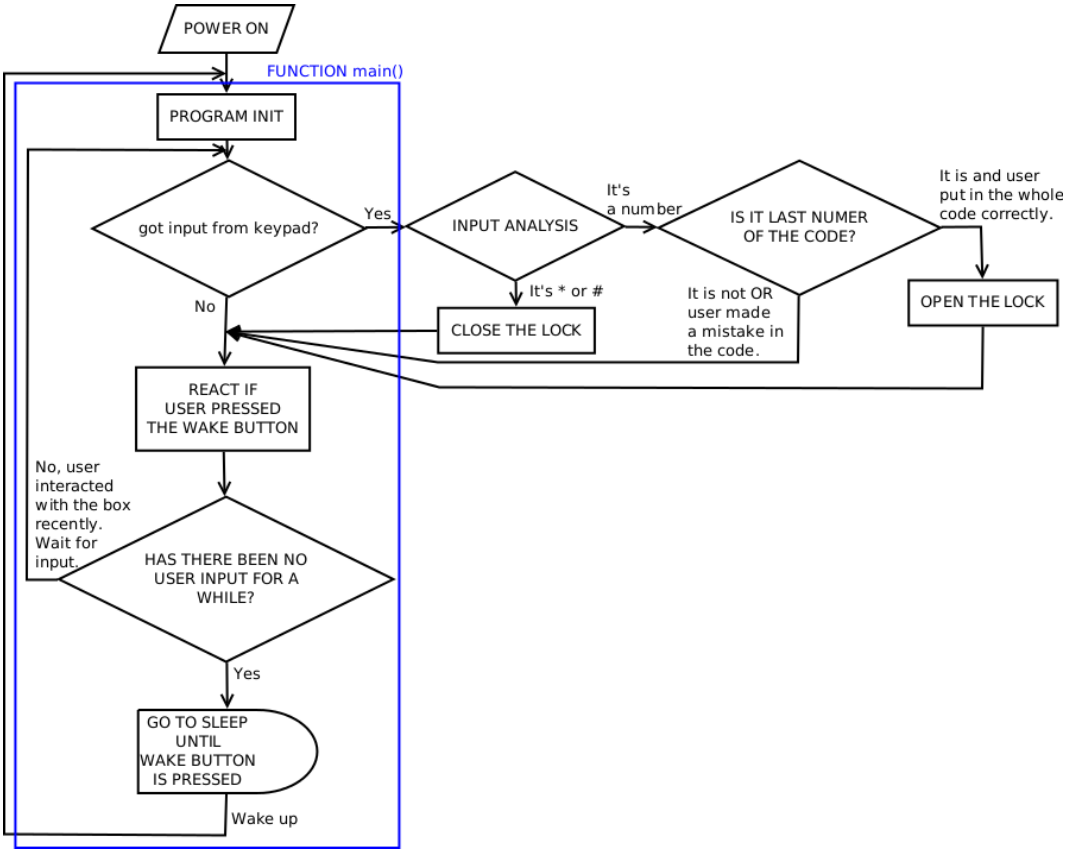

Figure 4.5: Simplified logic of the bike lock program.

4.6.1 Main

Function main() is a part of the software that the microcontroller will keep executing in a loop unless it has been disturbed by another task. In this project, the main() started as a large, monolithic function and was subsequently refactored to performing small, atomic actions. In the beginning, all numeric inputs were caught in one run of the function and the code numbers were compared one by one on the spot.

While changing the way user input is handled, also keypad event listeners were temporarily tried out. The listeners are quite powerful and abstract feature of the library that allow catching various keypad events, not only key presses. Each key press could be caught and handled in 3 separate moments: when the key gets pressed, when it is held and when it is released. While it is

a very powerful tool, it came with a price of higher size of the compiled code and much higher SRAM usage. It had to be abandoned when new features were added that would not fit in the memory size limit for the ATmega328p.

Now, main() catches only key and button presses with a simple function and forwards it to another function to handle it. The functions save some values in global variables and process them in suitable loops.

Aside from handling user input, function main() is keeping track of user inactivity and puts the device to sleep when there are no actions for a specified period of time. Figure 4.5 marks the functions the main() function goes through every time.

4.6.2 Input

User input is caught in main() and processed in later functions and the result is returned to the function to give the user feedback. The way the input is processed changed a few times in a significant way.

In the first prototype, code was accepted prefixed with # symbol. That made the code extremely easy, but at the same time added more work for the user. After a rewrite the code, it became more complex. In the following code iterations the feedback logic was created to flash a red light every time a wrong button is pressed. While it is not a secure logic as it could lead to a malicious user easily cracking the code and opening the lock, it was quite useful during the first tests of the code comparison logic.

The logic originally also would reset the comparison of the input against the code once an error has been made. During internal tests, user feedback about this feature was quite negative. Users reported that it is confusing and it would be better if each input series was the length of the numeric code, not of variable length depending on the user input.

That comparison logic was removed during refactoring due to security and usability concerns to be replaced with a simple check the length of the numeric code. The user receives feedback after they have finished putting in all numbers and if the input does not agree with the numeric code, user can put in the correct code right away.

One remaining security concern that has not been yet addressed is detecting multiple failed attempts to unlock the device. Setting a reasonable limit for failed code input attempts in a row would be a good security improvement. It is a goal of future iterations to add that feature.

4.6.3 Code validation

The numeric code to compare the user input against is generated when user starts interacting with the bike lock. It is calculated using the current time and a secret value using the TOTP library for Arduino. A small deviation from the library is the period in which one numeric code is valid as it has been customized to match the situation.

Additionally an advanced feature was added later in the development to prevent a special case and increase the lock's reliability in face of unreliable RTC chip. The change was made for the corner case of the user getting an old code from the mobile application and putting it with a delay long enough for the bike lock to already generate a new code and evaluate the user input against it. The device has been programmed to have a short period of dual validity of old and new code, called BOTH_VALID in Figure A.2. This feature improves the usability but also lowers the security of the device, since for a short time there is a higher possibility to guess the code as two are valid instead of one. The code has been written to make sure only the new or old code are accepted, not a mix of both.

The code entered by the user is sent from main() to a function that generates a fresh numeric code and compares the input against the first number in the numeric code. Two global variables are incremented then: one is incremented for each input, one when the input matches the numeric code. Next input will be matched against the next number of the code until the length of the code is traversed. If all numbers typed in by the user match the numeric code, the increment and the comparison variables will be equal to the code length which will mean the code is correct. Otherwise the comparison variable will be lower, which means the input was not equal and the comparison failed. The mechanism is detailed in Figure A.1.

4.6.4 Locking

Once the user input is confirmed to be same as the numeric code, the Arduino sends a signal to the servo to move into a position allowing the user to open the lid of the box and retrieve the key. That position of the servo is a number that has been determinated through experiments. Different servos have different values for their positions. To make sure the same code will work with the same servo, the latter also has to be put in the correct position with a correctly attached ending. These are important factors to remember if more prototypes are going to be assembled.

The project now has two different servomechanisms, and they are both using different values. However the code base is common for all the bike

locks, which means it has to be written in a way to allow easy switching of the definitions of the values before compiling and uploading the code to the boards.

Currently, the bike lock tracks internally if the lock has been opened or closed to prevent powering up the servomechanism unnecessarily. However it is possible to press the button to close the lock when the lid is not closed. If a user were to press the button in that situation, they would have to input a correct code again to relock it. Work has been done to prevent this scenario by detecting if the lid is closed or not. The work is however still in an incomplete state and needs more work both in electrical design and programming.

4.6.5 Sleep

Table 10-1. Active clock domains and wake-up sources in the different sleep modes.

