CHALMERS
UNIVERSITY OF TECHNOLOGY

Evaluation of possible positioning of devices in a low energy mesh network

Bachelor’s thesis in Electrical Engineering

Robert Jegerås and Erik Eklund

Department of Signal and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Evaluation of possible positioning of devices in a low energy mesh network

Robert Jegerås and Erik Eklund

Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31 – 772 1000
Fax: +46 (0)31 – 772 3663

Department of Signals and Systems
Gothenburg, 2016

Abstract

This report deals with positioning between devices in a low energy mesh network. It uses established technologies such as Bluetooth Smart communication. With the increasing usage of IoT devices the demand for new applications also increases. One desirable application of Bluetooth communication is to extract the relative positions of devices in close proximity. The purpose of this thesis is to come up with a solution to this problem. The received signal strength will be used for the problem because electromagnetic waves decrease in energy the further it travels and can be used to determine the distance travelled. The reader will be presented with both theoretical and practical explanations for the solution. That includes the mathematical and physical considerations that had to be made. The reader will also be presented with the code written in C used for the solution. The results shows that the exact positions and distances are not very reliable however the relative positions of each device can still be extracted successfully. A discussion and proposed future work will be presented at the end of the thesis. This thesis was made at the request of the System and Architecture division of Semcon AB, Gothenburg, Sweden.

Sammanfattning

Den här rapporten handlar om att uppnå relativa positioner mellan enheter i ett lågenergikommunkationsmeshnätverk. Den drar nytta av etablerade tekniker som Bluetooth Smart. Ett önskat användningsområde för Bluetooth kommunikation är att ta ut positioner mellan enheter som är i närheten. Syftet med denna uppsats är försöka lösa det problemet. Den mottagna signalstyrkan från den elektromagnetiska strålningen kan användas för att ta reda på avstånd mellan två enheter då strålningens styrka avtar med avståndet som den färdas. Läsaren kommer att bli presenterad med både teoretiska och praktiska förklaringar till lösningen. Det innefattar både matematiska och fysiska överväganden som gjordes. Läsaren kommer även bli presenterad med programkoden som skrevs i C. Resultaten visar att det inte blir pålitliga resultat när exakta positioner och avståndet beräknas. Däremot kan de relativa positionerna mellan varje enhet extraheras ur lösningen. En diskussion och förslag till framtida arbete presenteras i slutet av rapporten. Uppsatsen gjordes i samarbete med avdelningen System and Architecture hos Semcon AB, Göteborg, Sverige.

Acknowledgements

This report was written as a part of the bachelor thesis done at Chalmers University of Technology in Gothenburg, Sweden at the institute of Signals and Systems. The thesis was a side project conceived by Semcon AB who we cooperated with during the thesis. Semcon AB also provided desk space for both of us.

We would like to give our thanks to our supervisor at Semcon AB: Anna Funke, Magnus Andersson and Peter Nordin who gave us this opportunity. A thanks to our supervisor at Chalmers, Sakib Sistek for guidance and moral support during rough times and finally a thanks to our examiner Manne Stenberg.

Table of contents

Abbreviations

BLE	Bluetooth Low Energy
RSSI	Received Signal Strength Indicator
FSFM	Free Space Friis Model
FSPL	Free Space Path Loss
RF	Radio Frequency
SoC	System on a Chip
CPU	Central Processing Unit
RAM	Random-accessed Memory
SIG	Special Interest Group
IoT	Internet of Things
IDE	Integrated Development Environment

1. Introduction

1.1 Background

It is widely believed that the future home will include a wide variety of connected devices. The future can already be seen today with media centres and computers but eventually even lamps, doors and other simple things could also be connected. This development that is seen today is commonly called “Internet of Things” (IoT).

The garden will not be an exception and even here the simple and complex devices like sprinklers and autonomous lawnmower. What differentiate the garden against the home is that the access to a reliable power source is limited, and will rely on battery power and or solar power.

These power constraint make having an energy efficient product the outermost of importance, a fitting solution would be “Bluetooth Smart”, also known as “Bluetooth Low Energy”(BLE). Another challenge is that a garden is in the most cases a fair amount larger than what could be covered by a single antenna. With the use of BLE another problem appears at BLE sends with a lower bitrate than regular Bluetooth.

Therefor to solve all these problems a mesh network will be established where all the devices are both receivers and transmitters. This solution could then produce coverage over a larger area than what a single antenna ever could.

Husqvarna AB together with Semcon AB are researching the possibility for this kind of solution to work. In addition to the main project Semcon AB thought of another smaller side project where the goal is to achieve relative positions of devices using low energy communication in a mesh network.

1.2 Goal

The main goal is to have a functional position system in network of stationary devices.

1.3 Purpose

The purpose of having the positions of devices in a mesh network is many. To know where the devices are without remembering exactly where they are laid, so that the devices are never forgotten. Another purpose of knowing the positions of the devices is that an autonomous lawnmower does not run over sensitive equipment when mowing the lawn. Yet another purpose for this thesis is to investigate of how accurate the system can be.

1.4 Delimitations

The environment during the testing phase will be flat with no obstacles. Four devices in the mesh network will be tested and all the devices will be stationary.

1.5 Scope

The system will be built upon Nordic semiconductor development board NRF51422 as it has been chosen by Semcon AB as the device for their main project. Investigation about what is the ideal distance measurement when regarding power consumption. The program will be written in an existing mesh network template. A device shall be able to calculate its neighbours position.

2. Technical Background

A number of different physical properties has to be taken into account as well as some mathematical techniques.

2.1 Nordic Semiconductor nrf51422

The nrf51422 is manufactured and developed by Nordic Semiconductors, which is a company based in Norway. The nrf51 series is a part of ultra-low power system on a chip (SoC). The board is less popular in the maker community than other competing brand because of this niche solution. Due to this the setup complexity is quite high as there are not that many users who have had similar problems setting the board up.

The development board uses an ARM Cortex-M0 32-bit CPU with 156/128kb memory and 32/16kb RAM. The board has a 2.4GHz transceiver with a low power sensitivity of -93 dBm. The board is designed to be ultra-low power and can be run by a 3v coin cell battery (C2032) [1].

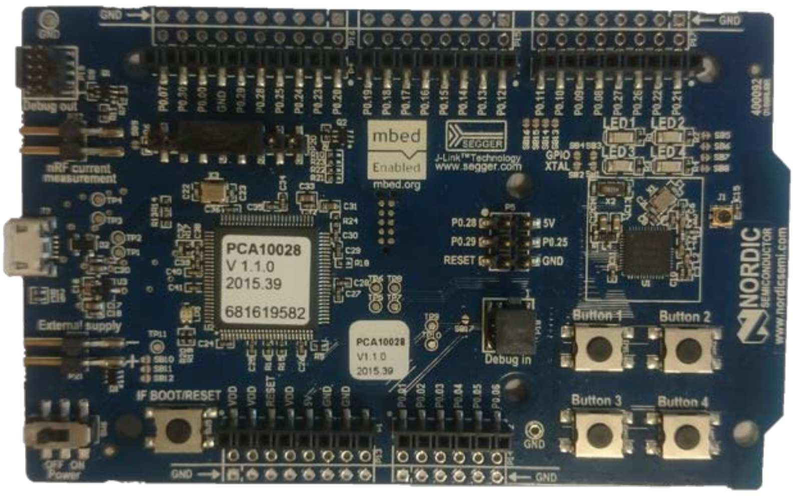


Figure 2-1

2.2 Distance measurement

To eventually calculate the angles of the devices in the network a linear distance between the devices were needed to make a calculation possible

A number of different ways to evaluate distance between several devices was researched. The advantage and limitations of these different techniques was investigated before the final implementation was done.

2.2.1 Received Signal Strength Indicator (RSSI)

The received signal strength in a RF wireless communication system can be presented as a RSSI value. The signal strength that is received from a package can be converted to a distance using

Free Space Friis equation. The advantages of this solution is that the device who receives the package and its signal strength knows exactly which device the package came from. Another big advantage is that no external device needs to be implemented for this solution to work, in a final version an antenna might be added for extra reliability.

2.2.2 Laser

Measuring distances using laser is an accurate and reliable technique, and with their millimetre precision a reasonable solution to solve the distance problem. However without knowing at what angle the other devices are at, it would be next to impossible to know where to measure. A solution to that problem would be to implement a reflector on the devices and have the laser gauge spinning and detecting the stronger ray from the reflector.

Even with this solution there would still be a problem because the devices have no idea from which device that specific ray came from.

And to implement this solution would be very costly and would consume too much power to be considered a low energy solution [2, 3].

2.2.3 Ultrasound

Using sound waves to measure distances is an old technique and was popularized in the early nineteen hundreds with echo sounding to be able to determine depth out at sea. Using ultrasound is an accurate measurement tool similar to laser measurement but is even harder to control due to the sound waves reflecting in a less reliable way. This solution would also need external devices added on to the existing device to be able function. The ultrasound function is likely not that power efficient as it would need to send and listen to soundwaves.

2.2.4 Photo

Measuring the distances between the different devices is possible to do with the use of photography, however to implement a self-sufficient system that does not require external help would be problematic. The camera that takes the photo needs to be connected to the system and must be in line of sight of every device. Even if the photo is taken, reference points needs to be present to be able to calculate the distance between the devices. This solution is expensive and has a high power consumption which would make this solution unsuitable for this project.

2.2.5 Magnetism

Measuring distances using magnetism is an accurate technique but its limitations far exceeds its advantages. The technique uses a system called 3d magnetic tracking with its high accuracy it is still too hard to implement in a small form factor and requires more power than what is available. The technology is not open source and hard to implement in a system as it requires additional hardware on the device [4].

2.3 Physics and mathematics

In this chapter all the physics and formulas that was used will be explained.

2.3.1 Filtering

The values in the system are quite inconsistent and fluctuating between each measurement, to stabilize the system some sort of filter needed to be implemented. Two simple filters are used in this project and the two are described below.

2.3.1.1 Moving Average

Moving average is like a normal average calculation but the oldest values after a set amount is discarded.
a is a set of data at discrete time.

$$a = [a_0 \; a_1 \ldots a_{n-1}] \tag{2.1}$$

$$MA(n)_m = \frac{a_{n-m} + a_{n-(m-1)} + a_{n-(m-2)} + \cdots + a_{n-(m-m)}}{m} \tag{2.2}$$

Where n is the n´th step of the process and m is the length of the desired calculation.

2.3.1.2 Kalman filter

One way to “filter” the noise from a lot of data is Kalman filtering. Rudolf E. Kálmán introduced this filter in 1960. Big changes in the measured value doesn’t affect the estimated value. Kalman filtering requires some knowledge or assumptions about the system like initial estimate(EST_0), error in the estimate(E_{EST}) and error in the measurement (E_{MEA}) [5]. The measured value in the system is MEA.

