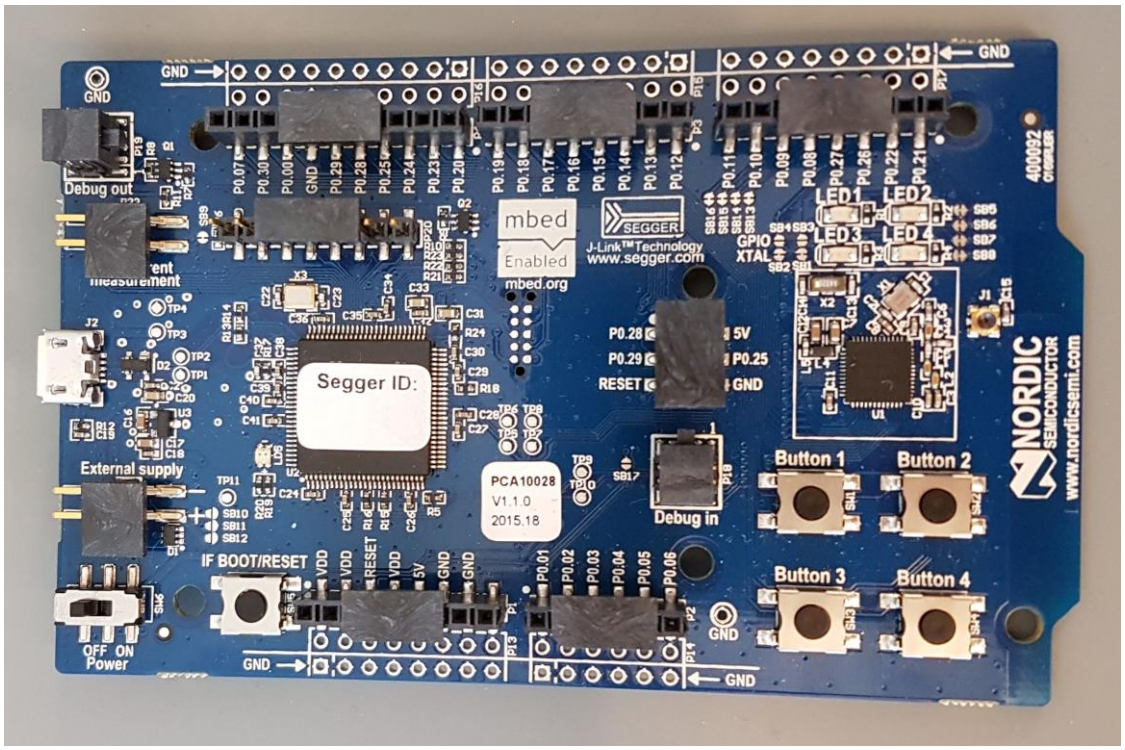
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Applying Multi-Role Communication for Bluetooth Smart Devices

An Evaluation of Using a Low Energy Wireless Network to Position a Mobile Device in Short Range Applications

Master's thesis in Computer Systems and Networks

FILIP BERTILSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Master's thesis 2016

Applying Multi-Role Communication for Bluetooth Smart Devices

An Evaluation of Using a Low Energy Wireless Network to Position a Mobile Device in Short Range Applications

FILIP BERTILSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2016

Applying Multi-Role Communication for Bluetooth Smart Devices
An Evaluation of Using a Low Energy Wireless Network to Position a Mobile Device in Short Range Applications
Filip Bertilsson

Supervisor: Sally McKee, Department of Computer Science and Engineering
Company Supervisors at Consat: Niklas Sundin and Måns Breitholtz
Examiner: Lars Svensson, Department of Computer Science and Engineering

Master's Thesis 2016
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The nRF51 Development Kit, one of the devices used to create the Bluetooth Smart network presented in this thesis.

Printed by Department of Computer Science and Engineering
Gothenburg, Sweden 2016

Applying Multi-Role Communication for Bluetooth Smart Devices
An Evaluation of Using a Low Energy Wireless Network to Position a Mobile Device in Short Range Applications

Filip Bertilsson
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Many new cars can communicate with a smartphone and use it as a music player, a navigation unit, or use it to let the driver make a telephone call. Consat Engineering AB has pushed such communication technology even further by developing a prototype system in which the car uses Bluetooth Smart/Bluetooth Low Energy (BLE) connections to track the position of the smartphone in relationship to the car itself. The prototype allows for the phone to automatically unlock the car when close to it and start it when inside.

In this thesis I improve on that prototype by developing and evaluating the use of a scalable, energy-efficient, wireless network of multi-role BLE devices for positioning. The results show that the system works well with a distance of 10 meters between the nodes, and that the signal strength can reach distances of up to 30 meters. The resulting system is not as scalable as was first intended but still more so than the Consat prototype. However, by building the system with newer hardware with more RAM, the scalability problem can be solved.

The results of this project demonstrates that a network of multi-role BLE devices can be used as a scalable, energy-efficient solution for positioning of a device through continuous flow of data.

Keywords: Bluetooth Smart, communication, positioning, energy-efficient, multi-role

Acknowledgements

I would like to thank Consat Engineering AB for facilitating the project. I especially would like to thank my supervisors Måns Breitholtz and Niklas Sundin at Consat for all their support and guidance.

I would also like to thank my supervisor Sally McKee, at Chalmers University of Technology, for her insight and expertise in the writing process.

Lastly I would also like to thank my wonderful fiancée for all her support and helpful remarks.

This thesis would not have been possible without the help from these individuals.

Filip Bertilsson, Gothenburg, June 2016

Contents

List of Figures

1

Introduction

Many new cars can communicate with a smartphone and use it as a music player, as a navigation unit, or use it to let the driver make a telephone call. Consat Engineering AB are pushing such technology even further by enabling the car to know where the smartphone is in relation to the car, which opens up for more functionality [2]. Consat have built a prototype system in which the car uses Bluetooth Smart/Bluetooth Low Energy (BLE) connections to track the position of the smartphone in relationship to the car itself. The current purpose of the system is to use a smartphone as a car key. This allows people to share their car with family members more easily, and it can also be used by car rental companies to let a customer book a car and download the key directly to a phone. The current system works in all vehicles that are roughly the same size as a car.

In this thesis, I improve on the Consat prototype by making the system more scalable, easier to install, and more energy-efficient. My system uses a network of wireless BLE devices [8], allowing the new prototype to work in larger vehicles or other areas of use, and it is easy to install since no wires are needed. I also evaluate and test the resulting prototype in order to determine whether this kind of system is a feasible solution.

