NTNU

Norwegian University of
Science and Technology

# Solution for testing and evaluation of asset tracking for construction sites

Karl Jørgen Svantorp

Master of Science in Electronics
Submission date: June 2018
Supervisor: Torbjørn Ekman, IES

Norwegian University of Science and Technology
Department of Electronic Systems

# Abstract

The goal of this project has been to make a system for asset tracking at construction sites. Among many outdoor positioning systems, Global Navigation Satellite System (GNSS) is a common choice. However, GNSS systems are vulnerable to multipath effects, and at places where the satellites appear low on the horizon, both unavailability and bias is prevalent. To counter this, a local outdoor positioning system using Ultra Wideband (UWB) is proposed, which uses stationary anchor points determined with Global Positioning System (GPS) to place the assets in a global context. A device is designed which functions both as a comparison test bench for GPS and UWB accuracy and power consumption, and as either anchor or tag in the proposed local positioning system. The results indicate that the proposed system have an accuracy comparable to consumer grade GPS receivers when using on board GPS to position the anchor nodes, at up to 5m horizontal error. This error decreases to maximum 1.3m when accurate anchor points are used. The power consumption is however higher than low power GPS receivers. This project was carried out during the spring semester 2018 at NTNU, in collaboration with Aventi Intelligent Communication AS, from this point referred to as AIC.

# Problem description

Develop a system and devices for the evaluation, comparison and testing of commercially available Ultra Wideband positioning systems and GNSS systems for asset tracking on construction sites.

# Sammendrag

Målet med dette prosjektet har vært å lage et system for sporing av gjenstander på byggeplasser. Blandt mange utendørs posisjoneringssystemer, er Global Navigation Satellite System (GNSS) et vanlig valg. GNSS er derimot sårbart for flerveiseffekter, og på steder hvor satelitter ligger lavt på himmelen er både utilgjengelighet og skjevheter vanlig. For å motvirke dette presenteres det et lokalt utendørs posisjoneringssystem som bruker Ultra Bredbånd (UWB) og stasjonære ankerpunkter bestemt med Global Positioning System (GPS) for å plassere gjenstandene i en global kontekst. Det er designet en enhet som fungerer både som et sammenligningsgrunnlag for GPS og UWB nøyaktighet og effektforbruk, og som enten anker eller tag i det presenterte posisjoneringssystemet. Resultatene indikerer at systemet har en nøyaktighet som er sammenlignbar med rimelige GPS mottakere når anker nodene er posisjonert med en egen GPS, med en horisontal feil på opptil 5m. Denne feilen reduseres til maksimum 1.3m når nøyaktige ankerpunkter er benyttet. Effektforbruken er derimot høyere enn en lav-effekts GPS mottaker. Dette prosjektet ble gjennomført på vårsemestert 2018 på NTNU, i samarbeid med Aventi Intelligent Communication AS, heretter refert til som AIC.

# Table of Contents

# Abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| AoA | Angle of Arrival |
| API | Application Programming Interface |
| CEP | Circular Error Probable |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| LPWAN | Low Power Wide Area Network |
| LTE | Long Term Evolution |
| NB-IoT | Narrowband Internet Of Things |
| OFDM | Orthogonal Frequency Division Multiplexing |
| OTDOA | Observed Time Difference Of Arrival |
| QPSK | Quadrature Phase Shift Keying |
| PAPR | Peak-to-Average Power Ratio |
| PCB | Printed Circuit Board |
| PSM | Power Saving Mode |
| RMSE | Root Mean Square Error |
| RSSI | Received Signal Strength Indicator |
| RTK | Real-Time Kinematic |
| RTLS | Real Time Locating System |
| RTT | Round Trip Time |
| SDS-TWR | Symmetrical Double Sided Two Way Ranging |
| TDOA | Time Difference of Arrival |
| ToA | Time of Arrival |
| USB | Universal Serial Bus |
| UWB | Ultra Wideband |

Chapter 1

# Introduction

In industrial applications that rely on accurate and reliable positions, Real Time Location Systems (RTLS) [1] are viable solutions. RTLSs usually track mobile tags attached to assets using anchor nodes as fixed reference points, that exchange wireless signals to determine the tags' location. Common technologies on which RTLS is implemented today includes bluetooth, through the location and cell size of the closest anchor, and WiFi, through trilateration similar to the system implemented in this project, only with less bandwidth. The most accurate RTLS systems are however often based on UWB wireless signals. This allows for robust, short range and energy efficient signalling. The locations of anchor nodes in a RTLS system needs to be known in order to determine the tag location, and in this project GPS is used by the anchors to position themselves. RTLSs will usually collect measurements at a central station in the system that presents the locations to the user. In this project the anchor nodes send their measurements to a server using NarrowBand Internet of Things (NB-IoT), and the locations are both calculated and presented to the user at this server. The system is specified in chapter 3, along with the functional requirements and a description of operation for the user. Chapter 4 draws upon the specification and in greater detail explains why the respective technologies was chosen and discusses alternatives, thus outlining a design that can be implemented. The implementation of the system is described in chapter 5, where the specification serves as background for choices of hardware, circuit layout and code libraries. Simulations are presented in chapter 6, and show relevant relationships between accuracy of tag positioning and both anchor quantity and position accuracy. The results presented in chapter 7 show that the tags in the system has a positional accuracy comparable to consumer grade GPS in the horizontal plane

when using the anchor nodes' on-board GPS receiver. Chapter 8 presents possible further work and improvements are discussed, along with sources of error that can be improved.

The most novel part of this system is the anchor nodes' ability to position themselves, as systems available for purchase usually require the user to manually determine and register the position of anchor nodes during installation.

Chapter 2

# Theory

## 2.1 Positioning

One common element used in positioning in otherwise differing systems such as GPS, cellular networks or local UWB systems, are the anchors, or reference points. For GPS these reference points are satellites, in cellular networks they are base stations, and in UWB systems they are referred to as anchor points. To be able to find the position of a tracking device in these systems based on the known reference points, a few trigonometric properties can be used together with the knowledge of either distances through Time of Arrival (TOA) or Round Trip Time (RTT), Time Difference Of Arrival (TDOA) for distance differences, and Angle of Arrival (AoA) for angles. This chapter describes the relevant technologies and theoretical background for this project.

### 2.1.1 Trilateration

Trilateration is a method for determining the location of points by using distances from known points. A common example is the calculation of the intersection point of three circles in 2D plane, or four spheres for 3D space, with known location and radius. For the case where the location and radius of the circles are exactly known, and the circles intersect at an exact point, the solution is trivial. However, if the circles do not intersect at exactly one point, or either the location or radius of the circles are uncertain, the solution becomes an area with a distribution of probability, as shown in figure 2.1. Further, if there are more than N+1 known distances and points for N-D space, the problem is over-determined with no exact solution. It can however

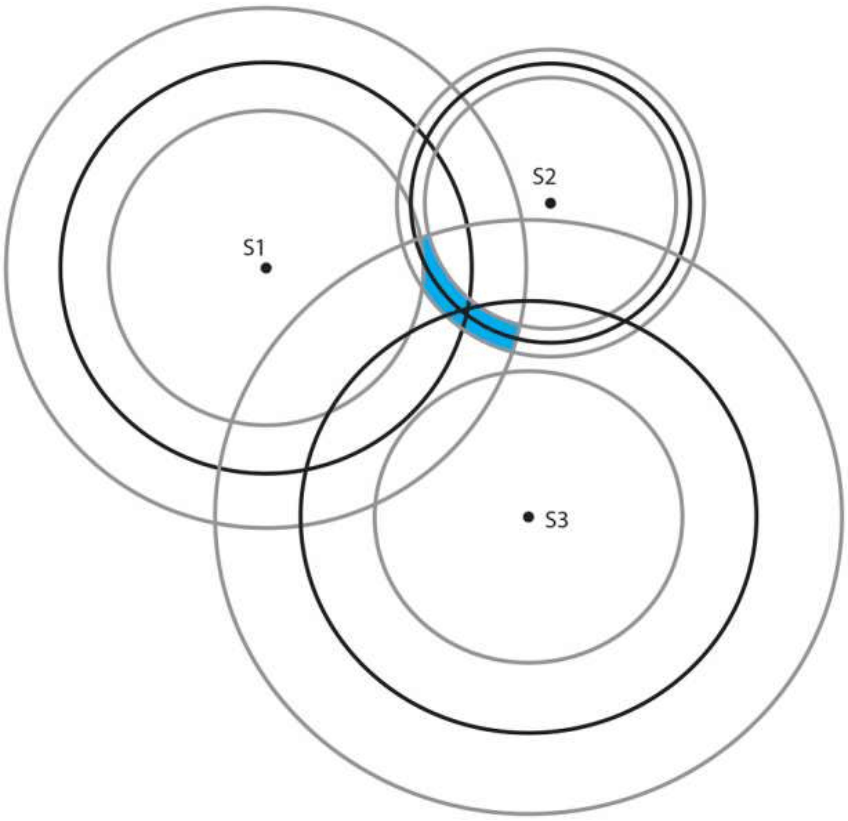


**Figure 2.1:** Trilateration with three circles from anchor points S1,2,3, each with a threshold of probability in grey, giving and area of most probable location in blue

be estimated with non-linear least squares algorithms. The two algorithms Gauss Newton and Levenberg Marquardt is evaluated in chapter 6, and the most successful is used in the final system to calculate the positions of tags. A non-linear least squares algorithm is necessary to be able use more than four anchors in the calculations. More anchors increases coverage and accuracy of positions, so the chosen algorithm plays an important part in the system.

### 2.1.2 GPS Positioning

GPS is a positioning and navigation system that uses estimated distances, or pseudoranges, to satellites to calculate positions on earth using trilateration. A GPS receiver will receive wireless signals from satellites on orbit around earth, and based on the time of arrival of these signals, along with knowledge of the satellites positions, the position of the receiver can be calculated. The error of this position is mainly due to the error sources listed in table 2.1.

For single frequency (C/A code) receivers the main source of error is usually the ionosphere, which can lead to an positional error on the order of four meters. This may or may not have

**GPS sources of error**

- Ephemeris data. Errors in the transmitted location of the satellite
- Satellite clock. Errors in the transmitted clock
- Ionosphere. Errors in the corrections of pseudorange caused by ionospheric effects
- Troposphere. Errors in the corrections of pseudorange caused by tropospheric effects
- Multipath. Errors caused by reflected signals entering the receiver antenna
- Receiver. Errors in the receiver's measurement of range caused by thermal noise, software accuracy, and interchannel biases

**Table 2.1:** Sources of error in GPS positions [15]

changed since 1996, when [15] was released. Low power consumer grade receivers are usually single frequency receivers. For dual frequency receivers, the effects of the ionosphere is reduced and the dominant sources of error become ephemeris data and satellite clock. For single frequency receivers a total horizontal error of 10.2 m (at $1\sigma$, 68% confidence level) can be expected. The GPS receiver chosen for this project, discussed in chapter 5, have a reported error of 2.5m Circular Error Probable (CEP) (50% confidence level).

See figure 2.2 for an illustration of common sources of error for GPS measurements.

### 2.1.3 UWB Positioning

The term Ultra Wideband usually refers to a radio technology that uses very high bandwidth for low-energy, short range communication. The bandwidth is typically 500 MHz or more, or above 20% of the center frequency. Since UWB is relatively short range, with most systems operating at ranges of tens to hundreds of meters between each tag and anchor, the system proposed here can be considered a local positioning system. This means that it needs reference points to be able to place all the points in a larger, or global, positioning system. This is usually done by placing anchors at known points in a larger coordinate system, and finding the tag location in the local system before translating that location to the larger coordinate system using the positions of the anchors.

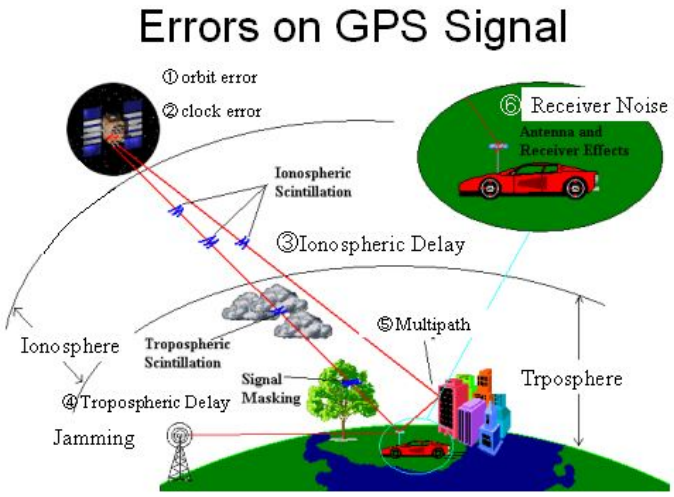


**Figure 2.2:** Visualization of sources of error in GPS measurements, from [19]

### UWB distance measurements

To use trilateration, you need to know the distances between nodes. In this project symmetrical double-sided two-way ranging (SDS-TWR) is used to find the distances, mainly because it alleviates the need for time synchronization and reduces the impact of clock drift. Time of flight is measured by RTT, $T_{roundA}$ in round A and $T_{roundB}$ in round B in figure 2.3. This means that the original transmitter is also the end receiver of the signal, and the same clock is then used for timestamping the transmission and reception. This makes time synchronization unnecessary[9], but the RTT includes a processing delay in the responding node, $t_{reply}$ in figure 2.3. It is therefore necessary for the originating node, Device A in round A and Device B in round B, to know the processing delay in the other node, in order to subtract this from the RTT. The processing is done in the physical layer, and for known hardware the processing delay can be measured and accounted for in development. For one RTT round, round A, the distance d between Device A and Device B is then given by expression 2.1.