It is an iterative process that consists of 3 steps:
1. Calculation of the Kalman Gain:

$$KG = \frac{E_{EST_{t-1}}}{E_{EST_{t-1}} + E_{MEA}} \tag{2.3}$$

2. Calculate the next estimated value:

$$EST_t = EST_{t-1} + KG(MEA - EST_{t-1}) \tag{2.4}$$

3. Calculate the next error in the estimated value:

$$E_{EST_t} = E_{EST_{t-1}}(1 - KG) \tag{2.5}$$

And then the process is repeated.

2.3.2 Signal propagation

If P_t is the transmitted power and P_r is the received power, the distance between transmitter and receiver can be calculated like the following equation [6].

$$d = \frac{c}{4\pi f}\sqrt{G_r G_t \frac{P_t}{P_r}} \qquad (2.\ 6)$$

G_r and G_t is the receiver and transmitter antenna gain. f is the frequency of the electromagnetic wave and c is the speed of light in a vacuum.

This formula assumes ideal components. Real components have losses and will be taken into account below. This is called Free Space Friis equation or Friis transmission equation, this equation will be called FSFM henceforth [14].

This can also be done on a logarithmic scale stated in dB for simpler computation [7].

$$20\, log(d) = P_t - P_r - CL_r - CL_t + G_t + G_r - 20\, log(4\pi f) + 20 log(c) \qquad (2.7)$$

CL is the cable loss.

This simplified form of the equation holds true for distances $d >> \lambda$ and the antennas are correctly aligned and their respective polarization are the same [15].

2.3.3 Angle calculation

If three sides of a triangle is known, all three angles can be calculated using these simple formulas.

$$A = arccos(\frac{b^2+c^2-a^2}{2bc}) \qquad (2.8)$$

$$B = arccos(\frac{a^2+c^2-b^2}{2ac}) \qquad (2.9)$$

$$A = arccos(\frac{b^2+a^2-c^2}{2ab}) \qquad (2.10)$$

Where A, B and C are angles given in radians.

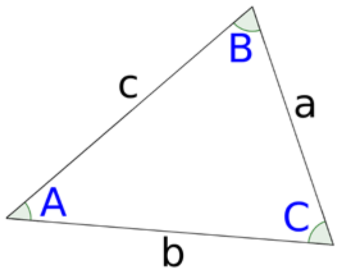


Figure 2-2

2.4 Bluetooth

Ericsson AB invented the technology now known as Bluetooth with the intention to be used with wireless headsets. The technology has since been developed and managed by a special interest group (SIG) that consists of over 25 000 companies. Bluetooth has evolved from a specific solution to a standard that used by millions of devices.

2.4.1 Bluetooth Low Energy

One of these advancements in Bluetooth technology is BLE which stands for Bluetooth Low Energy also known as Bluetooth Smart. The technology was developed by Nokia in 2006 under the name of Wibree but was later implemented in the Bluetooth standard with Bluetooth 4.0 in 2010 [11, 12]. Internet of Things is something that was made commercially viable with the implementation of Bluetooth Smart as it made it cheaper and more energy efficient to implement connectivity to a number of devices.

2.4.2 Advertising (Broadcast)

Advertising is used to establish connection between devices. It contains information about the device to make the connection process faster and simpler. This can be used to transmit data across devices without having an established connection.

3. Implementation

3.1 Code

The program was written in C using Keil uVision5 for development and testing. It uses Softdevice S110 and SDK 8.1 with an existing template with mesh functionality that was made by Nordic Semiconductor [13]. The program uses the BLE advertising message to achieve mesh functionality

3.1.1 Overall structure and functionality

The program is divided into three parts, one that is sending an array filled with data collected during runtime. The second part is receiving the same array but from the other devices and process the data and continue building the array to be sent again.
The third part of the program converts signal strength to distances and calculates the angles between them.

The transmitter part of the program is quite simple, two timers with different lengths and then a transmit functions that sends the array every time one of the timers is triggered. The timers are set to one short and one long where the long timer is initiated at start and the short timer is initiated when the long timer is triggered. Whenever a short timer has been triggered a function to send the array is called and transmits the package over the advertisement channel.
The receiver part of the program is far more complex as it needs to keep track of every other devices package, as the data in the package varies and the data from the advertisement itself is important.

The array is built up with the specific device id in the first place followed by the signal strength captured during runtime in the data segments as follows, each place in the array is the signal strength between two specific devices. The signal strength between device A and device B is located on the second place in the array and the signal strength for device A and C is in the third place and so on.

The array that is send by device A looks like this: $[A, 57,54,60,55,67,65]$, the places next to the device ID is the signal strength between the other devices in the mesh network.

The program checks from which device the array came from and the signal strength it had when it arrived. This signal strength is then put through a Kalman filter and after a set amount of times this process has been done the filtered value is mapped onto the array on a specific location and is now ready to be transmitted with the new value on it.

The code as a whole can be viewed in appendix 7 with a simple flowchart for the program structure in appendix 1.

3.1.2 Functions

Short descriptions of the most important parts and functions in the program.

3.1.2.1 Main

In the main function of this project essentially all other functions are called. But the main functionality of the main function is the receive part of the function as it needs to happen during an interrupt from a BLE mesh event. The data that arrives from the BLE mesh event is checked if it is a package that is meant for positioning or not and what device the package came from. Depending on what device the package came from a filtration is made using a Kalman filter on the received signal strength (RSSI). The remaining data on the array is processed in a similar fashion.

3.1.2.2 Timers

The program has two timers that will trigger an interrupt. One long and one short. When the device is powered on the long timer is started. When the long timer is triggered it starts the short timer. When the short timer is triggered it sends the final Kalman array in a BLE advertising message. If the short timer has been triggered x number of times it stops itself and will not resume until the long timer is triggered again.

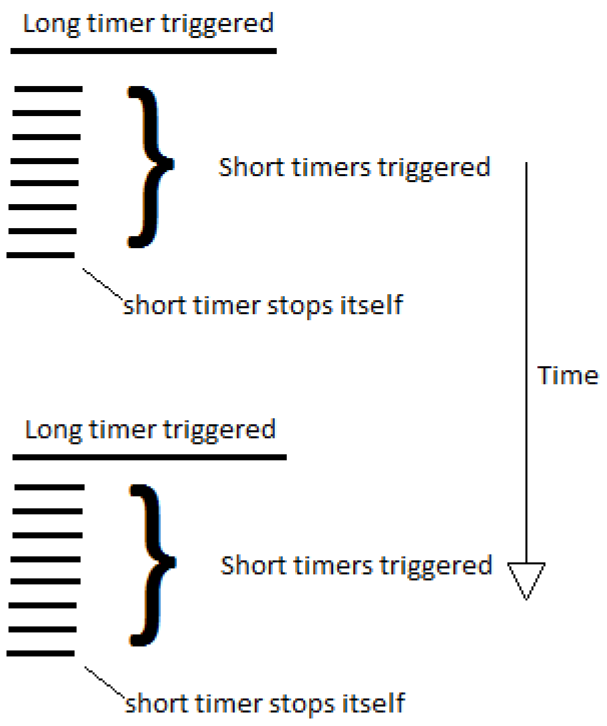


Figure 3-2
Illustration of how the timers operate.

3.1.2.3 Kalman

The Kalman function is called from the switch statement in the main function of the program. It takes three arguments and those are the new received signal strength, the previous estimated value and the previous estimated error. It performs the Kalman calculation and returns two values, the new estimated value and the current estimated error. When enough iterations has been executed the estimated value will be assigned to a corresponding location in the final array. The program has to keep track of all the different values for every case and does so by having four lists of values for every case.

3.1.2.4 Distance conversion

The distance conversion uses the FSFM equation to convert the signal strength to a distance in meters. The function takes in a signal strength from a specific location in the array and return the distance for that signal strength to a different array but in the same location as the one before.

3.1.2.5 Angles

To compute the angles in a triangle for three or more devices this function is called with the distances calculated before in the distance conversion function. To calculate the angles three sides of a triangle must be present for the equation to work. To be able to do this the function must be called with three specific values, it cannot just be three distances in the array as some cannot form a triangle in reality. Distances between device A and B, A and C, and A and D does not form a triangle as an example. Because of this a special iteration of what distances to calculate must be done before calling the function.

The calculation for the angles in the function is not that complicated, the functions puts the distances in three different equation, one for each angle and then returns the angles.

3.2 Kalman filter compared to moving average

A simulation was done to compare the Kalman filter and a moving average to determine which is the most effective for the task. A small program was written in MATLAB to compare the two. The program generates 20 random values between -50 and -60 which symbolizes the fluctuation in RSSI value. There is also a 1% chance that the value will have another -40 added to itself. This is because sometimes a signal is received that has been reflected in the surrounding environment and therefore has significantly lower signal strength.

The Kalman filter was applied to both the simulated RSSI value as well as the linear value from the generated RSSI. This was done to see if there were any differences in the end result.