Unlike positioning systems such as GPS, Bluetooth is sufficiently accurate to be used in indoor positioning systems [20] [4]. As Wang et al. [20] and Yih-Guang et al. [7] show, the distance determination error can be narrowed to approximately 0.5 meters or less, making Bluetooth positioning a feasible technology to use in a short-range positioning system. The prototype on which my project is based uses BLE to get a position in relation to a vehicle. The system is embedded into the vehicle and is constantly scanning for a paired device. The system therefore has to be energy-efficient to not drain the battery if the vehicle is not used for some time.

While my system is more scalable, many improvements remain that fall outside the scope of this thesis. Security is an important aspect of most systems, but in this thesis I do not implement security features because of the limited time frame. Since the BLE nodes are advertising in all directions, the resolution of the positioning could be improved. To accomplish such improvement, the Bluetooth nodes antennas could be replaced with directional antennas, making it easier to determine of which side of the car the smartphone is. However, doing so is beyond the scope of this project. When developing an energy-efficient system like the one I propose in this thesis, it

is important to think about possible code optimizations. For instance, there are many ways to improve the code executed on the BLE devices such as using better register management, using loop unrolling, and using the low-power features of the processor, e.g., sleep modes [18] [23]. However, because of time constraints I will not do so in this project and so I leave it open as a possibility for future work.

2

Background

In order to understand the technical details of the following chapters, I first explain the basics of the technologies I use in this project, beginning with Bluetooth Low Energy. I also explain the prototype system by Consat on which this project is based to increase the understanding of what I try to accomplish.

2.1 Bluetooth Low Energy/Bluetooth Smart

Bluetooth Low Energy (BLE), also known as Bluetooth Smart, is a short range communication technology introduced in Bluetooth 4.0 [6]. While ordinary Bluetooth is also used for short range wireless communication, BLE is aimed at ultra low power devices. BLE allows a device to communicate wirelessly for several months or years on a single coin cell battery. To achieve such low power, BLE uses its own protocol stack instead of the ordinary Bluetooth stack [5].

A BLE device that runs functionality (other than just *advertising*, as defined below) has one or more *services*. A service is a set of data that describes the functionality of the device. There are primary and secondary services, where a primary service is usually the main functionality of the device and a secondary service is auxiliary functionality used by at least one primary service. A service can have one or more *characteristics* that are used for storing data. The characteristic has a set of properties, a value, and information about the representation of the value. Services and characteristics have their own Universal Unique Identifier (UUID), so devices know what services and characteristics other devices have. The UUID also enables another device to read from and write to the right service/characteristic.

When a device broadcasts data, it is called advertising. An advertisement packet can consist of many different fields, such as what type of Bluetooth the device is running, if the device is connectible or not, and the UUIDs of the services run on the device. If a device is connectible, it includes that information in the advertisement packets. Another device can then send a Connection Request message in order to establish a connection. At the Link Layer, master and slave roles are given to the devices, where the advertiser is the slave. As a means to conserve power, the slaves are usually in sleep mode and wake up at an interval given by the master to listen

for packets sent by the master. A master can be connected to many slaves.

The protocol stack of BLE is divided into two parts, the host and the controller. The upper layer protocols, belonging to the host, are the Generic Access Profile (GAP), the Generic Attribute Profile (GATT), the Attribute Protocol (ATT), the Security Manager Protocol (SMP), and the Logical Link Control and Adaptation Protocol (L2CAP) [3]. The controller consists of the Link Layer and the Physical Layer. Communication between the host and the controller is done through the Host Controller Interface (HCI). See Figure 2.1 for a visual representation of the host part of the BLE protocol stack. The protocols are explained in a top down approach as follows:

- **GAP:** The GAP protocol controls advertising and connections and decides what role the device has. There are four roles in BLE: the broadcaster role, the observer role, the peripheral role, and the central role. A broadcaster device only broadcasts messages while an observer only receives messages from a broadcaster. The two most important roles to know in order to understand this project are the peripheral role and the central role. Central devices usually take care of the more performance intensive tasks while the peripherals are often devices performing simple tasks and transmitting data to a central device. For example, heart rate monitor watches often have the peripheral role and connect to a smart phone which acts as a central device. In order to connect, a central device is scanning for a peripheral device's advertisement.

- **GATT:** GATT is a framework that uses ATT to discover the services run on another device and to transfer characteristic data between the devices.

- **ATT:** The ATT is used to send data between devices. The data is represented by a data structure called an *attribute*, which is then used by the GATT to transfer the characteristic data between the devices.

- **SMP:** SMP is responsible for the security of a connection. It supports AES128-encryption and offers device authentication, device authorization, among other features.

- **L2CAP:** The L2CAP protocol is used to multiplex data on top of a Link Layer connection. The data that is multiplexed comes from ATT, SMP and Link Layer signaling. L2CAP in BLE is derived from the L2CAP protocol used in standard Bluetooth but it is tweaked for low power consumption, and thus works in a best-effort fashion without any flow control or retransmissions.

Finally, in order to understand the positioning technology, the concept of Received Signal Strength Indicator (RSSI) is important. As the name suggests, RSSI is an indication of the signal strength from a device. In BLE, the RSSI value is between -127dBm to 20dBm. RSSI can be used to estimate the distance to a device or a position in relation to some devices.

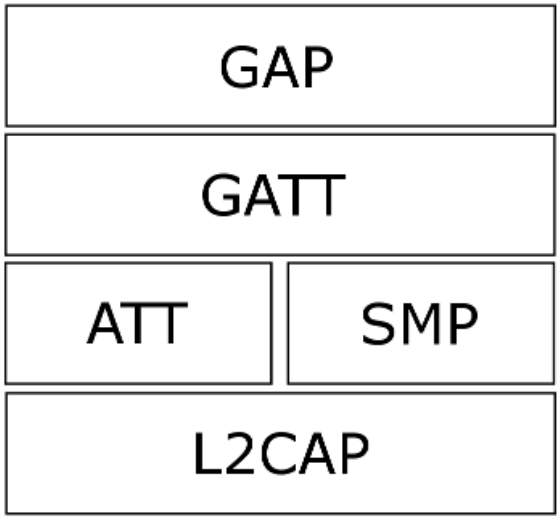


Figure 2.1: The host part of the BLE stack from a top-down approach.

2.2 The Consat Prototype System

Consat Engineering AB have developed a system that enables a smartphone to act as a car key. The system consists of a central computer, five BLE nodes, and a smartphone. In order for the reader to understand my new prototype, I describe the existing Consat prototype system briefly in this section.