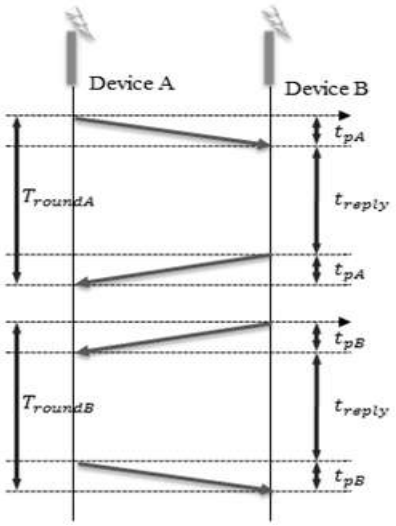


**Figure 2.3:** SDS-TWR signal exchange and timing

$$d = v \times (T_{roundA} - t_{reply})/2 \tag{2.1}$$

Where $v$ is the speed of light, $T_{roundA/B}$ is the RTT in each round and $t_{reply}$ is the processing delay. Still, using only one such round to find the distance would make the measurements suffer from clock drift because readily available hardware clocks have finite resolution. By measuring RTT over two rounds, with one round originating from each node, first order error from clock drift will be zeroed out [9]. After the two rounds, round A and B, the distance d is given by expression 2.2

$$d = v \times (T_{roundA} - t_{reply} + T_{roundB} - t_{reply})/4 \tag{2.2}$$

This expression shows how the distance is found between anchor nodes and tags in the proposed system in this project, and the mean of several such distances is used to trilaterate the position of the tags in the system through the implementation in chapter 5.

## 2.2 Narrow Band Internet of Things

NB-IoT is the new cellular network intended for Internet of Things devices and sensors and standardized by 3rd Generation Partnership Project (3GPP) in release 13. The small bandwidth of 180 kHz both extends the range and reduces the needed transmission power from the de-

vices on the network. NB-IoT is based directly on Long Term Evolution (LTE) and can be deployed 'in-band', using free resource blocks of a normal LTE signal. Like LTE, NB-IoT uses Orthogonal Frequency Division Multiplexing (OFDM), a multiplexing scheme known to be quite power-ineffective, due to the high Peak-to-Average Power Ratio (PAPR) [14]. In practical terms, this leads to a higher power consumption than a conventional coin cell battery can provide for. Lithium-ion batteries are the best replacement that both power relatively high current loads and keeps size to a minimum. Prior to this project, NB-IoT was considered for a positioning system with the IoT network as the primary positioning service. 3GPP release 14 introduces Observed Time Difference Of Arrival (OTDOA) for NB-IoT, which has been used as a positioning technique over cellular networks for some years, but then only for LTE. Using NB-IoT, the positioning system would have been very different from the system proposed in this project, with already deployed cellular towers acting as anchor nodes, potentially allowing for less energy usage and great coverage for devices, at the cost of accuracy. Unfortunately, the telecom operators in Norway need more time to implement OTDOA for NB-IoT, and are not ready to deliver this service in time for this project. The accuracy of NB-IoT OTDOA positioning is defined in [2] in terms of the timing unit $T_s$ which is defined as $\frac{1}{15000*2040} \approx 33ns$. The minimum timing accuracy is defined as $\pm 20T_s$ which leads to a worst case positioning uncertainty of

$$3 \cdot 10^8 m/s \cdot \frac{2 * 20}{15000 * 2048} s \approx 391m$$

This relatively low accuracy show that a NB-IoT OTDOA positioning system is not suited for a local system at construction sites.

## 2.3 Plane Earth Model

The plane earth path loss model, also commonly referred to as the two-ray ground reflection model, is a common model to use when calculating the path loss for line of sight, multipath communication systems. It assumes a plane, perfectly reflective earth, and only two paths for the signals to travel between the two communicating points. Figure 2.4 shows the geometry and paths assumed with the plane earth model. The shortest path is naturally the line of sight, and the wave reflected of the ground is slightly longer, causing a phase shift that can be destructive for the received signal. The path loss $L_{PE}$ for this environment given by expression 2.3 is

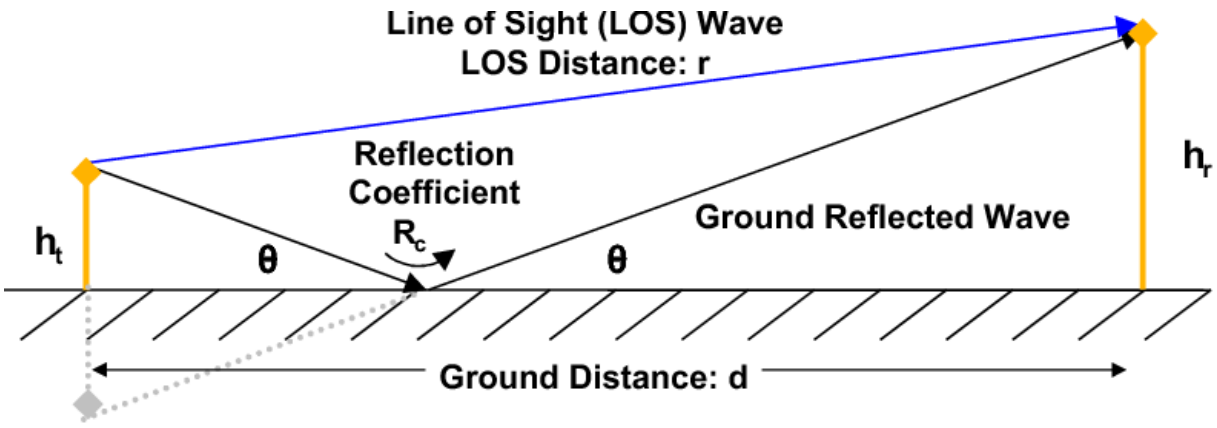


**Figure 2.4:** Plane earth model geometry, from [10]

only determined by the distance d between the transmitter and receiver, the wavelength $\lambda$ of the signal and the heights of the transmitter and receiver, $h_t$ and $h_r$ [3]. This is relevant for the system proposed further in this project because the UWB nodes in the system will be placed at different heights in order to pursue line of sight, making it a suitable model for the path loss between the nodes.

$$L_{PE} = \frac{(4\pi d)^2}{4\lambda^2 sin^2(\frac{2\pi h_t h_r}{\lambda d})} \tag{2.3}$$

This model is not utilized in this project at this time, but should be considered as a foundation for further work on optimization of anchor placement during deployment.

Chapter 3

# Specification

## 3.1 Functional Requirements

The project request made by AIC was to develop and evaluate a positioning system using UWB for high accuracy, outdoor, local measurements, and NB-IoT for data exchange with remote parts of the system. The use case was specified as a $100m \times 100m$ constructions site, where keeping track of any non-stationary assets is important. The solution should should be designed with ease of use and deployment in mind.

## 3.2 System Spec.

The system is divided into server and devices both to move the computationally heavy tasks away from the battery driven devices, and for the server to function as the link to the user. As the anchors do not exchange measurements between themselves, and they could potentially be spread over large distances, a central server is necessary. The following list summarizes the major parts of the system:

- UWB positioning devices, IEEE 802.15.4a standard compliant
- Server, capable of gathering measurements from UWB devices and calculate positions
- Server frontend, to present calculated positions to the user
- Communication between devices and server

These parts are further defined below. The communication between devices and server are mainly handled by the cellular network and in this project the specification of this part is limited to the NB-IoT module on the device.

## 3.3 Device Spec.

The device will function both as a test bench for UWB and GPS positioning, enabling comparison of these technologies, and a proof of concept positioning product. The UWB system requires two types of devices, this is discussed in chapter 4. The GPS was added to the device for two reasons. The tags need to have a well established and widely used positioning system to compare with, both on position accuracy and energy consumption, and to be able to position the anchor points. The global position of anchor points needs to be known to place the tags in a global coordinate system. The device should have the following key features:

- UWB and GPS module, both for comparison and for self-positioning of anchors for ease of deployment
- NB-IoT module, requested by AIC as the data link to server
- Potential to measure energy consumption of each part of device individually (in lab, not normal operation), to potentially expand battery life
- Small size, for mobility and ease of deployment
- Software for both tag and anchor operation, for ease of development and production.

These features are further detailed and discussed in chapter 4, and the implementation is discussed in chapter 5.

## 3.4 Server Spec.

The server should receive the UWB distance measurements from all the anchors, along with their GPS coordinates. This data should be stored in a database on the server before it will be used to calculate the positions of the UWB tags. The server should also act as a webserver to function as a frontend to the end user, displaying the device positions. The server should have the following key features:

- Asynchronous UDP reception from potentially many devices simultaneously
- Database for storage
- Web frontend with positions on map
- Implementation of trilateration algorithm

Chapter 4 later defines what technologies should be used to implement these features, which is presented in chapter 5. The implementation of the trilateration is further detailed in chapter 6.

## 3.5 Description of Operation

Although the hardware is the same for both UWB anchors and tags, they will be configured differently during operation. First, anchors are installed evenly across the area, at heights above 2m. This is to increase the probability of line-of-sight between each anchor and tag. When the anchors are turned on, their position will be determined using GPS, and start sending their coordinates to the server using NB-IoT. The server will store the coordinates in a database. Then tags will be installed on the assets that will be tracked. When the tags are turned on, the anchors will measure the distance to each of them, and send these measurements to the server along with the anchors' position. The tags do not use GPS or NB-IoT. The server will receive this data and store it in a database before using it to calculate the position of each tag. The calculated tag positions is stored in a database, and made available to the user through a website with a map.

Chapter 4

# Design

This chapter extends the specification by explaining how the functionality that was presented in chapter 3 should work. This includes motivating why several design choices were made over others, and detailing the specification enough so that concrete hardware and software can be selected on the proper background in chapter 5, implementation.

## 4.1 Device

As explained in section 2.1.3, positioning systems using trilateration usually have two types of devices: tags and anchors. This is because anchor positions generally need to be known. Additionally, since in this case the anchors do the distance calculations, they hold all the information that the rest of the system is dependent on. This means the tags do not strictly need any other hardware than the UWB module, while anchors need the NB-IoT module for data exchange and a way to be positioned in a global or external context, with GPS in this case. However, if the tags have the same hardware as the anchors, and the possibility to turn off all unnecessary modules, the devices become much more versatile. This way they can be configured as anchors as needed, without consuming more power than necessary when functioning as tags. As this project also serves as a comparison between UWB and GPS positioning systems, the ability to position tags with GPS is needed to compare the positioning accuracy of both systems. That is why only one type of device is developed for this project, all of them with the option to mechanically switch off the GPS and NB-IoT modules.

While the size of the device was not a priority for AIC at this point, high mobility of the tag

devices is an obvious advantage for this system. The ease of deployment also benefits from the anchors being small enough to be hand held. A small board size was therefore pursued during hardware layout design, as mentioned in section 5.2.

### 4.1.1 Communication

NB-IoT as communication channel between anchor and server was specifically requested for this project, as mentioned in section 3.2. Still, compared to the most common alternatives such as WiFi, 4G LTE and even cabled ethernet, there are several advantages to NB-IoT. Below is the two main reasons NB-IoT was chosen as the communication channel to the server:

- On a construction site power cables are already widely installed and a few extra for this device won't make much difference. Network cables are less common and thus deployment takes more time and money if network cables were used.
- Wifi and LTE supports much higher data rates that is not needed for this project. These higher data rates can be made possible with a loss of range, and at higher prices and energy consumption.

A disadvantage to NB-IoT, along with LTE, is the dependence on the cellular network which is a paid service out of the control of this system. The results in Chapter 7 shows no problems with the cellular connection.

### 4.1.2 Reference Points

The position of the anchor nodes in the system needs to be known as they act as reference points between the local UWB positioning system and the global GPS. Further, there are several good reasons why UWB with GPS as reference is potentially more suitable than simply using GPS directly:

- Higher accuracy. GPS positioning devices at consumer prices are often accurate to a few meters. UWB distance measurements are accurate down to a few centimeters.
- Areas with high rise buildings. GPS quickly becomes inaccurate in densely populated areas with tall buildings because of the lack of line-of-sight to satellites. UWB anchors

could be installed on or between buildings and obstacles, avoiding this drawback. This might however require a more manual approach to anchor positioning.

- Indoor use. While not a part of this project, the anchors could be installed and manually positioned indoors, and the system would still work depending on propagation environment.

As mentioned in the specification in section 3.3, the GPS also serves as the metric which the proposed system will be measured against.

### 4.1.3 Distance Measurements

Having decided on a local positioning system in favor of a global one, as explained above, the motivation for using UWB and distance measurements remains. Common alternatives to distance measurements through SDS-TWR in RTLS include AoA or TDOA as mentioned in chapter 2, and distance through signal strength. Regardless of the fact that UWB was requested for this project, there are several benefits of using time of flight UWB:

- No time synchronization. Using TDOA or one-way ToA, good time synchronization is crucial for the accuracy of the measurement. With SDS-TWR, the synchronization between nodes do not greatly impact the accuracy of the measurement [9]. Time synchronization often need cables with known lengths in order to be accurate, which would severely decrease the ease of deployment. Possible use of TDOA and time synchronization due to spectrum regulations is further discussed in section 8.

- Price. Angle of Arrival requires special antenna arrays that are more expensive than a single UWB antenna.

- Accuracy. Received signal strength is a very common method for distance indication, due to the fact that many communication systems calculate a received signal strength indicator (RSSI) at the physical level and presents this to the user. This is however very susceptible to error from both shadow fading and multipath fading. UWB SDS-TWR mitigates both of these effects by doing two measurements separated in time, and spanning over a large frequency band.

Although distance measurements can probably be done cheaper than accurate angle measurements, UWB is still quite expensive in itself, and price could possibly be brought down further by choosing a less accurate and more widespread wireless technology.

### 4.1.4 Power Delivery

As the tags are mobile units, the possibility to power the tags with a battery is essential. The anchor nodes are however not moved around after deployment and would benefit from a more stable power source than a battery. A solution that can accommodate both would therefore be preferable, and as many battery powered units today are charged and powered by an external power source without removing the battery, many such solutions exist. Charging the battery during operation was not a goal for this project, as it would complicate the circuit without adding any real benefit to a prototype that ultimately will not be sold to an end user. It should however be considered a benefit if added to a to a finished product, both because of convenience for the user and uptime of each node in the system.

Batteries that can deliver enough current for the relatively high burst that radio modules often draw during transmission or reception are not typically found in convenience stores, and a quick survey of the available battery technologies revealed that a lithium based battery would be the best choice both in terms of size and possible rate of discharge. The choice of lithium battery opened up a range of voltage regulators that are focused on the mobile market, where devices are commonly powered by a lithium battery with the possibility for charging and power through a Universal Serial Bus (USB) port. The voltage regulators for such devices therefore allows for voltages from 2.7V, typical voltage of an almost drained lithium battery, to 5V, the standard USB voltage. The device implemented in this project need an output voltage between 3.1V and 3.6V from the regulator, and thus needs the regulator to act as both a buck converter, converting the input voltage down from the range 3.7V-5V, and a boost converter, converting the voltage from a lithium battery that is low on energy up from 2.7-3V.