The Kalman filtering and moving average computation are shown below

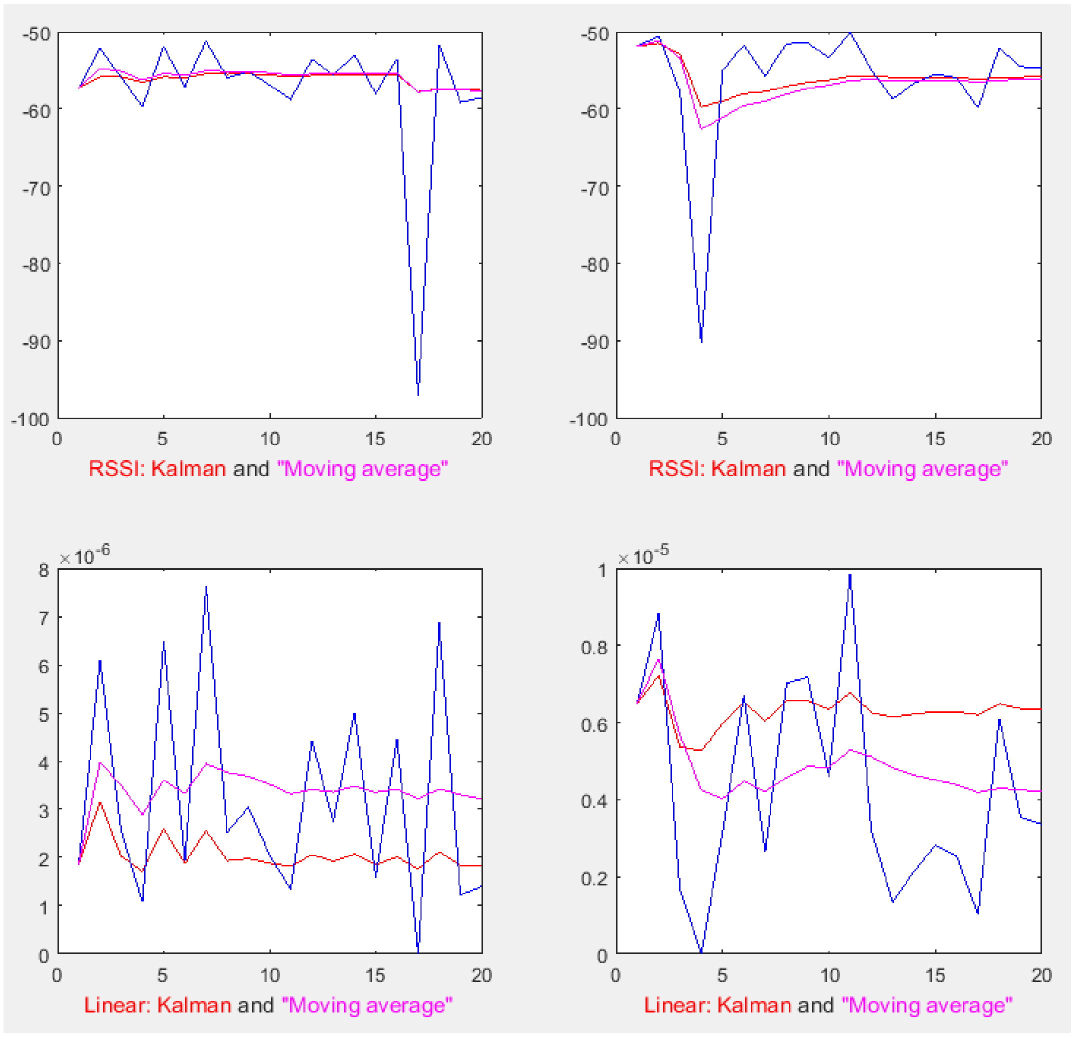


Figure 3-1. Two different sets of 20 values (in blue). Both Kalman filter and moving average were applied and shown in red and pink

As seen in the graphs the differences between Kalman filtering and moving average is not very big. However big changes in the values has a lesser effect on the Kalman filter than moving average.

It appears that the logarithmic and linear calculations does not make a difference in the end result. However the difference between the Kalman calculation and the moving average calculation can be seen more clearly.

The difficulty in making a good Kalman filter is the estimates that have to be made before the filtering begin. In this example the first value of the Kalman filter is the same as the first generated value. This makes the resulting estimation provided by the Kalman filter not as accurate if the first value is far away from the "true" value. If a better estimate can be generated the performance of the Kalman filter will increase.

There is also an estimate about the error in the measurement that can improve the performance of the Kalman filter. This is how much the measured value is known to fluctuate between measurements. In this example the error in measurement is 3 which is approximately how much the RSSI value varies between measurements when testing the system.

The Kalman filter was chosen for this solution because of its ability to “ignore” big changes in values even though moving average was very close to the Kalman filter.

3.3 Distance calculation

After some rearrangements of the Free Space Friis equation formula the distance d can be calculated like this:

$$d = 10^{(P_t - P_r - CL_r - CL_t + G_t + G_r - 20\,log(4\pi f) + 20\,log(c))/20} \tag{3.1}$$

The nrf51422 development board has about 0,5dB cable loss and 0dBi antenna gain [8, 9]. The frequency is 2426 Mhz for BLE advertising channel 38 [10]. c is the speed of light in vacuum. This reduces the formula down to:

$$d = 10^{(P_r - P_t - 1 - 40{,}14)/20} \tag{3.2}$$

The transmitted power P_t in this case is 4dBm and P_r is the received RSSI value. This results in the final calculation for the distance d:

$$d = 10^{(-37{,}14 - P_r)/20} \tag{3.3}$$

This resulted in the calculated distance being about ten times bigger than the actual distance when the devices were close to each other. For demonstration purposes the following conclusion was made. If the average RSSI values is greater than -60 then the distance will be divided by ten. If the average RSSI values is less than -60 then the distance will be divided by three.

3.4 Testing

3.4.1 Data Processing from Mesh Network

To demonstrate the results from the calculated distances and angles a simple JavaScript with a canvas to display the results from the collected and calculated data.

For the JavaScript to be able to connect to the mesh network a simple Python script was written to extract the data from the serial port on the local computer connected to one of the devices. The script was reading the data that comes from the serial port and writes a log file with that exact data.

The data that is logged comes from the program and is the array that is calculated and filtrated during run time. This array differs a little bit from the previous declared array that was mentioned before. Instead of only the RSSI value for each of the distances which are of the type 8 bit unsigned integer. As it is an integer only whole number is available and many decimals that is calculated from the Kalman filter is thrown away. The data that is logged does not require to be of that same type as it is only logged to the serial port, this lead to the data in this array to be of the type float. The type float allows for significantly more accurate values, with up to 16

decimals. For this demonstration only 4 decimals were used as the data became too small to make a difference in the end results.
After the JavaScript reads the latest data in the log file, it does the same equation that the devices does, it converts the signal strength to distance and then calculates the angles for each triangle in the mesh network. This is a bit redundant to do the same calculation twice but this is done as it was easier to export a simple line from the mesh than all of the calculated results and by doing it like this the speed to show a result on the canvas increased significantly.

The results from the calculation is then used to display the result in the canvas with using the distances and angles to calculate what x and y coordinates the devices have. With a different colour for each device lines are drawn between each device to show all the triangles made up by the devices.

A table consisting of all the relevant data is also shown, with the distances between each device and angles.

3.4.2 Method

To test the system as a whole testing was done a number of times to showcase different scenarios the system could be exposed to.

The test was done with four different layouts in three different scales, near (~1m), medium (~5m) and far (~15m) resulting in a total of twelve measurements. The results were then compared to the actual distances and angles between the devices, measured with a measuring tape.

The different layouts is designed to fully test the capabilities of the system and show how in real life the system may function.

Each test was done for 5 minutes each to get a more reliable result as the values becomes more exact as the times goes on.

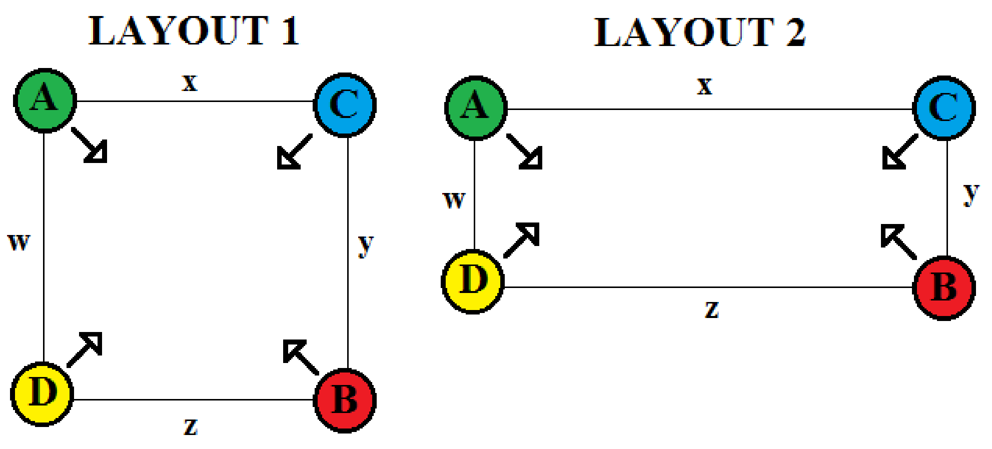


Figure 3-2

In the first layout the system is tested to perform using units in a uniform structure where the devices are the same distances from each other. The second layout is quite similar to the first layout but here they are paired up and each pair is separated by a larger distance.

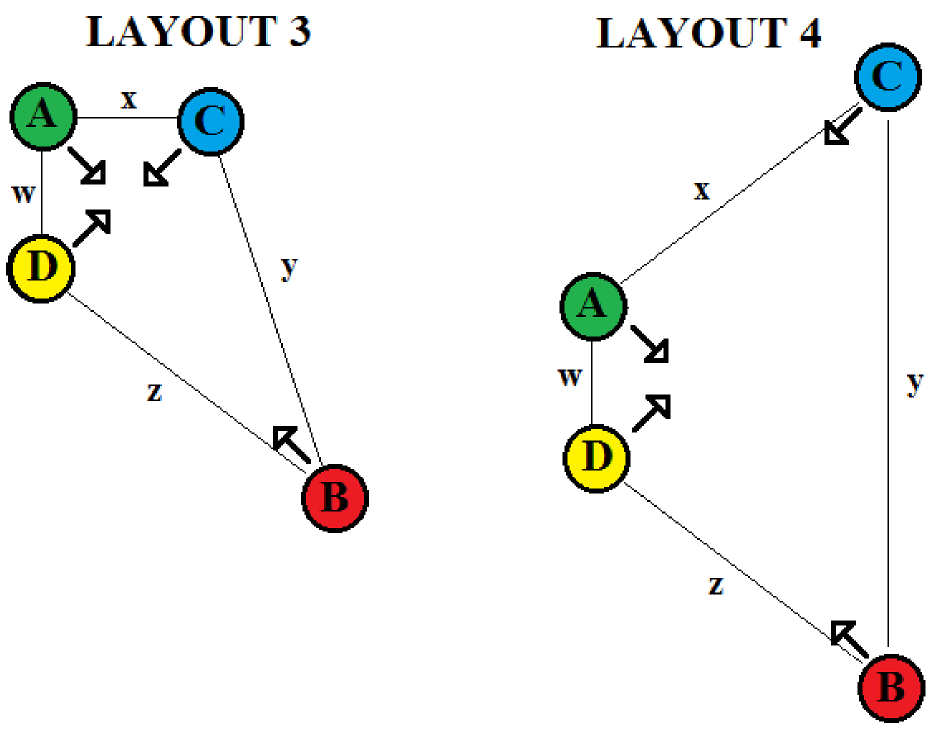


Figure 3-3

Both layout 3 and 4 is made in this form to see how well the system handles when one or two devices is significantly further away from the rest.

In every measurement done the orientation and the height of the devices is the same to eliminate any inconsistencies in the result. The devices sit flat on a table at height of 80 cm with the part of the device with the antenna pointed towards the centre of the layout as shown in the pictures above, the testing area with an undergoing test can be viewed in appendix 2.

4. Results

The results from the demonstration that was explained in the previous chapter are shown below. All measurements were made at five minutes each for the result to settle. A reset was done after each measurement. The testing area was an open space with high ceiling, the location is shown in appendix 2. The distances between each unit compared to the given result from the system is displayed in a table in the beginning of each layouts appendix.

