# A wireless ultrasonic positioning network using off-the-shelf devices

Martin Järve


Thesis submitted for the degree of
Master in Electrical Engineering, Informatics and Technology
60 credits

Department of Physics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

# A wireless ultrasonic positioning network using off-the-shelf devices

Martin Järve

A wireless ultrasonic positioning network using off-the-shelf devices

http://www.duo.uio.no/

Printing: Reprosentralen, Universitetet i Oslo

# Abstract

The Global Positioning System (GPS) is the standard technology for outdoor positioning, but it cannot be used for accurate indoor positioning due to scattering and attenuation of GPS signals. A popular alternative is to use ultrasound for indoor positioning because of the high accuracy and low cost of these systems, but few such systems are readily available for users. This thesis presents a wireless ultrasonic indoor positioning system that is built using commercial off-the-shelf devices. This system is easily adaptable for different applications that require accurate positioning with room-level coverage and has a low deployment effort. The positioning system presented here is implemented using a wireless sensor network, and uses Micro Bit V2 microcontroller boards as the wireless node control units, which have an onboard ultrasonic microphone. This means that an ultrasonic transducer for transmission is the only external component needed, and the Murata MA40S4S transmitters are used for this purpose. The system estimates the ultrasonic time-of-flight (ToF) between transmitting and receiving wireless nodes which are synchronized using a radio communication link. By using four transmitter nodes with known positions and the estimated ToF distances to the receiving node, the receiver is positioned by applying multilateration. The results show a 90th percentile sub-centimeter accuracy and precision in ToF distance estimation, and achieved a mean positioning error less than 3 cm in the estimated 3D coordinates with a standard deviation less than 0.41 cm.

# Acknowledgments

I would like to thank the people that made this thesis possible. To start off, I would like to thank my supervisors Ketil Røed and Sverre Holm for their guidance throughout this project. I would also like to thank SINTEF for collaborating in making this interesting project possible. It has been a fun and interesting journey to develop this ultrasonic positioning system. Thanks to ELAB for being helpful with developing the PCBs used in this project.

Thanks to my fellow students for engaging academic and non-academic discussions, and for useful tips. Finally, thanks to my girlfriend for giving feedback on my writing, and for the support and help throughout the last five years. You have made this education much more manageable.

Thank you all for being part of my academic journey and helping me achieve this milestone.

# Contents

# 1 Introduction

Location-based services have become an integrated part of our everyday lives. Its use has evolved from surveillance and navigation to become a fundamental part of social networking, entertainment, advertisement and education (Huang & Gartner, 2018). The main enabler of location-based services is the Global Positioning System (GPS), which has become the standard technology for outdoor positioning. The typical GPS positioning accuracy with smartphones is in the range of several meters when located in the line-of-sight of the positioning satellites, but it cannot be used for accurate positioning indoors due to scattering and attenuation of GPS signals (Puricer & Kovar, 2007). There is therefore a need for reliable and accurate indoor positioning standards that could serve as an extension of the satellite positioning system. Applications include navigation in shopping centers and airports, and localizing objects in warehouses and virtual reality.

While many technologies exist for indoor positioning, they all have their own drawbacks and one must consider a balance of cost, accuracy and coverage area among other design considerations when deciding for a positioning technology. In contrast to outdoor positioning, indoor positioning comes with major challenges due to the variety of indoor space layout, multipath signal propagation, and a lack of standards. As a result, there is a demand for low-cost, easy to deploy positioning system that is applicable for most indoor environments.

Ultrasonic positioning can provide high accuracy at a low cost (Ureña et al., 2018), mainly due to the lower wave speed and frequency which means cheaper hardware with lower frequency processing can be used. While electromagnetic waves propagate through walls and can cover multiple rooms and floors, acoustic waves are confined to the room of the transmitter and require more nodes to cover the same area. This can be an issue for scalability, or can be exploited using fingerprinting[1] or if the application requires binary room-detection[2].

[1] Refer to chapter 2.1.1 under the explanation of RSS for a brief explanation of fingerprinting.
[2] A binary room-detection positioning system works by simply giving a “yes” or “no” answer based on the presence or absence of a signal.

There are many ultrasonic indoor positioning systems based on wireless sensor networks that can achieve sub-centimeter positioning accuracy (Medina et al., 2013b; Qi & Liu, 2017), but using ultrasonic signals require custom designed hardware capable of transmitting and receiving ultrasound, and are thus not readily available for users. There has not been much focus on designing acoustic positioning systems using commercial off-the-shelf (COTS) devices and further research in the field is needed (Liu et al., 2020). Some proposed systems have used COTS speakers, amplifiers and soundcards, which may be pre-installed in a room, to achieve accuracy below 2 cm using signals in the audible frequency range (Moutinho et al., 2016; Sertatıl et al., 2012). These systems have the advantage of using already existing infrastructure to achieve accurate positioning, but they lack the advantages of wireless sensor networks; which are usually much smaller in size, easily deployable and scalable.

With improvements in availability, reduced cost and increased processing power of commercial microcontroller boards in the recent years, small wireless sensor node networks can be made with low cost. The aim of this thesis is to investigate if a wireless sensor network based ultrasonic indoor positioning system can be built using COTS hardware, which has a low deployment effort, centimeter-level accuracy and room-level coverage. The system presented here is illustrated in Figure 1 and consists of ultrasonic transmitting and receiving wireless nodes. The wireless nodes consist of a Micro Bit V2 microcontroller board which has an embedded microphone with specified frequency response up to 80 kHz, meaning an ultrasonic transducer for transmission is the only external component needed. The transmitting nodes act as anchor points with known positions, while the receiving nodes will measure the time-of-flight (ToF) of ultrasonic pulses from the transmitters. The ToF measurements are then converted into distances between the transmitter-receiver pairs, and by applying multilateration the receiving nodes can be positioned and tracked.

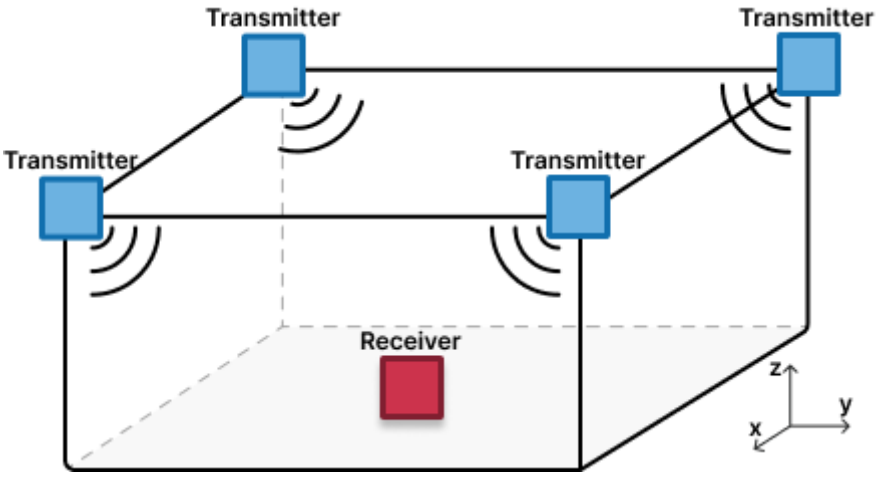



**Figure 1:** Illustration of a wireless ultrasonic positioning network. The transmitting nodes are placed at known positions and the receiving node measured the ToF of ultrasonic pulses from the transmitters. The receiver's position is found by applying multilateration on the estimated ToF distances.

The positioning system will be evaluated with regards to the main evaluation metrics of accuracy, coverage area, update rate, multi-device positioning and deployment effort. Further, this thesis presents the main challenges of building such a system, the limiting factors of the performance metrics, and discusses how improvements can be made.

# 2 Background Theory

## 2.1 Acoustics

### 2.1.1 Acoustic positioning techniques

Acoustic positioning can be achieved several ways by exploiting the physical nature of acoustic waves. By using the controlled transmission and reception of acoustic signals, and by applying models of the acoustic properties, positioning can be achieved using different techniques. The technique, or observable that is employed determines how positioning is calculated, and below are some of the most common techniques of how acoustics can be used for positioning.

- **ToF:** Time-of-flight (ToF) or time-of-arrival (ToA) is the time between the transmission and reception of a signal, and requires accurate time synchronization between the transmitter and the receiver. The measured time is used to estimate the distance between the transmitter and the receiver using the speed of sound. By using three or more distances 3D positioning can be estimated using lateration. The advantages of using ToF based positioning are the low system complexity and high accuracy, making it the most common technique for acoustic positioning and there are many variations of the ToF estimation technique.

- **TDoA:** Time-difference-of-arrival is a version of ToF which is usually implemented by using the target as the transmitter, and several anchor nodes as the receivers. The transmission time is unknown, and the relative time differences at the receivers are used to determine the distances. The advantage of this method is that the synchronization between the transmitter and the receivers can be omitted, requiring only the receivers to be synchronized, but at the cost of requiring one extra receiver node for the comparison of relative arrival times.

- **AoA:** Angle of arrival estimates the angle at which the sound waves are received by measuring the arrival time difference between several microphones. By using the estimated angles positioning can be estimated using angulation. This technique is

based on the TDoA, but instead of using the arrival time difference of individual receiver nodes, multiple receivers are placed in an array on one node. The advantage of AoA is that it does not require synchronization between nodes since the time of transmission is not used and the receivers are on the same node. However, this comes at the cost of higher computational requirements of array processing.

- **RSS:** In contrast to the positioning techniques above, received signal strength (RSS) does not measure time, but the strength of the received signal. This requires knowledge of the transmitted signal power and an accurate model of the propagation and absorption of the acoustic waves, which can be used to convert the transmitted and received signal strength into distance. Positioning can then be estimated using either lateration or fingerprinting, where the latter is a method of characterizing and storing the received echo structure at various points in a room (referred to as fingerprints), and comparing the measured signal with the stored fingerprints. RSS systems can be more robust to noise than ToF based systems and can cover an area with fewer nodes, but the positioning is usually less accurate than ToF.