	Active clock domains					Oscillators		Wake-up sources						
Sleep mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main clock source enabled	Timer oscillator enabled	INT1, INT0 and pin change	TWI address match	Timer2	SPM/EEPROM ready	ADC	WDT	Other/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADC noise reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Figure 4.6: The sleep modes of the ATmega328p microcontroller[6]. The modes differ by the numer of features that are turned off to conserve energy. The most power saving mode is called *Power-down* listed third in the table.

The ATmega microcontrollers are capable of sleep to save power. Sleep means setting up wake up conditions and turning off all other functionalities and entering a power saving mode. The ATmega328p offers various stages of sleep (see Figure 4.6), which can be picked depending on the powered down features and available wakeup conditions. In this project, the wake up condition is an external interrupt on one of two special pins of the microcontroller. These interrupts are able to wake up the ATmega328p from power-down, the deepest sleep, which is used in this project.

The sleep is entered after a period of no input from outside and the signal to wake up is given by the user in the form of a button wired to the

microcontroller's interrupt pins INT0 and INT1 (see Figure 4.6). During development there were multiple attempts to wire the numeric pad to the Arduino in a way that would allow wake up on any keypress from the user and also on pin change interrupt. All attempts however produced unreliable solutions due to the how the numeric pad is wired in a matrix. The work was moved to using a button, which initially was also unreliable due to electrical reasons: the floating of the pins. Solution turned out to be adding a simple pull-up resistor to the button.

Managing sleep and interrupts can be done in Arduino using the embedded C libraries and functions as well as the default Arduino libraries. There are also projects trying to make a layer on top of that offering power saving functions. One of them is the Narcoleptic library[22]. There seem to be many ways to go around coding that behaviour since both Arduino and native C libraries provide a variety of functions. In this project, we use both native functions and the Arduino functions.

The project uses a large number of Arduino libraries to manage all the hardware elements: serial communication for debugging, keypad, servomechanism and RTC. There are also two embedded C libraries for sleep: avr/sleep.h and avr/power.h as well as a hash library to work with a TOTP library found online. Since they are all tailored for embedded programming, it is expected that only the necessary functions are included in the final binary uploaded to the Arduino to optimize for size.

Chapter 5

Evaluation

This chapter starts with technical evaluation, which includes the power consumption measurements of the device. It then proceeds to report the results of user surveys from internal testing.

5.1 Evaluation methods

We evaluated the project and the service from different sides. The first side is the hardware suitability for the task which was noted during development of the smart lock.

The next side is an evaluation which looked at the long-term use of the device and how suitable the product is to be used by the service in the condition it is in now. We take into account the security, maintainability and power-management.

The last side, described at the end of the chapter is an evaluation of the service as a whole done. It is achieves through two surveys. These surveys were conducted in the office on the participants of the internal beta testing. This trial was running for 2 weeks at the end of April, 2015.

5.2 Hardware limits

During the development of the project the limitations of the ATmega328p have been reached more than once. The project is still missing certain features that could prove to be too much for the architecture of the current chip. ATmega328p is an 8bit microcontroller which is an older and less powerful architecture type compared for example to ARM chips.

Aside from the architecture, it is also possible that the large software size and high SRAM use are caused by the inefficiency of the libraries provided for

Arduino. Unfortunately the second hypothesis is impossible to check exactly at this stage as it would most likely take too much time to implement the same functionalities better than the libraries do. However, faced with the limitations, the firmware has been optimized a few times to reduce the use of resources. The four main issues have been listed in following subsections.

Table 5.1 illustrates the code size and the SRAM use for each major iteration. The table also gives a short list of features per iteration, such as basic operations (serial connection, keypad logic, numeric code generation and comparison), sleep, microSD handling, box lid position checking, servo power switch and improvements in code.

5.2.1 Logic optimisation

The code of the bike lock has been growing as more features were added. Table 5.1 compares the code size in bytes. The maximum size of the code that can be uploaded to the device is 30 720 Bytes. While none of the numbers in the table get that exact number, during development of each iteration, the code has been very close. Each time it was optimized as much as possible resulting in the sizes given in the table.

The largest increases in the code size are the addition of microSD functionality, sleep and the keypad listeners. All of these functions require additional library functions which add to the code size as well as the logic required to use them.

Creation date	Code size [B]	SRAM use [B]	Features
27 Jul 2014	14 916	1255	basics
2 Sep 2014	15 064	1219	basics, logic improvements
15 Oct 2014	17 968	1277	keypad listener, sleep
20 Oct 2014	28 496	1724	keypad listener, sleep, microSD
20 Oct 2014	20 062	1680	basics, sleep, improved debugging
17 Nov 2014	28 084	1739	basics, sleep, microSD, lid check
29 Apr 2015	20 152	1688	basics, sleep, power switch
	30 720	2048	max. values for ATmega328P

Table 5.1: Resources used by each large firmware iteration. The table contains the dates on which each program was created, the size of the most recent version of the code after compiling and the use of SRAM calculated after the program was loaded on the board. Last column lists main features implemented in the code. Basics mean the serial connection, basic numeric code check logic and running RTC

The microSD functionality causes the largest increase of code size but there is not much code written in the application to use it, which means the libraries are the issue here. On the other hand, sleep functionality mostly uses direct C programming commands to the processor, so the library size cost is low. Last are the keypad listeners which are taking only little extra space in comparison.

5.2.2 SRAM optimisation

Another resource drain was discovered later, and it was the use of the SRAM. The SRAM size on the boards used in the project is 2KB or 2048 Bytes. The attached Table 5.1 lists the SRAM usage per iteration. The measurement are collected right after initialization of the code and before the system starts processing tasks, so the processor is idle.

We can see in the table that the SRAM use increases with each iteration. Largest increases were caused by introduction of the microSD card handling and the keypad listener. Decreases in the use can be noted at the improvements in logic and debugging.

The debugging in this project are messages sent over serial connection to the pc. They exist in the code as plain-text strings. By default, Arduino programmer stores all the static values in SRAM, even though they do not change and could be therefore stored in static memory. It is possible to change that default using a macro F("string"), which causes the "string" value to be stored inside flash memory. After applying that patch SRAM use dropped. It is not noticeable in the table, as the iteration afterwards also included new, resource-consuming features.

While testing the SRAM, we also checked an example code for writing to a microSD card and the SRAM use was at 1054 Bytes when the program was writing to the card and sending debugging messages over the serial connection. We checked that the base features such as serial connection should use about 100-200 Bytes. This means that microSD handling functions in the example would need between 800-900 Bytes of SRAM free. Looking at the Table 5.1, the largest increase of SRAM use between any iterations was of about 500 Byte, which is smaller than expected 800-900 Bytes. On the other hand, there is only one iteration with enough unused SRAM. All other had larger SRAM use.

We deduced the microSD libraries were using the SRAM but as there was not enough SRAM, they would fail. We based this also on the observation that no iteration that includes microSD card support managed to write anything to the microSD card. Later we learned that because the microSD cards are formatted in FAT32, the library has to write a sector of 512 Bytes,

which is reserved as a buffer in SRAM in advance. As there was not enough SRAM, the buffer was never created and writing to the card failed.

Logging

The microSD board was incorporated into the hardware prototypes and the example projects were ran to determine if it is set up correctly. This was a success so the project moved to incorporating the logging to microSD functionality into the main project. Adding the microSD logging to the code did not work and the section above detailing the SRAM use explains why.

An alternative to microSD logging could be making use of the EEPROM memory of the Arduino chip. This has not yet been implemented or tested, but from research on the topic, it is noticeable that extracting the information from the board might be problematic. It might require loading a special program for extracting the data from the EEPROM and then loading back the original program. Additionally, EEPROM can be written only a limited amount of times, so it would have to be done efficiently.