2.2.1 The Nodes

There are five BLE nodes placed inside the car, one in each door and one in the center of the car. The one in the center is the main node. This node is always active and used to detect an approaching smartphone. If the phone has the proper identification needed to unlock the car, the main node will tell the central computer to wake the other nodes. When the nodes are awake, they will start tracking the RSSI of the phone signal and send it to the central computer for positioning.

2.2.2 The Central Computer

The central computer is located inside the vehicle as a standalone unit. It is responsible for downloading information to allow the correct phone to act as the key. Without the right verification and encryption keys, the phone will not be able to unlock or start the car. The central computer is also responsible for positioning the phone based on the RSSI values seen by the five BLE nodes. The system is accurate enough to decide by which door the phone is located and if the phone is inside the car. If the phone is close to a door, that door will unlock when a person grips the door handle. If the phone is inside the vehicle, the engine can be started.

2.2.3 The Smartphone

In order for a smartphone to act as a key, it needs the correct verification as proof to the system that it is allowed to unlock and start the car. When the phone approaches the car, it scans for a specific service that runs on the central computer. If the phone has the proper identification, the application that runs on the phone can connect to the system in the car and the phone can act as a key.

3

Goals and Challenges

My main purpose with this project is to improve on Consat's prototype system by creating a scalable and energy-efficient wireless prototype using multi-role BLE devices. However, the focus does not lie in attaining a perfect prototype, but rather in the evaluation of whether this is a feasible solution for this kind of system. To achieve this, I have set the following goals:

- To arrange the BLE nodes in a network to allow scalability and reliability. Ideally this would be in a mesh-network.
- To test the hardware and resulting system in order to find out how effective BLE is for this kind of system and to see how scalable the system is.

These goals bring several challenges to the project. Since a BLE device usually is either a peripheral device or a central device, there will be a challenge to create a multi-role device that is both a peripheral and a central simultaneously, and to make a number of these devices communicate in a network. Furthermore, factors such as data transfer times and signal strength on different distances can limit the functionality of the system. How these goals and challenges have evolved and formed into the final prototype will be shown in the coming chapters.

4

Methods

In this chapter, I describe the hardware I use, the specific software needed to program the hardware, the Consat prototype system, and lastly how I conducted tests to find out how the signal strength changes over distance and to find out transmission times.

4.1 Hardware

The hardware choices I make in this project are based on some basic constraints. The first is that the hardware must use BLE technology and it must be fully programmable. Furthermore, the nodes of the network must be small to enable easy installation. Lastly, the hardware must be economical for me to use in this project. There are several BLE chips on the market that can be used in this kind of application. For me, the choice was between Nordic Semiconductor's nRF51 chips [13], STMicroelectronics' SPBTLE-RF [19], and Silicon Labs' BT121 Bluetooth Smart Module [17]. I chose the nRF51 chips from Nordic Semiconductor since there are many small modules using those chips on the market. Furthermore, the documentation about the nRF51 chips is better and more extensive than those of the other choices.

The system consists of a central computer and five nodes where one acts as a main node. The central computer of the prototype is placed inside the vehicle. The central computer is responsible for the positioning calculations and for downloading key information, allowing connections from only the correct smartphone. For this project, I'm simulating that central computer on a Raspberry Pi 2 along with a USB Bluetooth adapter [16]. Although the Raspberry Pi can simulate the prototype system, I have chosen to run a simpler program, thus ignoring security factors. This is a choice I made in order to be able to program the nodes to establish connections without having to focus on the much more complicated encryption programming of the hardware.

The second component, the main node, is responsible for finding a nearby phone, establishing a connection to it, and establishing a connection to a positioning node. The main node has to be power efficient and able to play the roles as a central device and a peripheral device at the same time. Based on this and the reasons stated in the

beginning of this chapter, I chose the nRF51 Development Kit v1.1.0 from Nordic Semiconductor as the main node [11]. The nRF51 DK, also known as PCA10028, is a development board with an on-board SEGGER J-Link OB Debugger that can be used to program the positioning nodes. The board has a fully programmable nRF51422 chip with a 32-bit ARM® Cortex™ M0 CPU, 32kB RAM and 256kB embedded flash memory [1] [12]. Programs developed for the kit can be run on both nRF51422 and nRF51822 chips. Figure 4.1 shows the nRF51 DK.

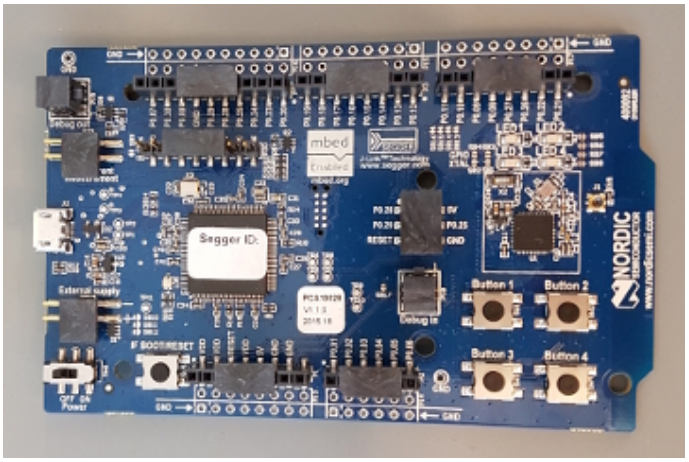


Figure 4.1: The Development Kit v1.1.0 from Nordic Semiconductor.

The third component is the positioning nodes. The need for small size, and compatibility with the nRF51 DK, made me choose nRF51822 Beacon kits from Nordic Semiconductor for the positioning nodes [14]. The nRF51822 Beacon kit is the size of a coin-cell battery and is powered by either a CR1632 coin-cell battery or from an external power source. The beacon kit has an nRF51822 chip with the same CPU as an nRF51422 but with 16kB RAM and 128kB embedded flash memory [13]. Figure 4.2 shows the nRF51822 Beacon Kit Rev. 1.2.0. Later in the project, I realized that the beacon kit did not have enough RAM to actually handle the needed number of connections. Since I already had the finished code for the nodes at this time I needed hardware with an nRF51 chip to continue the project. Lack of time and the fact that Consat had them available made me choose the BLE400 along with three Core51822 from Waveshare [21] [22]. The Core51822 is based on the same nRF51822 as the beacon kit but with 32kB RAM instead of 16kB RAM which enables the same functionality as the nRF51 DK. The BLE400 is a motherboard used for programming the Core51822 devices. Figure 4.3 shows the Core51822 module from Waveshare.