### 2.1.2 Time-of-flight distance estimation

The positioning system presented here uses ultrasonic ToF between anchor nodes and a receiver node to estimate the position of the receiver using multilateration. The estimated distances using ToF are obtained indirectly by estimating the propagation time of sound, which is directly proportional to the distance covered. By using the nominal speed of sound in air $c = 343\ \mathrm{m/s}$ at 20° C, the estimated time $t$ can be converted into distance $d$ using equation (1). Using the nominal speed of sound of 343 m/s at $20℃$ can be good enough for indoor positioning, but for applications that require accurate positioning sound speed modeling should be considered.

$$d = c \cdot t \tag{1}$$

### 2.1.3 Speed of sound modeling

The sound speed depends on the temperature, relative humidity, atmospheric pressure and frequency, but temperature is the most significant factor and the other factors are most commonly neglected when modelling the sound speed. Using only temperature gives the simple model in equation (2) (Pierce, 2019, p. 30).

$$c = \sqrt{\gamma RT} \tag{2}$$

Where $\gamma = 1.4$, $R = 287\ \mathrm{J/(kg \cdot K)}$ and $T$ is the temperature in K. At $20°\ \mathrm{C}$ the speed of sound is

$$c = \sqrt{1.4 \cdot 287\ \mathrm{J/(kg \cdot K)} \cdot 293.16\ \mathrm{K}} \approx 343.2\ \mathrm{m/s}.$$

Equation (2) can also be approximated around $0°\ \mathrm{C}$ by (3) (Holm, 2019, p. 294), where the sound speed is given by only the temperature $T'$ in Celsius, giving a simpler model to use. The speed of sound is 343.4 m/s at $20°\ \mathrm{C}$ using (3).

$$c \approx 331.3 + 0.606 \cdot T' \tag{3}$$

Further, the ToF distance estimations are sensitive to changes in temperature, and using equation (2) this sensitivity can be found. The difference in sound speed due to $1°\ \mathrm{C}$ temperature difference is

$$\frac{dc}{dT} = \sqrt{1.4 \cdot 287\ \mathrm{J/(kg \cdot K)} \cdot 293.16\ \mathrm{K}} - \sqrt{1.4 \cdot 287\ \mathrm{J/(kg \cdot K)} \cdot 292.16\ \mathrm{K}} \approx 0.586\ \frac{\mathrm{m/s}}{\mathrm{K}}$$

Normalizing this gives

$$\frac{\frac{dc}{dT}}{c} = \frac{0.586\ \mathrm{m/s}}{\sqrt{1.4 \cdot 287\ \mathrm{J/(kg \cdot K)} \cdot 292.16\ \mathrm{K}}} \approx 1.7\ \mathrm{mm/K/m} \tag{4}$$

Thus, the error in the estimated ToF distance due to temperature variation is 1.7 mm per °C and per meter, meaning 1° C error gives an absolute error of 1.7 mm at 1 m, and 8.5 mm absolute error at 5 m.

### 2.1.4 Resolution

Generally, the resolution of acoustic positioning is limited by the wavelength, which is inversely proportional to the frequency. For example, in ultrasonic echo based imaging, the wavelength must be smaller than the distance between two adjacent objects in order to distinguish between the two. In ToF estimation, the number of wave peaks per second (frequency) determines the time resolution in which the received pulse can be said to be above a detection threshold, or the precise time of the peak of the received pulse. Figure 2 attempts to illustrate this by showing two pulses with the same number of periods, but one with twice the frequency of the other, and is thus much narrower in time. The wavelength of 40 kHz ultrasound at 20° C that is used here can be found with equations (1) and (2).

$$d = 343.2\ \mathrm{m/s} \cdot \frac{1}{40\ kHz} = 8.58\ \mathrm{mm}$$

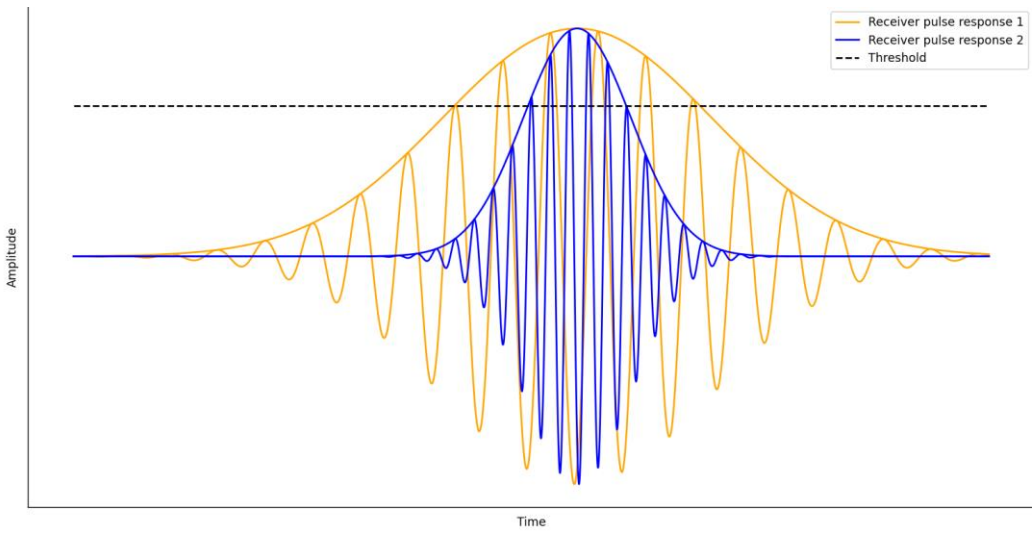



**Figure 2:** Ultrasonic receiver pulse response. Both pulses have the same number of periods, but the blue pulse has twice the frequency of the orange pulse.

### 2.1.5 Attenuation

The high attenuation of ultrasonic frequencies significantly limits the SNR at distances, and thus the positioning range. This attenuation is caused by the relaxation process of the molecules in air and is dependent on the temperature, humidity and pressure (Holm, 2019, p. 56). Acoustic wave attenuation in air is shown in Figure 3 at 20° C. Attenuation in the audible frequency range can be negligible, but at 40 kHz it may reach 1 dB/m, and 3 dB/m at 100 kHz.

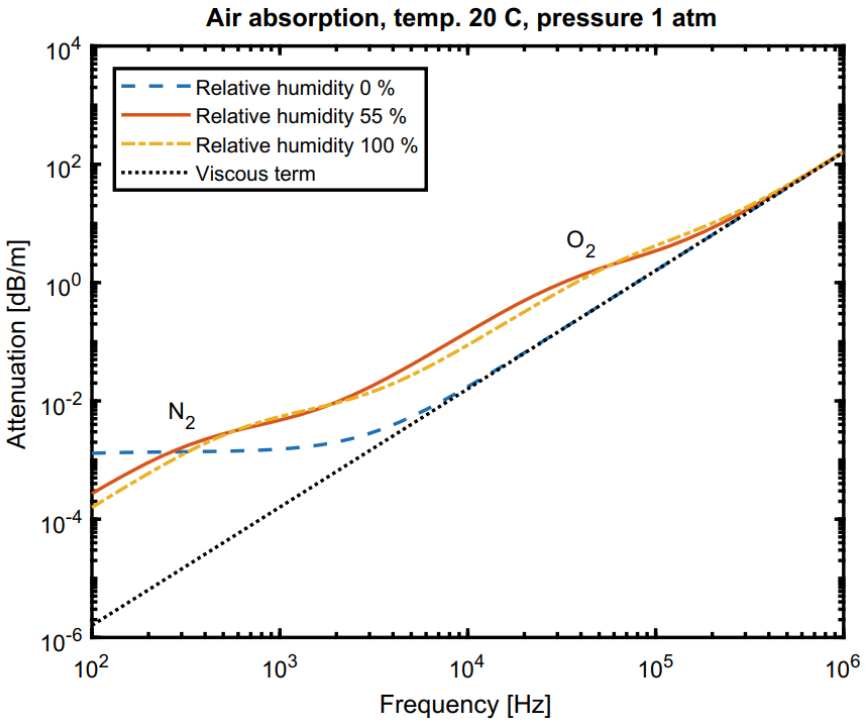



**Figure 3:** Attenuation in air at 20℃, 1 atm and for 0, 55, and 100% relative humidity. From *Waves with Power-Law Attenuation* (p. 57), by S. Holm, 2019, Springer Nature Switzerland AG. Reproduced with permission.

Since the ToF accuracy is usually limited by the wavelength, and because of the high attenuation of ultrasonic frequencies, there is a tradeoff between accuracy and range in acoustic positioning systems. Using higher frequencies for positioning means the coverage area is reduced because of the higher attenuation, but the accuracy is increased because of the shorter wavelength. However, the fact that ultrasonic frequencies are inaudible for humans is an additional advantage of using ultrasound for positioning.

## 2.2 Trilateration

Trilateration in acoustic positioning is the process of using simultaneous distances between anchor nodes and a target to calculate the position of the target. The anchor node $i$ has a known position $x_i$, $y_i$, $z_i$ and the distance to a target is $d_i$, with a total number $k$ anchor nodes. For determining the 3D coordinates $x$, $y$, $z$ of a target, three anchor nodes with distances to the target it needed. If more than three are used, then the positioning is called multilateration.

The distance $d_i$ is estimated using acoustic waves, which are spherical in nature. A system of equations (5) can be constructed using Pythagoras' formula where the distances $d_i$ are the radii of spheres, with the anchor node positions being the centers, that intersect at the target with coordinates $x$, $y$, $z$. The assumption here is that the distances $d_i$ are obtained simultaneously such that all spheres intersect at a position $x$, $y$, $z$.

$$\begin{aligned} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2 &= d_1^2 \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2 &= d_2^2 \\ \cdots+\cdots+\cdots &= \cdots \\ (x-x_k)^2+(y-y_k)^2+(z-z_k)^2 &= d_k^2 \end{aligned} \tag{5}$$

If only three anchor nodes are used, then (5) can be solved exactly by referencing the plane defined by the three anchor nodes (Fang, 1986). By choosing to place the first anchor node in the origin $(0, 0, 0)$, the second on the x-axis $(x_2, 0, 0)$, the third on $(x_3, y_3, 0)$ and all three in the z-plane, (5) can be solved with equations (6), (7) and (8), but this solution is limited by requiring exactly three anchor nodes which are placed in the same z-plane.

$$x = \frac{d_1^2 - d_2^2 + x_2^2}{2x_2} \tag{6}$$

$$y = \frac{d_1^2 - d_3^2 + x_3^2 + y_3^2 - 2x_3x}{2y_3} \tag{7}$$

$$z = \pm\sqrt{r_1^2 - x^2 - y^2} \tag{8}$$

The system of equations (5) is non-linear, but can be linearized by referencing the position of the first anchor node (Meyer & Elaksher, 2021). Then the equations can be expressed in matrix form as

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b} \tag{9}$$

where

$$\boldsymbol{A} = \begin{pmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) & 2(z_1 - z_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) & 2(z_1 - z_3) \\ 2(x_1 - x_4) & 2(y_1 - y_4) & 2(z_1 - z_4) \\ \vdots & \vdots & \vdots \\ 2(x_1 - x_k) & 2(y_1 - y_k) & 2(z_1 - z_k) \end{pmatrix}$$

$$\boldsymbol{x} = (x \quad y \quad z)^T$$

$$\boldsymbol{b} = \begin{pmatrix} x_1^2 - x_2^2 + y_1^2 - y_2^2 + z_1^2 - z_2^2 + d_2^2 - d_1^2 \\ x_1^2 - x_3^2 + y_1^2 - y_3^2 + z_1^2 - z_3^2 + d_3^2 - d_1^2 \\ x_1^2 - x_4^2 + y_1^2 - y_4^2 + z_1^2 - z_4^2 + d_4^2 - d_1^2 \\ \vdots \\ x_1^2 - x_k^2 + y_1^2 - y_k^2 + z_1^2 - z_k^2 + d_k^2 - d_1^2 \end{pmatrix}$$

The cost of doing this is that one more anchor node is needed, but they can now be placed freely. If four anchor nodes are used then $\boldsymbol{A}$ is a square matrix and can be solved exactly using (10).

$$\boldsymbol{x} = \boldsymbol{A^{-1}b} \tag{10}$$

If more than four anchor nodes are used, meaning the system of equations is overdetermined, the solution can then be obtained using least squares with (11).

$$\boldsymbol{x} = (\boldsymbol{A}^T \cdot \boldsymbol{A})^{-1} \cdot (\boldsymbol{A}^T \cdot \boldsymbol{b}) \tag{11}$$

# 3 System Design

This chapter presents the design considerations and requirements of the wireless sensor network based ultrasonic positioning system, their importance in this thesis, and how they are addressed in the design of the system. The choice of hardware components and methods used to realize such a positioning system are also presented here. The following are some common design requirements of acoustic positioning systems:

- **Synchronization:** The positioning technique determines if time synchronization is needed between the wireless nodes. In ToF systems the transmitters and receivers have to be synchronized to measure the time from the transmission of signal, to the reception. This is usually achieved by a radio communication link between the nodes since the radio signal propagation time can be neglected. Other techniques such as TDoA, AoA and RSS do not use the transmission time and do not require synchronization between the transmitters and receivers. Accurate time synchronization is an important part in ultrasonic positioning networks and solutions to the challenges are presented in Medina et al. (2013a). There are different methods for synchronizing wireless nodes, and chapter 3.4.1 covers synchronization in more detail.