## 4.2 Server

The server can be split up into four parts: incoming, calculation, storage and outgoing. The database should be a commercially available database system, as this does not need any cus-

tomization beyond normal database use. As the relations between the stored data are relatively simple, a non-relational database is chosen. This is further discussed in chapter 5. The incoming part of the server will receive UDP data packets from the anchors' NB-IoT modules, and should store each of these until all the anchors have sent their information, then use this to calculate the positions of the tags in the system. The outgoing part of the server should function as a classical web server that serves a map application and queries the database for the location of all the devices, displaying them on the map. As event based, scalable web server software can be written in frameworks that can function as an event based UDP recipient simultaneously, it makes sense to write a single piece of server software that incorporates both the incoming and outgoing parts of the total server. The chosen framework and server implementation is further discussed in chapter 5.

Chapter 5

# Implementation

Chapter 4 can be summarized in a set of primary design principles that should be in focus when choosing and making specific parts in the system. The principles that guide the implementation of the system in this chapter are as follows:

- Low power consumption
- High positional accuracy
- High mobility
- High power efficiency

These principles are found throughout the hardware and design choices made below.

## 5.1 Hardware Choices

The device implemented for this system is primarily a platform for three independent systems to communicate over. In this section the hardware for each of these subsystems, communication, reference points and distance measurements, is selected according to the design principles above.

### 5.1.1 Communication

As the performance of NB-IoT was not critical in this project, neither in terms of radio performance or power consumption, a chip used in an earlier project [21] was chosen to both limit

the impact on development time and cost of development tools. A development kit using BC95 from Quectel was readily available for testing, and was therefore chosen for ease of development. The reason power consumption was not considered important for the NB-IoT hardware is because it will only be used on the anchor nodes, that are not battery powered. While the anchor devices will support battery power as well, the system usage will benefit from the simplicity of always powered stationary anchors.

### 5.1.2 Reference Points

| **Model** | **current consumption, acquisition mode** | **acquisition time, cold start** | **horizontal accuracy, CEP** |
|---|---|---|---|
| Quectel L96 | 22mA | 15s | 2.5m |
| ublox UBX-M8230-CT | 20mA | 26s | 2.5m |
| OriginGPS ORG1518-R02 | 41mA | 35s | 2.5m |
| GNS 601uLP | 21mA | 31s | 1.5m |

**Table 5.1:** Comparison of GPS modules

A survey of the latest low power, consumer grade GPS modules on the market was performed, and Quectel L96 was chosen because of shorter acquisition time. This leads to less energy consumption overall, although two other chips draw less current in acquisition. Table 5.1 show the GPS modules and chips that was considered.

On the final device implementation the GPS module is set to 'static' navigation mode. It is not uncommon for GPS devices to have several modes to choose from that helps optimize the accuracy and power consumption for a level of movement, such as a 'vehicle' mode or dynamic mode for rapidly changing positions that needs frequent updates, and conversely, a static or stationary mode that increases accuracy over time and uses less energy. This is done by employing different Extended Kalman filters in the GPS receiver that is optimized for a respective receiver mode, as explained in [16]. As the anchor nodes does not move during a single deployment, this static mode is a suitable setting.

### 5.1.3 Distance Measurements

There are several UWB products and systems for positioning available on the market, but most of them are locked down and proprietary, without any possibilities to change the system, neither hardware nor software. A lot of these off-the-shelf systems are built around the same few chips. There are only a handful of integrated UWB positioning chips that are used in available products, the most common of which is DecaWave DW1000 and Beespoon UM100. DW1000 has a higher pulse repetition frequency and thus supports tracking of higher velocity objects. UM100 claims to have successful measurements at 800m range, while DW1000 only claims a 230 range maximum. While the range of both products is enough for this project, high velocity tracking is not a part of this project and decreased anchor density is an obvious advantage in terms of cost and deployment. On the other hand, the DW1000 has a superior market share. This means that open source firmware is available for specific microcontrollers. This leads to the Radino32 DW1000 from In-Circuit. This is an available hardware solution for UWB positioning, working both as a standalone system or as a part of a larger hardware solution. It contains both the DW1000 and a low power microcontroller, STM32L151CC, with the necessary firmware driver for communication between them, based on open source code. This availability of open, modifiable firmware and software was the main reason the Radino32, with DW1000, was chosen over UM100.

On a construction site there will of course be desirable to track multiple objects, not just one. The source code for the Radino32 defines the limit for how many devices that can be on the same UWB network as 64 devices. This was determined to be enough for this project, and further study of solutions and technologies for multiple access was not performed.

### 5.1.4 Power Delivery

The BC95 NB-IoT chip demanded that the power supply should be able to deliver 0.5A, despite its claim to not draw more than 230 mA in typical conditions. Added to this peak of 500 mA is the current draw of the microcontroller, and some small currents for the other modules to keep them alive. A minimum of 600 mA current delivery capability is then demanded of the supply circuit. To support both battery and USB connection as power source, the supply circuit also had to have a voltage range from 2.7 V (typical minimum for Lithium batteries) to 5V( USB 2.0 voltage). Given that the output voltage would be 3.3V, and the minimum input voltage

should be 2.7, the supply circuit needed to be a buck-boost converter, being able to convert the input voltage both up and down. To be able to assemble the device, the supply circuit had to be immediately available from at least one parts supplier, and come in a package with leads that can be easily hand soldered without special equipment (i.e. not QFN/BGA type packages). The LTC3521 from Linear technologies was chosen because of its ability to transition from Burst Mode™ to PWM mode automatically, thus maximizing power efficiency over the whole output current range.

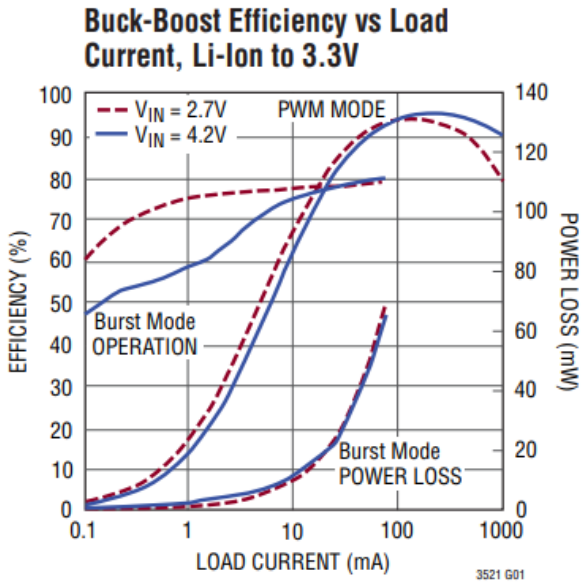


**Figure 5.1:** Efficiency of power regulator over load current, from datasheet [8]

This differentiates it from the smaller and cheaper LTC3536, which also have both bust mode and PWM mode, but lacks automatic transition. This means there will either be high efficiency on lighter loads but no ability to deliver current over 200mA, or very low efficiency for lighter loads, 20% efficiency at 1 mA, the order of magnitude at which the current draw of the device will spend the most time. Instead, as shown in figure 5.1, the LTC3521 has an efficiency of over 50% at lighter loads and over 90% for larger loads when employing both burst mode and PWM mode.

ADP2504 from Analog Devices and TPS63031 from Texas Instruments both offers comparable efficiency over the current range and meets all the criteria except the ability to be easily hand soldered. They are both cheaper and smaller, but their package have no leads and are intended for machine assembly. For larger scale production beyond prototypes, one of them would probably be the preferred choice.

## 5.2 Circuit Design

After picking out the main parts of the device, as detailed in section 5.1, a circuit was designed to maximize performance of the chosen parts according to specification. The very first layout proposal was a 8x5 cm single side Printed Circuit Board (PCB), but this was reduced to a 5x5 cm double sided PCB in order to make the device smaller and more mobile.

**Antenna traces** To get the best radio performance out of each of the radio modules, impedance controlled microstrip transmission lines was designed for each of them to their respective $50\Omega$ RP-SMA connectors. To find the line widths of the antenna traces the LineCalc tool in Keysight Advanced Design System was used, and all transmission lines was found to be just under 3.1 mm wide, with slight variations with the different frequencies. The transmission lines are clearly visible in figure 5.2, as the wider traces designated by A1 and A3 at the top layer, and the much shorter trace designated by A2 on the bottom layer. Because the transmission line from the NB-IoT module to the antenna connector became too short, 5.1 mm, this line was not tapered to the calculated width, and instead just tapered to the width of the antenna connector pad width.

**Individual module control** To be able to accurately compare the current consumption of the different modules, without leakage currents from deactivated modules influencing the results, mechanical power switches were installed on each power line to each module. Additionally, the Universal Asynchronous Receiver-Transmitter (UART) interface between each module was made available for external connection on the PCB, making it possible to control each module without the microcontroller drawing power. These two features, mechanical power switches and available UART ports, ultimately made it possible to measure the power consumption of each module accurately, as specified in section 3.3.

**Capacitance** In accordance with the datasheet of each module in the circuit, capacitors was added to the circuit physically close to the power supply pin of each module. A capacitor of $22\mu F$ was also added to the output pin of the voltage regulator, as specified in its datasheet [8]. This was done to be able to deliver enough power at sudden bursts of power consumption by the modules. As discussed later in section 7.1.3 there might not have been enough capacitance for all the modules.

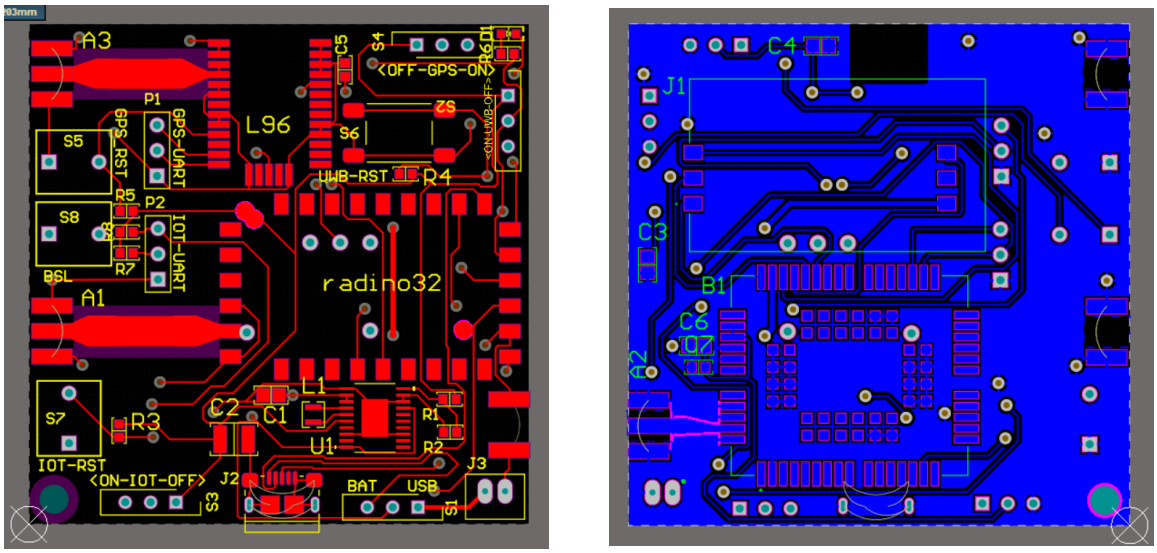


**Figure 5.2:** Top and bottom layer of PCB design

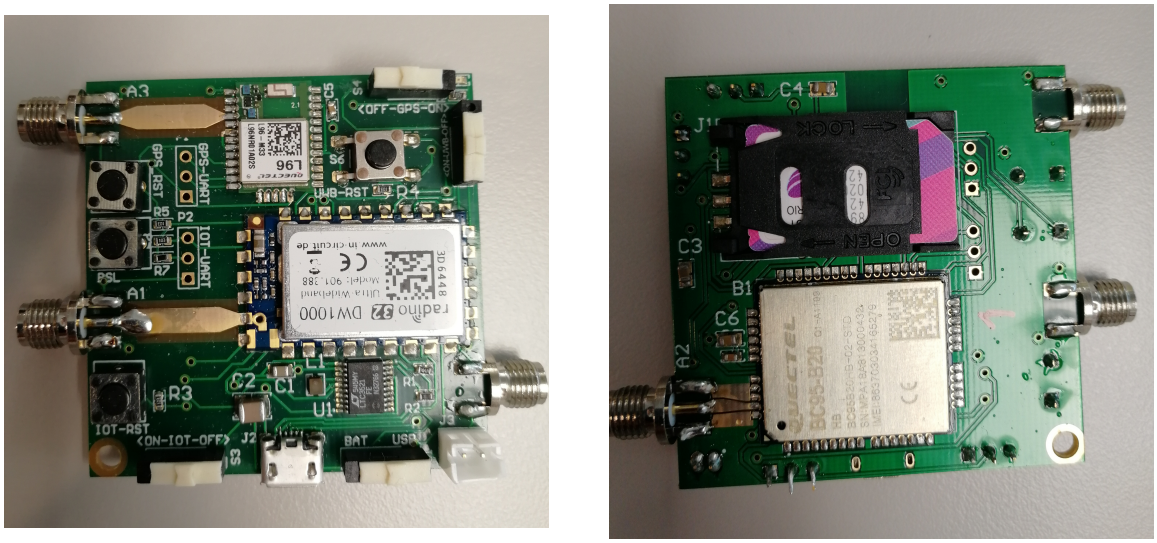


**Figure 5.3:** Top and bottom of assembled devices

**Manufacturing** After the circuit design and layout was finished, a single circuit board was produced at the prototype laboratory at NTNU, to verify that sizes and margins used in the layout could be used in production without causing short circuits between traces. Because the prototype laboratory at NTNU was unable to make plated via holes that could be placed below components, the fabrication files were sent to a PCB production company in China which produced 10 circuit boards. The PCB production parameters were as follows:

- Substrate material: FR-4 TG130
- Substrate thickness: 1.6 mm
- Copper trace thickness: 1 oz
- Surface finish: immersion gold (to avoid degradation of transmission line by copper oxidation)

The devices were finally assembled with components by the prototype laboratory at NTNU. They assembled 9 out of 10 devices, with one unsuccessfully attempt. One assembled device is shown in figure 5.3. The feedback from those who soldered the circuits was that it would have been easier if all the modules were on the same side of the board, and if the solder pads was not gold plated. This is something to consider for a potential next version.