4.1 Measuring the first layout

The first layout is a square with the sides x, y, z and w. The layout is shown in figure 3.2 with the title LAYOUT 1. The first layout has the simplest shape and therefore should have the best results. All figures for the first layout and its respective table with measuring data can also be found in appendix 3.

4.1.1 Near

The results for the first layout with a near distance shows how one device can disturb the result in the system. Here the device named C is sending with a weaker signal than the rest which causes the whole shape to become disfigured.

4.1.2 Medium

The results from the measurement with a distance at five meters from each other makes a much more accurate results and shows an almost complete square, this is one of the best result possible where an almost replica of the desired shape was given.

4.1.3 Far

At ten meters away from each other the results are an almost complete square with only a few degrees off.

4.2 Measuring the second layout

The second layout shown in figure 3.2 with the title LAYOUT 2. The basic form is of a rectangle with A and D on one short side and B and C on the other. This layout and its relative results can be viewed in appendix 4.

4.2.1 Near

As the distances is so short the result are not that truthful to the real devices as the signals are far too unreliable at short range.

4.2.2 Medium

The rectangle is beginning to take shape at this distance.

4.2.3 Far

Even if the results are not perfect a clear rectangle is present in this result.

4.3 Measuring the third layout

The third layout shown in figure 3.3 and has the title LAYOUT 3 is where three units are closer together than the fourth one. The fourth device in these measurements are the device named B. The layout and its results can be found in appendix 5.

4.3.1 Near

Both B and C got further away from the rest in this result, not that surprising when the devices are so close together.

4.3.2 Medium

This result does not show that B is further away from the rest. The signal strength between A and B was to strong and that distorted the result a bit.

4.3.3 Far

In this result it is clear that the device named B is further away from the group, which is a successful result.

4.4 Measuring the fourth layout

The fourth and final layout tested on the system is shown in figure 3.3 with the title LAYOUT 4, the layout is design to see if the system can recognise if two devices is further away than two who are closer together. Devices B and C are the devices who are further away during this test. Results from this layout can be viewed in appendix 6.

4.4.1 Near

The results here are not that surprising as the difference in distance is not that great and with the short distances signal strength is higher.

4.4.2 Medium

Here the result clearly show that both device B and C is further away from the other two.

4.4.3 Far

With the devices further away from each other the signal strength varies more and the result is not that clear.

5. Discussion

5.1 Results discussion

To only use RSSI as an indicator of distance estimation is clearly not very accurate.
The signal strength vary by a fair amount among the different test scenarios. However this is nothing that could have been foreseen as most problems with using signal strength as a base for distance calculation is that the signal strength itself have a tendency to jump up and down in strength.

This is eliminated with an implementation of a Kalman filter in this solution. No matter how good conversion made from the signal strength to distance the problem still remains with the signal strength itself. The unreliability of RSSI is shown in the results as it is next to impossible to determine how the signals propagate through the room, which signal reflects on the walls, floors or ceiling and how these reflections affect the result. Because of the unreliability of the RSSI two stationary devices with the same distance between them and pointed towards each other can still receive different signal strength.

The solution that was made in this project does convert the distance, these values are not close to what the real world distances are but they are however close to their relative distances to each other. Therefore the angles between the devices are calculated with these distances given from the conversion.

That makes it possible with this technique to find out an estimate on where devices are relative to each other and what form they take on in a mesh network.

The results shows that an accurate representation of a real world layout is not possible, but all tests shows that it is not far of the intended layout.

The program is accurate in finding the signal strength between devices but no matter how accurate the signal strength is the signal strength itself does not always stay the same from device to device. What that means is that for this system to have more accurate distance estimation another system needs to be implemented that can gather more data on the devices.

5.2 Antenna discussion

This solution uses the simple version of Friis Free Space equation where it is assumed that the antennas are impedance matched (Γ_t and Γ_r are zero), correctly aligned

$$|a_t \cdot a_r^*|^2 = 1 \qquad (5.1)$$

and zero absorption in the medium ($\alpha = 0$). The full equation looks like this [15]:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2 (1 - |\Gamma_t|^2)(1 - |\Gamma_r|^2)|a_t \cdot a_r^*|^2 e^{-\alpha d} \qquad (5.2)$$

That the antennas are correctly impedance matched is a fair assumption given that the nrf51422 development kit was used. However, the polarization vectors a_r and a_t are almost impossible to know exactly at any giving time with respect to each other. The absorption coefficient $\alpha$is also very difficult to calculate. All of these has a significant effect on the received signal strength. A possible solution to this will be presented in the next chapter.

6. Future work

6.1 Additional methods

All these methods has the possibility to make the distance estimate more reliable but requires additional research.

6.1.1 Particle filter

Another possible approach to this problem is instead of using filtered signal strengths is to use something called particle filter. If implemented in a similar way with four devices the particle filter would use the signal strength from all devices to figure out where one device is located. With this technique it should be more accurate as it uses the signal strength of three devices to find the location of the fourth. Then repeat this until all the devices location is known. What makes this better than the solution made in this project is that it uses more measuring points to determine one location. This solution would also be more efficient as it would not need to calculate angles to know the location of the devices.

6.1.2 Joint probabilistic data association filter

This method takes all probable targets into account when creating the statistically most probable solution from the data gathered [16].

6.1.3 Compressed sensing

Another way of creating a better estimation from noisy and sparse signals is compressed sensing. It can recover a more precise representation of the data than the sparse measurement [17].

6.2 Real world situation

Testing the system in a real world environment is something that can be interesting to add to this project as all the collected data was measured inside. Moving the system outside to a lawn for example may show a variety of different results that is not included in this system.

6.3 Moving device

To triangulate a moving device in the vicinity of the mesh network would need little modification of the original code. But as the technique used in this project an accurate estimate of the moving device would not be possible. With the particle filter mentioned earlier a moving device would not need any modification at all to be functional.

6.4 Close quarter RSSI

Something that could produce better results when devices are close to each other is to implement a few decimals in the RSSI itself. As it is now only integers are allowed which makes it difficult to determine any changes when the devices are within one meters of each other.

6.5 Antenna

The nrf51422 development board has a built in antenna that is parallel to the ground. Because of this the antenna aperture will not be optimal between devices depending on how it is aligned. This greatly affects the signal strength on the receiving device. One possibility is to have an antenna that is pointing straight up from the device. This will make the antenna aperture the same regardless of how the devices are aligned with each other.

7. Conclusion

The end goal for this thesis was evaluate the possibility to find relative positions of devices in a low energy mesh network. This thesis evaluate the possibility to do so by using the received signal strength that is already existing in communication between two devices using Bluetooth Smart. The system is built on Nordic Semiconductor´s nrf51422 development board, and the system itself is written with C on an existing scalable mesh network template.

In the end the result gained from this thesis were that to achieve an exact position of any device more forms of evaluating distances were needed. With the RSSI as an only point of reference only a relative shape was possible.

Acknowledgment

We like to give a final thanks to Semcon AB for financial support during the thesis.

References

[1]Nordic Semicondutor (2016) [Online] Available: https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51822

[2] Bosch (2016) [Online] Available: http://www.bosch-professional.com/gb/en/glm-80-26171-ocs-p/

[3]Coherent (2016) Measuring Laser Power and Energy Output [Online] Avaiable: http://www.coherent.com/downloads/aboutmeasuringlaserpowerndenergyoutputfinal.pdf

[4]A. Plotkin and E. Paperno (2003). 3D magnetic tracking of a single subminiature coil with a large 2D-array of uniaxial transmitters

[5] Emmanuel Candes (2005, October 28). The Kalman filter [Online]. Available: http://statweb.stanford.edu/~candes/acm116/Handouts/Kalman.pdf

[6] Mike Golio, The RF and Microwave handbook, CRC Press, 2001, ch 9, pp 19

[7] Laird Technologies. (2012). Understanding Range for RF Devices [Online]. Available: http://www.digikey.se/Web%20Export/Supplier%20Content/Laird_776/PDF/laird-wireless-understanding-range-rf-devices.pdf?redirected=1

[8] Øyvind Karlsen. (2015, November 13). Nrf51 conducted power[Online]. Available: https://devzone.nordicsemi.com/question/56859/nrf51-conducted-power/

[9] Petter Myhre. (2015, December 22). Nrf51 DK Antenna gain[Online]. Available: https://devzone.nordicsemi.com/question/61709/nrf51-dk-antenna-gain/

[10] Bluetooth SIG. (2016). Technical considerations [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/technical-considerations

[11] Ed Grabianowski. (2006) Is Wibree going to rival Bluetooth?[Online] http://electronics.howstuffworks.com/wibree.htm

[12] Bluetooth SIG, inc (2016) Low Energy [Online] https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy

[13] NordicSemiconductor (2016). nRF51-ble-bcast-mesh [Online]. Available: https://github.com/NordicSemiconductor/nRF51-ble-bcast-mesh , (Accessed 17-02-2016)

[14] Mathuranathan (2013, September 13). Friis Free Space Propagation Model [Online]. Available: http://www.gaussianwaves.com/2013/09/friss-free-space-propagation-model/ , (Accessed 17-5-2016)