- **Accuracy:** A positioning system's accuracy is measured by how large the error in the obtained positioning is with regards to the ground truth. High accuracy means small error and the accuracy requirement is application dependent. The accuracy limit is determined by the wavelength of the signal, but in normal indoor conditions multipath effects, in-band noise and dilution of precision are usually the limiting factors. Dilution of precision is the error in positioning due to the geometric distribution of transmitters and receivers (Langley, 1999). Sub-centimeter accuracy has been achieved with wireless sensor networks using ultrasonic ToF estimation (Medina et al., 2013b; Qi & Liu, 2017)

- **Update rate:** This is how often a positioning system can update the location of the target. For tracking of fast moving targets, a faster update rate is required compared to locating static objects. The update rate depends on the multiple access protocol that is

employed, the complexity of the signal processing, and is limited by the propagation speed of the signal and its multipath reflections.

- **Coverage area:** The system operating area is determined by the distribution and the number of transmitters and receiver, and is affected by dilution of precision. Due to sound being reflected off walls and solid surfaces, the coverage area is limited to the room size the system is deployed in.

- **Privacy:** Positioning systems can be divided into centralized and decentralized systems. In centralized systems the positioning of targets is computed on a central unit, whereas in decentralized systems the targets compute their own positions which are kept privately at the target. Whether a centralized or a privacy-oriented positioning system is requires is application dependent.

- **Deployment effort:** The required coverage area and the complexity of the positioning technique affect the deployment effort. Whether the positioning system needs proprietary hardware, or can be setup using existing infrastructure or using COTS devices affects the deployment effort, time and cost at the user end.

- **Cost:** Low cost is always desirable. The cost is mainly determined by the cost of the hardware, installation and the number of nodes needed to build the positioning system. Higher accuracy usually means faster processing requirements and more costly hardware. The size of the positioning area determines the number of nodes needed and reception range requirements of the hardware.

- **Multi-user capability:** The number of devices that can be located simultaneously with the positioning system. A positioning system should be easily scalable for multiple users that go in and out of the positioning area. Depending on the positioning technique a multiple access protocol might be necessary to share the ultrasound in air medium between the targets.

- **Scalability:** Scaling acoustic positioning systems to new areas can be costly since the signals do not propagate through walls and because of the high attenuation of ultrasound frequencies. Depending of the positioning technique and the number of nodes needed it can be more or less deployment effort and costly to scale the system to new areas.

There are no absolute criteria for this positioning system since this system is not application specific, but the goal is to achieve best performance with regards to the evaluation metrics presented above, with the most important begin accuracy, coverage area, update rate, multi-device positioning and deployment effort, in addition to being built using off-the-shelf hardware. The hardware components, positioning methods and techniques are chosen with regards to the above mentioned considerations.

## 3.1 Wireless node control unit

The choice of wireless node control unit is mainly determined by the cost, availability and processing power. The control unit needs to perform processing required to control an ultrasonic transducer and perform analog-to-digital conversion and ToF computation. Based on the design considerations, a general-purpose microcontroller is the better alternative over a custom microcontroller board and FPGA devices because of the greatly reduced cost and shorter development time, and the advantage of accurate parallel data processing of an FPGA does not improve the system node proposed here.

### 3.1.1 Micro Bit V2

Different microcontroller boards were considered for this thesis, with the main requirements being that it should be easy and fast to program, and it should have digital and analog interfaces such as a pulse width modulation (PWM) module and analog-to-digital converter (ADC) capable of a sampling rate at least twice the operating frequency. Arduino boards are most commonly used for amateur prototyping as they support many hardware and software modules, and have a huge supporting community which makes programming microcontrollers accessible for novice programmers. The Micro Bit V2 is the latest version of the educational boards designed by the British Broadcasting Corporation (BBC). It was designed to teach schoolchildren computer programming and is readily available and easy-to-use open-source platform, which is great for prototyping. Besides the support for different programming

languages, it does not offer any additional functionality from the Arduino platform. Perhaps because of that, and because it is targeted at schoolchildren, it lacks the user community.

The main reasons for choosing the Micro Bit V2 is because it has on-board Knowles SPU0410LR5H-QB-7 MEMS microphone which is rated up to 80 kHz, and it supports Bluetooth Low Energy (BLE) and low level radio communications, which means all of the necessary hardware is onboard the Micro Bit, except for the ultrasonic transmitter. The 2.4 GHz transceiver supports the microbit-radio protocol which enables broadcasting of small radio packets between Micro Bit boards. The support for radio communication is important because this is used to send synchronization messages between the wireless nodes. The board's target microcontroller is a 64 MHz nRF52833 with an embedded 12 bit, 200 ksps ADC, and several GPIO pins with support for PWM output and analog input. The ADC will be used to sample the analog microphone output and the PWM will be used as excitation signal for the ultrasonic transducer. Figure 4 shows the Micro Bit V2 board and its main hardware components.

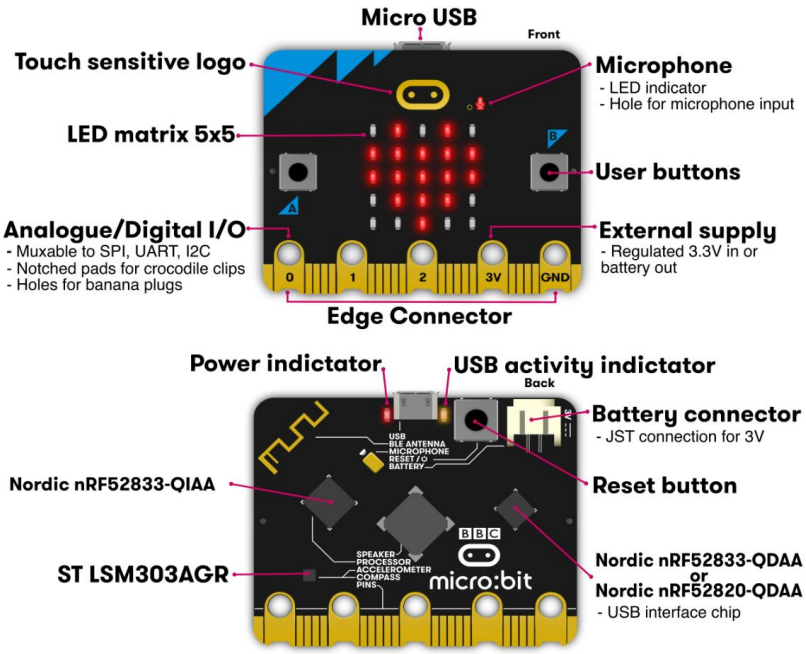



**Figure 4:** Micro Bit V2 hardware overview, 2022, by microbit-mark. (https://github.com/microbit-foundation/dev-docs/blob/master/hardware/assets/microbit-overview-2-2.png).

### 3.1.2 Micro Bit V2 programming

There are many options when it comes to the programming language to control the Micro Bit (Figure 5). It was decided to use the Micro Bit runtime as development environment for programming the Micro Bit. The runtime is a low level C/C++ tool developed by Lancaster University which provides a Component Oriented Device Abstraction Layer (CODAL) (Devine et al., 2019). Using the CODAL allows for easier and faster software development by abstracting each hardware component as a software component, represented by a C++ class.

The tools to compile C/C++ code for the Micro Bit V2 were installed from the GitHub repository by Finney et al. (2023). The main program that implements the positioning system designed here can be found in appendix A.

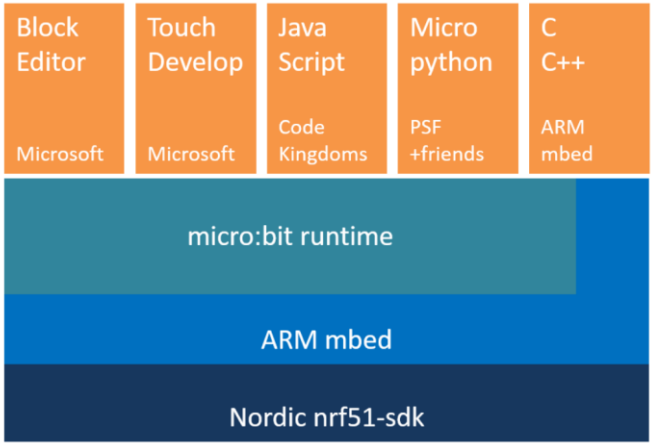



**Figure 5:** The different software layers running on the Micro Bit, 2016, by Finney. (https://github.com/lancaster-university/microbit-docs/blob/master/docs/resources/examples/concepts/architecture.png).

## 3.2 Ultrasonic transducer

### 3.2.1 PMUT from SINTEF

Piezoelectric micromachined ultrasonic transducers (PMUT) developed by SINTEF were first considered to be used for ultrasonic transmission in this thesis. These are state-of-the-art ultrasonic transducers, and the goal was to explore their performance and possible applications. These PMUTs are not off-the-shelf devices, which goes against one of the requirements of this positioning system, but they are about to be commercialized by Sonair (n.d.), and we should expect to see more use of these transducers in the near future.

These devices have the advantages of small footprint, low power consumption and low cost. Thus, this technology is ideal for future smart devices and Internet of Things (IoT) devices. The miniature size of the PMUTs makes them easy to integrate onto small, low-power microcontroller boards like the Arduino and Micro Bit (see Figure 6 for the size scale of the PMUTs).