5.2.3 Arduino connectors

Another hardware limit of the board turned out to be the number of pin connections. With the current setup, all the pins Arduino Pro Mini offers are in use and adding new hardware functionalities would require attaching more pins.

There is a possibility to add more extensions via the SPI connector, where they would be managed in a master-slave mode by the microcontroller, however that would mean that the new extension would have to use this connection and that all the devices connected via SPI would have to take turns communicating to the microcontroller. Currently the microSD board occupies the SPI pins. Either it would have to be removed permanently from the design or the program on the microcontroller would have to manage this situation.

Alternative option for that would be adding a pin extender or moving to use a board or a microcontroller with more pins.

The Arduino Pro has 2 pins less than Arduino Pro Mini, so Viola based on it has to work now with the microSD board completely disconnected to make space for the power supply pin.

5.3 Internal hardware testing

In addition to the software testing, the prototypes have been continuously tested during development.

Enclosure

The enclosure shape and functionality was difficult to plan and predict. In first iteration of the enclosure, the materials and shape were sub-optimal, which lead to employing a designer to make a professional design. However the design turned out to have a few design flaws that were hard to foresee before the enclosure was used.

One of the interesting issues was the problem of screws touching the electronics and making the lock inoperable. This was fixed by wrapping the electronics in bubble wrap. Another issue with the design was lack of battery holder, which at the beginning was remedied with soldering of the batteries together. However the soldering points would not hold together which prompted us to find and order a suitable small enclosure online (see Section 4.5.1).

Another seemingly small issue was the size of the bike key compartment in the lock. As it turns out only very small keys fit in it, which has limited us to use lower quality bike locks in the project. In the future the design will have to be corrected if we wish to use or example U-locks.

While some of the problems were solved with temporary fixes, some still remain. Among them there is low security as the box can be opened with a particular screw-driver type. Additionally when the lid of the box is open, the compartment with electronics of the box can be also accessed through the hole in the lock. This is a danger in case the user would try to access it.

The many issues are mostly due to the fact the enclosure was still in its first phase of prototyping. It did not get enough iterations and attention, which resulted in continued security and design issues.

5.4 Power measurements

The power consumption is among the biggest issues in the project. The smart bike lock needs to be able to work for as long as possible without a change of batteries, or it will be too tedious to maintain it. Being able to measure that power consumption would have been a great advantage. However the only sensitive enough tool, a multimeter, would require checking the power readings multiple times to be able to average it over time. Best for

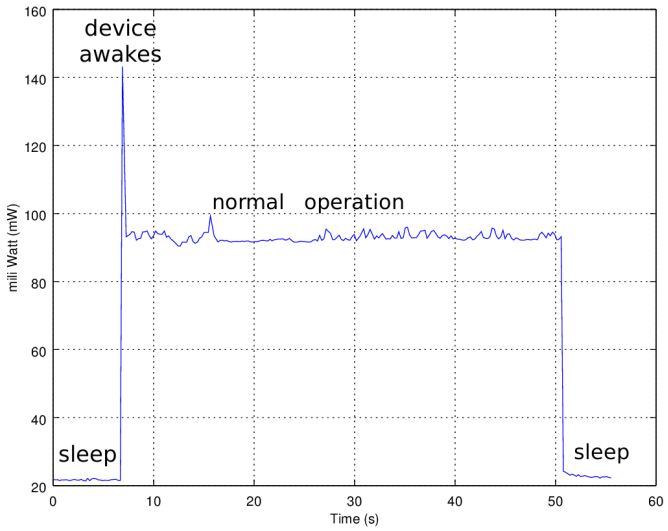

Figure 5.1: Power consumption of the bike lock while in use. The figure is a measurement of opening and closing of the lock. It was done with a lock not equipped in a servo power switch. The servo is therefore connected and drawing power all the time. There are no spikes of power for the two moments the servo moves. The overall power draw is high. This measurement is the worst case scenario in which the servo is connected the whole time the device is awake.

measurements are devices that supply the power while measuring the draw at the same time.

To measure the power consumption, we have visited the Helsinki Hacklab to use their hardware for power monitoring. Measurements were done with a HP 6632b power supply on the latest hardware and software iterations.

Results

The results from measurements reveal that there is a very large power draw from external elements that are not in use, but as they are directly plugged to Arduino or the battery, they use power while idle. The measurements have shown which elements those were and allowed for separating them.

In the Figure 5.1 we can see how the bike lock uses the power. At first the bike is asleep and the power consumption is low. Then as it wakes up, starts to charge its capacitors, flashes the led light and adjusts the servo position. These actions show up as a momentary power increase. Then the power draw lowers to a certain level and stays there during normal operation: receiving input from the users, flashing the light, opening the lock and waiting for more input from the user to close the lock again. After a certain amount of

time passes without user interaction, the lock goes back to sleep.

Variations	**P. awake [mW]**	**P. awake Δ [mW]**	**P. sleep [mW]**	**P. sleep Δ [mW]**
board, RTC	82.34 mW	-	1.26 mW	-
board, RTC, power switch	83.42 mW	1.08 mW	1.23 mW	-0.03 mW
board, RTC, servo	118.18 mW	35.84 mW	36.34 mW	35.08 mW
board, RTC, microSD card	99.41 mW	17.06 mW	9.50 mW	8.24 mW

Table 5.2: Average power used by the elements of the smart bike lock. The first row has the values measured and calculated for the necessary basic elements: the Arduino board and the RTC chip. Following rows have connected additional elements. The P.awake column has values measured while the device was awake, while the P.awake Δ calulcates the increase of power consumption using first column as a base value. Similarly P.sleep column has values of the device that is sleeping and the P.sleep Δ column has the difference. All these values have been calculated from measurements done on a recent iteration of a bike lock by disconnecting and reconnecting various elements to get detailed measures for each feature.

Issues identified in the measurement

Table 5.2 contains multiple average values of power consumption of the device with different hardware elements connected in order to determine their power consumption. The first and third columns are the total average for measurement. First column is the power of the awake and idle device. Third column is the power of the sleeping device. Second and fourth column are the difference calculated by subtracting the power consumption of the base, which is Arduino Pro Mini board with RTC. All measurements were done on an idle device over a span of 3 minutes.

From the measurements we can see there are a few things that increase the power consumption of the device. The main issue is the servo, which draws the most power even when idle. Connecting it through a power switch to the board reduces that consumption significantly. The second issue is the microSD board, which when it has a microSD card in the socket, draws a lot more power when awake. When asleep, it draws less power, but still a significant amount.

There is one negative value in the table for the difference between no devices connected and servo connected through a power switch. It might be that it is due to floating pins being reduced as the pin driving the power switch disconnected without the power switch and may float.

5.4.1 Power consumption model

From the measurements we can create a model to determine how long the battery will last with known power usage. We can use the known average idle power and the power of the two actions: unlocking and locking of the smart bike lock. The scheme follows assumptions that once a bike lock is opened, it has to be also closed. This means we have a simple Equation 5.1 that depends on the number of times n these two actions are performed.

$$Energy_{\text{total}} = n \cdot Energy_{\text{move}} + Power_{\text{idle}} \cdot time_{\text{idle}} \tag{5.1}$$

From measurements shown in Figure 5.1 we can calculate that act of opening and closing the lock uses 4.06J over 43.6s, which in Equation 5.1 is called $Energy_{\text{move}}$. When unused the bike is sleeping and the lock consumes 1.23mW, called $Power_{\text{idle}}$. The bike lock is powered by a set of batteries that we assume have about 2000mAh and 6V. That is 43200W, which is our $Energy_{\text{total}}$.
From these assumptions we get values listed in Equations 5.2. We then put them in Equation 5.1 and get the result in Equation 5.4.

$$\begin{aligned} Energy_{\text{move}} &= 4.06J = 4060mJ \\ Power_{\text{idle}} &= 1.23mJ \\ time_{\text{idle}} &= time_{\text{total}} - n \cdot 43.6s \\ Energy_{\text{total}} &= 43200J = 43200000mJ \end{aligned} \tag{5.2}$$

$$43200000mJ = n \cdot 4060mJ + 1.23mW \cdot (time_{\text{idle}} - 43.6s) \tag{5.3}$$

$$time_{\text{idle}} = 35121994.81s - n \cdot 3300.81s \tag{5.4}$$

n	**yearly use**	***time***
0	none	1 year 41 days
365	1 a day	1 year 27 days
730	2 a day	1 year 13 days
1095	3 a day	1 year
1460	4 a day	11 months 16 days

Table 5.3: Battery expectancy with bike use over time with daily frequency of use projected for a year.