## 5.3 Device Software

The UWB module came with open source software that run on the integrated STM32L151 microcontroller. That code is written in C++, but Arduino libraries are utilized to provide very easy interfacing with serial communications hardware. This code from the manufacturer of the UWB module was then expanded on to communicate with the GPS and NB-IoT module. These are both meant to be controlled over UART and the respective commands can be found in their datasheets, [17] and [18] respectively. The source code for this interface can be found in the appendix section III.

## 5.4 Server

The server is divided into four software parts: incoming, database, calculation and front end. The Incoming part simply receives data, in the form of UDP packets, from the anchors over NB-IoT and stores this data in the database. The anchors send two main types of messages, GPS data and distance measurements. The GPS data is stored in a collection of its own, which hold all the anchor IDs, and a log of their respective GPS data. Each log item holds a position, the number of satellites that was used to find that position, and horizontal dilution of precision (HDOP), which is a number indicating the quality of the data. The distance measurements are stored in another collection, which holds the anchor and tag ID that the distance was found between, and a log of the last distance measurements that was done between these two IDs, or points. Each log item holds the distance in meters, the received signal power at both the anchor and tag at the time of the measurement, and a time stamp from the anchor. The topology is outlined in figure 5.5. The incoming server is implemented using Nodejs, with its included UDP server library. The database is a MongoDB database, with three collections: Anchors, Distances and Tags. The native MongoDB driver for Nodejs was used for interfacing between the database and both incoming and frontend server. An image of the user frontend can be seen in figure 5.4.

The calculation part is done in python, mostly because the least squares algorithm that was tested and chosen in Matlab, Levenberg Marquardt, was readily available through the python library SciPy. Both the Matlab and SciPy libraries use the same FORTRAN implementation of the Levenberg Marquardt algorithm from the MINPACK library [13]. The calculation code collects all the data from the database, then loops through all the tags and calculates their positions using the distances and respective anchor positions that is linked through the ID fields in the database. As opposed to the approach from simulations, described in chapter 6, the initial guess at each tag position is the position of the closest anchor. All server source code can be found in the appendix.


**Figure 5.4:** User frontend, showing the anchors as black dots and tags as blue dots on a map

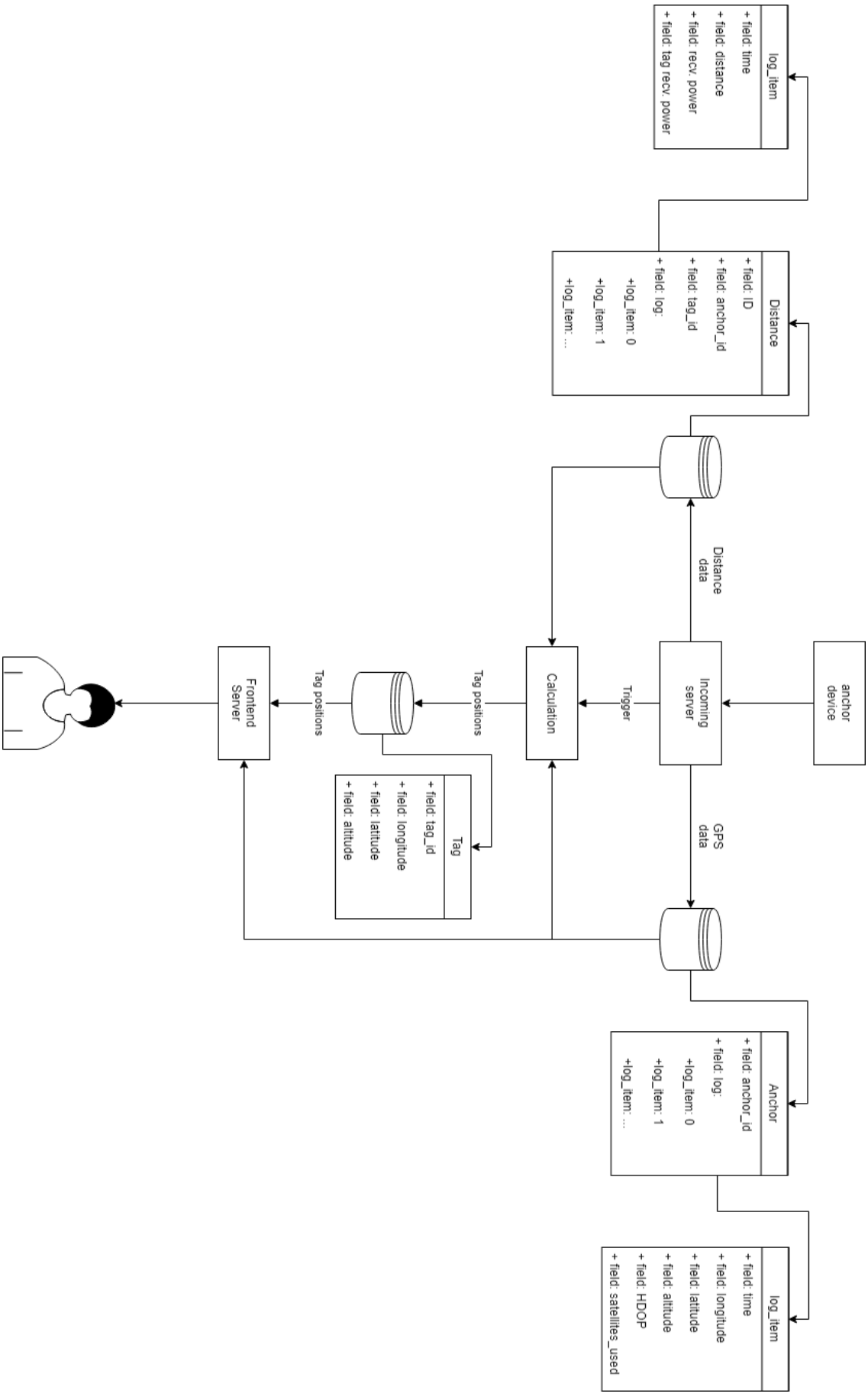


**Figure 5.5:** Server flowchart with database topology, from incoming device data to user interface

Chapter 6

# Simulations

This chapter details the simulations that was done both to characterize some of the parameters that impact the system performance, and to evaluate what algorithm should be used to solve the overdetermined trilateration problem and calculate the tag positions.

The simulation environment can be configured with either anchor nodes uniformly distributed on a disc with radius 50m as shown in figure 6.1, or evenly ordered on a circle with equal distance between all neighbouring anchor nodes. The randomly distributed option provides a more realistic setup on a construction site where topography and infrastructure defines the optimal setup, while the perfectly ordered option provides the ability to explore a scenario where a tag has equal distance to all anchor nodes and avoiding cases where the randomly distributed anchors could be clustered close together.

The simulation start with placing a number of anchors and tags inside a specified area, 100x100m in this case. This is the true, unknown positions that we are trying to estimate. The true distances between the tags and the anchors are calculated. An error is added to each anchor position to account for the inevitable uncertainty in GPS measurements. This error takes the form of a uniform distribution on all three axes within a specified limit, the limit equal on all axes. An error is added to the distance, to account for noisy and inaccurate measurements. This error is proportional to the distance, in this case each error is within 10% of the distance, to account for the path loss and measurement noise. This correlation between distance, or signal strength, and measurement error was later found not to be as strong in the final UWB measurements in section 7.1.2.

Using the inaccurate measurements and anchor positions, the chosen algorithm is applied to

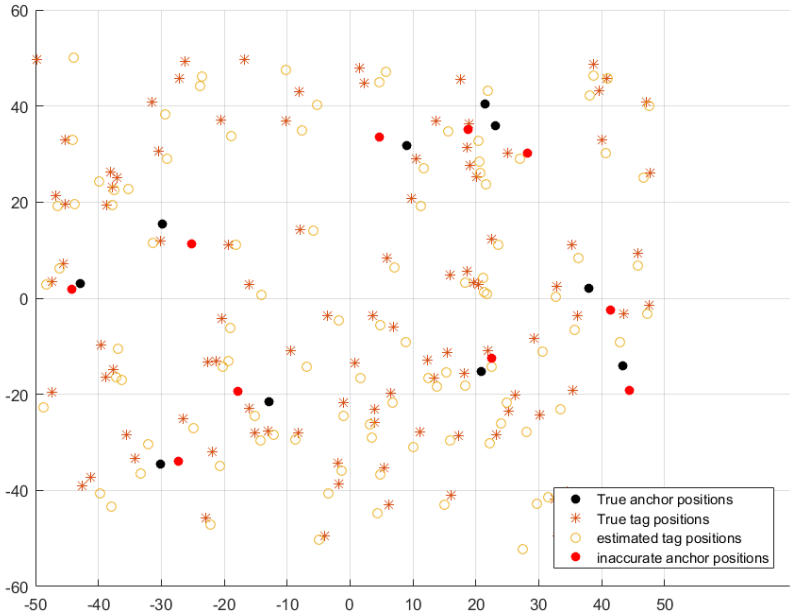


**Figure 6.1:** Layout of simulated nodes

find the tag position. This begins with an initial guess on the tag position, which is in the middle of the area in this case. A possible improvement is to make a guess closer to the true position using the 4 shortest measurements, i.e the 4 closets anchors, and solve the trilateration equations only using these. This is an arithmetic problem that can be solved before any estimation is applied. After each tag position is estimated, the distance from the estimated position to the true position is calculated, and a Root Mean Square Error (RMSE) is calculated from all of these. The RMSE for the anchor positions is found the same way.

Using the above simulation we can find how the positioning accuracy depend on various parameters. The results is generally found by varying the concerning parameter over 200 simulations.

First the simulation is run 200 times with an increasing limit for anchor position error each time. The area is kept at 100x100m, with 100 tags and 10 anchors. The anchor height is distributed evenly between 2-10m.

Next, the number of anchors is analyzed. With 100 tags, 2.5m limit on anchor position error on each axis, and height distribution on anchors at 2-10m.

The anchor position height is also analyzed by varying the length of the range of heights. The heights of the tags and anchors are interesting because of the Plane Earth Models' path loss

dependence on height, as mentioned in section 2.3, and because line of sight should be pursued during deployment. The plane earth model is not simulated here, but as it is relevant for a real setup, it is interesting to see how height affects the simulated algorithms.

## 6.1 Gauss Newton Algorithm

Initially, the Gauss Newton algorithm was chosen to solve the nonlinear least-squares problem. This choice was primarily based on the availability of open source code that could be utilized without much alteration, and was later replaced after testing.

Figure 6.2 shows that the 2D tag RMSE increases quadratically with the anchor RMSE with a minimum at 1m. This is likely the minimum error that can be achieved with 10 anchors with this algorithm, as it is approximately the same error we see in figure 6.3 at 10 anchors.

Figure 6.3 show that on an area of 100x100m there is a substantial improvement in tag position RMSE from 3 anchors to 4. The improvement is much less from 4 to 5, and after 5 the improvement is below 1m. This makes 4 anchors the most cost efficient number of anchors, and anything above 5 is likely to be wasteful.

It can be observed in figure 6.4 that by increasing the range of the anchor position height distribution, the tag position RMSE gets a lot better from a length of range of 1m to 5m, before the improvement diminishes after a range length of 5m. This means that no matter the height the anchors are installed in, it should vary by at least 5m. Why this phenomena occurs is not known, but it could possibly be related to the ambiguity that would occur if all the anchors were placed on a horizontal plane. This would make it impossible to determine just from distances to anchors if the tags were above or below that plane. Inaccuracies in measurements and anchor positions could cause this same ambiguity when anchors are close to a planar topology. A proper sensitivity analysis should be performed in further work in order to characterize this phenomenon.

## 6.2 Levenberg Marquardt Algorithm

After the first, indoor test of the UWB equipment was performed, as described later in section 7.1.1, the Gauss Newton algorithm was found to diverge and produce distant outliers in the calculated tag positions. After the algorithm was changed to Levenberg Marquardt [11] and tested

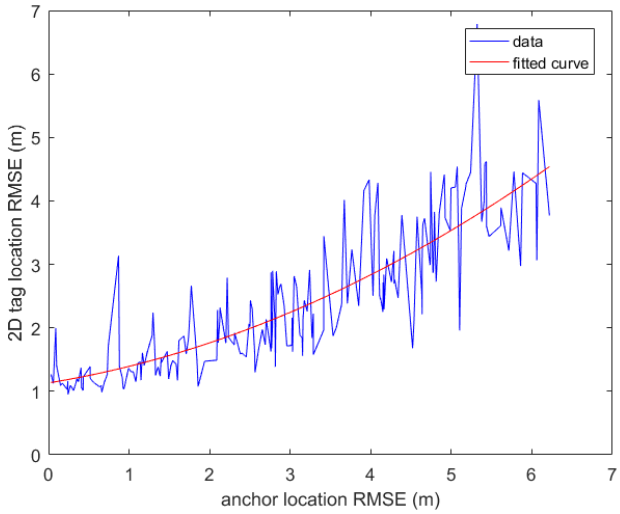


**Figure 6.2:** Gauss-Newton, squared relationship between anchor RMSE and tag RMSE for 10 anchors

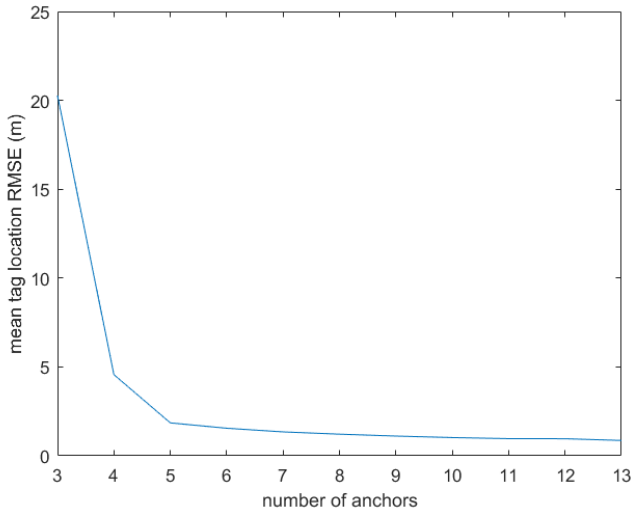


**Figure 6.3:** Gauss-Newton, tag RMSE by number of anchors

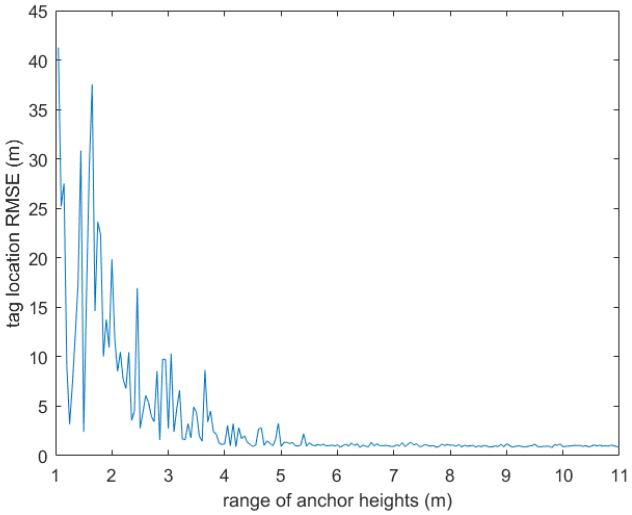


**Figure 6.4:** Gauss-Newton, tag RMSE by length of range on anchor heights

with measurements with good results, the simulations was repeated with the new algorithm.