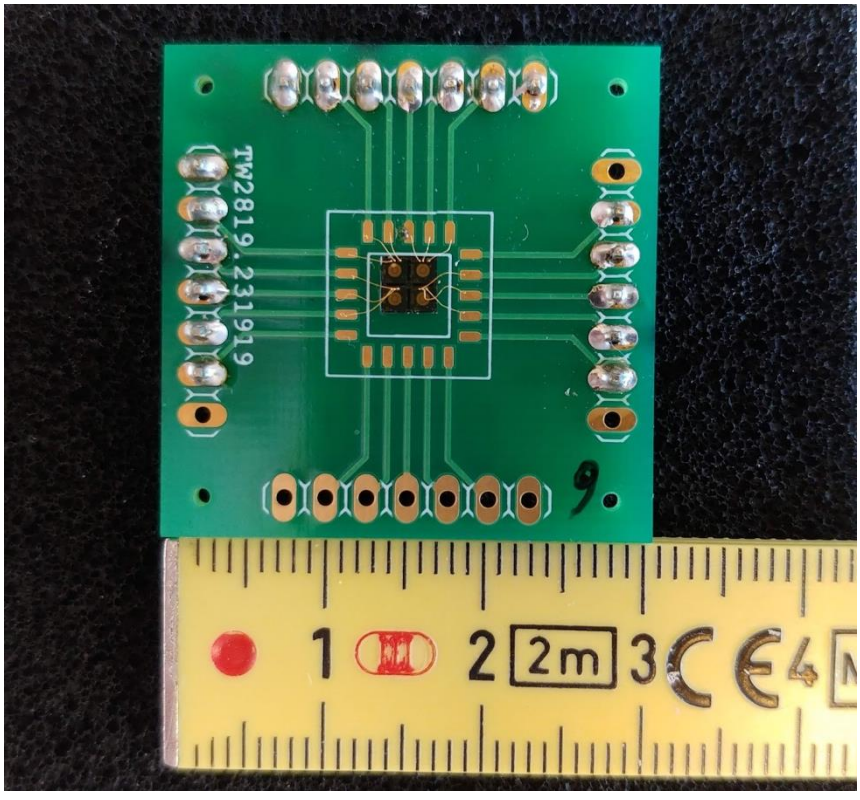


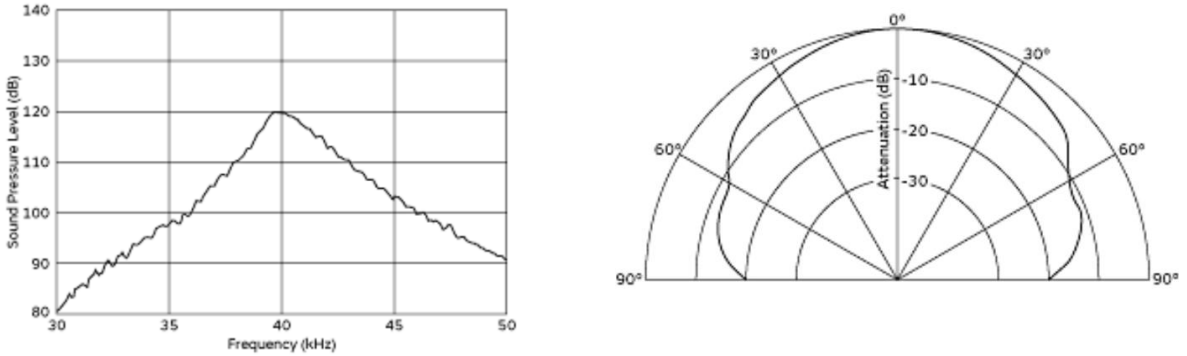

**Figure 6:** Four PMUTs can be seen in the center of the circuit board, each with the size of about 2 mm x 2 mm.

Since the PMUTs are newly developed, only preliminary documentation and characterization measurements are available. According to SINTEF we can expect 110 dB sound pressure level (SPL) at the resonance frequency of 95 kHz, and a narrow frequency bandwidth around the resonance frequency (SINTEF, personal communication, 2022). However, due to the production variability each PMUT should be characterized to find their resonance frequency and SPL for best performance.

However, using PMUTs for this application presented some challenges. First, the sensitivity of the Knowles microphone on the Micro Bit V2 drops rapidly after 80 kHz, and using the 95 kHz signal from the PMUT gives poor signal-to-noise ratio (SNR). The PMUTs are not packaged, meaning the silicon die is simply bonded to a printed circuit board (PCB), the membrane is open to its surroundings and the electrodes are wire bonded to the circuit board. This means the PMUTs are prone to failure if not handled carefully. Due to these challenges, it was decided to use commercial alternatives that are more convenient for this application, such as the MA40S4S ultrasonic transducer from Murata.

### 3.2.2 Murata MA40S4S

In ultrasonic positioning a transducer that generates a high sound pressure level (SPL) means the pulse can be detected at greater distances because of greater signal-to-noise ratio (SNR). Therefore, a high SPL transducer with a large radiation pattern is desirable to cover greater areas. The MA40S4S from Murata was chosen since they meet these requirements, and are cheap and readily available. Figure 7 a) shows they have a narrow bandwidth with peak SPL at 40 kHz and can generate up to 120 dB SPL. Those characteristics were measured with a 10 $V_{RMS}$ sine wave excitation signal and measured at 30 cm (Murata, n.d.-b). In this system the Micro Bit will generate a 3 V PWM signal which will be connected to the transducer. To maximize the transmitted SPL the 3 V PWM signal should be amplified, but in order to keep the complexity of the transmitter low, an amplifier is not used between the Micro Bit PWM output pin and the transducer. Figure 8 shows that even with the 3 V (2.1 $V_{RMS}$) signal the transducers can generate almost 110 dB.




a) b)

**Figure 7:** Murata MA40S4S ultrasonic transducer SPL a) and directivity b) characteristics, (n.d.-b), by Murata. (https://www.murata.com/products/productdetail?partno=MA40S4S).

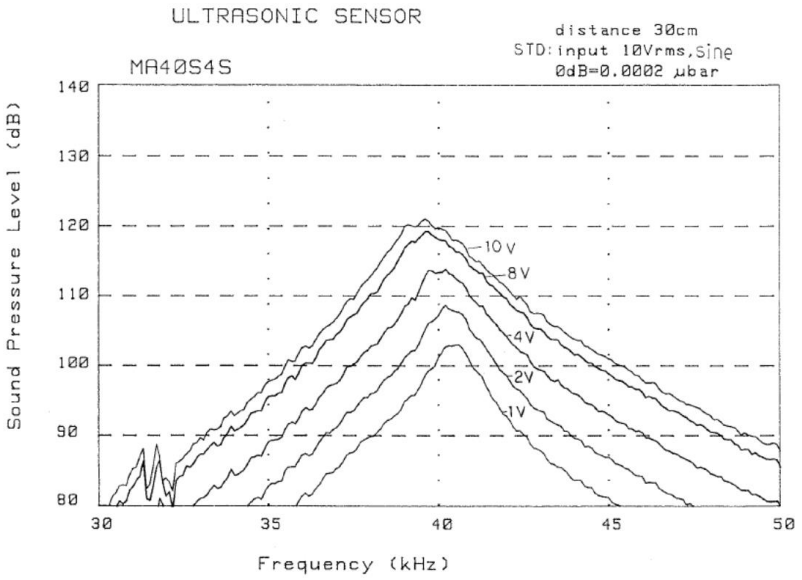



**Figure 8:** Frequency vs sound pressure level characteristic of MA40S4S , (n.d.-a), by Murata. (https://www.murata.com/-/media/webrenewal/products/sensor/ultrasonic/open/applinote_maopn.ashx).

## 3.3 Ultrasonic receiver

The main requirements for the ultrasonic receiver are to match the frequencies of interest and bandwidth of the transmitted signal, have good SNR, and omnidirectional sensitivity to ensure good coverage area. In addition, to comply with the positioning system requirements it has to be a low cost COTS device.

The Micro Bit V2 board is equipped with an omnidirectional SPU0410LR5H-QB-7 MEMS microphone that has a flat ultrasonic frequency response up to 80 kHz (Knowles Electronics, 2013). During development, it was found that the onboard microphone does not have any front-end amplification and its output is connected directly to the target microcontroller unit (MCU). This and the fact that the microphone is surrounded by many noisy digital components give poor sensitivity and SNR.

It was therefore decided to use an external ultrasonic receiver which can be connected to the Micro Bit general-purpose input/output (GPIO) pins. Such a receiver was already available from an earlier project from the course FYS4260 by Järve (2022). This ultrasonic receiver is a heterodyne bat detector which can frequency-shift the received ultrasound down to audible range and output the sound through a 3.5 mm audio jack. However, it also has an output pin that is connected to the pre-amplified microphone output. The microphone is the same SPU0410LR5H-QB-7 that is on the Micro Bit V2, but the microphone output is pre-amplified and high-pass filtered to attenuate frequencies below 20 kHz. The bat detector can be powered from the Micro Bit or with a 3 V CR2032 battery. The full schematic is shown in Figure 9, but only the circuit with the microphone and amplifiers are used in this positioning system. The hardware functionality is illustrated in Figure 10 and the printed circuit board is shown in Figure 11.

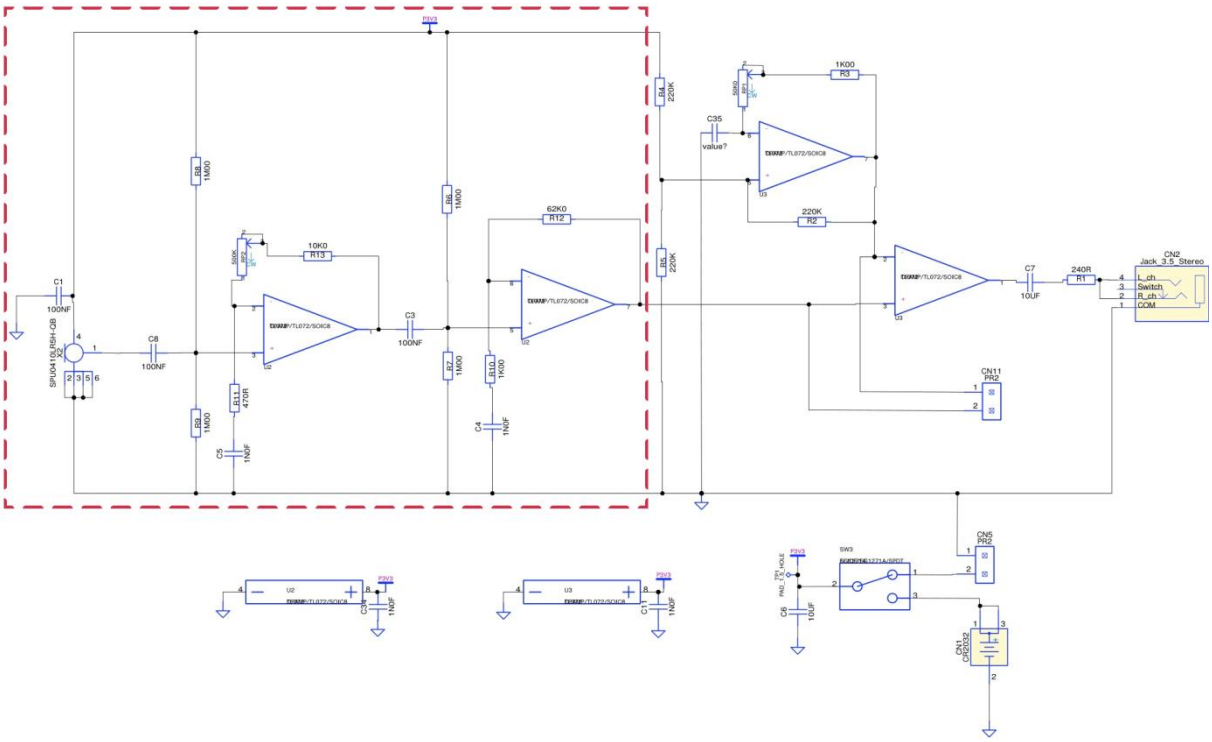


**Figure 9:** Schematic of the bat detector. From *Making a Arduino stackable bat detector*, by M. Järve, 2022. Based on the schematics from Berber (n.d.).