The results presented in Table 5.3 show that with the bike used once a day on average, the battery should last it a full year. With larger amount of rentals per day the battery time lowers slowly. The results show that with moderate use, the bike lock can work without issues for 8 months which is close to the length of the biking season in Finland.

5.5 Feedback from users

In last 2 weeks of April 2015, the project ran a closed internal beta at InlineMarket Evolutions. The beta used prototype of the mobile application for iPhone and a cloud server. The test system offered two bikes, a male mountain bike called Ilpo and a female city bike called Viola (see Figure 5.2). Both were equipped with a prototype bike lock.

The beta was a test run before external testing could be started. It was to check stability of the solution and gather more feedback on software and hardware. At the end 5 people from the company decided to use the company bikes. Almost all of the testers in this beta own bikes so the bike sharing service competes against their own bikes. Additionally two users have a technical background, while the rest are marketers and managers.

After the beta was over, we gave users two surveys. One of them is the standardised System Usability Scale. The second one was a custom, project-related survey asking questions about the usability of the elements of the system and the functionalities that are not implemented yet.

5.5.1 The System Usability Scale

The System Usability Scale (SUS) survey was developed in 1980s by John Brooke[9] as a quick way to evaluate system's usability. The recommended use[34] for it is right after the users had a chance to use the system but before any discussion about it. The testers are presented then a set of 10 statements. They have to evaluate the statements on scale of 1 to 5 if they agree with them or not. The statements are arranged so every odd one is positive and every even one is negative. The answer values are normalized: answers for negative statements subtracted from 5, so 2 becomes 3, and for positive ones are lowered by 1 to get numbers from 0 to 4. Results can be seen in the attached Figure 5.3. Five open ended questions were added to the survey to gather also free form feedback about the system.

Despite the recommendation, due to project limitations, the test had to be ran a few days after the testing was over. Moreover the tests were internal

Figure 5.2: The second iteration bike lock, Viola, mounted on a woman's city bike. It was one of two bikes used for the internal testing.

and testers were from the same workplace as developers of the system. These circumstances may have impacted how the testers graded the system.

Lastly, the users were grading the system as a whole, not just the smart bike lock which is the focus of this thesis. To get more specific answers we have created the custom survey described below.

System Usability Scale results

The values on the scale are all higher than the middle value, but at the same time the error bars are very long. It is possible that the positive feedback might be inflated by the environment explained above. Similarly the error bars can be caused by the different situations of the testers. Further explanation can be found in the open text questions.

Open text feedback

The open text questions were asked in the same survey as SUS questions to complete the users' image of the system. These questions were:

- What was good in the bike sharing system?
- What was bad in the bike sharing system?
- What would you change or add?
- If the system would work perfectly, how would you integrate it into your schedule?

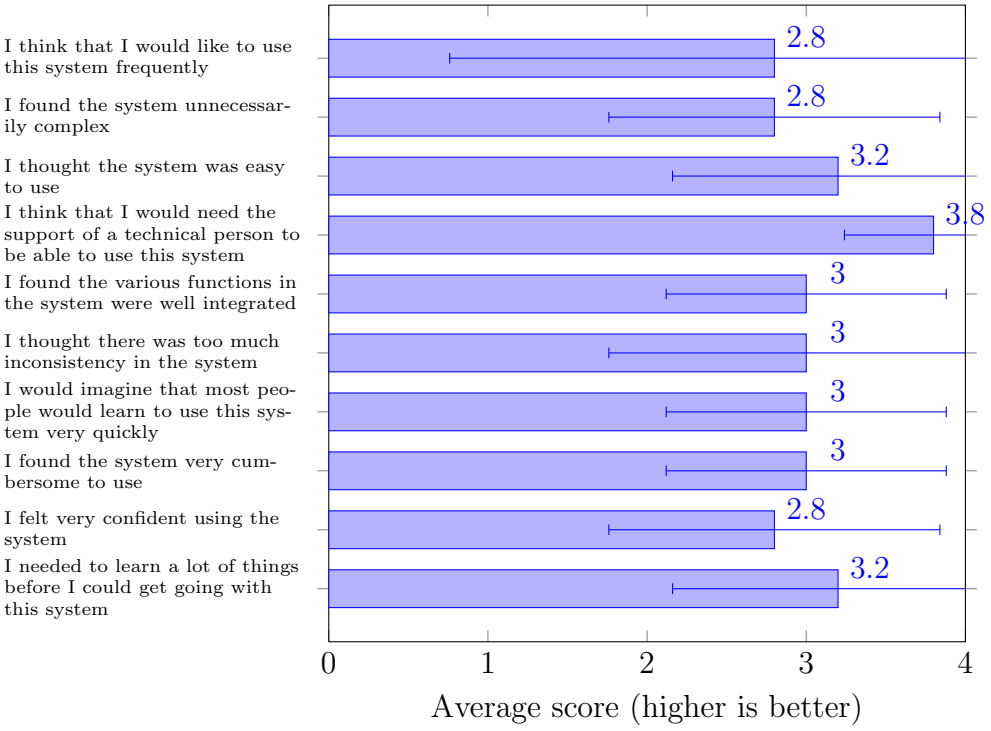

Figure 5.3: The mean SUS scores by question. Error margin bars represent a 95% confidence interval.

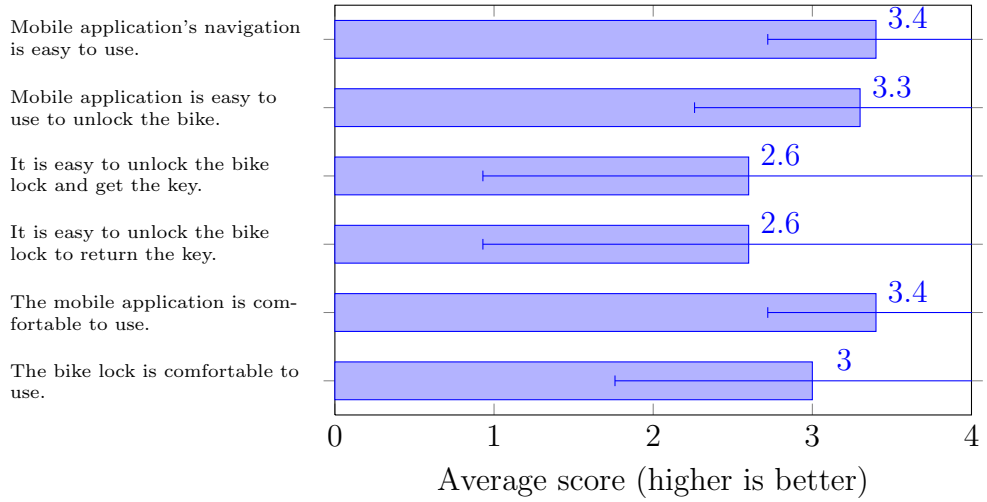

Figure 5.4: The custom survey's mean scores for system evaluation. Error margin bars represent a 95% confidence interval.