The results of simulating increased anchor position uncertainty was much the same as before with a quadratic relationship between anchor position RMSE and tag position RMSE, see figure 6.5.

The results of varying the number of anchors was improved quite a lot for a small numbers of anchors, while a larger number of anchors converged at about the same accuracy as before, 0.8m. At 3 anchors, the tag 2D RMSE was 20m for Gauss Newton, but 3m for Levenberg Marquardt, as shown i figure 6.6.

The range of anchor position height distribution did not impact the tag position RMSE in the same way as for Gauss Newton at all. There seems to be no apparent relationship between the length of range of anchor heights and tag position error for the Levenberg Marquardt Algorithm, as can be seen in figure 6.7.

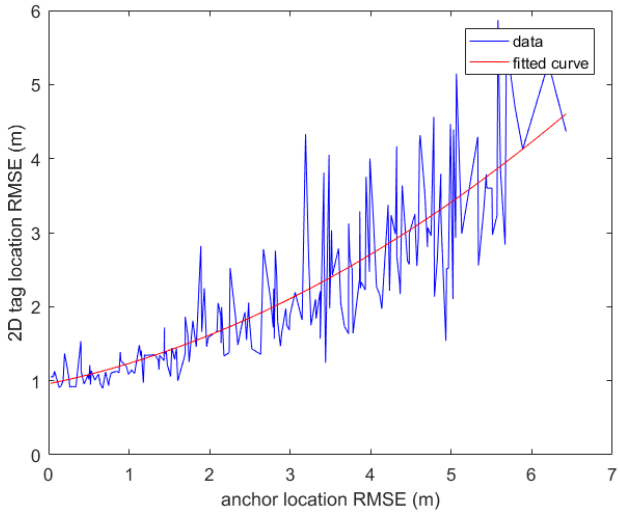


**Figure 6.5:** Levenberg-Marquardt, squared relationship between anchor RMSE and tag RMSE for 10 anchors

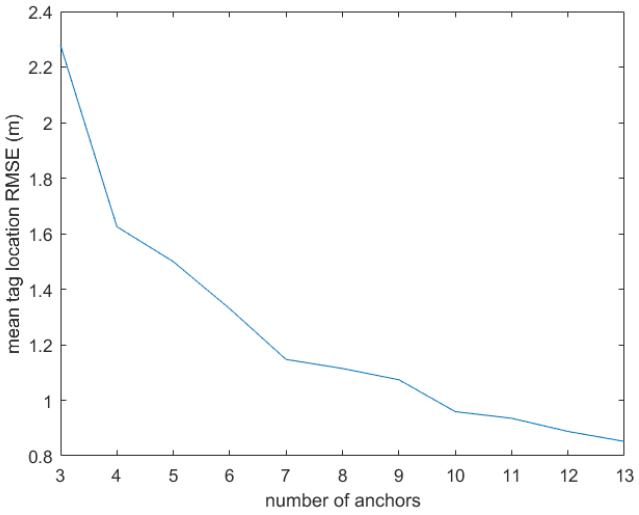


**Figure 6.6:** Levenberg-Marquardt, tag RMSE by number of anchors

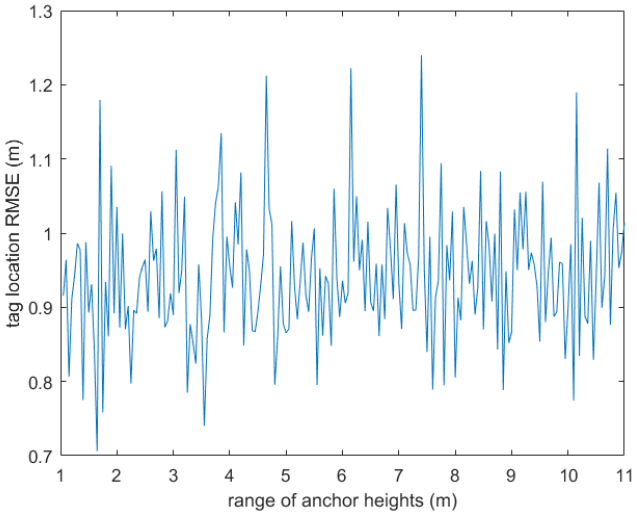


**Figure 6.7:** Levenberg-Marquardt, tag RMSE by length of range on anchor heights

Chapter 7

# Tests & Verification

This chapter describes the tests that was performed both during development and after the final implementation was done, and the results that was observed. The tests include simple initial test to see if the UWB modules and algorithm worked, and a final test to verify that the whole system works together. The results focus on position accuracy and power consumption.

## 7.1 Subsystem and System Tests

In order to verify the functionality of different parts of the system while development was still ongoing, several test was performed. The most important of these test are described in this section.

### 7.1.1 First Test of UWB Modules, Small Indoor Room

To verify the functionality, and get an idea of the accuracy of the chosen UWB modules, a small test was performed indoors. The test was conducted in a small room measuring approximately 5x5m. 3 anchors nodes were each placed in a corner, making a right-angled triangle. A single tag node were placed close to the middle of the room, with line of sight to all anchor nodes. The test ran for about 6 minutes, and gave 685 ranging samples at each anchor node. This was imported in matlab, and the standard deviation in the calculated positions were found to be between 10-50 cm in horizontal plane, but this was due to some outliers which appeared after position calculation. These outliers were several kilometers away from the true position. The cause of this is still not determined, but the Gauss Newton algorithm is known to diverge if the

residuals becomes large [12]. The calculation was later done using the Levenberg Marquardt algorithm instead. Using this algorithm the results were much more accurate, with no outliers, and a standard deviation of 1-2cm in the horizontal plane. Doing the same calculations with the same data in the python code implemented for the server got the same exact results down to millimeter level. The results of the two calculations using Gauss Newton and Levenberg Marquardt is shown in figure 7.1 and 7.2 respectively.

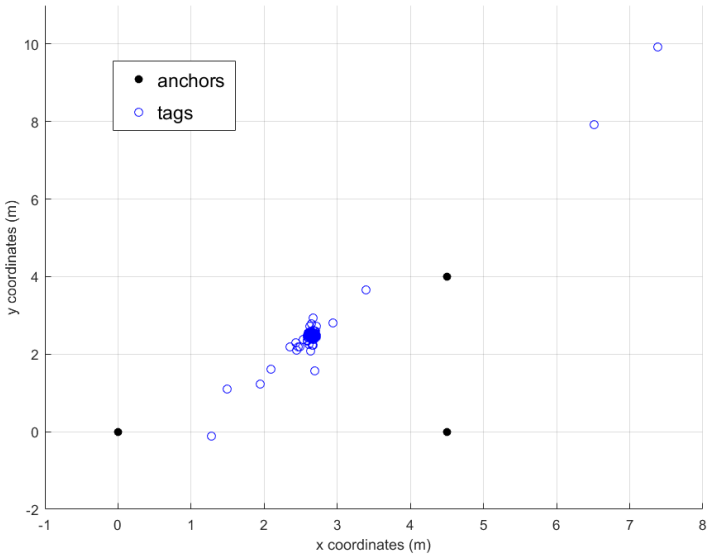


**Figure 7.1:** Preliminary indoor test of UWB system using Gauss Newton, the worst outliers not visible

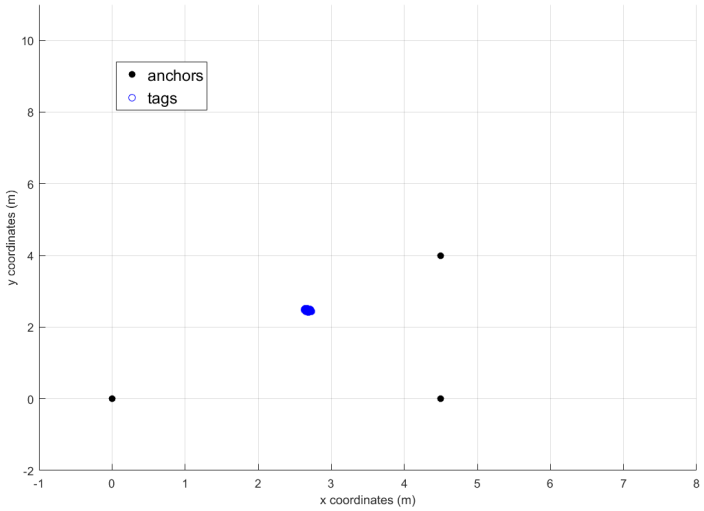


**Figure 7.2:** Preliminary indoor test of UWB system, using Levenberg Marquardt algorithm.

### 7.1.2 Second Test of UWB Modules, Fixed Distance, Varied Transmit Power

To characterize the distance measurement error, a test was performed inside at a fixed distance of 9m. The transmit power of both tag and anchor was then decreased from 30 dB to 1.5 dB to see if the measurements became less accurate with less received signal power, and to find the probability distribution that best fit the data. The reference for these decibel values are not known, they are not specified further in the datasheets of neither the Radino32 nor the DW1000. As shown in table 7.1 and figure 7.3, the standard deviation of the distance measurements went a little higher when the received power went down, from around 0.02m to around 0.08m. However, the received power of neither tag or anchor decreased with the decreased transmit power for a range of over 25 dB, which seems unlikely. If the receiver happened to be in saturation, unable to utilize all the transmitted power, the results should be less accurate as a saturated receiver will distort the signals. If the receiver was not saturated, the reported received power should be proportional to the transmitted power. Therefore the reported received power is likely wrong.

| TX power (dB) | mean distance(m) | standard deviation distance(cm) | mean anchor RX power(dB) | standard deviation RX power(dB) | mean tag RX power(dB) | standard deviation tag RX power(dB) |
|---|---|---|---|---|---|---|
| 30 | 8.988 | 1.9 | -77.177 | 1.2 | -76.565 | 1.2 |
| 25 | 9.000 | 2.1 | -77.376 | 1.2 | -76.767 | 1.3 |
| 20 | 8.990 | 1.9 | -76.809 | 1.2 | -76.158 | 1.2 |
| 15 | 8.991 | 2.1 | -77.119 | 1.3 | -76.478 | 1.3 |
| 10 | 8.987 | 1.8 | -76.622 | 1.2 | -75.938 | 1.3 |
| 5 | 8.975 | 1.9 | -75.976 | 1.2 | -75.576 | 1.2 |
| 4 | 8.972 | 9.2 | -81.935 | 1.1 | -81.222 | 1.2 |
| 3 | 9.022 | 7.5 | -87.837 | 0.8 | -87.281 | 1.0 |
| 2 | 9.038 | 7.5 | -95.820 | 0.6 | -95.546 | 0.6 |
| 1.5 | 9.011 | 9.6 | -100.771 | 0.8 | -100.565 | 0.6 |

**Table 7.1:** Results of distance measurements at different transmit power levels

Some possible probability distributions for the distance measurements is shown in figure 7.4. The two best fitting distributions, that was found through the distribution fitting tool in Matlab, are not commonly used to characterize line of sight, multipath wireless channels, such as the Rician distribution. This could be due to the distance measurement accuracy mainly

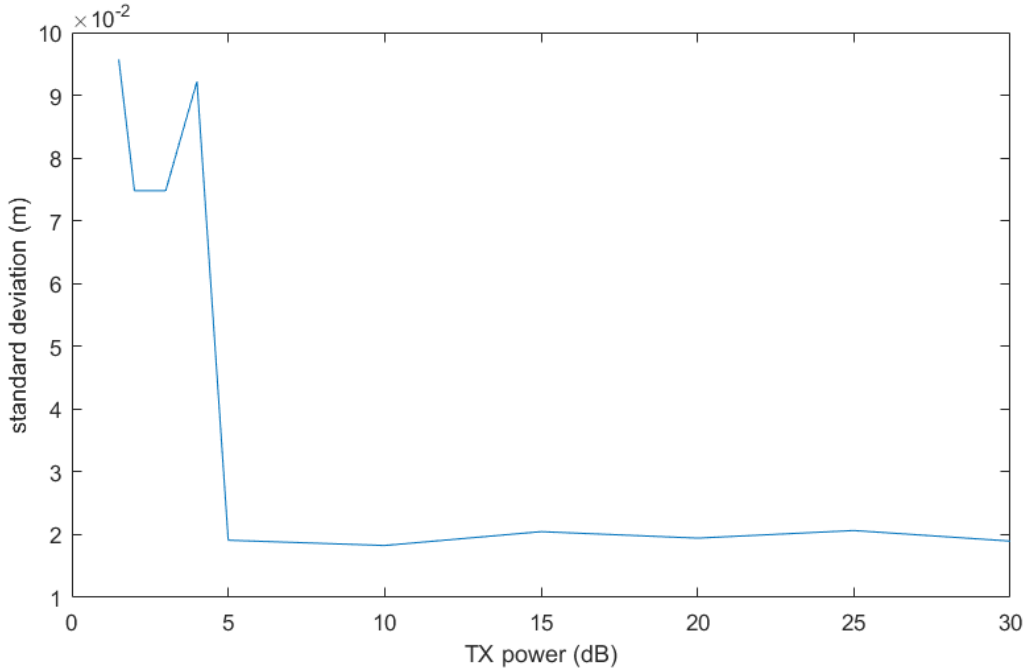


**Figure 7.3:** Standard deviation in distance measurements as a function of transmitted power

being limited by the algorithms used to calculate them, rather than the fading of the wireless signals.