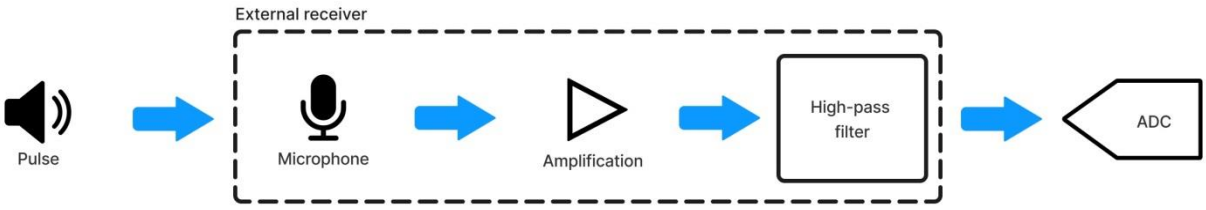



**Figure 10:** Overview of the hardware functionality of the external ultrasonic receiver.

It is possible to use the same ultrasonic transducer used in this design to both transmit and receive ultrasound. Using the MA40S4S would in effect make a band-pass filtered receiver with a center frequency at 40 kHz, which would be desirable to attenuate out-of-band noise. The bat detector is not a COTS device, but there are some other ultrasonic receivers such as the HC-SR04 Arduino range detector which could be used instead. However, the bat detector has the same microphone as the Micro Bit V2, and it can be used to demonstrate how a simple analog front-end would significantly improve the performance of the onboard microphone.

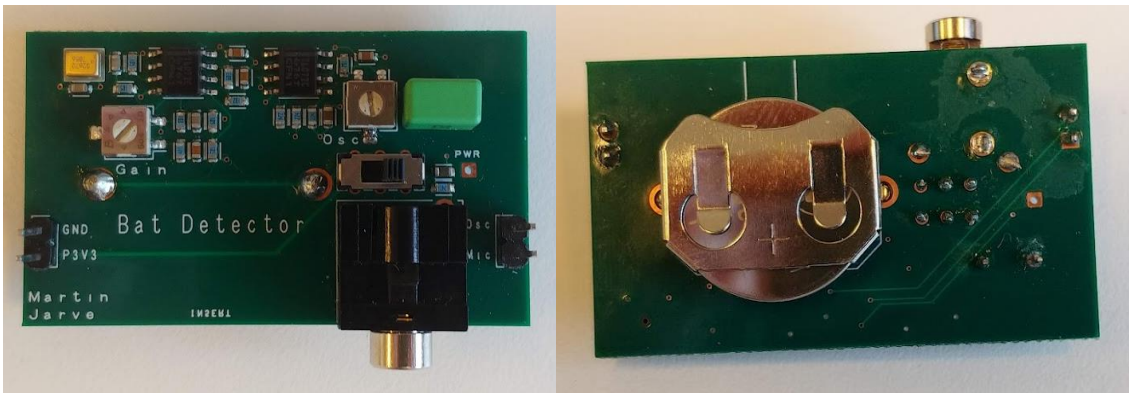



**Figure 11:** The bat detector that was used as the external receiver.

## 3.4 System overview

### 3.4.1 Synchronization of wireless nodes

The concept of time is essential in wireless node networks. Each node has at least one crystal oscillator or a hardware clock with a given period, but these clocks are temperature sensitive and are prone to clock drift, and they can only be used for internal processing. The only time hardware clocks can measure is the time between their internal events, like the time since startup, and the time one node experiences is not the same another node. To measure the time sound travels from one node to another, the nodes have to be synchronized to a common clock. The accuracy of this clock should be in the microsecond range since a latency of 100 µs will result in a positioning error of approximately 3.4 cm. The Global Positioning System (GPS) provides such timing accuracy, but it requires a GPS receiver for each node which is costly. It also requires line of sight with GPS satellites which make it non-feasible for indoor use, and if GPS signal is available there would not be a need for the positioning system designed in this thesis!

One option could be to use one Micro Bit node as a synchronization node. This node broadcasts its local time using a radio communication link to the other nodes which will synchronize to the broadcasted time. This method works only if the time for sending and

receiving a radio packet is negligible, but as shown in Roche (2006) there are many sources of non-deterministic delays. Figure 12 shows the breakdown of the time delay for transmitting and receiving a radio package. Send time is the time it takes for the transmitter to construct a radio package with time information, and the access time is the medium access control (MAC) delay in accessing a radio channel, both are delay sources present at the sender. The propagation time is the time it takes to send all the information across the medium. Finally, the time it takes to receive and process the package is the receive time. Not only do we have a packet delay, but it is also non-deterministic.

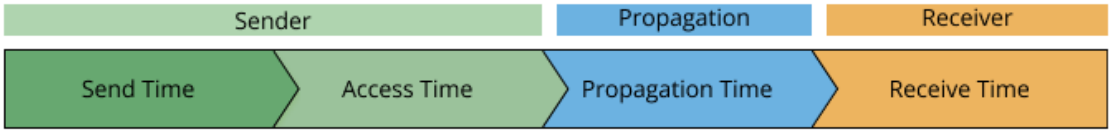



**Figure 12:** Sources of radio packet delays, adapted from Roche (2006).

The paper by Roche (2006) presents different synchronization methods that try to eliminate or reduce any of these delay components. Reference Broadcast Synchronization (RBS) is one of these methods and it can achieve accuracy in the microsecond range. The method uses one reference node to broadcast a synchronization message and the other nodes will take a timestamp at the reception of this message. The receiving nodes can then compare their timestamps to find the offset and synchronize their clocks to the reference node. The method works by assuming that all nodes are close enough, so that the propagation time of the radio signal can be neglected. With this method the delay sources at the sender will be eliminated and assuming the propagation time is instantaneous, the only source of delay is at the receiver.

To synchronize the wireless nodes in this thesis, the ultrasound transmitting node will broadcast a radio packet to the ultrasound receiving node. This packet contains the Micro Bit identification character of the transmitting node. Immediately after the package is transmitted, an ultrasonic pulse is also transmitted. When the receiving node receives this packet it knows to expect an ultrasonic pulse from Micro Bit associated with the identification character. That means for every measurement the transmitting node has to send a synchronization packet.

### 3.4.2 ToF estimation

Estimating the ToF is one of the least complex acoustic positioning techniques, with low receiver complexity and good accuracy. As explained in the previous section, to measure the ultrasound ToF between two nodes in this thesis, the transmitting node sends a radio packet to signal to the receiver of incoming ultrasonic pulse, and the ToF is measured entirely on the receiver side.

Figure 13 shows an overview of how ToF measuring is implemented in software on the wireless nodes. First, the transmitting node broadcasts a radio packet containing the identification character of the transmitter, and immediately after generates a PWM pulse which is connected to the ultrasonic transducer. The receiving node takes a timestamp at the reception of this radio packet and starts the ADC which starts sampling the ultrasonic receiver output. The ADC is configured for continuous sampling at 200 kHz with double buffering, meaning when a buffer is full it will start filling a new buffer while the full one can be processed. A counter $i$ keeps track of the number of ADC samples since startup. Since the sampling period is a fixed time interval of 5 µs, we can count the number of samples until pulse detection. The ADC samples are stored in a direct memory access (DMA) buffer with size $M$, and when a buffer is full it is sent downstream for processing. First step of processing is finding the average value of the samples in the buffer, that is the DC offset, and subtracting it from the samples. Then, the magnitude of each sample is compared with a threshold value, and the corresponding sample index that is above threshold is the measured ToF. The sample index is multiplied with 5 µs to get the ToF in microseconds, and thus the resolution of this system is limited by the sampling frequency ($5\ \mu s \cdot 343\ m/s \approx 1.7\ mm$). After the pulse is detected, the ToF data is sent over radio to a central node for data logging.

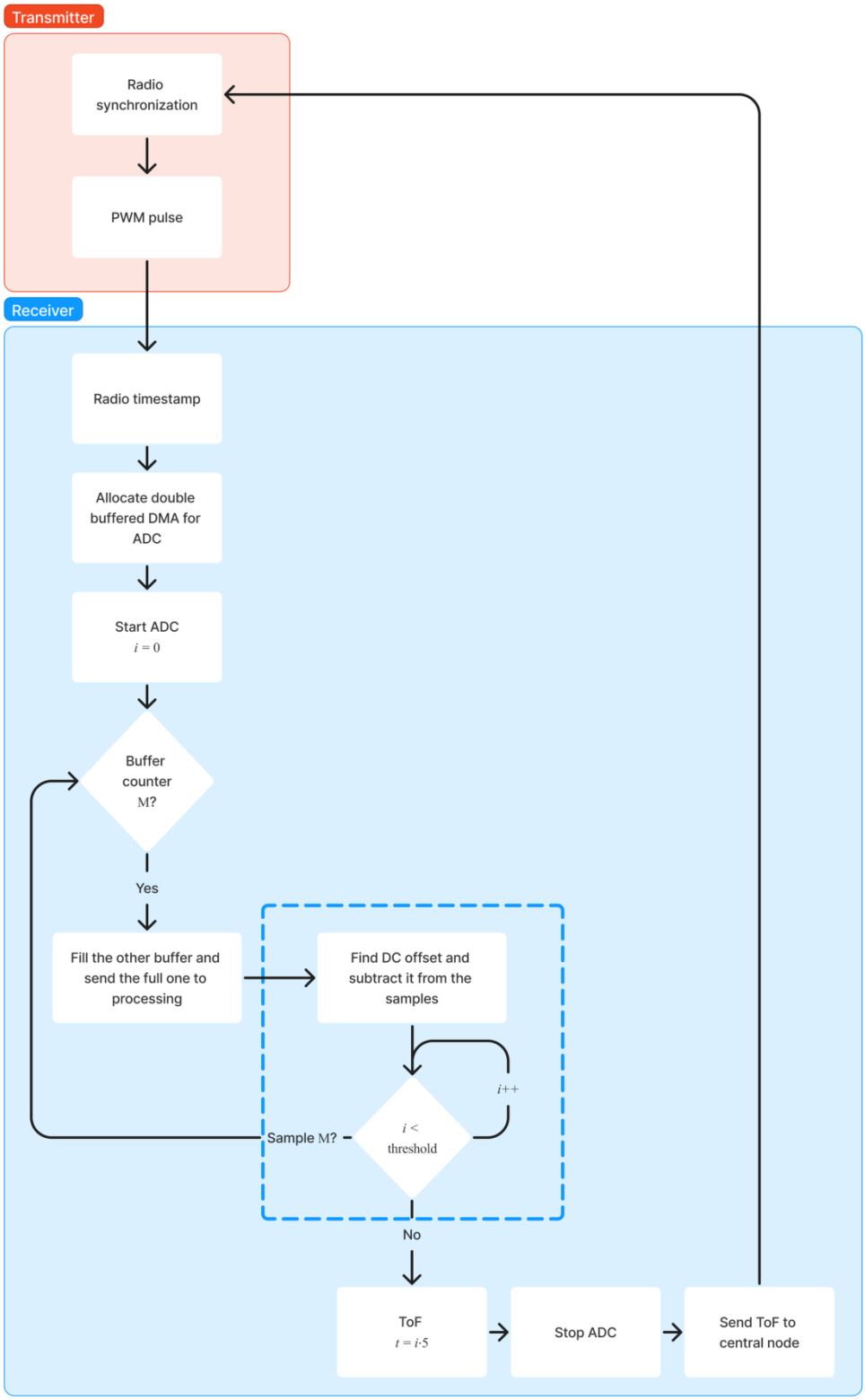



**Figure 13:** Overview of how the ultrasound ToF measuring is implemented in software on the wireless nodes.

Figure 14 illustrates the receiver pulse response and the threshold value which must be chosen such that it is above the noise floor whilst not too high to not detect an ultrasonic pulse at greater distances. Usually an envelope detector is implemented such as Hilbert transformation, but for simplicity and reduced development time this is not implemented here. The drawback of not extracting the ultrasonic envelope is shown in Figure 15 a) and b), which illustrates the pulse detector that is implemented on the Micro Bit. The signal period is five times the ADC sample period, which is the case when sampling a 40 kHz pulse at 200 kHz. The intersection points between the vertical ADC sample lines and the signal envelope and receiver pulse response illustrates the discrete value of the envelope and the rectified pulse response on the Micro Bit, respectively. Figure 15 a) and b) show the same received pulse, but shifted in time, and we see that in a) the envelope detector is above threshold five sample indexes earlier than the received pulse response. In b) the envelope detector is above threshold one sample index earlier than the received pulse response. Depending on how fast the received pulse rises, threshold level and ADC sampling timing it is possible get a detection delayed by a few sampling periods when not using envelope detection.