- For what sort of purpose or what activities would this work best?

In answer to the first questions all testers wrote that the system is easy to use. The downsides of the system were mostly targeted at the setup and the mobile application. One user wished for native applications for other platforms than iPhone and one reported the tracking on the iPhone was not satisfactory. The system setup issues found by one user were too few test bikes and no bike helmets.

When asked what could be added, most users commented on the mobile application, poor instructions and bike identification. Another user asked for reservation system and another for better map. Only one user asked for a smarter bike lock that could communicate wirelessly with the server and the smartphone.

Lastly, asked how frequently the users would use it if it was flawless, three users responded they would definitely use it, and two users would probably use it. The supplied use-cases were lunch trips and visits to clients. One user said the bike serves better than a taxi in the city center.

Overall the feedback gathered with these questions looks quite positive. The SUS closed questions give an overall high score. The answers to additional open questions are directed mostly at the smartphone application.

5.5.2 Custom survey

The second survey that was sent to the testers was created as a set questions that were though worth asking. The survey is larger and asks more questions. The main reasoning behind it is to collect project-specific feedback with a section dedicated only to wireless questions.

This survey contained a matrix question asking to evaluate the elements of the system (see Figure 5.4). This question shows more in detail which parts of the system were better than others. We can see again that while all the values pass over the middle, similarly to previous survey the error margin bars are very large. The bike lock is graded lower than the mobile application showing that this part of the system still could be improved usability-wise.

Open questions in custom survey

The open questions in the survey asked about the issues encountered while using the system and improvements to introduce. Some testers listed more things in this survey than in the previous survey.

Two users reported they had issues with entering the code into the lock. One user also complained during the testing period as well as in the survey about the lock not always flashing the led when the wakeup button was

pressed. This was fixed during the testing period. The same user also complained that the box does not close itself and the user is forced to click a button to make it close. This feature is currently under development.

In suggestions about improvements, one user asked for simplified pickup logic, which might mean also the need to press buttons on the lock. Another user noted that the system is more prone to errors such as key loss and there is no backup plan when that happens. This is an important remark which should be taken into account in further development.

Custom survey's wireless section

The last section of the custom survey asked only about users' smartphones and their wireless capabilities. As the answers were anonymous, it is hard to guess which answers were given by developers and which by non-technical team members. However we can see clearly that two testers did not know what new wireless technologies their phones had, along with another user they did not care about getting them in their next phone.

One tester had a BLE smartphone and only two testers expressed interest in getting a smartphone with BLE. It is possible though that more people answering the survey have a BLE-enabled phone but have never used it and do not know about it.

Two users were fine with using a keyfob if their phone could not be used with the smart lock. One tester would consider getting a phone that would support the wireless standard. Only one user would abandon the service altogether if their smartphone could not be used with the smart bike lock.

Overall if we were to base our smart lock design only on answers from the survey, we would have to forgo the use of short range wireless. However knowing that over half of the testers lack the knowledge about the wireless standards we can assume at least a few do have BLE in their smartphones but do not know about it.

Chapter 6

Discussion

The main topic of this chapter are the future improvements of the smart bike lock. The options are explained based on the progress of the work and the feedback.

6.1 Project improvements

The project is currently in the prototyping phase and the largest issue that should be discussed are the future directions and improvements that can be made to it. The paths for the project are: improving on the current design, starting a parallel project with a different architecture and switching to using wireless technologies.

6.1.1 Design improvements

The current prototype of the bike lock hardware is nearing its final form with the embedded design using the ATmega328p. While it lacks some features and only relies on numeric input, most of the functionalities are close to stable now. The list of significant changes to improve the design is small and low-effort now. These changes would be, for example, switching to use a 3.3V Arduino Pro Mini with suitable peripherals, a more robust RTC chip, a more solid numeric keypad or an enclosure with improved design.

The improved design of the enclosure is an important element in the list in large part because the company has no design specialist on the team. The design was done by an contractor but still is not perfect. Further design would again require support from a specialist who could work closely with the team on improvements and changes to the overall design.

The list of changes contains no major software changes as the largest

changes would most likely be too much of a time investment, like rewriting the libraries, or not possible in the current architecture, like logging to the microSD card turned out to be. Adding support for larger features like wireless could mean moving to a different architecture.

6.2 Alternative solutions

As the ATmega328p has proven to be too limited for expanding this project, alternative architectures should be considered. There are many alternative options but with initial research, they can be limited to two major possibilities.

6.2.1 Arduino Mega

One solution would be to simply use an ATmega model with more SRAM, pins and flash, like ATmega2560, which is used as the core of the Arduino Mega board. This would make it possible to maintain the similar architecture reusing the code and redesigning the current arrangement to suit the new main board. SparkFun shop offers a Pro Mini version of the Arduino Mega, where improvements were made similar to those in the Arduino Pro Mini compared to the original Arduino Uno. There are however some changes that would have to be taken into account, like different pin connectors, a larger size and a higher price.

6.2.2 ARM chips

While working on the project, we have consulted various embedded programmers like Kliment Yanev and Teemu Hakala. Their advice was to have a look at the ARM architecture and use it in another prototype. ARM is an architecture that offers faster chips with more RAM and flash memory as well as additional features. The ARM architecture offers chips more powerful than this project requires. However this means that it would be possible to implement wireless communication as well as better security for the communication protocols, logging and all the existing features.

The downside to using the ARM chips is a less accessible user community and seemingly worse access to training and materials in comparison to Arduino. This was one of the reasons ARM was not the architecture first selected for this project. The second reason was that at the beginning, the ARM chips seemed too overpowered for such a small project. However this

turned out to not be true as the project has reached the hardware limits of ATmega328p more than once.

For beginners starting with ARM, there are ready-made boards such as STM32 Nucleo. However it is not optimized to size or functionality for this particular project, so a programming prototype could be made partially with the board, but eventually the project would need a custom board design. Moving to the ARM architecture would also mean completely new code and assembly. It would be a completely separate project where only the hands-on experience from the ATmega iterations would be useful. The advantages of custom design would however include the possibility to add all the electrical improvements found while developing the Arduino iterations.

6.3 Other hardware issues

Aside from alternative architectures, there are other interesting improvements and modifications that could be researched. Following are the most interesting issues, such as power generation, wireless and mass production.

Alternative energy sources

The project has encountered many issues with powering the hardware and it needs more improvements to be energy efficient. However there is another approach to consider, which are energy sources that could gather power for the hardware while it is in use. One of the possibilities would be to use a dynamo, like the Lock8 product. It would, however require strategic placement to give the lock access to the bike tire to use its movement.

Another possibility is to use solar power, like the SkyLock does. It would require that part of the bike lock box be covered in solar cells. This is challenging to do with the current setup relying on numeric buttons. It would be possible to implement it with a wireless lock, but with wireless the power draw would be significantly higher posing a question if the solar panels would be sufficient. Also, just like dynamo, it would be nearly useless while the bike is stored away in dark room.

Adding alternative energy sources to extend the battery life of the bike lock is an interesting idea. It would require a lot of work to develop, measure and determine a suitable solution.

Wireless solutions

While no implementation has been made in the wireless area, the possibility has been considered the whole time the project was in development. This

is partially because wireless is seen as the most user-friendly technology in this context compared to numerical input. Another reason is that in the oncoming smart lock market all solutions use a short-range wireless.

The user feedback given in two surveys has shown that users were interested in using short-range wireless to open the lock. No user was against it and most answers showed more interest in BLE than in NFC.