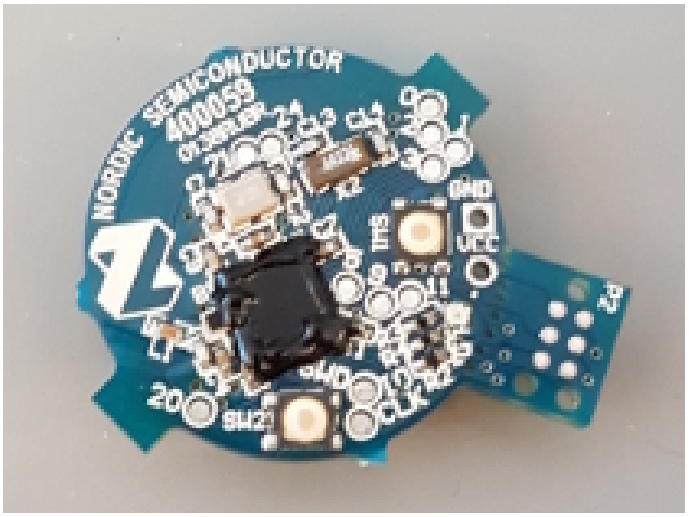


Figure 4.2: The nRF51822 Beacon Kit Rev. 1.2.0 from Nordic Semiconductor.

Figure 4.3: The Core51822 from Waveshare.

4.2 Software

In order to program the nRF51 chips, two important components are needed: the softdevice and the Software Development Kit (SDK). In this section I provide basic information about these components.

The nRF51 chips have two regions of flash memory, one for the application and one for the softdevice. The softdevice is a pre-compiled software stack for Bluetooth Low Energy. In this project, I have chosen the S130 softdevice from Nordic Semiconductor [9]. S130 enables the device to act as both central and peripheral

simultaneously, which is needed for the nodes to connect to several other nodes and to a phone. The softdevice can handle at most three central connections and one peripheral at the same time. For the positioning nodes to communicate in a mesh network and reading the RSSI of the smartphone at the same time, three central connections and one peripheral connection are needed. To program these multi-role devices, there is a module called the peer manager from the SDK which needs an S13x softdevice to work. The S130 softdevice reserves 10kB of RAM for its own purpose.

The SDK I have used is developed by Nordic Semiconductor and is of version 11 [10] and it is written i the C programming language. The SDK provides functions for advertising, scanning, connecting, sending data, encrypting transmissions and more. The SDK is very low level and provides many options for the programmer to set up the BLE functionality, but the learning curve of the SDK is steep.

4.3 Testing

To see how well the hardware performs, I have tested the change in signal strength over distance as well as the time it takes for two bytes of data to travel through the system when the nodes are connected in a line. The reason for these tests of the system can be read in 6.1.1. In order to minimize surrounding interference, I conducted the testing in the countryside.

4.3.1 Signal Strength

I tested the signal strength of the nRF51 Development Kit and a Core51822 every meter up to a total distance of 30 meters. The signal strength was measured with a smartphone using Nordic Semiconductor's nRF Master Control Panel application. All the tests lasted for 15 seconds. The results from the signal strength tests are presented as mean values, median values, and standard deviations and can be seen in 6.2.

4.3.2 Data Transfer Times

I tested the time it takes for two bytes of data to travel through a sequence of nodes by using two Raspberry Pi computers. One Raspberry Pi runs a central application and the other runs a BLE service. The one running the service (the peripheral Raspberry Pi) waits for a connection to one of the BLE nodes before starting a timer and sending data. The data in this test consists of messages sent every 300ms. When the BLE node gets the data it sends it forward to the next node and so on until the last node sends the data to the Raspberry Pi running the central application. The Raspberry Pi computers are linked together with a cable between

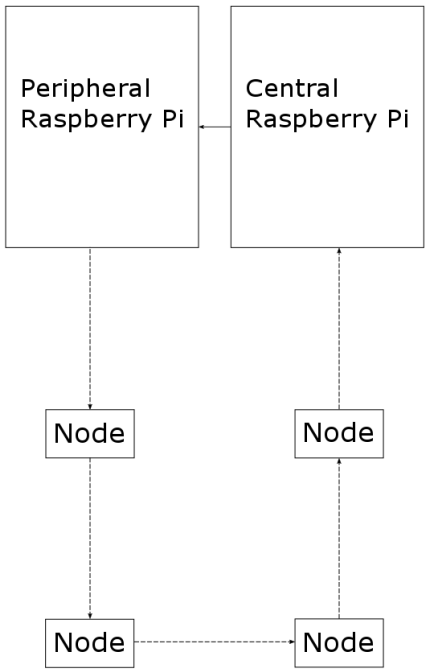


Figure 4.4: The test setup when measuring the data transfer times. Data is sent from the peripheral Raspberry Pi through the four BLE nodes to the central Raspberry Pi. The dotted lines are BLE connections and the solid line is a wired connection.

pins. When the central Raspberry Pi receives the data it sets its pin to high, which also sets the corresponding pin to high on the service Raspberry Pi. This transition triggers an interrupt on the service Raspberry Pi and the timer stops. Figure 4.4 shows the test setup with four nodes.

I ran this test with different distances between the BLE nodes and with a different number of nodes. I started with one node at a one-meter distance to both Raspberry Pi computers. I then did the test with two, three, and four nodes, all at a one-meter distance from the neighbour nodes. After that, I did the same testing with a three-meter distance between the nodes and finally with a 10-meter distance.

To get representative data, the service Raspberry Pi sends data 1000 times, where the messages consist of two bytes each. The first byte is a sequence number used to show which messages are received by the central Raspberry Pi. The second byte starts with zero and is incremented by each device it passes through, resulting in a number equal to the number of connected devices. Since the Core51822 nodes do not have any way to show that they are connected, the second byte of the message is used to verify that the message has passed through all devices. The library used to control the GPIO-pins of the Raspberry Pi computers may have introduced noticeable overhead in the testing. This is discussed in 7.2. The results of the tests can be seen in 6.2.

5

Software Implementation

In the previous chapter, I described the hardware and the SDK I used. In this chapter, I describe the implementation of the software I have written for the hardware, and some of the problems I had.

In the early stages of the project, I used a device manager module from the SDK. The device manager manages the connected peer devices by saving GATT configurations, security keys, and more. Later on, I made the transition to the peer manager module mentioned in the previous chapter. For me, the most interesting difference between the device manager and the peer manager is that the latter allows multi-role devices. When I still used the device manager, I considered making the devices act as multi-role devices by changing the role of each device when necessary. I soon realized this would be complicated, unreliable, and slow, which made me read more about the peer manager. Since the peer manager allows true multi-role devices, I knew roughly how to structure my applications using it. However, when running my application, nothing happened. I had to debug the entire code, read the SDK documentation, and restructure the code to try to find a solution. At this point in the project timeline, I used an alpha-version of the SDK which later showed to be the reason for my applications did not work as intended. When updating the SDK, the application started working at once and the long period of error searching and debugging was over.