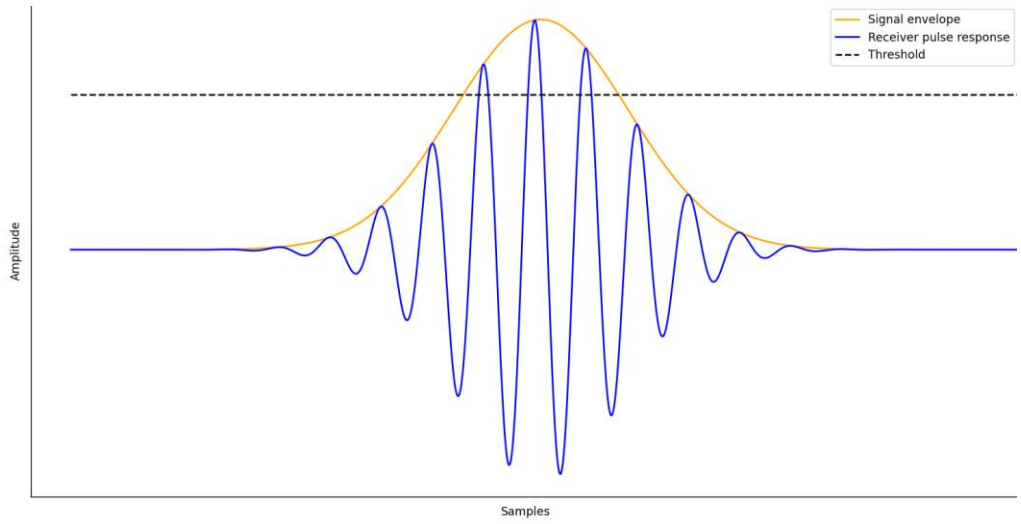
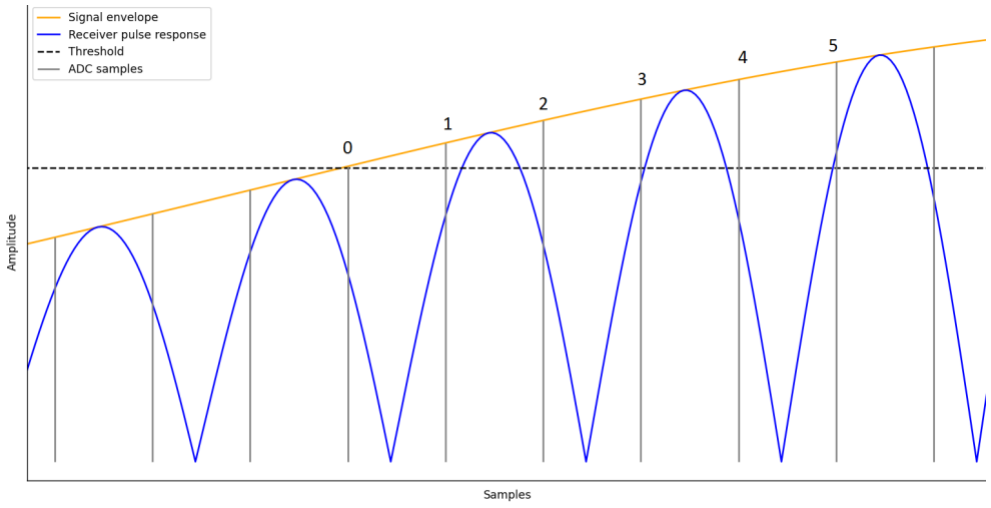
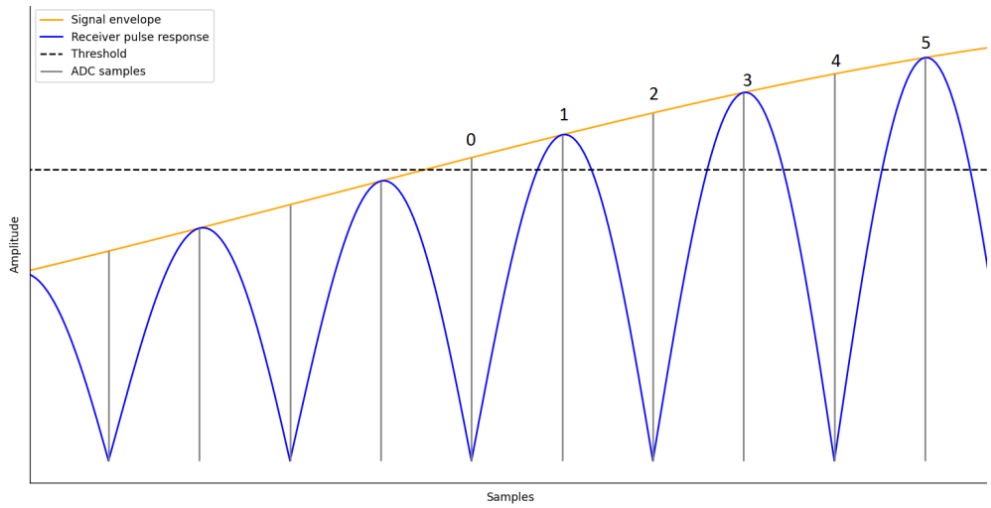



**Figure 14:** Ultrasonic receiver pulse response with threshold value for pulse detection.




a)




b)

**Figure 15:** Both a) and b) contains the same pulse but shifted in time. The intent is to showcase how envelope detector is more consistent than comparing ADC samples directly with threshold.

The time $t$ between the arrival of the radio packet and the ultrasonic pulse detection can be modelled as

$$t = t_{\text{pulse}} - t_{\text{radio}} = t_{\text{ToF}} + t_{\text{bias}} \tag{12}$$

where $t_{\text{pulse}}$ and $t_{\text{radio}}$ are the timestamps of their respective events, $t_{\text{ToF}}$ is the true ToF value we are trying to measure, but the actual value measured includes the addition of a bias $t_{\text{bias}}$. This bias does not come from the radio packet delays in Figure 12, because $t_{\text{radio}}$ timestamp is taken when a complete radio packet has been received. Thus the bias (illustrated in Figure 16) includes processing overhead such as any additional transmission side delay that comes from switching from radio transmission task to the PWM signal generation task,

including the processing required to generate the PWM signal. That is if all of that takes longer time than the propagation and receive delays. From $t_{\text{radio}}$ timestamp there will also be some delay in the direct memory access (DMA) buffer allocation and ADC startup. Some of the bias is also a result of where the threshold level is set, as the pulse will be detected after its actual arrival.

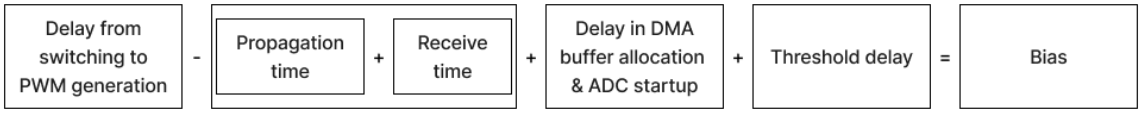



**Figure 16:** Processing overhead and other contributions to $t_{\text{bias}}$.

It is desirable that this bias value is constant across measurements since it can be characterized and removed by calibration. The bias can be found by taking ToF measurement at a known distance and finding the difference between the measured and true ToF values. By modeling the sound propagation speed, we can convert the known distance to the true ToF value. With enough measurement it is possible to estimate $t_{\text{bias}}$ by finding the average value for the bias. Subtracting $t_{\text{bias}}$ from the measured values will give the estimated ToF.

### 3.4.3 Wireless node network control protocol for positioning

There are two types of nodes in this positioning system; ultrasonic transmitters and ultrasonic receivers (Figure 17), where the transmitters will be acting as anchor nodes with known positions, and the receivers are to be located. The transmitting nodes have an ultrasonic transducer connected to the PWM output pin, and the receiving nodes have an ultrasonic receiver connected to the pin that is configured for ADC input. All nodes are powered by a 3 V AAA battery pack, except for one anchor node which will be acting as a data sink and transfers the ToF data over serial communication to a laptop for calculation of the receiver coordinates according to equation (11).

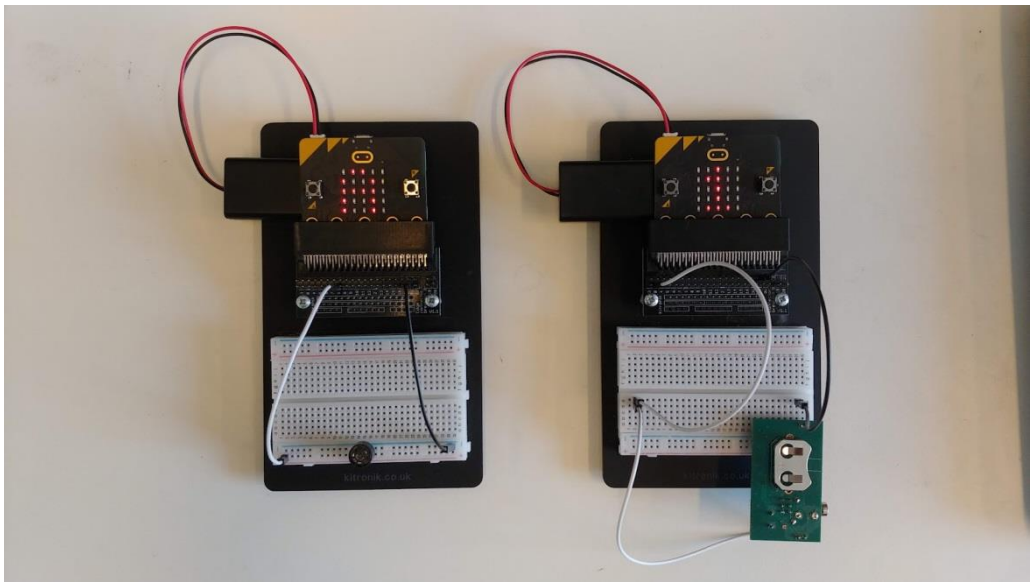


**Figure 17:** Transmitter node (left) with MA40S4S ultrasonic transducer and receiver node (right) with the external ultrasonic receiver.