Size of the numeric pad

The current prototypes are using a large numeric pad, which includes all numbers from 0 to 9 and additional two characters. The size could be however decreased to fewer digits to shrink the size of the enclosure. Fewer numbers to use in the numeric code may require increasing the length of the code and a modification to the algorithm that generates it. However the benefit would be drastic decrease in size of the enclosure. This modification is another intersting prospect to try out in the future iterations.

Location tracking with GPS

This project has already been in planning and development for a year. At the beginning of the project, the search for a combined GPS and GSM modules turned up only one solution that was bulky, expensive and packed with features that were not needed for this project, such as a loud-speaker and buttons for a phone.

Since then new and more suitable products have become available and they could be used in the future to develop alternative solutions for this project. The existence of a complete module for GPS and GSM functionality will considerably expedite the prototyping.

Large-scale production

The current prototypes are all based on ready-made boards which have been carefully designed and tested. They still require modifications to be of use in this project. However if an opportunity arose to manufacture this product in large quantities, the design of Arduino would have both redundant and missing features. Additionally, buying ready-made boards to incorporate into a custom board would be inefficient and expensive. The design would most likely have to be custom, and therein would lie possibility to move to ARM microcontrollers or another alternative.

6.4 Future use case: Campus bikes

During the development of the project another, non-corporate use case has appeared, namely bike sharing on the Otaniemi campus of Aalto University in Espoo. There are currently bikes for rent called Campus Bikes[20], but they are only for staff members and borrowing requires asking the building personnel for assistance. The bikes are branded with Aalto logo and placed in a number of larger buildings to wait for users.

Due to the shape of the campus, which is made of scattered buildings, an automated bike service for not only staff but also students could be beneficial. It is possible that also a classic, 3rd generation station-based bike sharing system could be implemented there. This would however require a lot of planning and work to build the official bike stations. A smart bike lock based solution would be free of that issue and could be implemented quickly and with existing bike racks on the campus.

Although we have not analyzed the campus use case deeper, we expect it to be slightly different from company bikes. First it would be on a much smaller area, so localization would have to be quite accurate. Second, if not only staff but also students were allowed to use the bikes, it would have a very high turnover. It is hard to predict how the use case would advance as it was noticed later in the project and prioritized less.

Chapter 7

Conclusions

This chapter presents the project work and lessons learned from it in distilled form. These conclusions are intended to be a guiding light in the future, related projects.

7.1 Lessons learned

This project was started with the goal of creating a complete solution which enables bike sharing for. It has been scoped to be a manageable project using as many ready elements as possible, in order to iterate fast over stable and usable versions. At the current moment, the prototype is close to being completed in the originally planned scope. The bike lock prototypes have been used internally and user testing shows that the usability is acceptable, but also that it should be improved in certain ways.

Still, the development has reached the architectural limits of the Arduino platform, that has been used in these prototypes. This means that adding more functionality to the current prototype would likely be challenging. As a consequence, other alternatives should be considered if features such as wireless communication are to be added to the solution.

The main lesson learned from this thesis project is related to the development in a constrained environment, an embedded platform in this case. We have learned valuable lessons while trying to optimize the memory footprint, design and power consumption of the device. The knowledge from the latter part especially will be used in the design of new devices that use external elements such as servomechanisms. The challenges related with power consumption have taught us how to design for improved battery life which is quite important in these kind of projects.

Power efficiency

The power efficiency of the project has been a considerable problem. The causes were discovered only after meticulous power measurements. Overcoming that limitation is a major step towards turning the design into a functional product. In this project, it was necessary to solder a custom-made board to add more power control. This exemplifies the limitation of relying on ready elements which are not equipped with such features.

Some devices come with manuals that have an estimated power consumption given by the manufacturer, like the ATmega chips. Other devices are provided without this information, like the servos used in the project. Overall, it is difficult to predict the power usage of the complete system before building a prototype first.

Still, creating a design that is highly power efficient would require strict power control for all elements connected to the board like the servomechanisms. The power control would most likely require a custom-design board that would do the power control for the main board. Such functionality could also be incorporated into a custom main board if the prototype of this project is to be manufactured.

Voltage

From the beginning of the project we used the default 5V voltage of the Arduino Uno. We continued with it even when we moved to the professional series of boards, which support the lower voltage of 3.3V. In retrospect, we should have moved to the lower voltage. It would have given us many advantages, such as lower power consumption and the possibility of connecting 3.7V lithium ion batteries directly, without voltage converters.

We noticed how problematic the higher voltage is after buying parts that required 5V. This prevented in practice the transition to the lower voltage. Moreover, we originally thought 5V parts were cheaper or that it would be difficult to obtain low voltage alternatives. Later, with more research, we found that is not the case. While some parts would indeed be more expensive, supplying the system with a steady 5V would be also expensive.

Product development and user feedback

From working in this project, we learned also that users value the external elements of the system, such as the design of the box and presentation of the application. A functional prototype was not well received because it did not have an elegant enclosure. The elegance of the device should have been given a higher priority from the beginning, to make a better impression.

Another lesson learned from the user feedback was how much they preferred a simplified system. The system is considered subpar if it takes too long to access the service and start using it. Any mechanical or technological improvement that could speed it up would be welcome. The speed of service should be a high priority for future system design.

In this project, that simplified design would mean combining a mechanical bike lock and the smart bike lock into one device. This has not been done due to lack of experience in designing mechanical locks. Instead, a simple box was built to house a key for a regular bike lock.

7.2 Future work

Although the project for the thesis is over, the work on the smart bike lock will continue. The previous chapter described the possible directions of development, starting from simple power improvements to different boards to finally a custom design. As working on the project has been rewarding, we plan to continue working on the development in the near future.

7.3 The future of bike sharing

During this project, we have found that this and other similar smart lock projects show the development of bike sharing is progressing. The products will enter the markets in the next few years and, possibly, change how bike rental and sharing is done. Technology is now advanced enough to create democratized bike sharing, which can lower the entry barrier to biking in more areas than bike has sharing already done. Moreover, it can be a base for creating small, local enterprises. While it is not a certainty that such a change will happen, it is still an interesting possibility to consider and worthy of support.

The success of bike sharing systems that use smart bike-mounted locks depends heavily on the users. If the users find the system easy to work with and feel incentivized to care, the bikes will be well secured and in good condition. In contract, if the users do not care and lock the bikes in bad, insecure ways or just leave them too far away from where other users could access them, the bikes will be easily lost and broken. This is one of the reasons why the smart bike lock system is suitable for small sharing programs, where users share a connection, by belonging to the same company, for example.

Bibliography

[1] Adafruit Industries. Adafruit. http://www.adafruit.com/, 2015. Accessed 15 Apr 2015.

[2] af Heurlin, E. Series ultracap balancer. https://github.com/rambo/ultracap_balancer, 2015. Accessed 10 May 2015.

[3] Akar, G., and Clifton, K. J. Influence of individual perceptions and bicycle infrastructure on decision to bike. *Transportation Research Record: Journal of the Transportation Research Board 2140*, 1 (2009), pp 165–172.

[4] Alli, G., Baresi, L., Bianchessi, A., Cugola, G., Margara, A., Morzenti, A., Ongini, C., Panigati, E., Rossi, M., Rotondi, S., Savaresi, S., Schreiber, F., Sivieri, A., Tanca, L., and Depoli, E. Green Move: Towards next generation sustainable smartphone-based vehicle sharing. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2012* (Oct 2012), pp. 1–5.

[5] Arduino LLC. Arduino. http://arduino.cc, 2015. Accessed 15 Apr 2015.

[6] Atmel Corporation. *Atmel 8-bit microcontroller with 4/8/16/32KBytes in-system programmable flash datasheet*, 2014.