Another limiting factor when implementing the applications was the RAM of the devices. The nRF51822 Beacon Kit did not have enough RAM to establish the necessary connections for the software to work. When I switched to use the Core51822 devices instead, the application worked smoothly and at least some of the connections needed was attainable.

In the end, I developed two central applications and a service for the nodes, two test applications to the Raspberry Pi computers in the tests, and a small phone service application that run on a Raspberry Pi instead of a phone. The software theoretically enables the BLE nodes to form a mesh-network, but because of RAM constraints, I only tested the nodes in a line network. In the next chapter, I describe the resulting system and its components.

6

Results

In this chapter, I present the resulting prototype system and how it works. I also present the data gathered from the signal strength testing as well as from the transfer time testing.

6.1 The Prototype System

The resulting prototype consists of the same components as the Consat prototype: the central computer, the main node, the other four nodes, and the smartphone application. Since my focus is on the nodes, the central computer and the phone application are small applications lacking real functionality that are run on Raspberry Pi computers. I have developed a relay service and two BLE central applications that run on each node. Figure 6.1 shows the resulting system layout with the nodes connected in a sequence. A more detailed view of the layout of a node can be seen in Figure 6.2, which shows how the central applications and the relay service are working on a node and how they connect with the previous node and the phone to the left, and to the next node on the right.

When a node starts, it will begin advertising and scanning simultaneously. If another node is detected by the scanning, or if the advertising is detected by another node, a connection will be established.

6.1.1 The BLE Central Applications

There are two BLE central applications on each node: one for handling the connection to the previous node and one for handling the connection to the phone. I call them the node central application and the phone central application. The central applications are also responsible for the handling of received data. A notification event is triggered on the node when there are data incoming from either a node or the phone. When this event is triggered, each central application will check whether the data belong to the application itself. If not, it will ignore the data. For example, if the phone sends data to a node, both the node central application and the phone central application will receive the data. The node central application determines

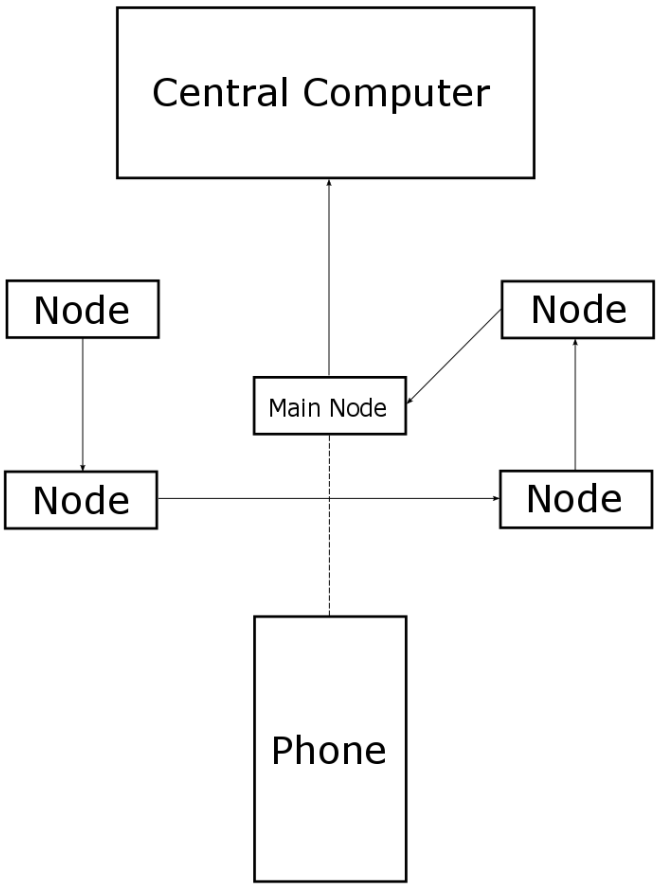


Figure 6.1: An overview of the system layout. The nodes are connected in a line network. Data is relayed from node to node. The phone can connect to one node and send data through that to the central computer.

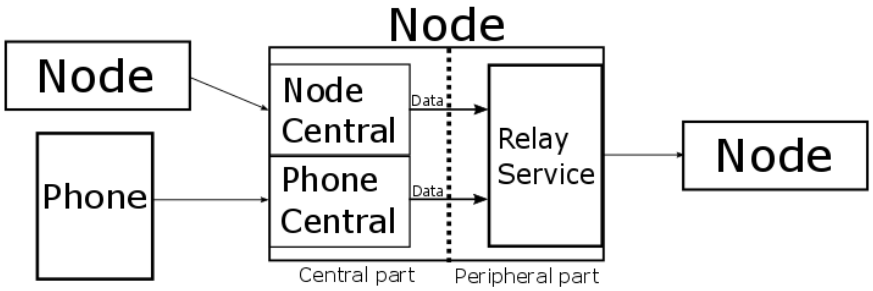


Figure 6.2: The service and the central applications run on a node. To the left are the previous node and the smartphone. The previous node connects to the node communication central application and the phone connects to the phone central application. The data is then passed to the relay service and sent to the next node.

that the data come from the phone by checking the connection handle and thus ignores the data. However, the phone central application will find that the connection handle is indeed associated with the phone and so it accepts the data. The last step for the phone central application is to pass the data to the relay service to forward it to the next node.

The node central application can, in theory, handle two connections at a time and the phone central application can handle one connection. This would make it possible to arrange the nodes in a tree structured network. Because of RAM issues I had problems connecting the second node however, making it possible to only connect one node on each end along with the phone. The result is a line network of nodes instead of a mesh or a tree structured network.

6.1.2 The Relay Service

Every node advertises the UUID of the relay service. The relay service is registered in the GATT enabling it to be visible to the central application of the next node. The service receives data from the central applications, i.e., data received from another node or a phone, and sends the data to the next node. As mentioned, a notification event is triggered on the node whenever another node or the phone send data to the node. When the event is triggered, the received data is forwarded to the relay service. The relay service then sends it forward to the next node. If the received data is the RSSI value of the phone, the relay service adds the node's ID to the data array before sending the data.