For 3D positioning of a node, at least four anchor nodes with known $(x, y, z)$ coordinates are needed, and the ToF from each of those anchor nodes to the receiver node when performing multilateration with equation (11). Since the pulses are not encoded and this system uses a simple pulse rising edge detector, there is only one acoustic channel available for the ToF measurements, which means for 3D positioning of a node just one transmitter can send a pulse at a time, and the next transmitter has to wait for the multipath signals to die out before sending a new pulse. Considering this, the transmitters will send out a $0.5$ ms pulse and a $200$ ms wait period is enough for reflections to die out ($200$ ms $\cdot$ $343$ m/s $= 68.8$ m). A time-division multiple access (TDMA) protocol is implemented to share the frequency channel (see Figure 18). One measurement cycle is one time frame, and each frame is made up of four time slots where each transmitter gets a $200$ ms time slot to transmit their pulse. That means the positioning update time is $800$ ms for this system.

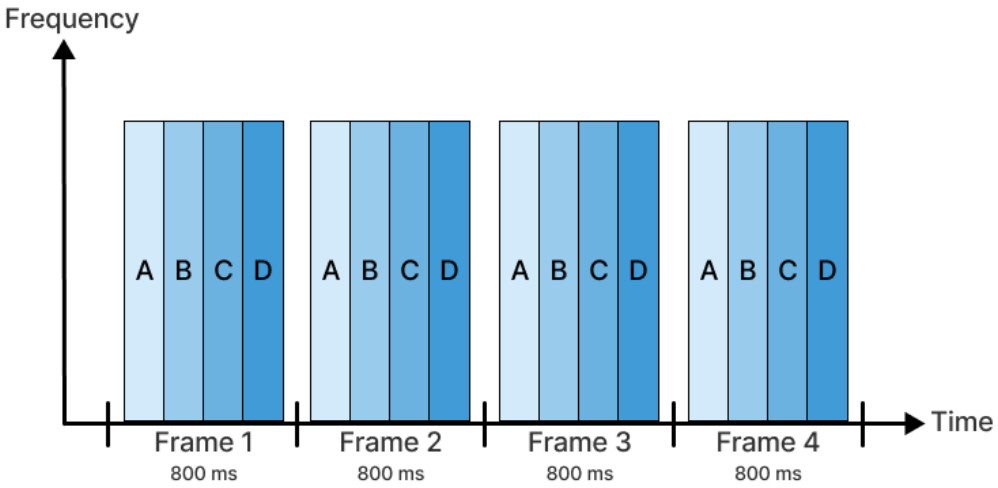



**Figure 18:** A time-division multiple access protocol (TDMA) is used to share the same frequency channel by dividing time into block that are dedicated to individual transmitters.

Figure 19 shows an overview of one measurement cycle for positioning one receiver. The four anchor nodes are labelled “A”, “B”, “C” and “D”, and the receiver is labelled “1”, and all nodes share the same radio channel for communication. Node “A” starts the measuring cycle by broadcasting a radio packet containing its identification character “A”, and proceeds immediately by sending an ultrasonic pulse. The receiving node measures the ToF between the radio synchronization packet and the ultrasonic pulse, and broadcasts a radio packet containing the measured ToF, the transmitter identification character, and its own identification number. That way the measured ToF is linked to the transmitter-receiver pair. Since all nodes receive this packet, the next node, node “B”, knows that a successful ToF was measured between “A” and “1”, and proceeds by waiting 200 ms before it initiates a new ToF measurement. After all four transmitter-receiver pairs have measured the ToF, the receiver’s position is calculated, and a new measurement cycle begins.

Node “A” receives all the measured ToF data and transfers it to a laptop over serial communication. There, the bias is subtracted and the estimated ToF is converted into distances using equations (1) and (2), before the position is estimated using equation (11). The Python code that is running on the laptop is included in Appendix B. This makes it a centralized positioning system, but we need the data on central device anyway in order to present it. It is also possible for the receiver to calculate its own position without sharing it to make the positioning system privacy oriented.

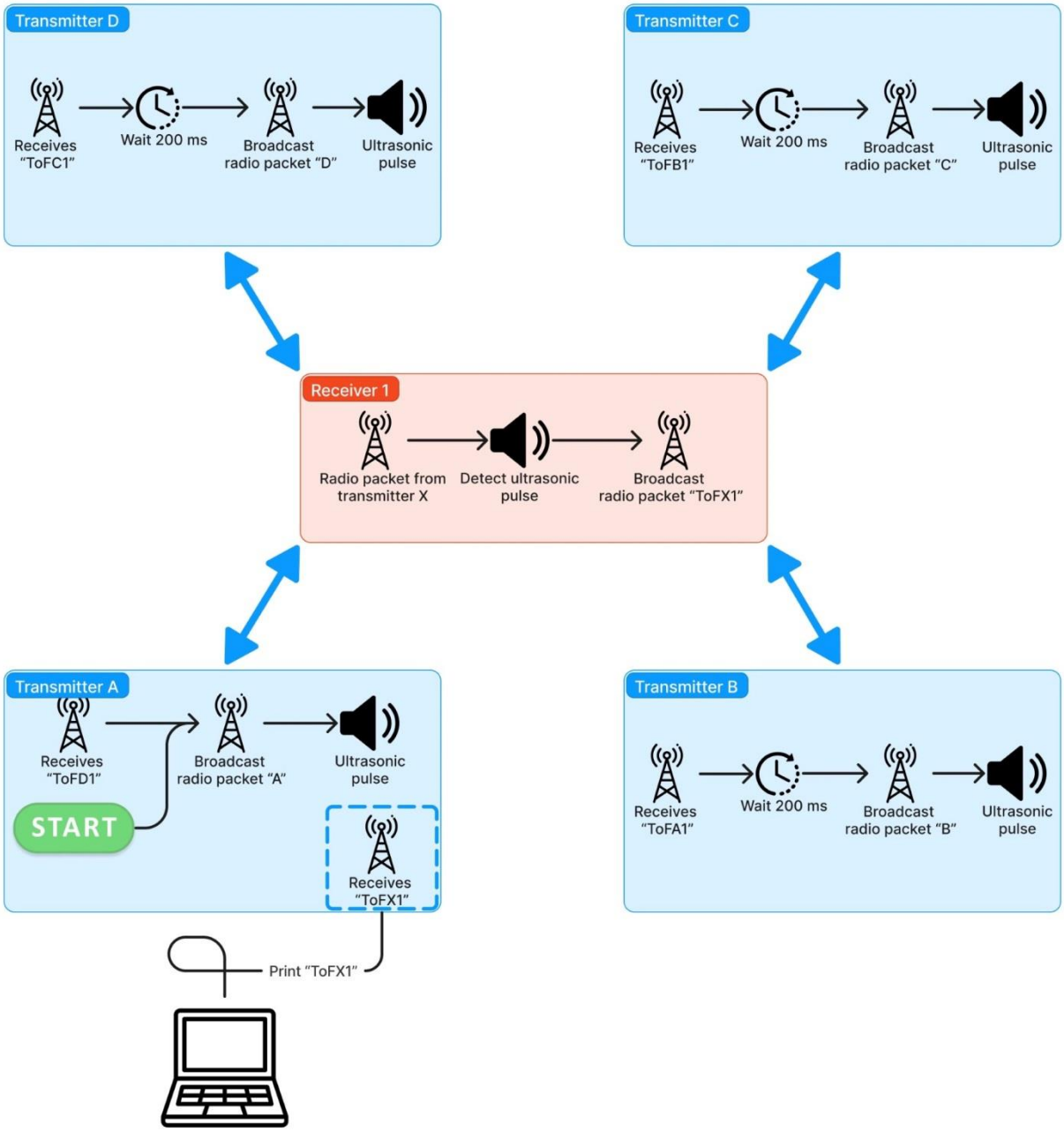



**Figure 19:** Overview of the measurement cycle for positioning using ultrasonic ToF.

# 4 Results

This chapter describes the experimental setups that were used to evaluate the wireless ultrasonic positioning system according to the design considerations, and presents the results of those experiments. Effort was done to isolate the sources of error that contributed to the overall inaccuracy of the system, find their significance and look at how these errors could be reduced. The measurements used a Rohde & Schwarz HMF2525 arbitrary function generator to generate test signals, a Tektronix DPO2004B oscilloscope to gather the data and a TFA LT-102 digital thermometer to measure the ambient temperature.

## 4.1 Synchronization

### 4.1.1 Variance in radio synchronization packet receive time

As discussed in chapter 3.4.1 about synchronization, the radio packet receive time is non-deterministic, which implies a variance in this delay. Since the receive time is a part of the bias in the measured ToF, it is important to characterize the variance in the receive time to estimate how much it contributes to the inaccuracy of the measured ToF. The variance in the receive time cannot be measured directly since the reception timestamp is taken on a different node than the transmitting node, and they do not share the same clock. That means the time difference between two events must be measured on the same node. A way to estimate the entire radio packet delay is to measure the round trip of a radio transmission and divide that time by two. This was done by taking a timestamp right before a radio packet was transmitted, a second node received this packet and immediately sent a radio packet back, and the first transmitter took another timestamp at the reception. With this method the entire radio packet delay (transmitter and receiver side) could be estimated, not the receive time itself, but it can still be a good indicator of time synchronization error between the nodes.

Figure 20 shows 900 measurements of the estimated radio packet delay and Table 1 contains some statistical values of the measurements. We can see that most of the time the delay is 324 µs or 324.5 µs. The accuracy of the clock used to take the timestamps is 1µs and it is

also therefore the minimum time resolution in the measurements, but since the measured time is divided by two to estimate the one-way radio delay, the effective time resolution is 0.5 µs.

The mean value is 324.6 µs with two standard deviations being 3.96 µs, meaning if the data was normally distributed, we could expect the transmitter-receiver nodes to be synchronized within 3.96 µs around 95% of time ($3.96\ \mu s \cdot 343.2\ m/s \approx 1.3$ mm). The maximum measured difference in the radio packet delay is 20 µs ($20\ \mu s \cdot 343.2\ m/s \approx 6.8$ mm).