[7] Berezdivin, R., Breinig, R., and Topp, R. Next-generation wireless communications concepts and technologies. *Communications Magazine, IEEE 40*, 3 (2002), pp 108–116.

[8] Brandon, J. Review: Fuz Designs Noke. http://www.wired.com/2015/02/review-fuz-designs-noke/. Accessed 17 Mar 2015.

[9] Brooke, J. SUS-A quick and dirty usability scale. *Usability evaluation in industry 189*, 194 (1996), pp 4–7.

[10] Buck, D., Buehler, R., Happ, P., Rawls, B., Chung, P., and Borecki, N. Are bikeshare users different from regular cyclists? *Transportation Research Record: Journal of the Transportation Research Board 2387*, 1 (2013), pp 112–119.

[11] Cavill, N., and Davis, A. Cycling and health. *What's the evidence* (2007).

[12] DeMaio, P. Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation 12*, 4 (2009), pp 41–56.

[13] DeMaio, P. J. Smart bikes: Public transportation for the 21st century. *Transportation Quarterly 57*, 1 (2003), pp 9–11.

[14] Dicola, T. Low Power WiFi Datalogger — Adafruit Learning System. `https://learn.adafruit.com/low-power-wifi-datalogging/`, April 2015. Accessed 19 Apr 2015.

[15] Firnkorn, J., and Müller, M. What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm. *Ecological Economics 70*, 8 (2011), pp 1519–1528.

[16] Free2Ride. The Lock Box - built for YOU by Free2Ride. `https://www.kickstarter.com/projects/free2ride/the-lock-box/`, 2014. Accessed 17 Apr 2015.

[17] FRENZEL, L. E. NFC DON'T LEAVE HOME WITHOUT IT. *Electronic Design 60*, 12 (2012), pp 30–37.

[18] FUZ Designs. Noke U-lock World's Smartest U Lock. `https://www.kickstarter.com/projects/fuzdesigns/noke-u-lock-worlds-smartest-u-lock`, 2015. Accessed 10 May 2015.

[19] Goasduff, L., and Rivera, J. Gartner Says Smartphone Sales Surpassed One Billion Units in 2014. `http://www.gartner.com/newsroom/id/2996817`, Mar 2015. Accessed 2 May 2015.

[20] Häkkinen, H., and Löyttyniemi, M. Sustainable transportation - Campus services - Aalto Inside. `https://inside.aalto.fi/pages/viewpage.action?title=Sustainable+transportation`. Accessed 13 May 2015.

[21] Jamieson, P. Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? *Proc. FECS* (2010), pp 289–294.

[22] Knight, P. narcoleptic - Sleep library for Arduino. `https://code.google.com/p/narcoleptic/`, 2010. Accessed 10 May 2015.

[23] Lella, A. comScore Reports January 2015 U.S. Smartphone Subscriber Market Share. `http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-January-2015-US-Smartphone-Subscriber-Market-Share`, Mar 2015. Accessed 2 May 2015.

[24] LOCK8. LOCK8 - the World's First Smart Bike Lock. `https://www.kickstarter.com/projects/lock8/lock8-the-worlds-first-smart-bike-lock`, Dec 2013. Accessed 9 Mar 2015.

[25] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and Ranen, O. Hotp: An hmac-based one-time password algorithm. *The Internet Society, Network Working Group. RFC4226* (2005).

[26] M'Raihi, D., Machani, S., Pei, M., and Rydell, J. Totp: Time-based one-time password algorithm. *Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 6238* (2011).

[27] Margolis, M. *Arduino cookbook*. "O'Reilly Media, Inc.", 2011.

[28] Martin, C. iPhone 6 NFC chip is restricted to ApplePay but may open to developers soon. `http://www.pcadvisor.co.uk/news/apple/3572112/iphone-6-nfc-chip-is-restricted-applepay/`, Sep 2014. Accessed 08 May 2015.

[29] Mesh Motion Inc. BitLock: Turning your smart phone into your bike key. `https://www.kickstarter.com/projects/126495570/bitlock-turning-your-smart-phone-into-your-bike-ke`, Nov 2013. Accessed 9 Mar 2015.

[30] Midgley, P. The role of smart bike-sharing systems in urban mobility. *Journeys 2* (2009), pp 23–31.

[31] Noland, R. B., and Ishaque, M. M. Smart bicycles in an urban area: Evaluation of a pilot scheme in London. *Journal of Public Transportation 9*, 5 (2006), pp 71–95.

[32] Pololu Corporation. Pololu 5V Step-Up Voltage Regulator U1V11F5. `https://www.pololu.com/product/2562`, 2015. Accessed 10 May 2015.

[33] Rustie0125. Arduino low Power Project. `http://www.instructables.com/id/Arduino-low-Project-and-code/`, September 2014. Accessed 19 Apr 2015.

[34] Sauro, J., and Lewis, J. R. *Quantifying the user experience: Practical statistics for user research*. Elsevier, 2012.

[35] Seidle, N. Adventures in Low Power Land - SparkFun Electronics. `https://www.sparkfun.com/tutorials/309`, August 2011. Accessed 19 Apr 2015.

[36] Seik, F. T. Vehicle ownership restraints and car sharing in Singapore. *Habitat International 24*, 1 (2000), pp 75–90.

[37] Siekkinen, M., Hiienkari, M., Nurminen, J. K., and Nieminen, J. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE* (2012), IEEE, pp. 232–237.

[38] Urdiain, L. O., Romero, C. P., Doggen, J., Dams, T., and Van Houtven, P. Wireless Sensor Network Protocol for Smart Parking Application Experimental Study on the Arduino Platform. In *AMBIENT 2012, The Second International Conference on Ambient Computing, Applications, Services and Technologies* (2012), pp. 45–48.

[39] Velo Labs Inc. SKYLOCK. `http://skylock.cc/`, 2015. Accessed 17 Mar 2015.

[40] Vogel, M., Hamon, R., Lozenguez, G., Merchez, L., Abry, P., Barnier, J., Borgnat, P., Flandrin, P., Mallon, I., and Robardet, C. From bicycle sharing system movements to users: a typology of Véloflv cyclists in Lyon based on large-scale behavioural dataset. *Journal of Transport Geography 41* (2014), pp 280–291.

[41] Vorkoetter, S. Choosing and Using Nickel-Metal-Hydride (NiMH) Rechargeable Batteries. `http://www.stefanv.com/electronics/using_nimh.html`, March 2008. Accessed 19 Apr 2015.

[42] Wikimedia Commons. A Vélib station with its distinctive grey bicycles. `http://upload.wikimedia.org/wikipedia/commons/f/f7/Velibvelo1.jpg`, 2007. Accessed 12 May 2015.

[43] Wu, G., Talwar, S., Johnsson, K., Himayat, N., and Johnson, K. D. M2M: From mobile to embedded internet. *Communications Magazine, IEEE 49*, 4 (2011), pp 36–43.

[44] Wu, X.-g., and Zhang, R.-h. The Popularization and Application of Bicycle Sharing System in Urban Transportation System. In *Proceedings of the 10th International Conference of Chinese Transportation Professionals* (2010), pp. 2616–2628.

[45] YLE. Helsinki Suspending Free City Bike Programme. `http://yle.fi/uutiset/helsinki_suspending_free_city_bike_programme/1597732`, 2012. Accessed 10 May 2015.

[46] YLE. Helsinki haluaa kaupunkipyörät osaksi joukkoliikennettä. `http://yle.fi/uutiset/helsinki_haluaa_kaupunkipyorat_osaksi_joukkoliikennetta/7467249`, 2014. Accessed 10 May 2015.