6.1.3 The Phone

I have not developed an application for the phone. Instead, I have simulated the service that should run on a phone on a Raspberry Pi. The resulting service does not, however, provide the same functionality as the real phone service does. All the Raspberry Pi service does is to advertise the phone service UUID, allow a connection from the node, and then send some random data to the node. Since the roles of the devices are inverted in my prototype compared to those of the Consat prototype, the phone is a peripheral instead of a central. This has caused problems since the phone can only connect as a peripheral to one central at a time.

6.1.4 The Central Computer

I simulated some basic functions of the central computer on a Raspberry Pi. Recall that in the Consat prototype, the central computer runs the positioning algorithms and is responsible for the security of the system. However, in my simulated version the only functionality I have implemented is a central application that connects to the main node and can receive data sent from the nodes.

6.2 Test Data

I did the tests as described in section 4.3. The results from the distance tests with both devices are presented in Figure 6.3. The blue line represents the signal strength of the Core51822 and the orange line represents the signal strength of the nRF51 DK. As can be seen, the Core51822 drops in signal strength at 15 meters and is between -90 dBm and -100 dBm at 30 meters, which is low but acceptable. This is mostly because of cheap components. The nRF51 DK is significantly better with a slightly higher signal strength at 30 meters distance.

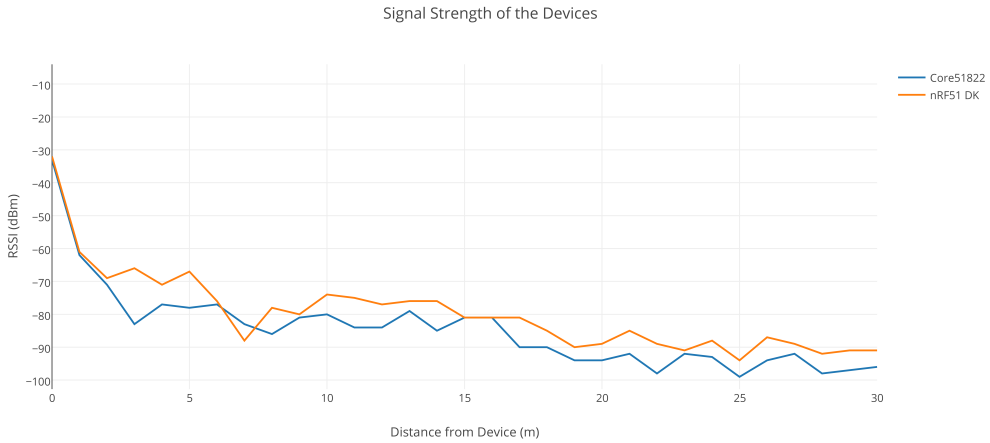


Figure 6.3: The signal strength of the devices, measured every meter to a maximum of 30 meters. The blue line represents the signal strength of the Core51822 and the orange line represents the signal strength of the nRF51 DK.

The mean, median and standard deviation of the transfer times with a different number of connected devices at a one-meter distance from each other are presented in Table 6.1. The corresponding results for the tests with distances of three meters and ten meters are presented in Table 6.2 and Table 6.3 respectively. The results differ more when using more devices than when increasing the distance with the same amount of devices. This was expected since the signal waves travel at the speed of light. The increase in transfer time when using more devices is because the extra overhead each device introduces. The results are further analyzed and discussed in section 7.2.

Table 6.1: The mean, median and standard deviation of the transfer times with one, two, three, and four devices connected to the Raspberry Pi computers with a distance of one meter between all devices. The results are discussed in Section 7.2.

Number of devices	Mean (ms)	Median (ms)	Standard deviation (ms)
1	94,98	97.50	42.78
2	161.70	159.25	78.86
3	242.20	233.00	136.38
4	295.35	311.00	106.21

Table 6.2: The mean, median and standard deviation of the transfer times with one, two, three, and four devices connected to the Raspberry Pi computers with a distance of three meters between all devices. The results are discussed in Section 7.2.

Number of devices	Mean (ms)	Median (ms)	Standard deviation (ms)
1	95.96	97.50	40.30
2	160.29	154.00	68.34
3	228.25	239.00	80.29
4	343.62	331.50	98.94

Table 6.3: The mean, median and standard deviation of the transfer times with one, two, three, and four devices connected to the Raspberry Pi computers with a distance of ten meters between all devices. The results are discussed in Section 7.2.

Number of devices	Mean (ms)	Median (ms)	Standard deviation (ms)
1	94.92	92.50	43.71
2	183.54	188.50	62.57
3	203.44	204.50	71.10
4	308.92	306.00	105.13

7

Discussion

In this chapter, I discuss and reflect upon the resulting prototype, the testing of the prototype, and how well I kept to the time plan. I also address the subjects of future work, both what is needed to make a working prototype and what can be added to make it a better one, and of ethical aspects related to this work.

7.1 The Prototype System

Although I developed a successful prototype it did not entirely meet the specifications I stated in the beginning of the project. Recall that in the Consat prototype, the smartphone is a BLE central unit. In order to get the nodes in my prototype to connect to at least one node on their central part and one node on their peripheral part, the peripheral connection is needed for the relay service to the next node. This means that the nodes can not connect to the phone as peripherals and so they must use a central connection for communication with the phone. Thus, the already developed phone application used in the Consat prototype could not be used to connect a phone to my prototype. Because of time constraints I did not finalize my prototype, and there was no time to develop a smartphone application for my system. Because of that, I could not conduct enough testing to see that all nodes can communicate with the phone at the same time.

Another problem is that I have not been able to create a true mesh network of the nodes with the hardware I chose. The implementation of the GAP protocol in the nRF51 SDK does not allow enough connections for a mesh network of multi-role devices. Furthermore, the hardware does not have enough RAM to keep that many connections at the same time. My resulting system is instead built up by a line network of the nodes. Although that approach is probably good enough even in larger vehicles, it is not as fault-tolerant as a mesh network can be. If one node dies, the nodes before and after that node in the line will try to connect. If the distance between these nodes is too great, however, they will not be able to establish a connection and the system will lose all nodes after the dead node as well. Another aspect to consider when it comes to fault-tolerance is the fact that the systems, both mine and the Consat prototype, have a single point of failure. The main node, which is responsible for the exchange of all security related information between the

phone and the central computer, is crucial for the system to work. If it dies, the system dies and the car can not be unlocked with the smartphone.

7.2 Test Data

The signal strength testing shows that the nRF51 DK performs better than the BLE400 nodes. The reason for the lesser signal strength of the Core51822 can be because of a hardware filter constructed with components of lesser quality than those on the nRF51 DK. It can also be because the device itself introduce more interference than the nRF51 DK. Either way, the Core51822 and the nRF51 DK both show an acceptable signal strength even at a 30-meter distance, although the nRF51 DK is slightly better.