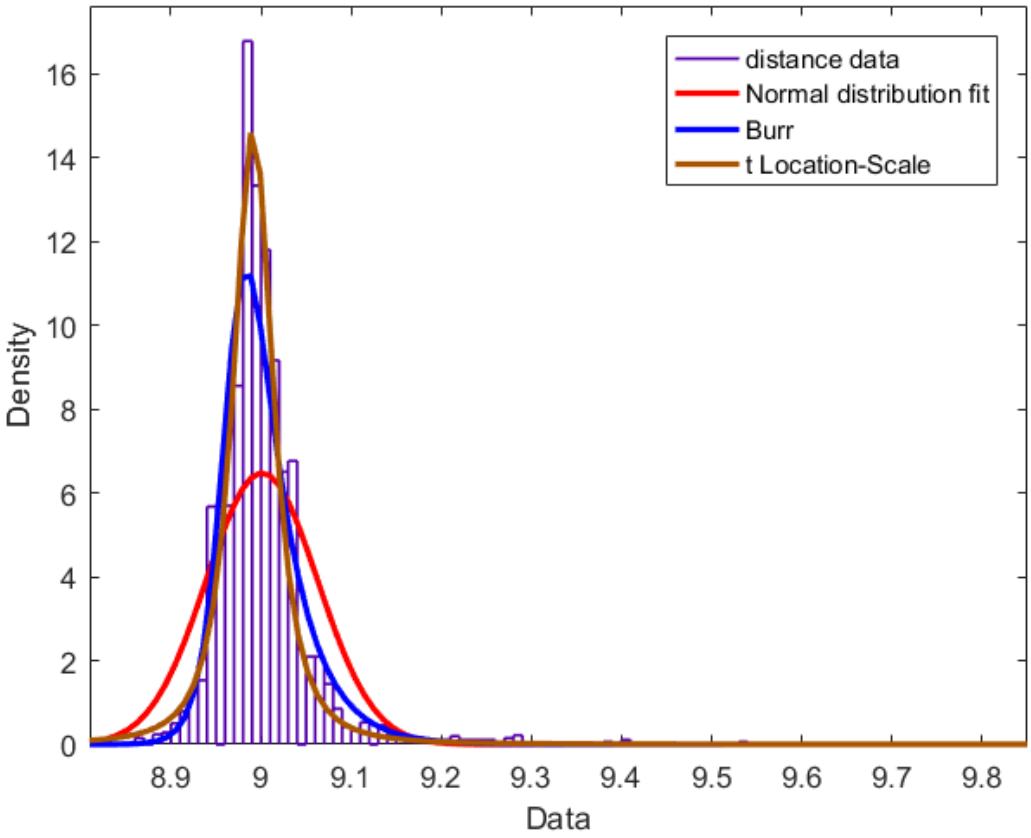


**Figure 7.4:** Possible distributions for distance measurements

### 7.1.3 Unit Test

When the devices were ready from assembly, a simple unit test was performed to verify that they worked as intended. With the device powered by a USB port through the voltage regulator, each module was powered on in turn to test their primary purpose while output was collected through their UART ports on a computer and verified. The NB-IoT module was set to connect to the cellular network, the GPS module was set to attempt to find its own position, and the UWB module attempted to receive UWB signals. This was successful for 8 of 9 assembled devices. When all modules on the devices was powered simultaneously however, the voltage regulator did not deliver enough power fast enough, and a voltage drop at over 1 V would occur when the radio modules activated, causing a reset of the NB-IoT and UWB modules, including the microcontroller. This can possibly be fixed in further work by adding more capacitance to the power supply net of the circuit, with charged capacitors functioning as a quick response energy reservoir for bursts of power consumption. Another possible cause of the lack of burst power delivery is the configuration of the voltage regulator, which was set to automatically transition from burst mode to PWM mode as described in section 5.1.4. The response time of this transition can have been overestimated, although the datasheet show a transition time below $50\mu s$ which was not observed. To be able to perform the verification of the rest of the system, the on board voltage regulator was bypassed with a laboratory power supply, delivering 3.3V directly to the power delivery net in the circuit. This kept the voltage drop to a minimum during operation and the next section show that all features were found to be functional.

### 7.1.4 Final Verification

As soon as the whole system was implemented, a complete setup and test was carried out. The setup had to take place outside to test the GPS, and due to the malfunction of the power delivery on the device the location was limited to places outside with access to the main power grid. The roof on the department of electronics and telecommunications of NTNU provided excellent GPS conditions with a broad horizon and access to power so the anchor nodes could be powered by an external power supply, bypassing the on board voltage regulator. To evaluate the accuracy the system against GPS during testing, the idea was to use the GPS and NB-IoT modules on the tags, in addition to the anchors. Tags would then upload its GPS position using NB-IoT just as the anchors normally do, and these could be compared to the calculated results


**Figure 7.5:** Test setup on roof of NTNU

later. Due to the need for an external power supply for all the modules to be powered this was not done, as this introduced a need for more cabling than a battery powered tag would need. The calculated positions were instead just compared to the accurate GPS Real-Time Kinematic (RTK) positions. Unfortunately the roof did not cover an area as large as $100m \times 100m$. The diagonal of the covered area measured approximately 45m, between the two anchor nodes furthest apart.

The anchor nodes was installed at appropriate heights to ensure a line of sight between all tags and anchors, without unnecessarily increasing the distance notably.

For reference to both the on board GPS on the anchor nodes, and the final, calculated positions of the tags, an industrial GPS RTK receiver capable of millimeter precision positioning was used to accurately determine the locations of both anchors and tags.

See figure 7.5 for a simple view of the verification setup, and figure 7.6 for an example of the setup of an anchor node. The results of the verification can be found below, in section 7.2.

**Figure 7.6:** Anchor device setup during final verification

## 7.2 Results

Here we find the final results of the verification process, and the ultimate metric in the comparison of accuracy between this local positioning system and the more available GPS. We see that the two tags are positioned by the system at a distance between 2.5m and 5.1m from their respective reference points. This is comparable to the accuracies reported by GPS systems and the errors observed on the GPS positioned anchors. It is however not possible to say much about the statistics of the tag positioning error, because of the low number of data points.

If the anchor points used in the calculation use the accurate positions of the reference points instead of the relatively uncertain positions found with the on board GPS, as reported in table 7.2, the results become much better. This is a clear indication that the inaccuracies of the GPS positioned anchor points will be heavily reflected in the tag positions and produce similar errors. This is further discussed in chapter 8.

Tag: 1

2D distance to reference using the mean of measured anchor position: 5.10 m

2D distance to reference using anchor reference positions: 0.62m

Tag: 2

2D distance to reference using the mean of measure anchor position: 2.49 m

2D distance to reference using anchor reference positions: 1.26m

The datasheet of the L96 GPS module reported a horizontal position accuracy of 2.5 CEP, but the average horizontal error of the anchor positions is over 4m. It seems as though the GPS module might not be as accurate as it claims. It is also important to note that when measuring distances between nodes at different heights, the vertical position accuracy is also important, as all positions must be calculated in 3D space. The datasheet of the GPS module does not mention vertical or 3D accuracy, but table 7.2 clearly show that 3D accuracy is notably lower than just horizontal.

| anchor id | 3D distance error (m) | 2D distance error (m) | std longitude (deg) | std latitude (deg) | std altitude (m) |
|---|---|---|---|---|---|
| 1 | 11.738 | 9.399 | 7.68e-03 | 3.54e-03 | 1.48e-01 |
| 2 | 8.056 | 1.973 | 1.43e-04 | 4.33e-04 | 1.74e-03 |
| 3 | 4.723 | 2.519 | 2.75e-03 | 6.32e-04 | 9.11e-02 |
| 4 | 8.490 | 3.420 | 1.01e-09 | 2.34e-05 | 1.31e-12 |
| 5 | 6.079 | 4.006 | 1.80e-04 | 8.50e-04 | 3.50e-03 |

**Table 7.2:** Results of anchor self positioning

Table 7.3 show the resulting measured distances between the tags and anchor nodes in the system. The true distance in this context is the calculated distance between the accurately positioned reference points. Important to note here is the fact that all distances have a positive error, meaning the anchors consistently overestimate the distance. This could be due to poor calibration. The UWB module firmware had a calibration procedure to find an appropriate parameter to account for antenna lengths and traces, which was performed.

### 7.2.1 Current consumption

Due to limited development time, the available power saving modes and configurations on each module was not taken proper advantage of. All measured currents in this section was at 3.3V.

| tag id | anchor id | mean measured distance (m) | std distance(cm) | true distance(m) | error distance(m) |
|---|---|---|---|---|---|
| 1 | 1 | 28.148 | 2.55 | 27.706 | 0.442 |
| | 2 | 11.126 | 15.9 | 10.956 | 0.170 |
| | 3 | 18.166 | 4.97 | 17.762 | 0.404 |
| | 4 | 35.440 | 37.7 | 34.988 | 0.453 |
| | 5 | 14.721 | 9.97 | 14.217 | 0.504 |
| 2 | 1 | 18.198 | 4.34 | 17.491 | 0.707 |
| | 3 | 38.282 | 4.01 | 37.648 | 0.634 |
| | 4 | 28.840 | 32.2 | 28.084 | 0.756 |
| | 5 | 16.949 | 12.8 | 16.603 | 0.346 |

**Table 7.3:** Measured distances

| Mode | typical current consumtion |
|---|---|
| PSM | 3.6 $\mu$A |
| Idle | 2 mA |
| Reception | 60 mA |
| Transmission | 220 mA |

**Table 7.4:** BC95 current consumption by mode, from datasheet [18]

**Quectel BC95, NB-IoT Module:** A period of varying current draw between 4-50 mA was measured, with quick bursts up to 300 mA, while trying to connect to the network. 260 uA was measured when connected to the network.

This corresponds well with the datasheet for this module, [18], which presents figure 7.7 as a diagram of the power consumption over time, along with the typical current consumption in each state in table 7.4. The timer values in figure 7.7, T3324 and T3412, are requested by the module and ultimately decided by the network upon connection. This was not configured for this implementation and should be included in further work on this system. When measuring current consumption on the implemented device, with only the BC95 module turned on and the other modules mechanically switched off, we can recognize the pattern in figure 7.7 during the network attachment period. The rapid variations between 4-50 mA can be recognized at the T3324 period in the figure and the corresponding current consumtions from the datasheet, 2 mA and 60 mA for idle and reception mode. The maximum 300 mA measured bursts corresponds to the typical value of 220 mA in transmission mode, and when connected, the 260 $\mu$A corresponds to the 360 $\mu$A in Power Saving Mode (PSM).

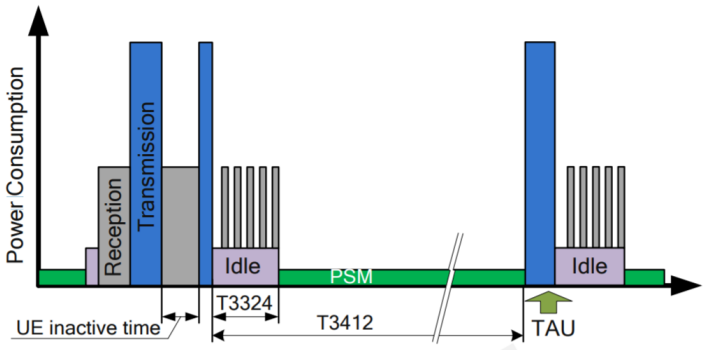


**Figure 7.7:** BC95 power consumption over time, from datasheet [18]

**Radino32 DW1000 UWB Module:** When the internal DW1000 radio chip was inactive, and the module actively tried to initialize the IoT and GPS modules through serial communication while forwarding the results through USB, the recorded current consumption was recorded at 65 mA. This corresponds well to the datasheets of both the Radino32 module [6], and the internal microcontroller [20], which states a total current consumption of the microcontroller of 60 mA when all inputs and outputs are active. For further work, the microcontroller should be put to sleep for a time instead of keep trying to initialize the other modules without end. As soon as the DW1000 radio chip became active, the current consumption became 230 mA, still with only the UWB module mechanically switched on. This should correspond to the current consumption from the DW1000 datasheet, [5], approximately 113 mA in reception mode, added to the 65 mA from the microcontroller, which is 178 mA. The DW1000 does however have 16 different modes for various frequency channels and data rates which have not been configured for this project beyond the default settings, which could explain the gap from 178 mA to 230 mA. At this point, the 230 mA is the typical current consumption of a device configured as a tag in the system. This consumption should be decreased, and probably can without much difficulty. This is discussed further in chapter 8.

**Quectel L96 GPS module:** A current consumption of 30 mA was measured for the GPS module during cold start acquisition mode, which is more than the consumption reported by the datasheet, [17], 22 mA. The current consumption after acquisition was not measured for this project, as the measurement equipment was not possible to bring outside.

**Full Operation** When all the modules were turned on and operated as a full function anchor node, the current consumption was consistently between 250 mA and 350 mA, with short burst

up to 400 mA. This was more than the voltage regulator was able to deliver, and the BC95 and UWB module would reset from low input voltage regularly when power was delivered from the LTC3521 voltage regulator. According to the datasheet of the LTC3521 [8], this current draw should not be a problem. The resets would occur during the bursts of higher power consumption, so a possible solution would be higher capacitance on the power supply line to have more energy stored there for rapid delivery.

Chapter 8

# Discussion

**Algorithm for inaccurate anchor positions** As showed in section 7.2, the inaccuracy of anchor positioning has a major impact on the accuracy of the tag positioning. This is not a novel discovery for positioning in wireless sensor networks, but as stated in [7], the main focus in error analysis for such systems are often inaccuracies in the distance measurements. The trilateration algorithm used in this project assumes that the mean of the anchor positions will be quite accurate, at least after some time when the GPS measurements are supposed to improve. The result show that one perhaps have to consider using an algorithm as the one proposed in [7], which assumes inaccurate anchor positions and tries to limit the propagation of these errors to the tag positions.