Appendix A

Flowcharts

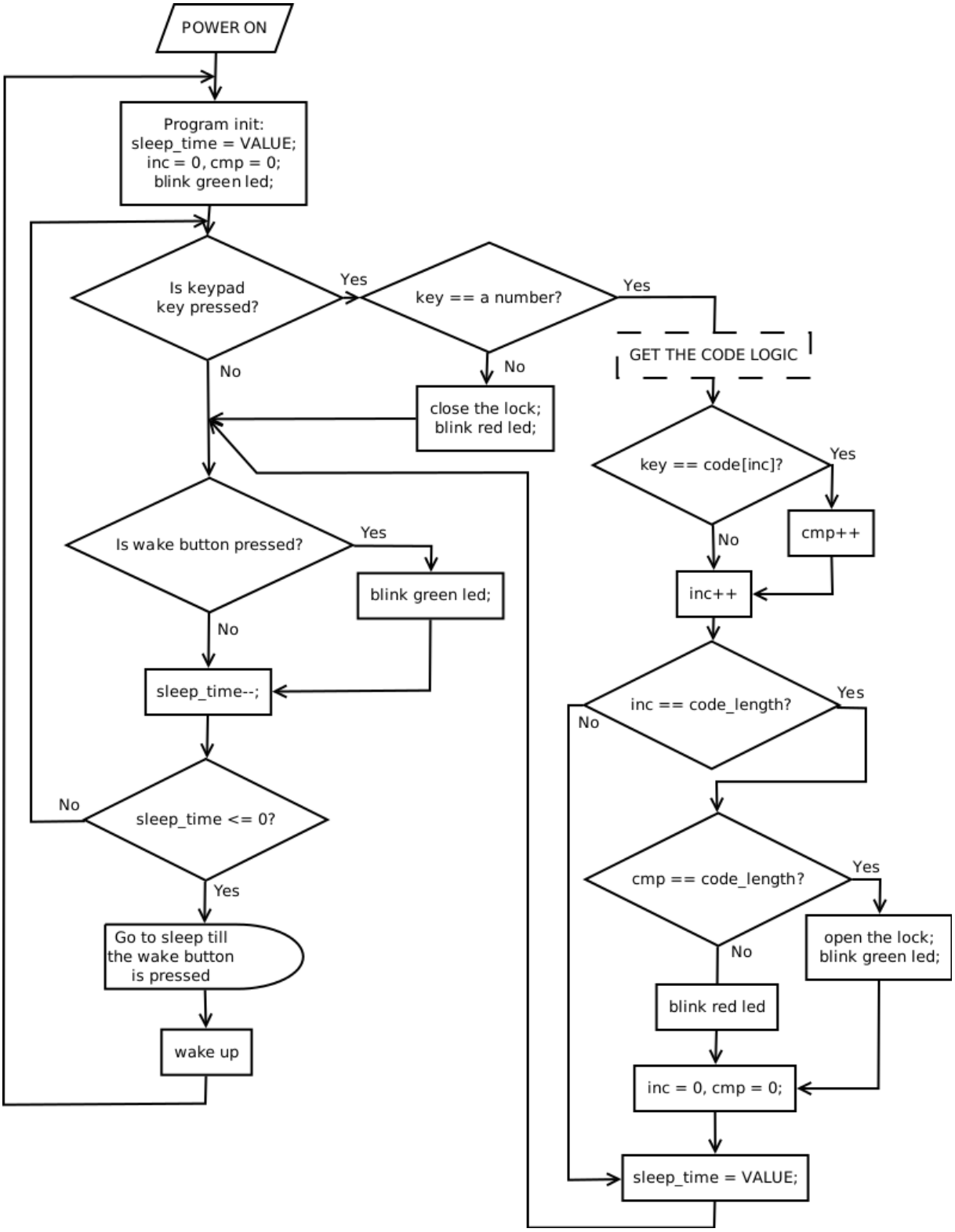

Figure A.1: Detailed logic the bike lock program.

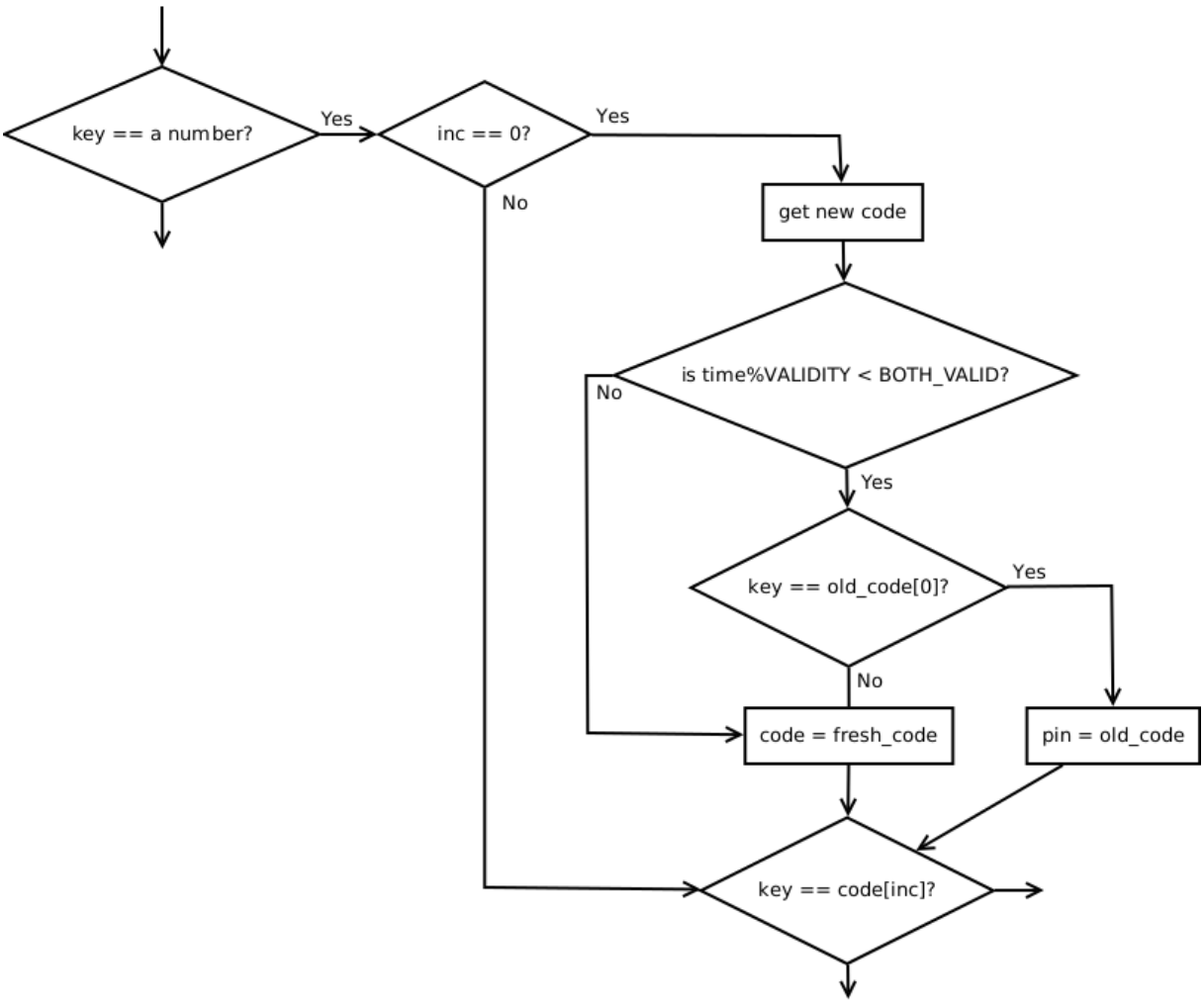

Figure A.2: Detailed logic for generating and validating the numeric code.