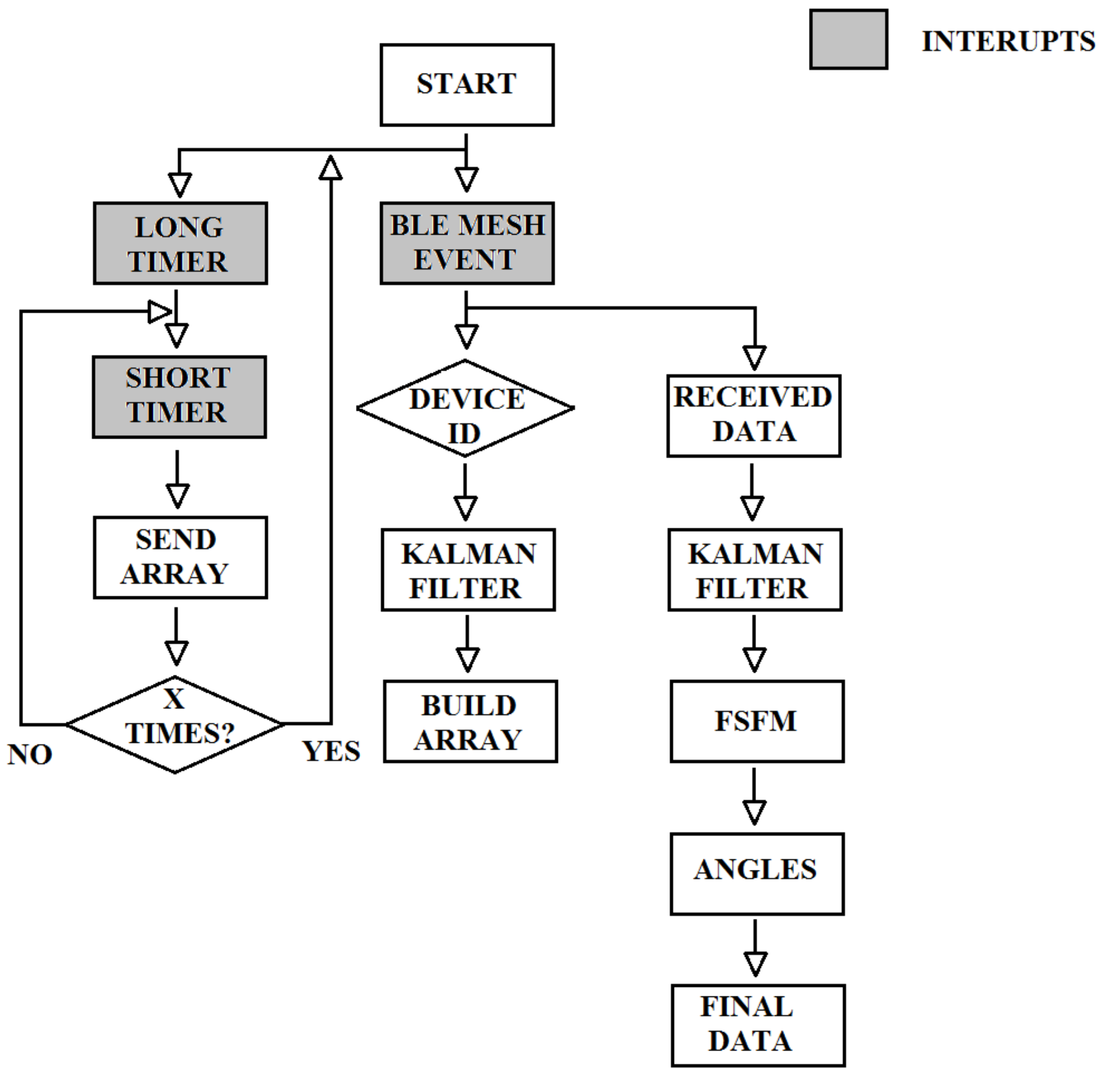
[15] Wikipedia (2016, April 22). Friis transmission equation [Online]. Available: https://en.wikipedia.org/wiki/Friis_transmission_equation#Modifications_to_the_basic_equation

[16] Wikipedia (2016, May 31) Joint Probabilistic Data Association Filter [Online]. Available: https://en.wikipedia.org/wiki/Joint_Probabilistic_Data_Association_Filter

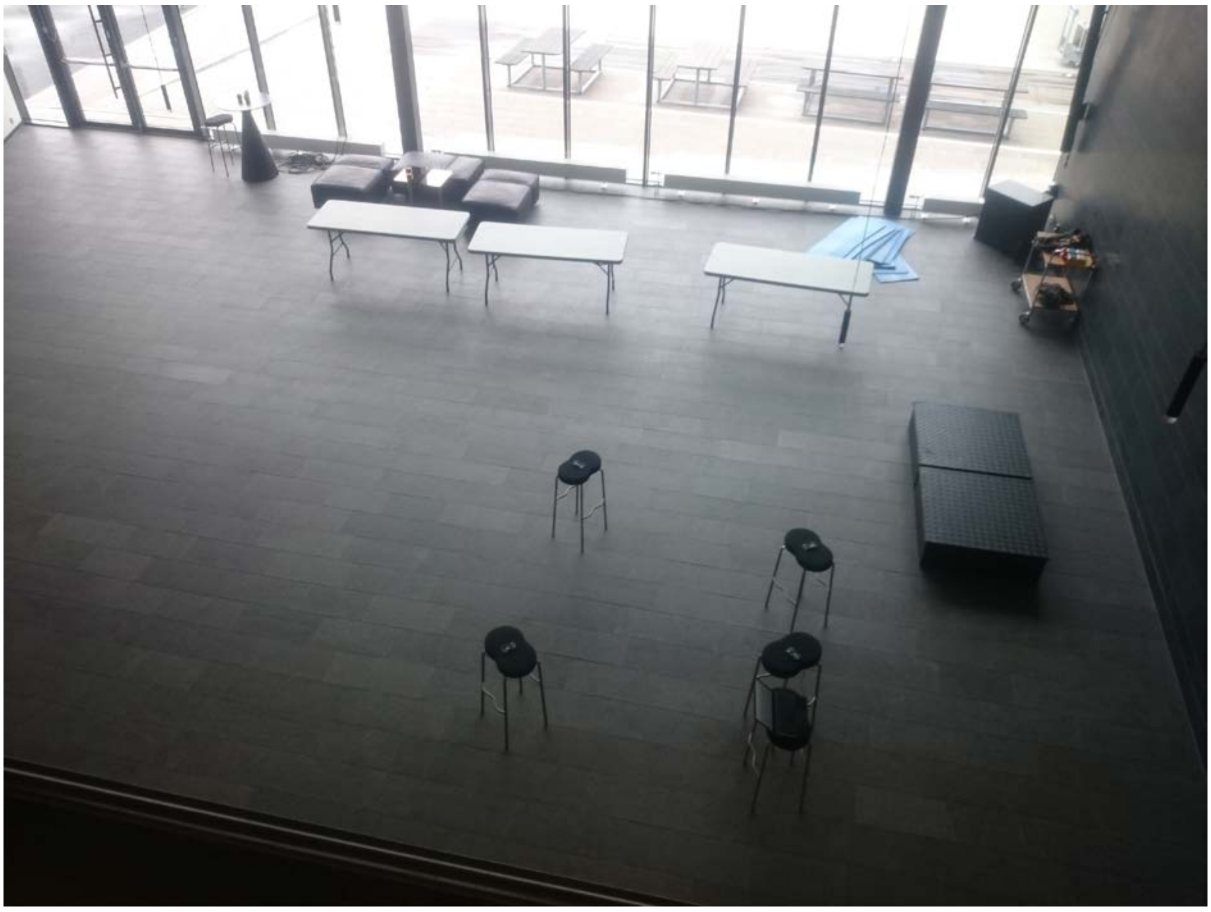
[17] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, "Introduction to Compressed Sensing," in Compressed Sensing: Theory and Applications, Cambridge University Press, 2012.

Appendices

Appendix 1 Flowchart



Appendix 2 Testing area


Testing area at Semcons lobby in Gothenburg

Appendix 3 First Layout

	x (AC) (m)	y (BC) (m)	z (BD) (m)	w (AD) (m)
Near	**1**	**1**	**1**	**1**
Measured near	1.74	0,73	1.50	0.97
Medium	**5**	**5**	**5**	**5**
Measured medium	6.3	7.63	7.33	8
Far	**10**	**10**	**10**	**10**
Measured far	8.73	9.27	10.57	10.47

Table appendices-1

Appendix 3.1 Near

	RSSI	Distance
x (AC)	-61.97	1.74
y (BC)	-60.67	1.50
z (BD)	-56.91	0.97
w (AD)	-54.45	0.73
AB	-57.71	1.07
CD	-59.76	1.35

Table appendices-2

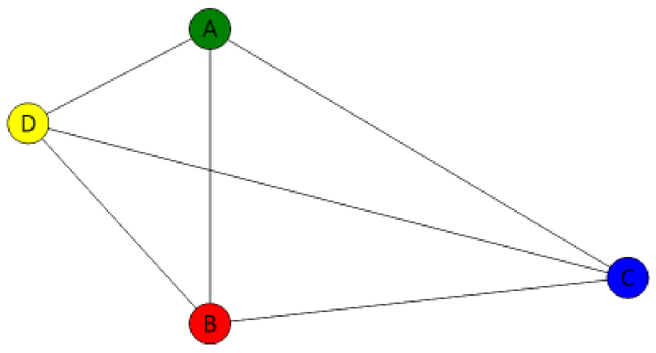


Figure appendices-1

Appendix 3.2 Medium

	RSSI (dBm)	Distance (m)
x (AC)	-62.69	6.30
y (BC)	-64.04	7.37
z (BD)	-64.77	8.00
w (AD)	-64.32	7.63
AB	-63.48	6.90
CD	-64.27	7.57

Table appendices -3

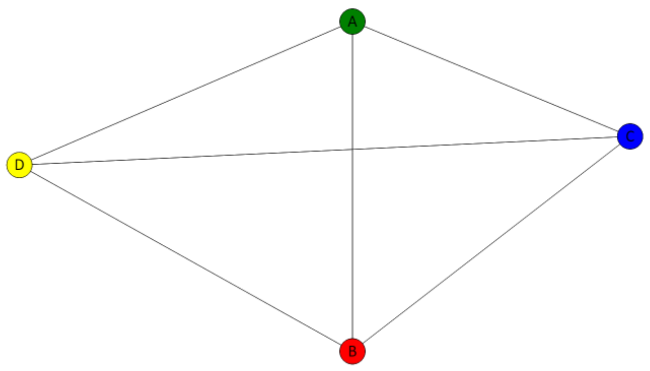


Figure appendices-2

Appendix 3.3 Far

	RSSI (dBm)	Distance (m)
x (AC)	-65.53	8.73
y (CB)	-67.16	10.67
z (BD)	-67.08	10.47
w (AD)	-66.03	9.27
AB	-66.68	10
CD	-66.87	10.2

Table appendices -4

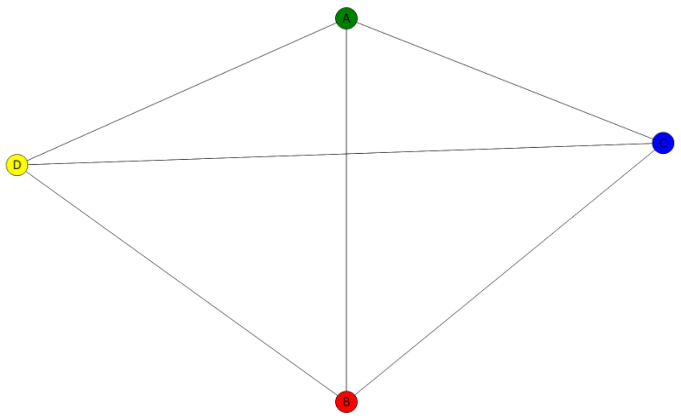


Figure appendices-3

Appendix 4 Second layout

	x (AC) (m)	y (BC) (m)	z (BD) (m)	w (AD) (m)
Near	**2**	**1**	**2**	**1**
Measured near	1.06	0.85	0.77	0.84
Medium	**5**	**2,5**	**5**	**2,5**
Measured medium	4.83	5.67	5.67	4.83
Far	**10**	**5**	**10**	**5**
Measured far	10.3	9.90	.79	6.30

Table appendices -5

Appendix 4.1 Near

	RSSI (dBm)	Distance (m)
x (AC)	-57.76	1.06
y (CB)	-55.70	0.85
z (BD)	-54.96	0.77
w (AD)	-55.65	0.84
AB	-53.76	0.99
CD	-57.11	0.68