A good signal strength is vital for the system to work. However, when using the prototype in a car, the distance between the nodes does not surpass three meters, meaning that the signal strength is good enough even in the cheaper Core51822 nodes. If the system is to be used in other applications, such as indoor positioning, the distance between nodes may be larger. In such cases better antennas, e.g., directional antennas, could make the system more reliable. Note also that in a finished product, the nodes would be tailored for the area of use, i.e., they would consist of hardware specifically chosen for that use case.

The transfer time tests show expected results with increasing transfer times the more devices there are, as stated in 6.2. Every node introduces extra overhead to the system since the received data has to be processed and relayed to the next node, which can be seen in the extra transfer times and the higher standard deviation. However, the results are not exact. The library used to control the GPIO-pins on the Raspberry Pi computers seems to be unstable. I noticed that even though the central Raspberry Pi did receive all data in order, the service Raspberry Pi sometimes did not get the interrupt. Since this interrupt is telling the service Raspberry Pi to stop the timer for a message, there are some missing timestamps and it is also possible that some end times have been linked to the wrong message. This basically means that the resulting data may not be as precise as I would have liked, but it is still possible to see that the transfer times are acceptable. Note also that the use of the pin-library itself increases the resulting times. A much better solution would have been to run the test with one multi-role Raspberry Pi instead. I tried to use one Raspberry Pi, but because of time constraints I did not manage to get it to fully work as a multi-role device.

When the system is used in a car, there are five nodes in total. I did not have enough hardware to test five nodes, but as the test results suggest, it is important to have reliable hardware with good antennas. The transfer time is short enough for the system to work correctly. As long as it does not take more than 1000ms the delay will probably not be noticed by the end user.

Recall that in the transfer test the data was sent every 300ms. This interval is tighter than necessary in a real product but it also shows how the system works if delays make the system congested to a small extent. If I had managed to get all nodes communicating with a phone as well, it would have been interesting to see how the system performs under pressure. As previously mentioned, the L2CAP protocol of the BLE stack does not offer any flow control or retransmissions, so testing how the system performs and how much data is lost would be necessary if the nodes could communicate with the phone simultaneously.

Even though the transfer time test was not as precise as I would have preferred, I consider the results encouraging. In a finished product, the nodes would most probably use better hardware and the distance between the nodes would not reach 30 meters in most cases.

7.3 Time Plan

I did not manage to keep to the original time plan, as seen in Figure A.1. The problems I had with the hardware during the entire project made the development part of the time plan much longer than intended. In turn, that meant that I did not have enough time in the end to finalize the mesh structure or to optimize the code further. I also had problems with testing the prototype, partly because the development consumed too much time but also because I would have needed more time to make the system connectible with the central computer. Now that the project is over, I realize I should have read more about the hardware in the planning stage which would have let me structure the project more around the limitations of the hardware. With more knowledge of the hardware, I might have been able to finalize a mesh or tree-like structure of the nodes and maybe even been able to connect the nodes to Consat's central computer and phone application. What I did learn, however, is how to develop Bluetooth Smart devices and the most common obstacles in doing so.

7.4 Future Work

There are many ways to improve on my prototype. For instance, in the planning phase I chose to not implement the security features that are needed in a finalized product. My prototype do not offer any kind of encryption between the nodes, the central computer, or the phone. This means that the system is susceptible to several kinds of security threats. In a complete product, this could have severe consequences and therefore implementation of security mechanisms should be prioritized. This implementation can be done by utilizing the SMP protocol of the BLE stack and also by utilizing the hardware support for encryption that exists on the nRF chips.

In order to connect the nodes in a mesh or tree network, hardware with more RAM

is needed. Nordic Semiconductor have released a chip called nRF52832 [15] that can easily run the code I have implemented and also handle at least one more connection because of more RAM. Although it might not be needed, synchronization between the nodes can be implemented as a means for better positioning. If the central computer knows exactly which RSSI value comes from which node and all nodes send their measurements synchronized, the resolution of the positioning may be better.

Making the system decentralized would make it more fault-tolerant as well. The current system has two single points of failure: the central computer and the main node. One way to make the system decentralized would be to allow some leader election between the nodes in order to establish which node is the main node. In this way, if the main node dies, another node will take its place through leader election. Furthermore, the functionality of the central computer could be moved to the nodes themselves since they all house a complete CPU. However, to make such a solution requires better hardware than I used in this project.

7.5 Ethical Aspects

Having the luxury of your car unlocking automatically when you get close and the convenience of sharing your car key by swiping on your phone may sound like great functionality. However, there are also some ethical aspects to consider. For instance, by using wireless technology the car can be hacked in more ways which can lead to easier car thefts. Furthermore, since the car is connected to the Internet for downloading verification the car can be a gateway to the servers containing information about users of the system. Depending on what the database includes, a hacker may even find information about family and friends that are allowed to use the hacked car. By hacking the database directly, the hacker may also find locations for specific cars and users.

8

Conclusion

This thesis presents an implementation of a network system of multi-role BLE devices used for positioning a smartphone, as well as an evaluation of that system for use in a product. The system is based on a prototype developed by Consat Engineering AB and the purpose of the system is to be more scalable, but still energy efficient, and easier to install than the Consat prototype. Even though the scalability of the system did not reach the level originally intended, it is still more scalable than the original prototype. The BLE technology itself makes the protoype energy efficient and since there are no wires it is easy to install.

The results show that the system works well when the distance between the nodes are 10 meters, and probably even more, and that the signal strength can reach distances of up to 30 meters. The intention was to make a mesh or tree structured network but in the end the hardware showed to be restrictive. With more RAM the system can maintain more connections, allowing for a tree- or mesh-structured network.

Even though the system does not work fully as originally intended, the results show that it still is a feasible solution for such system and can be seen as a proof of concept.

Bibliography

[1] ARM Ltd. Cortex-M0 Processor. http://www.arm.com/products/processors/cortex-m/cortex-m0.php, February 2016.

[2] Consat AB. Consat Engineering AB. http://www.consat.se/sv/consat-engineering-ab, February 2016.

[3] J. DeCuir. Introducing bluetooth smart: Part 1: A look at both classic and new technologies. *IEEE Consumer Electronics Magazine*, 3(1):12 – 18, 2014.

[4] R. Faragher and R. Harle. Location fingerprinting with bluetooth low energy beacons. *Selected Areas in Communications, IEEE Journal on*, 33(11):2418 – 2428, November 2015.

[5] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734 – 11753, September 2012.

[6] N. Gupta. *Inside Bluetooth Low Energy*. Artech House, 2013.