**TDOA alternative** A problem that the proposed system does not address is the fact that it is illegal in Norway to mount or fasten any UWB equipment outside, to any permanent or fixed infrastructure or installation for any period of time. This regulation can be found at [4]. The regulation does mention exceptions to this rule when the use of the interference reduction techniques 'Low Duty Cycle' or 'Detect and Avoid' are in use, which would need to be investigated and perhaps implemented in the system. Another possible solution to this restriction is to have the mounted anchor nodes only receive UWB signals, as opposed to both receive and transmit. The regulation does not specify whether it is limited to transmitters or not, which would need to be confirmed. In the case a system with permanently mounted UWB receivers would be allowed, one could use TDOA for positioning instead of SDS-TWR distance measurements as the tags would be the only required transmitter, which is a mobile unit. The anchors would then need to synchronize their clocks to be able to measure the time difference of arrival from the

tags, which could possibly be done with the time synchronized pulse-per-second signal from the GPS receiver, which is synchronized with the very accurate GPS satellite clocks. A TDOA positioning algorithm for inaccurate anchor positions and asynchronous transmitter is proposed in [22]

**GPS stationary mode** As described in section 5.1.2, the GPS module was set to static mode, which is supposed to make the positions more accurate over time. Unfortunately the verification setup was only allowed to run for a few hours, but the GPS measurements showed no improvements after the first 10-100 anchor position fixes, or first couple of minutes. A moving average of the anchor position error was calculated over all anchor position data sets, with a window size of 10 positions. The few anchor nodes who showed any notable change in accuracy at all did so within the first 100 measurements. One would perhaps observe an improvement over a longer time period than a few hours, long enough for the satellite constellation to completely change. When these first inaccurate positions were not a part of the mean, the anchor position error went down by 1 - 4 mm, which can be considered negligible when the remaining error is still several meters.

**Power consumption** Due to limited development time, the various configurations of the Radino32 and DW1000 UWB module was not explored and the power consumption could probably have been optimized further, if nothing else at least by turning the module of at regular intervals. This interval would then have to be synchronized on all tags in the network, and would possibly lower the number of tags that could be operated simultaneously. The high power consumption of the anchor nodes are not critical, as they are supposed to be mounted and connected to power infrastructure. The tags does not need other modules than the Radino32 to be powered, but the continuous draw of over 200 mA when the DW1000 active is too much to be able to compete with with the power consumption of even high accuracy GPS modules. This should be explored further, as very short bursts of current consumption at 200 mA could possibly compete with a GPS continuously drawing 20 mA in tracking mode. Important to note is the fact that GPS modules commonly do not have the possibility to transmit their measurements to base stations or anchor nodes like the UWB system does, so the 20mA would be added to the consumption of a communications module on a potential tracking device. This would further decrease the difference in power consumption between the system proposed here and a system

based solely on GPS positioning.

**Distances between anchors** A possibility to improve the positioning accuracy is to have the anchors measure the distance between themselves. This could be used to more accurately determine the anchor topology, and thus their positions. This is however not compatible with the suggestion to use TDOA with anchors in strict reception.

**GPS antenna** At the time of the final verification, proper GPS antennas were not available. They are typically right hand circular polarized, and in this case would be optimized for the GPS L1 frequency 1575.42 MHz. Instead, purely out of necessity, the same antennas that were used as NB-IoT antennas were also used for the GPS signals. They are linear polarized and optimized for the frequency ranges 698-960 MHz and 1710-2700 MHz. This should not affect the accuracy of the GPS positioning a significant amount other than decreasing signal strength, and the GPS modules reported they successfully used signals from 10-20 satellites to find their positions. Still, given that the GPS position error were larger than expected, this should be avoided in further work.

**Indoor use** The proposed system in this project could be altered to work indoors without much adjustment. A feature could be added to the user frontend allowing the user to manually type in the coordinates the each anchor. If coupled with a flag in the database, the trilateration implementation could be altered to calculate the position in either global coordinates or a local positioning system defined by the user. This could be implemented without any changes to the hardware, and would function similarly to several commercially available indoor positioning systems.

# Summary

The system proposed in this project works as intended by using a UWB local positioning system with self positioned anchor nodes and NB-IoT for data link to the remote calculation server. This functionality is demonstrated successfully, however with limited or no advantage over the already widely available GPS system. The horizontal error of the final calculations were 5.1m and 2.5m, which is slightly better than than the average horizontal error of the GPS positioned anchor nodes, although this is hardly conclusive with two data points. The power consumption of the tags is also higher than that of a low power GPS module, at over 200 mA versus 20 mA. This can probably be improved however, by further development on power cycling the UWB module and possibly utilize less power hungry modes of the DW1000 chip. Besides this, the NB-IoT communication is a success, the position calculation algorithm works well despite large anchor position errors, and the implemented device works as intended when coupled with a more capable power source.

# Bibliography


[1] Information technology — real-time locating systems (rtls) — part 1: Application programming interface (api), 2014. ISO/IEC 24730-1:2014.

[2] 3GPP. *TS 36.133; Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*, 4 2018. Rel. 15.

[3] L. Ahlin, Gazelle Book Services Limited, and J. Zander. *Principles of Wireless Communications*. Gazelle Book Services Limited, 1998.

[4] Norwegian Communications Authority. Forskrift om generelle tillatelser til bruk av frekvenser (fribruksforskriften), paragraph 31, 2012. `https://lovdata.no/SF/forskrift/2012-01-19-77/\T1\textsection31`.

[5] DecaWave. *DW1000 datasheet*, 2015. "`https://www.decawave.com/sites/default/files/resources/dw1000-datasheet-v2.09.pdf`".

[6] In-Circuit. *Radino32 DW1000 datasheet*, 6 2017. "`http://wiki.in-circuit.de/images/6/63/305000092A_radino32_DW1000.pdf`".

[7] B. Li, N. Wu, H. Wang, and J. Kuang. Nodes localization with inaccurate anchors via em algorithm in wireless sensor networks. In *2014 IEEE International Conference on Communications Workshops (ICC)*, pages 121–126, June 2014.

[8] Linear Technology. *LTC3521 datasheet*, 2013. "`http://www.analog.com/media/en/technical-documentation/data-sheets/3521fb.pdf`".

[9] Y. Liu and Z. Yang. *Location, Localization, and Localizability: Location-awareness Technology for Wireless Networks*. Springer New York, 2010.

[10] N. H. Lu. Linearized, unified two-ray formulation for propagation over a plane earth. *2005 Sensors for Industry Conference*, pages 95–100, 2005.

[11] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[12] Christian Mensing and Simon Plass. Positioning algorithms for cellular networks using tdoa. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 4, pages IV–IV. IEEE, 2006.

[13] J J Moré, B S Garbow, and K E Hillstrom. User guide for MINPACK-1. Technical Report ANL-80-74, Argonne Nat. Lab., Argonne, IL, Aug 1980.

[14] Md. Munjure Mowla and S. M. Mahmud Hasan. Performance improvement of papr reduction for ofdm signal in lte system. *CoRR*, abs/1404.2233, 2013.

[15] Spilker James J. Jr. Axelrad Penina Enge Per Parkinson, Bradford W. 11.5 standard error tables. In *Global Positioning System, Volume 1 - Theory and Applications*. American Institute of Aeronautics and Astronautics, 1996. ISBN:978-1-56347-106-3.

[16] Spilker James J. Jr. Axelrad Penina Enge Per Parkinson, Bradford W. 9.4. user process models. In *Global Positioning System, Volume 1 - Theory and Applications*. American Institute of Aeronautics and Astronautics, 1996. ISBN:978-1-56347-106-3.

[17] Quectel. *L96 Hardware Design; Rev. V1.1*, 2015. "https://www.quectel.com/UploadImage/Downlad/Quectel_L96_Hardware_Design_V1.1.pdf".

[18] Quectel. *BC95 Hardware Design; Rev. V1.5*, 12 2017. "http://quectel.com/UploadImage/Downlad/Quectel_BC95_Hardware_Design_V1.5.pdf".

[19] Mohit Sanguri. Gps survey techniques, 2010. "https://www.brighthubengineering.com/geotechnical-engineering/63637-gps-survey-techniques/#imgn_1".

[20] STmicroelectronics. *STM32L15xCC datasheet*, 8 2017. "http://www.st.com/content/ccc/resource/technical/document/datasheet/2a/6e/97/91/cd/c0/43/8b/DM00048356.pdf/files/DM00048356.pdf/jcr:content/translations/en.DM00048356.pdf".

[21] Karl Svantorp. Lte nb-iot otdoa positioning of roadside equipment, 2017. Semester project at NTNU, for AIC.

[22] Y. Zou and Q. Wan. Asynchronous time-of-arrival-based source localization with sensor position uncertainties. *IEEE Communications Letters*, 20(9):1860–1863, Sept 2016.

# Appendix

## I Server Source Code

### I.I Trilateration Algoritm Implementation

trilateration.py

```
1  import pymongo
2  import pyproj
3  import numpy as np
4  import numpy.matlib as npm
5  from scipy.optimize import least_squares
6
7  ecef = pyproj.Proj(proj='geocent', ellps='WGS84', datum='WGS84')
8  lla = pyproj.Proj(proj='latlong', ellps='WGS84', datum='WGS84')
9
10 def trilat(tagLocE,distanceN,anchorLocE):
11     distanceEst = np.sqrt(np.sum(np.square(anchorLocE - npm.repmat(tagLocE
12     res = np.absolute(distanceEst) - distanceN
13     return res.tolist()
14
15 client = pymongo.MongoClient('mongodb://localhost:27017')
16
17 db = client['UWBPositioning']
18 tags =  list(db['Tags'].aggregate([{
19     "$lookup":{
20         'from':"Distances",
21         'localField':"tag_id",
22         'foreignField':"tag_id",
23         'as':"distance"
```

```
24      }
25  }]))
26
27  anchors = list(db['Anchors'].find({}))
28  if len(anchors) < 3:
29      exit(0)
30  for idx,tag in enumerate(tags):
31      if len(tag['distance']) < 3:
32          continue
33      tags[idx]['distanceNoisy'] = np.array([])
34      tags[idx]['anchorLocEst'] = np.array([]).reshape(0,3)
35      for ind,dist in enumerate(tag['distance']):
36          anchor = next((item for item in anchors if item['anchor_id'] == d
37          if anchor is None:
38              continue
39          anchor_pos = {'lat':0,'lon':0,'alt':0}
40          anchor_pos['lat'] = sum([float(hist_item['latitude']) for hist_ite
41          anchor_pos['lon'] = sum([float(hist_item['longitude']) for hist_it
42          anchor_pos['alt'] = sum([float(hist_item['altitude']) for hist_ite
43          anchor_ecef = list(pyproj.transform(lla, ecef, anchor_pos['lon'],
44          tags[idx]['anchorLocEst'] = np.append(tag['anchorLocEst'],[anchor
45          dist_mean = sum([float(hist_item['distance']) for hist_item in di
46          tags[idx]['distanceNoisy'] = np.append(tag['distanceNoisy'],dist_m
47      f = lambda x: trilat(x,tags[idx]['distanceNoisy'],tags[idx]['anchorLo
48      sol = least_squares(f,tags[idx]['anchorLocEst'][np.argmin(tags[idx]['
49      tag_pos = list(pyproj.transform(ecef,lla,sol.x[0],sol.x[1],sol.x[2]))
50      db['Tags'].update_one({'tag_id':tag['tag_id']},{'$set':{'coordinates'
```

## I.II Incoming NodeJS UDP Server

server.js

```
1  var udp = require('dgram');
```

```
2  var MongoClient = require('mongodb').MongoClient
3
4  var server = udp.createSocket('udp4');
5  const {exec} = require('child_process')
6  var database;
7  MongoClient.connect('mongodb://localhost:27017', function (err, client) {
8    if (err) throw err
9
10   database = client.db('UWBPositioning');
11 })
12
13 server.on('error',function(error){
14   console.log('Error: ' + error);
15   server.close();
16 });
17
18 server.on('message',function(msg,info){
19   console.log('Recieved: '+msg.toString());
20   console.log('From: ' + info.address+':'+info.port);
21   var msg_str = msg.toString();
22   if(msg_str.startsWith('#')){
23     //parse UWB
24     var uwb_msg = msg_str.slice(1,-1).split(',');
25     var filter = {anchor_id:uwb_msg[0],tag_id:uwb_msg[2]};
26     var uwb_obj = {
27       time:uwb_msg[1],
28       distance:uwb_msg[3],
29       batlevel:uwb_msg[4],
30       remoteRXPower:uwb_msg[5],
31       RXPower:uwb_msg[6]};
32     database.collection('Distances').updateOne(filter,{$push:{history:{$e
```

```
33        if (err) throw err;
34
35      })
36      database.collection('Tags').insertOne({tag_id:uwb_msg[2]}).catch(func
37        throw err;
38      })
39
40
41    } else if (msg_str.startsWith('GPGGA')) {
42      //parse GPS
43      var gps_msg = msg_str.split(',');
44      var filter = {anchor_id: gps_msg[gps_msg.length-1]}
45      var gps_obj = {time:gps_msg[1],
46        latitude:gps_msg[2],
47        ns:gps_msg[3],
48        longitude:gps_msg[4],
49        ew:gps_msg[5],
50        fix_status:gps_msg[6],
51        sattelites:gps_msg[7],
52        hdop:gps_msg[8],
53        altitude:gps_msg[9],
54        alt_unit:gps_msg[10],
55        geoid_sep:gps_msg[11],
56        geoid_sep_unit:gps_msg[12]
57      };
58      var dotIndex = gps_obj.latitude.indexOf('.');
59      gps_obj.latitude =Number(gps_obj.latitude.slice(0,dotIndex-2))+ Numbe
60      dotIndex = gps_obj.longitude.indexOf('.');
61      gps_obj.longitude =Number(gps_obj.longitude.slice(0,dotIndex-2))+ Num
62
63      database.collection('Anchors').updateOne(filter,{$push:{history:{$eac
```

```
64        if (err) throw err;
65
66      })
67
68    }
69
70 });
71
72 server.on('listening',function(){
73    var address = server.address();
74    var port = address.port;
75    var family = address.family;
76    var ipaddr = address.address;
77    console.log('Server is listening at port ' + port);
78    console.log('Server ip :' + ipaddr);
79    console.log('Server is IP4/IP6 : ' + family);
80 });
81
82 server.on('close',function(){
83    console.log('Socket is closed !');
84 });
85
86 setInterval(function() {
87    exec('python trilateration.py',(err,stdout,stderr) => {
88      if (err) throw err;
89    })
90 },10000)
91
92 server.bind(1337);
```