Table appendices -6

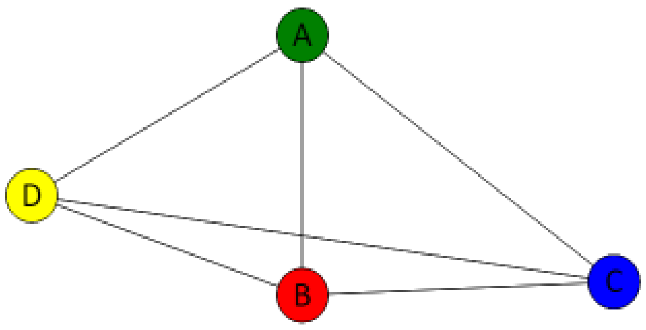


Figure appendices -4

Appendix 4.2 Medium

	RSSI (dBm)	Distance (m)
x (AC)	-60.38	4.83
y (CB)	-61.74	5.67
z (BD)	-61.74	5.67
w (AD)	-60.38	4.83
AB	-60.11	4.70
CD	-62.27	6.00

Table appendices-7

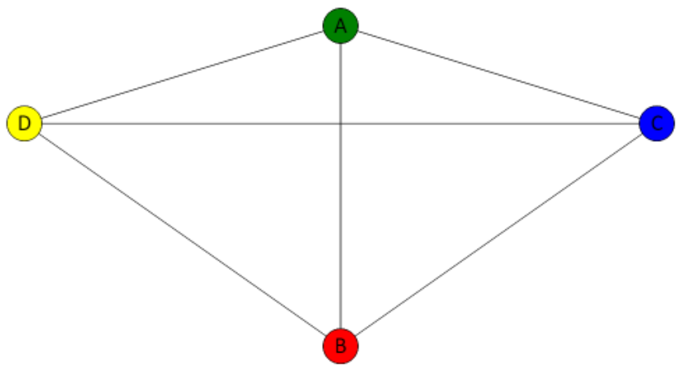


Figure appendices -5

Appendix 4.3 Far

	RSSI (dBm)	Distance (m)
x (AC)	-66.95	10.3
y (CB)	-66.59	9.90
z (BD)	-64.66	7.90
w (AD)	-62.67	6.30
AB	-66.68	7.80
CD	-65.45	8.67

Table appendices -8

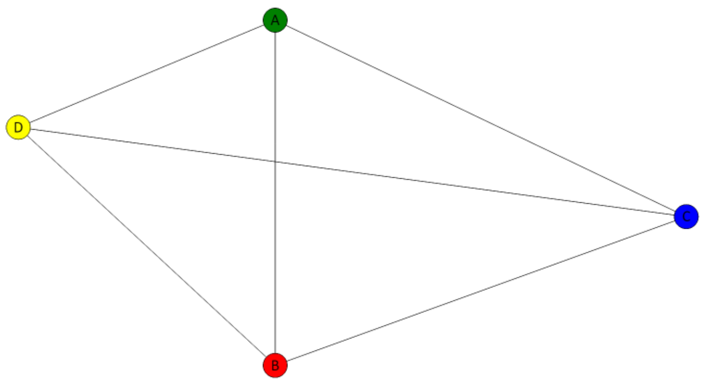


Figure appendices -6

Appendix 5 Third layout

	x (AC) (m)	y (BC) (m)	z (BD) (m)	w (AD) (m)
Near	**1**	**2,5**	**3**	**1**
Measured near	0.97	1.13	0.87	0.60
Medium	**2,5**	**6**	**7,5**	**2,5**
Measured medium	6.97	6.97	6.63	6.83
Far	**5**	**12,5**	**15**	**5**
Measured far	8.97	12.3	9.77	5.90

Table appendices-9

Appendix 5.1 Near

	RSSI (dBm)	Distance (m)
x (AC)	-56.88	0.97
y (CB)	-58.18	1.13
z (BD)	-55.96	0.87
w (AD)	-52.77	0.60
AB	-57.60	1.05
CD	-55.88	.86

Table appendices-10

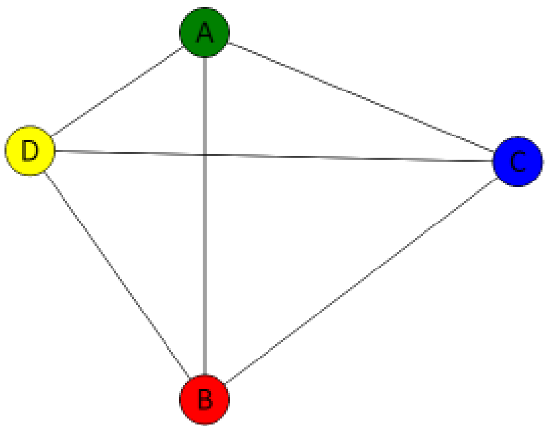


Figure appendices-7

Appendix 5.2 Medium

	RSSI (dBm)	Distance (m)
x (AC)	-63.53	6.97
y (CB)	-63.54	6.97
z (BD)	-63.11	6.63
w (AD)	-63.39	6.83
AB	-60.39	4.83
CD	-65.73	8.97

Table appendices-11

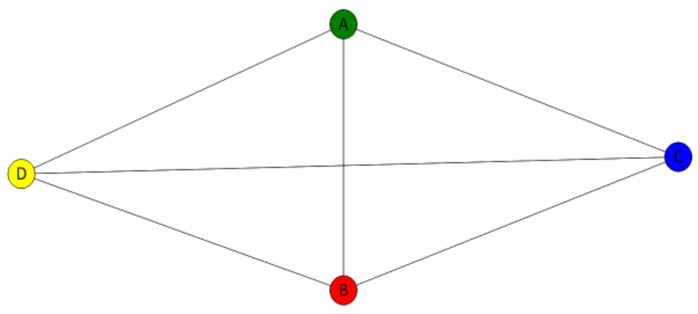


Figure appendices-8

Appendix 5.3 Far

	RSSI (dBm)	Distance (m)
x (AC)	-65.73	8.97
y (CB)	-68.51	12.33
z (BD)	-66.49	9.77
w (AD)	-62.12	5.9
AB	-68.79	12.73
CD	-65.05	8.27

Table appendices -12

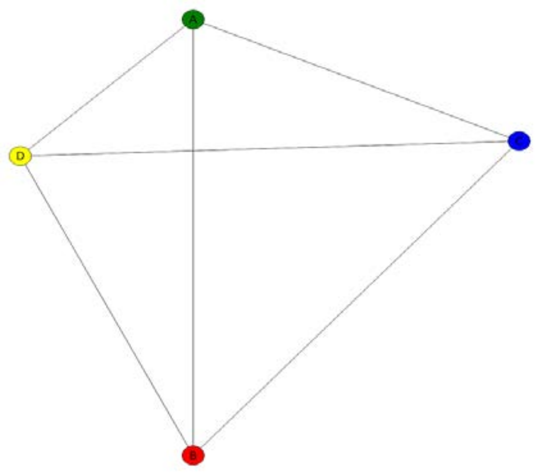


Figure appendices-9

Appendix 6 Fourth layout

	x (AC) (m)	y (BC) (m)	z (BD) (m)	w (AD) (m)
Near	**2**	**2**	**2**	**1**
Measured near	1.84	1.81	1.69	1.57
Medium	**5**	**5**	**5**	**2.5**
Measured medium	4.57	4.77	4.23	2.47
Far	**10**	**10**	**10**	**5**
Measured far	8.8	11.27	11.50	9.03

Table appendices-13

Appendix 6.1 Near

	RSSI (dBm)	Distance (m)
x (AC)	-62.43	1.84
y (CB)	-62.29	1.81
z (BD)	-61.72	1.69
w (AD)	-61.04	1.57
AB	-61.31	1.62
CD	-62.36	1.82

Table appendices-14

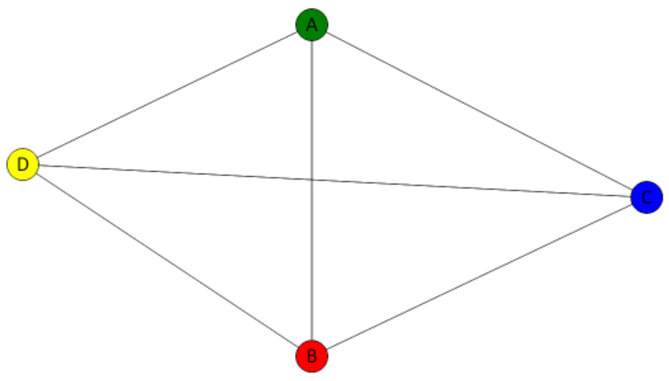


Figure appendices-10

Appendix 6.2 Medium

	RSSI (dBm)	Distance (m)
x (AC)	-59.87	4.57
y (CB)	-60.27	4.77
z (BD)	-59.25	4.23
w (AD)	-54.5	2.47
AB	-59.11	4.17
CD	-59.11	4.17

Table appendices-15

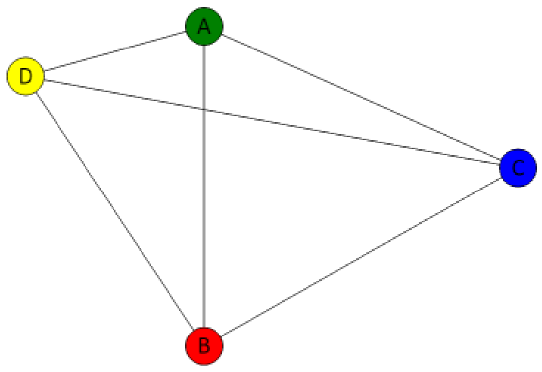


Figure appendices-11

Appendix 6.3 Far

	RSSI (dBm)	Distance (m)
x (AC)	-65.57	8.80
y (CB)	-67.73	11.27
z (BD)	-67.08	11.50
w (AD)	-67.91	9.03
AB	-67.73	11.5
CD	-67.08	10.43