[7] Y. Jan, H. Tseng, Y. Lee, C. Lo, and L. Yen. Accurate Bluetooth positioning using weighting and large number of devices measurements. *Wireless Personal Communications*, 79(2):1129 – 1143, November 2014.

[8] Nordic Semiconductor. Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.nordicsemi.com/eng, February 2016.

[9] Nordic Semiconductor. Nordic Semiconductor Infocenter. http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.s130.sds%2Fdita%2Fsoftdevices%2Fs130%2Fs130sds.html, April 2016.

[10] Nordic Semiconductor. Nordic Semiconductor Infocenter. http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v11.0.0/index.html?cp=4_0_0_0, April 2016.

[11] Nordic Semiconductor. nRF51 DK / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.

nordicsemi.com/eng/Products/nRF51-DK, April 2016.

[12] Nordic Semiconductor. nRF51422 / ANT™ / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.nordicsemi.com/eng/Products/ANT/nRF51422, April 2016.

[13] Nordic Semiconductor. nRF51822 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51822, April 2016.

[14] Nordic Semiconductor. nRF51822 Bluetooth Smart Beacon Kit / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51822-Bluetooth-Smart-Beacon-Kit, April 2016.

[15] Nordic Semiconductor. nRF52832 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy2/nRF52832, June 2016.

[16] Raspberry Pi Foundation. Raspberry Pi 2 Model B. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/, June 2016.

[17] Silicon Laboratories. Bluegiga DKBT Bluetooth Smart Development Kit | Silicon Labs. https://www.silabs.com/products/wireless/bluetooth/bluetooth-smart-modules/Pages/dkbt-bluetooth-smart-development-kit.aspx, June 2016.

[18] A. N. Sloss, D. Symes, and C. Wright. *ARM system developer's guide: designing and optimizing system software*, chapter 5, pages 102 – 155. Elsevier/Morgan Kaufman, 2004.

[19] STMicroelectronics. SPBTLE-RF STMicroelectronics. http://www.st.com/content/st_com/en/products/wireless-connectivity/bluetooth-bluetooth-low-energy/spbtle-rf.html#design-scroll, June 2016.

[20] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert. Bluetooth positioning using RSSI and triangulation methods. In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 837 – 842, January 2013.

[21] Waveshare. BLE400 BLE4.0 Bluetooth 2.4G Mother Board, provides I/O expansion connectors and various interfaces. http://www.waveshare.com/ble400.htm, June 2016.

[22] Waveshare. Core51822 BLE4.0 Bluetooth 2.4G Wireless Module, nRF51822 Onboard. http://www.waveshare.com/core51822.htm, June 2016.

[23] J. Yiu. *The Definitive Guide to Arm® Cortex®-M0 and Cortex-M0+ Processors*, chapter 19, page 511 – 557. Todd Green, 2nd edition, 2015.

A

Appendix 1

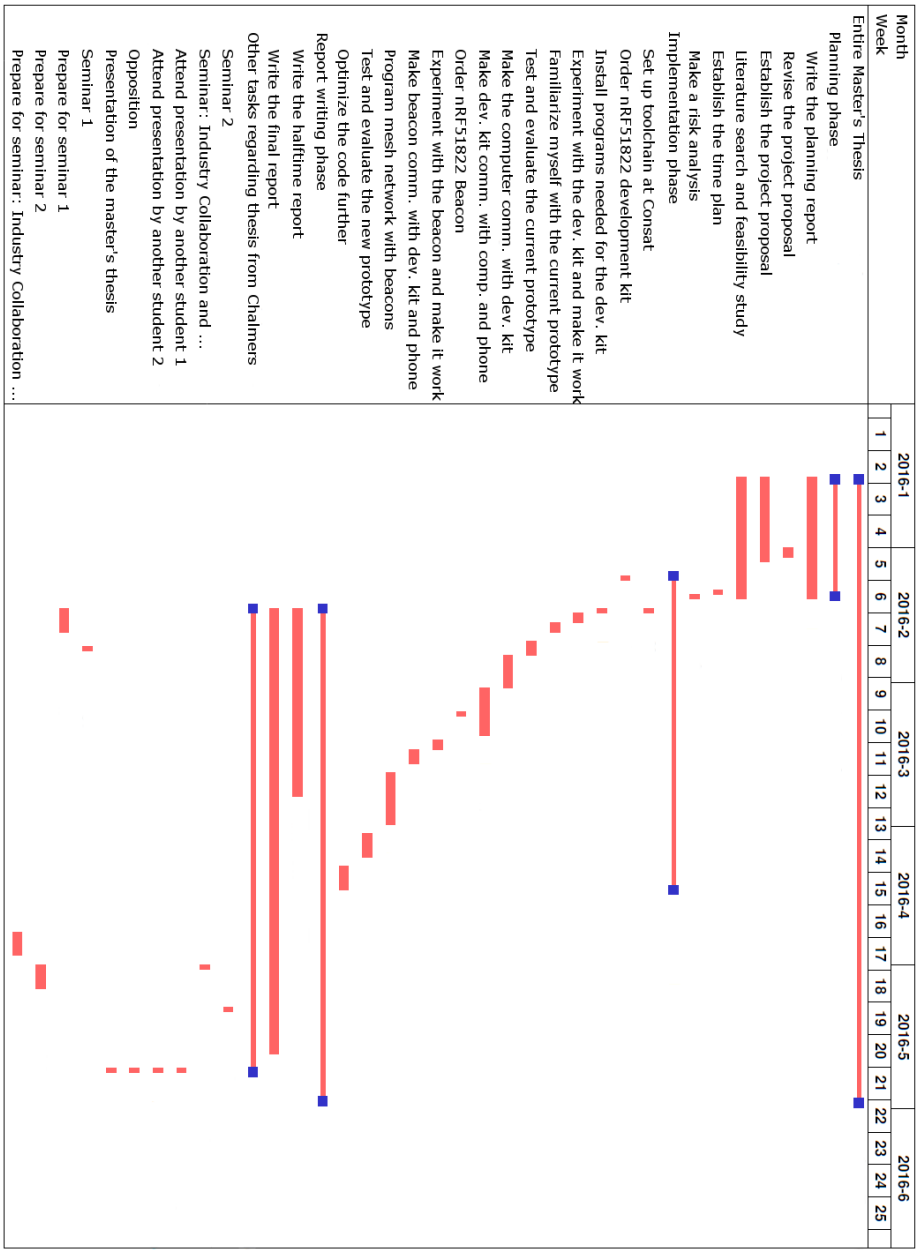


Figure A.1: The time plan for the project.