## I.III Server Backend Of User Interface

routes.js

```
router.get('/uwb-tags', function(req,res) {
  var db =req.app.locals.uwbdb;
  db.collection('Tags').aggregate([
   { $project:
     {
       coordinates:{ $slice: [ "$coordinates", -1 ] },
       tag_id: "$tag_id"
     }
  }]).toArray(function(err,arr){
    if (err) throw err;
    console.log(arr);
    geojson = arr.reduce(function(aggr,val,idx,arr){
      if (val.coordinates.length == 0) {
        return aggr;
      }
      aggr.features.push({
        type:'Feature',
        geometry:{
          type:'Point',
          coordinates:[val.coordinates[0].lat, val.coordinates[0].lon]
        },
        properties:{
          tag_id: val.tag_id
        }
      });
      return aggr;
    },{type:'FeatureCollection', features:[]})
    res.json(geojson)
  });
})
```

```
31
32 router.get('/uwb-anchors', function(req,res) {
33   var db =req.app.locals.uwbdb;
34   db.collection('Anchors').aggregate([
35    { $project:
36      {
37          coordinates:{ $slice: [ "$history", -100 ] },
38          anchor_id: "$anchor_id"
39      }
40   }]).toArray(function(err,arr){
41     geojson = arr.reduce(function(aggr,val,idx,arr){
42       let mean_lat = val.coordinates.reduce(function(sum,a){
43         sum += Number(a.latitude);
44         return sum;
45       },0)/val.coordinates.length;
46       let mean_lon = val.coordinates.reduce(function(sum,a){
47         sum += Number(a.longitude);
48         return sum;
49       },0);
50       mean_lon = mean_lon/val.coordinates.length
51       val.coordinates = [mean_lon, mean_lat];
52       aggr.features.push({
53         type:'Feature',
54         geometry:{
55           type:'Point',
56           coordinates:val.coordinates
57         },
58         properties:{
59           anchor_id: val.anchor_id
60         }
61       });
```

```
62          return aggr;
63      },{type:'FeatureCollection', features:[]})
64      res.json(geojson);
65    });
66 })
```

## I.IV Frontend Of User Interface

map.js

```
1 map.addSource('anchors',{type:'geojson',data:'/uwb-anchors'});
2       map.addSource('tags',{type:'geojson',data:'/uwb-tags'});
3       map.addLayer({
4           "id": "tag-points",
5           "type": "circle",
6           "source": 'tags',
7           "paint": {
8               "circle-color": "Red",
9           }
10      });
11      map.addLayer({
12          "id": "anchor-points",
13          "type": "circle",
14          "source": 'anchors',
15          "paint": {
16              "circle-color": "Black",
17          }
18      });
19      map.addSource('realtime',{
20        type:'geojson',
21        data:'/realtime',
22        cluster: true,
23        clusterMaxZoom: 12,
```

```
24      });
```

# II Device Source Code

The firmware implemented on the devices is an addition to the dw1000_advanced_demo library which can be downloaded at http://wiki.in-circuit.de/index.php5?title=radino_Library#Downloads Minor modifications was applied to this library for it to interface with the code in this section.

## II.I NB-IoT module interface

iot.cpp

```
1  #include "iot.h"
2
3  const byte numChars = 100;
4  char generalResponse[numChars];
5  char paramResponse[numChars];
6  char okResponse[3];
7  char errorResponse[6];
8  bool params = false;
9  bool ok = false;
10 bool error = false;
11
12 unsigned int timeout_counter = 0;
13 int socket;
14 char serverIP[] = "52.174.177.192";
15 char serverPort[] = "1337";
16
17
18 bool Iot::init() {
19   IotSerial.begin(9600);
20   if(!Serial){
21     Serial.begin(57600);
```

```
22    }
23    int i = 0;
24    while(i<10){
25      delay(200);
26      iwdg_reset();
27      i++;
28    }
29    connected = false;
30    IotSerial.println("AT+CFUN=1");
31    IotSerial.flush();
32    Serial.println("AT+CFUN=1");
33    if (!gotOK()) {Serial.println("error?");}
34    else {Serial.println("OK!");}
35    IotSerial.println("AT+COPS=1,2,\"24202\"");
36    IotSerial.flush();
37    Serial.println("AT+COPS=1,2,\"24202\"");
38    if (!gotOK()) {Serial.println("plmn error?");}
39    else {Serial.println("OK!");}
40    IotSerial.println("AT+CGATT=1");
41    IotSerial.flush();
42    Serial.println("AT+CGATT=1");
43    if (!gotOK()) {Serial.println("attach error?");}
44    else {Serial.println("OK!");}
45    if(!checkConnection()) {
46      Serial.println("connection error?");
47    }
48    socket = makeSocket(0);
49    if (socket == -1) {
50      Serial.println("socket error?");
51    }
52    requestInit = false;
```

```
53
54 }
55
56
57 bool Iot::checkConnection() {
58   IotSerial.println("AT+CEREG?");
59   Serial.println("AT+CEREG?");
60   receive();
61   lastCheck = millis();
62   if(params){
63     if (gotOK()){
64       int endIndex = strlen(paramResponse)-1;
65       if(paramResponse[endIndex]=='1' || paramResponse[endIndex]=='5'){
66         connected=true;
67         return true;
68       }
69     }
70   }
71   connected = false;
72   return false;
73 }
74
75 bool Iot::sendString(const char* str){
76   Serial.println("send start!");
77   if(!checkConnection()) {return false;}
78   if(socket == -1) {return false;}
79   int strlength = strlen(str);
80   //char* hello_counter_str;
81   //asprintf(&hello_counter_str,"%i",hello_counter);
82   //int counter_len = strlen(hello_counter_str);
83   int totlen = strlength;
```

```
84    char hex[totlen*2 +1];
85    int i;
86    for (i = 0; i < strlength; i++) {
87      sprintf(&hex[i*2],"%x",(unsigned int) str[i]);
88    }
89    //for (i = i; i < totlen; i++) {
90    //   sprintf(&hex[i*2], "%x",(unsigned int) hello_counter_str[i-strlengt
91    //}
92    //strlength++;
93    char *cmdStr;
94    asprintf(&cmdStr,"AT+NSOST=%i,%s,%s,%i,%s",socket,serverIP,serverPort,s
95    IotSerial.println(cmdStr);
96    IotSerial.flush();
97    Serial.println(cmdStr);
98    free(cmdStr);
99    //free(hello_counter_str);
100   receive(); //get number of characters transmitted
101   if(params && atoi(&paramResponse[2]) ==totlen){
102     Serial.print("sent:␣");
103     Serial.println(strlength);
104   }
105   if(gotOK()) {
106     //hello_counter++;
107     return true;
108   }
109
110 }
111
112 int Iot::makeSocket(unsigned int callerID){
113   char *cmdStr;
114   asprintf(&cmdStr,"AT+NSOCR=DGRAM,17,%s,1",serverPort);
```

```
115     IotSerial.println(cmdStr);
116     Serial.println(cmdStr);
117     receive();
118     if(params && (strlen(paramResponse) == 1)){
119       Serial.println("socket␣params␣ok");
120       if (gotOK()){
121         Serial.print("socket:␣");
122         Serial.println(atoi(paramResponse));
123         return atoi(paramResponse);
124       }
125     } else if (callerID != 0){
126       return -1;
127     }
128     IotSerial.println("AT+NSOCL=0");
129     Serial.println("AT+NSOCL=0");
130     if(gotOK()) {
131       return makeSocket(callerID +1);
132     } else {
133       return -1;
134     }
135 }
136
137 bool Iot::gotOK() {
138     receive();
139     if (ok && strcmp(okResponse,"OK")==0){
140       ok=false; error = false; params = false;
141       return true;
142     } else {
143       ok=false; error = false; params = false;
144       return false;
145     }
```

```
146 }
147
148 void Iot::receive() {
149     static boolean recvInProgress = false;
150     static byte ndx = 0;
151     char startMarker = '\n';
152     char endMarker = '\n';
153     char rc;
154     unsigned int waitCount = 0;
155     while (IotSerial.available() == 0) {
156       Serial.println("waiting...");
157       delay(50);
158       iwdg_reset();
159       waitCount++;
160       if (waitCount >= 4) {
161         Serial.println("iot receive timeout");
162         timeout_counter++;
163         if (timeout_counter >= 30){
164           requestInit = true;
165           timeout_counter=0;
166         }
167         return;
168       }
169     }
170     Serial.println("receiving...");
171     while (IotSerial.available() > 0) {
172         rc = IotSerial.read();
173         if (recvInProgress == true) {
174             if (rc != endMarker) {
175                 if(ndx == 0){
176                   switch (rc) {
```

```
177                     case 'O':
178                       ok = true;
179                       break;
180                     case 'E':
181                       error = true;
182                       break;
183                     default:
184                       params = true;
185                   }
186                 }
187                 generalResponse[ndx] = rc;
188                 ndx++;
189                 if (ndx >= numChars) {
190                     ndx = numChars - 1;
191                 }
192               } else {
193                 generalResponse[ndx-1] = '\0';
194                 if (ok){
195                   strcpy(okResponse,generalResponse);
196                 } else if (error) {
197                   strcpy(errorResponse,generalResponse);
198                 } else if (params){
199                   strcpy(paramResponse,generalResponse);
200                 }
201                 recvInProgress = false;
202                 ndx = 0;
203                 timeout_counter = 0;
204                 Serial.print("got:␣");
205                 Serial.println(String(generalResponse));
206                 if (strstr(generalResponse,"REBOOT") != NULL || strstr(ge
207                   requestInit = true;
```

```
208                     }
209                     return;
210                 }
211             }
212
213             else if (rc == startMarker) {
214                 recvInProgress = true;
215             }
216
217         }
218 }
```

## II.II GPS module interface

gps.cpp

```
 1
 2 #include "gps.h"
 3
 4
 5 bool Gps::init() {
 6   GpsSerial.begin(9600);
 7   //Serial.begin(57600);
 8   int i = 0;
 9   while(i<10){
10     delay(200);
11     iwdg_reset();
12     i++;
13   }
14   GpsSerial.println("$PMTK314,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0*28");
15   GpsSerial.flush();
16   //clearBuf();//get response of pmtk314 to clear buffer
17   GpsSerial.println("$PMTK886,4*2C");// set to stationary mode, for highe
```

```
18    GpsSerial.flush();
19    //delay(50);
20    //clearBuf();//get response of pmtk886 to clear buffer
21    iwdg_reset();
22    receive();//get RMC and GGA messages
23    if (strncmp(generalResponse,"GPRMC",5) == 0){
24      strcpy(rmc,generalResponse);
25    } else if (strncmp(generalResponse,"GPGGA",5) == 0){
26      strcpy(gga,generalResponse);
27    }
28    receive();
29    if (strncmp(generalResponse,"GPRMC",5) == 0){
30      strcpy(rmc,generalResponse);
31    } else if (strncmp(generalResponse,"GPGGA",5) == 0){
32      strcpy(gga,generalResponse);
33    }
34    //Serial.println("RMC:");
35    //Serial.println(rmc);
36    //Serial.println("GGA:");
37    //Serial.println(gga);
38    clearBuf();
39    lastCheck = millis();
40  }
41
42  void Gps::receive() {
43      static boolean recvInProgress = false;
44      static boolean checksumRcv = false;
45      memset(generalResponse,0,numChars);
46      static byte ndx = 0;
47      char startNMEA = '$';
48      char startChecksum = '*';
```

```
49      char endMarker = '\r';
50
51      char rc;
52      unsigned int waitCount = 0;
53      //Serial.print("gps available: ");
54      //Serial.println(GpsSerial.available());
55      while (GpsSerial.available() == 0) {
56        //Serial.println("waiting on gps..");
57        delay(50);
58        iwdg_reset();
59        waitCount++;
60        if (waitCount >= 20) {
61          //Serial.println("gps receive timeout");
62          return;
63        }
64      }
65      //Serial.println("receiving from gps...");
66
67      while (GpsSerial.available() > 1) {
68          rc = GpsSerial.read();
69          if (recvInProgress == true) {
70              if (rc != startChecksum) {
71                  generalResponse[ndx] = rc;
72                  ndx++;
73                  if (ndx >= numChars) {
74                      ndx = numChars - 1;
75                  }
76              } else {
77                  generalResponse[ndx] = '\0';
78                  recvInProgress = false;
79                  checksumRcv = true;
```

```
80                    ndx = 0;
81                    //Serial.print("got: ");
82                    //Serial.println(String(generalResponse));
83                }
84            } else if (checksumRcv) {
85              if (rc != endMarker) {
86                checksum[ndx] = rc;
87                ndx++;
88                if(ndx >= 3){
89                  ndx--;
90                }
91              }else{
92                checksum[2] = '\0';
93                checksumRcv = false;
94                ndx=0;
95                //Serial.print("checksum: ");
96                //Serial.println(checksum);
97                if(!checksumChecker(generalResponse,checksum)){
98                  memset(generalResponse,0,numChars);
99                }
100               return;
101             }
102           } else if (rc == startNMEA) {
103               recvInProgress = true;
104           }
105           delay(1);
106       }
107 }
108
109 void Gps::clearBuf() {
110   //Serial.print("clearing GPS buffer: ");
```

```
111   char rc;
112   while (GpsSerial.available() > 1) {
113     rc = GpsSerial.read();
114     //Serial.print(rc);
115   }
116   //Serial.println("cleared!");
117 }
118
119 bool Gps::getGGA() {
120   iwdg_reset();
121   if (millis()-lastCheck > 3000){
122     clearBuf();
123   }
124   receive();
125   lastCheck = millis();
126   if (strncmp(generalResponse,"GPRMC",5) == 0){
127     strcpy(rmc,generalResponse);
128     delay(50);
129     iwdg_reset();
130     receive();
131     if (strncmp(generalResponse,"GPGGA",5) == 0){
132       strcpy(gga,generalResponse);
133       return true;
134     }
135   } else if (strncmp(generalResponse,"GPGGA",5) == 0){
136     strcpy(gga,generalResponse);
137     return true;
138   }
139   return false;
140 }
141
```

```
142 bool Gps::checksumChecker(char* str, char* cs) {
143   int XOR = 0;
144   unsigned int i;
145   for (i = 0; i < strlen(str); i++)
146   {
147       XOR ^=    str[i];
148   }
149   //Serial.print("checked sum: ");
150   //Serial.println(XOR,HEX);
151   if (XOR == (int)strtol(cs, NULL, 16)){
152     return true;
153   }
154   return false;
155 }
```