Table appendices-16

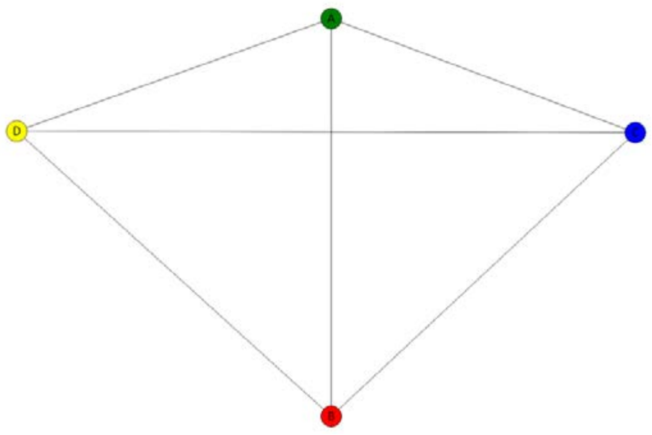


Figure appendices-12

APPENDIX 7 Code

```c
/*******************************************************************************
Copyright (c) Nordic Semiconductor ASA
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this
list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.

3. Neither the name of Nordic Semiconductor ASA nor the names of other
contributors to this software may be used to endorse or promote products
derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*******************************************************************************/

#include "rbc_mesh.h"
#include "nrf_adv_conn.h"
#include "timeslot_handler.h"
#include "app_timer_appsh.h"

#include "app_uart.h"
#include "softdevice_handler.h"
#include "app_error.h"
#include "nrf_gpio.h"
#include "nrf_drv_gpiote.h"
#include "SEGGER_RTT.h"
#include "boards.h"
#include "bsp.h"
#include "app_timer.h"
#include "app_timer_appsh.h"
#include "nrf_drv_clock.h"
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include <math.h>

#define PI    (3.141592653589793)
#define LIGHT (299792458)
#define FREQ  (2426000000) //BLE channel 38

//BLE transmitted power in dBm, 4 dBm max
#define TxPower 4

/**What device this unit is: A, B C or D*/
char device_id ='D';

//Kalman specs:
float err_meas=4.0;
int counterMax=10;

//timer specs
#define APP_TIMER_PRESCALER 15
#define APP_TIMER_MAX_TIMERS 8
#define APP_TIMER_OP_QUEUE_SIZE APP_TIMER_MAX_TIMERS

//to minimize Conflicting values, prime numbers are used for the timeouts
int low_prime_list[4]={1799,1877,1931,2067}; // prime number for the short timeout
int big_prime_list[4]={24999,25993,26591,26719}; // prime numbers for the long timeout
int short_counter=0;

```

```

/** Log transport medium flag. Set to 0 to use UART, 1 to use RTT */
#define LOG_RTT                   (0)

/** Logging predefines. Hides UART/RTT functions */
#if LOG_RTT

#define _LOG(str, ...)           SEGGER_RTT_printf(0, str, ##__VA_ARGS__)

#define _LOG_BUFFER(buf, len)    do{ \
    char hdr[] = {0, len};\
    SEGGER_RTT_Write(0, hdr, 2); \
    SEGGER_RTT_Write(0, (char*) buf, len); \
  } while (0)

#else

#define _LOG(str, ...) do{\
    char tx_str[128];\
    sprintf(tx_str, str, ##__VA_ARGS__);\
    char* c = tx_str;\
    while (*c)\
      app_uart_put(*(c++));\
  } while(0)

#define _LOG_BUFFER(buf, len) do{\
    uint8_t* c = buf;\
    uint8_t length = len;\
    while (length--)\
      app_uart_put(*(c++));\
    app_uart_put('\n');\
    app_uart_put('\r');\
  } while(0)

#endif

#define MESH_ACCESS_ADDR         (0xA541A68F)
#define MESH_INTERVAL_MIN_MS     (100)
#define MESH_CHANNEL             (38)
#define MESH_CLOCK_SOURCE        (NRF_CLOCK_LFCLKSRC_XTAL_75_PPM)
float KalmanArrayInit(uint8_t recieved_data, float rec_rssi);
int KalmanArrayLocation(uint8_t recieved_id, uint8_t transmitted_id);
float * kalman_calc(float RSSIvalue,float preEst,float errPreEst,float errPreMeas);
float * CallKalman(float rec_rssi,float prev_est,float err_est);
float * get_angles(float sideXdbm, float sideYdbm, float sideZdbm);
float * trig_function(float sideX, float sideY, float sideZ);
//float * FSFM_dist(float sig_str);

//timer id's:
static app_timer_id_t   timer_id_short,
                      timer_id_long;


typedef enum
{
  SCALING_CMD_TX = 'U',
  SCALING_CMD_DISABLE = 'D'
} scaling_cmd_t;

/** @brief parse a number from a string (similar to std::atoi) */
static uint8_t get_num(uint8_t** buf, uint32_t* len)
{
  uint8_t num = 0;
  uint8_t* p = *buf;

  while (*p >= '0' && *p <= '9')
  {
    num *= 10;
    num += *p - '0';
    p++;
    (*len)--;
  }
  *buf = p;
  return num;
}

/** @brief Parse an incoming set command */
```

```c
static void parse_set(uint8_t* buf, uint32_t* len, uint8_t* handle, uint8_t** payload)
{
  *handle = 0;
  uint8_t* p = buf;

  while (*p == ' ')
  {
    (*len)--;
    p++;
  }
  *handle = get_num(&p, len);
  while (*p == ' ')
  {
    (*len)--;
    p++;
  }
  (*len)--;
  *payload = p;
}

/**
* @brief Handle an incoming command, and act accordingly.
*/
static void cmd_rx(uint8_t* cmd, uint32_t len)
{
  if (len < 2)
    return;

  uint8_t* payload;
  uint8_t handle = 0;
  len--;
  switch ((scaling_cmd_t) cmd[0])
  {
    case SCALING_CMD_TX:
    /* got a new value */
      parse_set(&cmd[1], &len, &handle, &payload);

      if (rbc_mesh_value_set(handle, payload, len) == NRF_SUCCESS)
      {
        //_LOG("V[%d] %s\tL:%d\r\n", (int)handle, payload, (int)len);
      }

      break;
    case SCALING_CMD_DISABLE:
      rbc_mesh_value_disable(cmd[1]);
      _LOG("D[%d]\r\n", (int)cmd[1]);
      break;
  }
}
/**
* @brief Handle incoming character on the control channel (UART or RTT).
*   Holds text until a \r or \n has been received, upon which it calls cmd_rx()
*/
static void char_rx(uint8_t c)
{
  static uint8_t rx_buf[32];

  static uint8_t* pp = rx_buf;


  if (c != '\r' && c != '\n')
  {
    *(pp++) = c;
  }
  else
  {
    *(pp++) = 0;
  }
  uint32_t len = (uint32_t)(pp - rx_buf);

  if (len >= sizeof(rx_buf) || c == '\r' || c == '\n') /* end of command */
  {
    if (len > 0)
      cmd_rx(rx_buf, len);

    //_LOG("%s",rx_buf);
    pp = rx_buf;
  }
}


/**
* @brief General error handler.
*/
static void error_loop(void)
{
  NVIC_SystemReset(); /* reset the system.  */
}

/**
* @brief Softdevice crash handler, never returns
*
* @param[in] pc Program counter at which the assert failed
* @param[in] line_num Line where the error check failed
* @param[in] p_file_name File where the error check failed
*/
void sd_assert_handler(uint32_t pc, uint16_t line_num, const uint8_t* p_file_name)
{
  error_loop();
}

/**
* @brief App error handle callback. Called whenever an APP_ERROR_CHECK() fails.
*   Never returns.
*
* @param[in] error_code The error code sent to APP_ERROR_CHECK()
* @param[in] line_num Line where the error check failed
* @param[in] p_file_name File where the error check failed
*/
void app_error_handler(uint32_t error_code, uint32_t line_num, const uint8_t * p_file_name)
{
  error_loop();
}

void HardFault_Handler(void)
{
  error_loop();
}

/**
* @brief RBC_MESH framework event handler. Defined in rbc_mesh.h. Handles
*   events coming from the mesh. Propagates the event to the host via UART or RTT.
*
* @param[in] evt RBC event propagated from framework
*/
void rbc_mesh_event_handler(rbc_mesh_event_t* evt)
{
  char cmd = 'x';
  switch (evt->event_type)
  {
    case RBC_MESH_EVENT_TYPE_CONFLICTING_VAL: cmd = 'C'; break;
    case RBC_MESH_EVENT_TYPE_INITIALIZED: cmd = 'I'; break;
    case RBC_MESH_EVENT_TYPE_UPDATE_VAL: cmd = 'U'; break;
    case RBC_MESH_EVENT_TYPE_NEW_VAL: cmd = 'N'; break;
    case RBC_MESH_EVENT_TYPE_TX: cmd = 'T'; break;
  }
  _LOG("%c[%d][RSSI: %d] ", cmd, evt->value_handle, evt->rssi);
  _LOG_BUFFER(evt->data, evt->data_len);
}

/** Handler for incoming UART events */
void uart_event_handler(app_uart_evt_t * p_app_uart_event)
{
  uint8_t c;
  switch (p_app_uart_event->evt_type)
  {
    case APP_UART_DATA_READY:
      while (app_uart_get(&c) == NRF_SUCCESS)
      {
        /* log single character */
        char_rx(c);
      }
      break;
    default:
      break;
  }
}

  //perform kalman calculation
  //use pointers when calling this function
  //i.e int *X=kalman_calc(...);
  //will point to first place in return_array
  float * kalman_calc(float RSSI,float preEst,float errPreEst,float errPreMeas)
  {
    float KG,kalEst, errEst;
    static float return_array[2];
    KG = errPreEst/(errPreEst+errPreMeas); //Kalman gain
    kalEst=preEst+KG*(RSSI-preEst);     //current estimate
    errEst=(1-KG)*errPreEst;
    //returns both the kalman estiamte and the
    //new error estiamte to be used next time.
    return_array[0]=kalEst;
    return_array[1]=errEst;
    return (return_array);
  }

  //function to set a number of values and then call the actual Kalman function
  float * CallKalman(float rec_rssi,float prev_est,float err_est)
  {
    static float *kalman;
    if(prev_est==0)
    {
      prev_est=rec_rssi; //Sets the starting value to the same as the first RSSI
    }
    kalman=kalman_calc(rec_rssi,prev_est,err_est,err_meas);
    return kalman;
  }

  //Function to set the starting values to some arrays
  float *start_values()
  {
    static float first_time[4];
    first_time[0]=0;
    first_time[1]=1;
    first_time[2]=1;
    first_time[3]=0;
    return first_time;
  }

    //Function to find out what place the eventual Kalman values should be set to
    int KalmanArrayLocation(uint8_t recieved_id, uint8_t transmitted_id)
    {
      int place=8;
      //This device ID multiplied with the recieved ID.
      int position=recieved_id*transmitted_id;
      int test[6]={4290,4355,4420,4422,4488,4556};
      for(int i=0;i<=5;i++)
      {
        if(position==test[i])
        {
          return place=i+2;
        }
      }
      return place;
    }


    // Checks which values that should be a triangle
    bool whaTriangle(float *distArray)
    {
      static float finalTriangles[12];
      float *triangle;
      int plats[12]={2,3,5,2,4,6,3,4,7,5,6,7};
      for(int i=0;i<=11;i=i+3)
      {
        //checks if values for a triangle is present.
        if((distArray[plats[i]]!='#')&&(distArray[plats[i+1]]!='#')&&(distArray[plats[i+2]]!='#'))
        {
          //function is called to calulate the angles
          triangle=get_angles(distArray[plats[i]],distArray[plats[i+1]],distArray[plats[i+2]]);
          //Angles is set to the final triangle array
          finalTriangles[i]=triangle[0];
          finalTriangles[i+1]=triangle[1];
          finalTriangles[i+2]=triangle[2];
        }
      }
    }

    //Function to assemble all the data into one matrix
    void finalDataFunc(uint8_t *kalArr,float *distArr, float *triArr)
    {
      //This function takes all the data collected during runtime.
      int plats[12]={2,3,5,2,4,6,3,4,7,5,6,7};
      int start[5]={0,3,6,9,12};
      float finalData[4][9];
      for(int i=0;i<=3;i++)
      {
        int k=0;
        for(int j=start[i];j<=(start[i+1]-1);j++)
        {
          finalData[i][k]=kalArr[plats[j]];
          finalData[i][k+3]=distArr[plats[j]];
          finalData[i][k+6]=triArr[j];
          k++;
        }
      }
    }


    //arguments are signals strengths in dbm
    //will return angles in radians
    float *get_angles(float X, float Y, float Z)
    {
      float X2,Y2,Z2;
      static float trig_array[3];

      //angle_array is in radians
      X2=pow(X,2);
      Y2=pow(Y,2);
      Z2=pow(Z,2);

      trig_array[0]=((acos((X2+Z2-Y2)/(2.0*X*Z)))*(180/PI));
      trig_array[1] =((acos((Y2+Z2-X2)/(2.0*Y*Z)))*(180/PI));
      trig_array[2]=180-trig_array[0]-trig_array[1];
      return trig_array;
    }



    //Free Space Friis Model:
    float *FSFM_dist(float sig_str)
    {
      const int CableLoss=1;  // 0.5dBm for Rx and Tx = 1dBm
      sig_str=-sig_str;       //give the RSSI negative value again
      double dist_cons=20*log10(((4.0*PI)/LIGHT))+20*log10(FREQ); //
      float FSFM=TxPower-sig_str-CableLoss;   //FSFM in dB, Antenna Gain is 0dB

      static float distance;
      distance=FSFM-dist_cons;        //20log(d)
      distance=pow(10,distance/20.0); //distance in meters
      distance=distance/10.0;         //Calibration
      return &distance;
    }


    //Function to call Kalman with different values
    float *whichKalman(int8_t rssi, float *nod, int8_t id)
    {
      int counter=nod[2];
      if(counter>=counterMax)
      {
        //resets the counter
        counter=1;
        nod[3]=1;
      }
      if(nod[2] < counterMax)
      {
        nod=CallKalman(rssi,nod[0],nod[1]);
```

```
        counter++;
      }
      nod[2]=counter;
      return(nod);
    }

    //Long timer
    static void timer_timeout_handler_long(void * p_context)
    {
      app_timer_start(timer_id_short,low_prime_list[device_id-65], NULL);
    }
    //Short timer
    static void timer_timeout_handler_short(void * p_context)
    {
      short_counter++;
    }
    //Initializes the timer module.
    static void timers_init(void)
    {
      uint32_t err_code;

      // Initialize timer module, making it use the scheduler
      APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_MAX_TIMERS, APP_TIMER_OP_QUEUE_SIZE, NULL);

      // long timer:
      err_code = app_timer_create(&timer_id_long, APP_TIMER_MODE_REPEATED, 
timer_timeout_handler_long);
      APP_ERROR_CHECK(err_code);
      // short timer:
      err_code = app_timer_create(&timer_id_short, APP_TIMER_MODE_REPEATED, 
timer_timeout_handler_short);
      APP_ERROR_CHECK(err_code);
    }



int main(void)
{

  /* Enable Softdevice (including sd_ble before framework) */
  SOFTDEVICE_HANDLER_INIT(MESH_CLOCK_SOURCE, NULL);
  softdevice_ble_evt_handler_set(rbc_mesh_ble_evt_handler);
  softdevice_sys_evt_handler_set(rbc_mesh_sd_evt_handler);

  /* Init the rbc_mesh */
  rbc_mesh_init_params_t init_params;

  init_params.access_addr = MESH_ACCESS_ADDR;
  init_params.interval_min_ms = MESH_INTERVAL_MIN_MS;
  init_params.channel = MESH_CHANNEL;
  init_params.lfclksrc = MESH_CLOCK_SOURCE;

  uint32_t error_code = rbc_mesh_init(init_params);
  APP_ERROR_CHECK(error_code);

  nrf_gpio_range_cfg_output(0, 32);

  LEDS_CONFIGURE(LEDS_MASK);

#if LOG_RTT
  SEGGER_RTT_ConfigUpBuffer(0, NULL, NULL, 0, SEGGER_RTT_MODE_NO_BLOCK_SKIP);
#else
  app_uart_comm_params_t uart_params;
  uart_params.baud_rate = UART_BAUDRATE_BAUDRATE_Baud115200;
  uart_params.cts_pin_no = CTS_PIN_NUMBER;
  uart_params.rts_pin_no = RTS_PIN_NUMBER;
  uart_params.rx_pin_no = RX_PIN_NUMBER;
  uart_params.tx_pin_no = TX_PIN_NUMBER;
  uart_params.flow_control = APP_UART_FLOW_CONTROL_ENABLED;
  uart_params.use_parity = false;
  APP_UART_FIFO_INIT(&uart_params, 8, 256, uart_event_handler, APP_IRQ_PRIORITY_LOW, error_code);
  APP_ERROR_CHECK(error_code);
#endif

  //_LOG("START\r\n");


```

```
#if LOG_RTT
  rbc_mesh_event_t evt;
  while (true)
  {
      int8_t c = SEGGER_RTT_GetKey();
      if (c >= 0)
          char_rx(c);

      if (rbc_mesh_event_get(&evt) == NRF_SUCCESS)
      {
          rbc_mesh_event_handler(&evt);
      }

      sd_app_evt_wait();
  }
#else

      //Variebles used in this part of main
      sd_ble_gap_tx_power_set(TxPower);
      rbc_mesh_event_t evt;
      //initialize timers
      timers_init();
      bool newKalman=false;
      //These nod pointers holds all the information for each ID that is present in the mesh
      float *nodA;
      nodA=start_values();
      float *nodB;
      nodB=start_values();
      float *nodC;
      nodC=start_values();
      float *nodD;
      nodD=start_values();
      //This is the kalmanarray that is sent around in the mesh network
      uint8_t kalmanArray[8]={'U',device_id,'#','#','#','#','#','#'};
      //To make it easier to know which place is where the distance array is built in the same way
      float dist[8]={'U',device_id,'#','#','#','#','#','#'};
      float *final_triangles;
      //starting values to the logging part of the program
      float allOtherKal[2][6]={{50,50,50,50,50,50},{1,1,1,1,1,1}};
      int j=0;
      app_timer_start(timer_id_long,big_prime_list[device_id-65], NULL);


  while (true)
  {

    if(short_counter>j)
    {
      cmd_rx(kalmanArray,sizeof(kalmanArray)+1);
      if (j==15)
      {
        short_counter=0;
        app_timer_stop(timer_id_short);
      }
      j=short_counter;
      sd_app_evt_wait();
    }

    if (rbc_mesh_event_get(&evt) == NRF_SUCCESS)
    {
      //rbc_mesh_event_handler(&evt);
      rbc_mesh_packet_release(evt.data);
      //Arraykod
      int8_t rec_rssi=evt.rssi;
      uint8_t *rec_data=evt.data;
      if(rec_data[0]=='Z')//reset kalman calculation
      {
        nodA[1]=1;
        nodB[1]=1;
        nodC[1]=1;
        nodD[1]=1;
      }
      if(rec_data[0]>='A'||rec_data[0]<='D')    //Check if the recieved package contains the Kalman array
      {
        switch(rec_data[0])
        {
          case 'A':
            if (device_id=='A')
              break;
            int platsA=KalmanArrayLocation(rec_data[0],device_id);
            nodA=whichKalman(rec_rssi,nodA,device_id);
            if(nodA[3]==1)
            {
              kalmanArray[platsA]=nodA[0];
              newKalman=true;
              nodA[3]=0;
            }
            break;
          case 'B':
            if (device_id=='B')
              break;
            int platsB=KalmanArrayLocation(rec_data[0],device_id);
            nodB=whichKalman(rec_rssi,nodB,device_id);
            if(nodB[3]==1)
            {
              kalmanArray[platsB]=nodB[0];
              newKalman=true;
              nodB[3]=0;
            }
            break;
          case 'C':
            if (device_id=='C')
              break;
            int platsC=KalmanArrayLocation(rec_data[0],device_id);
            nodC=whichKalman(rec_rssi,nodC,device_id);
            if(nodC[3]==1)
            {
              kalmanArray[platsC]=nodC[0];
              newKalman=true;
              nodC[3]=0;
            }
            break;
          case 'D' :
            if (device_id=='D')
              break;
            int platsD=KalmanArrayLocation(rec_data[0],device_id);
            nodD=whichKalman(rec_rssi,nodD,device_id);
            if(nodD[3]==1)
            {
              kalmanArray[platsD]=nodD[0];
              newKalman=true;
              nodD[3]=0;
            }
            break;
          }
          if(newKalman==true)
          {
            for(int g=0;g<=5;g++)
          {
          if(rec_data[g+1]!=35)
          {
            float errPreEst=allOtherKal[1][g];
            float preEst =allOtherKal[0][g];
            float kalGain = errPreEst/(errPreEst+err_meas); //Kalman gain
            allOtherKal[0][g]=preEst+kalGain*(rec_data[g+1]-preEst);    //current estimate
            allOtherKal[1][g]=(1-kalGain)*errPreEst;
          }
        }
        int plats=KalmanArrayLocation(rec_data[0],device_id);
        //FSFM distance conversion
        //for(int i=2;i<=8;i++)
        //{
        //  if(kalmanArray[i]!='#')
        //  dist[i]=*FSFM_dist(kalmanArray[i]);
        //}
        //finalDataFunc(kalmanArray,dist,final_triangles);
        _LOG("demoRaw = '[{\"AB\":\"%.4f\", \"AC\":\"%.4f\", \"AD\":\"%.4f\", \"BC\":\"%.4f\",
\"BD\":\"%.4f\",
\"CD\":\"%.4f\"}]';\r\n",allOtherKal[0][0],allOtherKal[0][1],allOtherKal[0][2],allOtherKal[0][3],allOt
herKal[0][4],allOtherKal[0][5]);
        newKalman=false;
        }
      }
```

```
        sd_app_evt_wait();
        }
    }
#endif
}
```