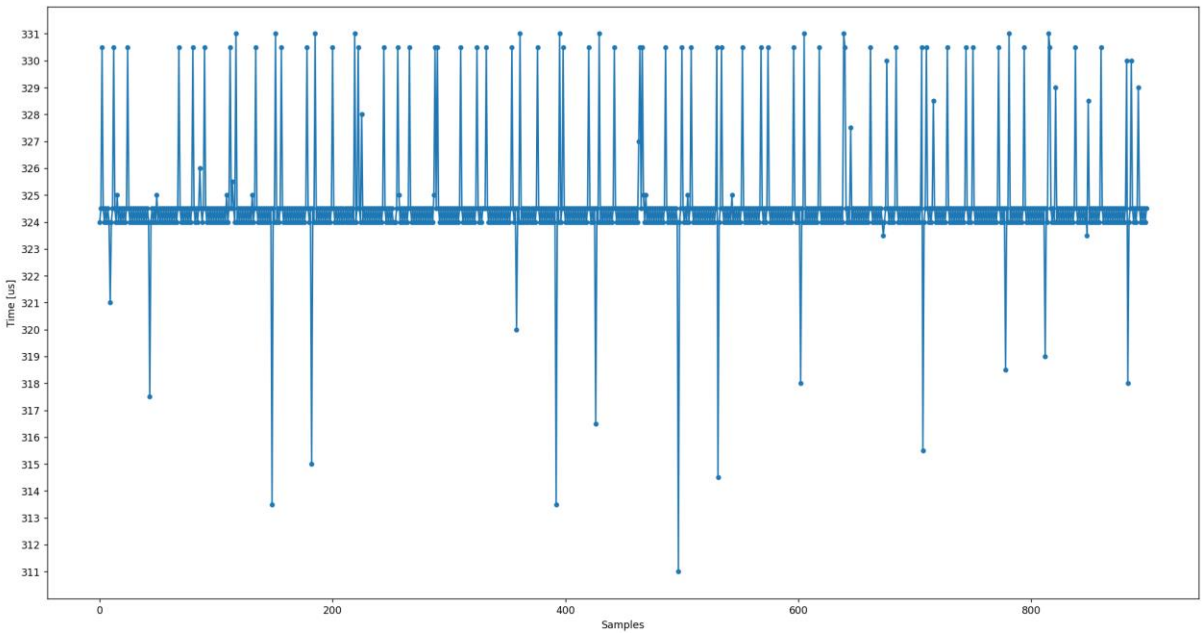



**Figure 20:** 900 measurements of the estimated radio packet delay (transmitter and receiver side).

**Table 1:** Some statistics of the estimated radio packet delay in Figure 20.

| **Mean value** | **Standard Deviation $\sigma$** | $\mathbf{max}(t) - \mathbf{min}(t)$ |
|---|---|---|
| 324.6 µs | 1.98 µs | 20 µs |

### 4.1.2 Node synchronization by radio vs by wire

To look more precisely at how much the wireless synchronization method reduced the system's precision, it was replaced with a wired synchronization method. Synchronization by wire was implemented by connecting together a GPIO pin on the transmitter and the receiver. The transmitter controls the logical value inside the wire by pulling it "low" from a "high" value right before it is about to transmit an ultrasonic pulse. The receiver detects this change and takes a timestamp, and is thus synchronized to the receiver. It is assumed that the change in logical value inside the wire happens instantaneously and the processing overhead is minimal. When the receiver detects the ultrasonic pulse, it sends the measured ToF back to the transmitter by radio, and the transmitter pulls the wire to a "high" state again. With this synchronization method it is possible to characterize the receiver's precision independent of a wireless synchronization method.

Figure 21 shows the transmitted PWM pulse, the received ultrasonic pulse, and the synchronization signal measured with an oscilloscope. The figure also illustrates the implementation of the ultrasonic ToF measuring system through the physical signals. We can see that there is a time delay between the "high" to "low" flank of the synchronization signal, and the start of the PWM pulse. Through repeating the same experiment it was found that this delay is constant around 72 μs within microsecond precision.

The pulse detection threshold is added to the plot at 91 mV above the receiver's DC level, which was the actual threshold level set on the Micro Bit. The intersection between the threshold and the microphone response in the plot does not correspond to the time instant when the receiving node detected the pulse, because the receiver's measured ToF includes a bias, which is characterized in the chapter under.

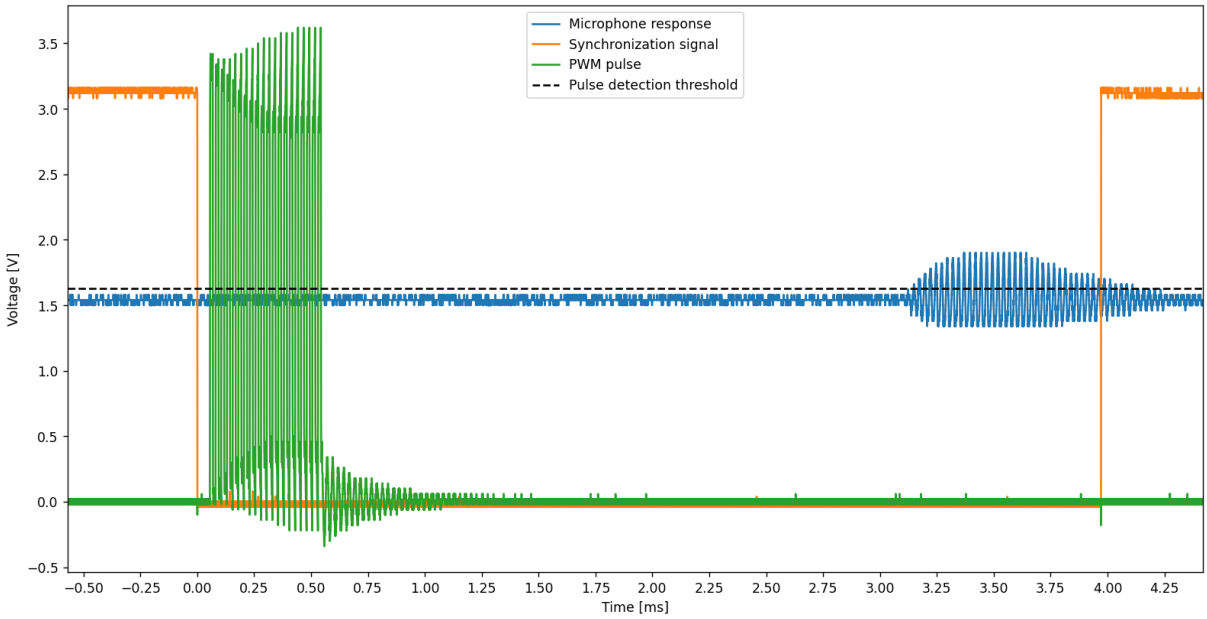



**Figure 21:** Ultrasonic ToF showing the transmitter's synchronization signal and PWM pulse, and the receiver's microphone response.

In order to look at the precision of the two synchronization methods, 500 ToF measurements were taken at distances 100 cm, 200 cm, 300 cm, 400 cm and 500 cm between the transmitter and the receiver, which were mounted on tripods at a height 120 cm above the floor (setup shown in Figure 22). The true distances were measured using a 1 m long measuring tape with millimeter markings by keeping it tight and straight between the transducer casing and the microphone package. The ultrasonic transducer and microphone was pointed at each other to ensure that the line-of-sight pulse was detected, and the interval between subsequent pulses was 200 ms.

**Figure 22:** Setup for measuring ToF between a transmitter (to the left) and a receiver (to the right).

Figure 23 and Table 2 shows the mean error of the distance measured using ToF with the two synchronization methods (the mean error here is the true distance subtracted from the average estimated distance). What is important is to look at is the standard deviation, not the mean error since the bias is not yet calibrated for in the figure. The table shows that the standard deviation of the two synchronization methods is roughly the same, except for distances 300 cm and 500 cm, when it is halved for the wired method compared to the radio method. This shows that the wireless synchronization method does not seem to reduce the precision of the ToF measurements, which makes sense since the radio packet delay from Figure 20 is near its mean most of the time, and the maximum delay difference is 20 µs (0.68 cm). 0.68 cm is below the 40 kHz ultrasonic wavelength (0.86 cm) which means the system's precision is still limited by the wavelength.

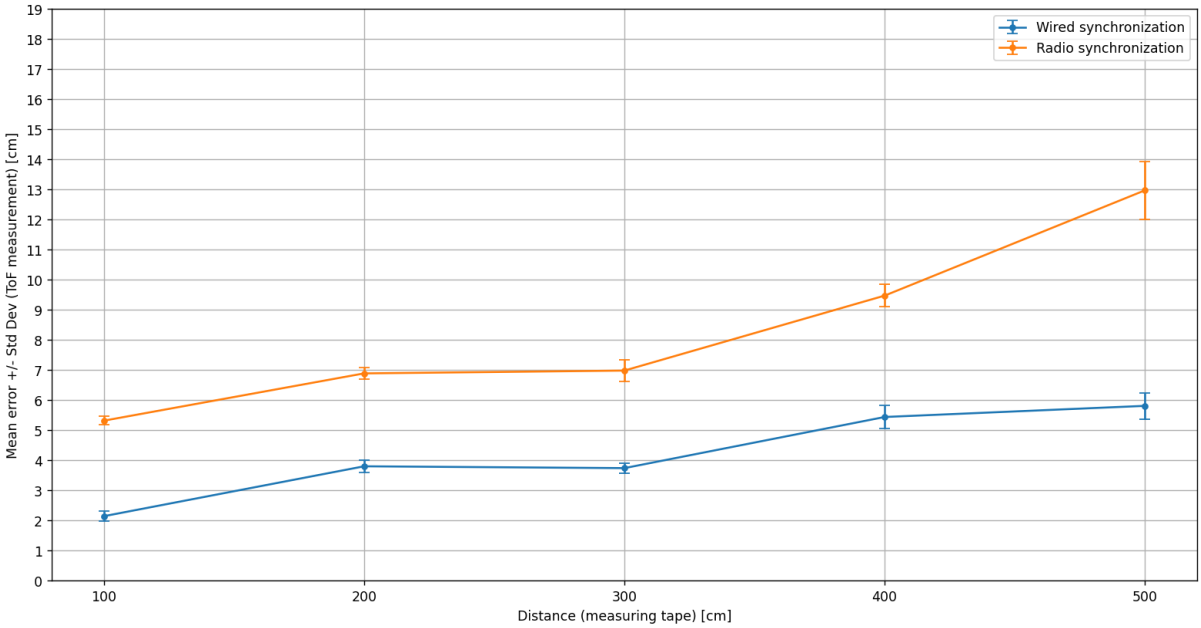



**Figure 23:** Mean error of the measured distances using ToF (with radio and wire synchronization) vs the true distance. The error bars show one standard deviation. 500 measurements were taken at each distance and the temperature was 20.7° C.

**Table 2:** Mean error of the estimated distances using ToF (with radio and wire synchronization) and the standard deviation (values of the over).

| **Distance [cm]** | **Mean error [cm]** | | **Standard deviation $\sigma$ [cm]** | |
|---|---|---|---|---|
| | **Wire** | **Radio** | **Wire** | **Radio** |
| **100** | 2.1 | 5.3 | 0.16 | 0.14 |
| **200** | 3.8 | 6.9 | 0.21 | 0.19 |
| **300** | 3.7 | 7.0 | 0.17 | 0.35 |
| **400** | 5.4 | 9.5 | 0.39 | 0.38 |
| **500** | 5.8 | 13.0 | 0.43 | 0.97 |

## 4.2 Accuracy

### 4.2.1 Calibration and accuracy of ToF distance measurements

Calibration of the ToF measurements is necessary to remove the bias that is explained in chapter 3.4.2. In order to characterize this bias and look at the accuracy of the ToF distance measurements of the system, the same experiment as in chapter 4.1.2 was performed using the same setup as in Figure 22. The ToF measurements were taken using both the onboard microphone on the Micro Bit and the external receiver to compare the difference in performance.

Figure 24 shows the bias (true distance subtracted from the average estimated distance) of the estimated distances using ToF for both receivers. It was expected that the bias would be constant for all distances, but the data shows almost linear increase with increasing distance. Therefore, linear regression of the bias is also shown in the figure, which for the external receiver had to be done in two steps to best fit the data. The linear regression is used to calibrate for the bias, and the results after calibration are shown in Table 3 and Table 4.

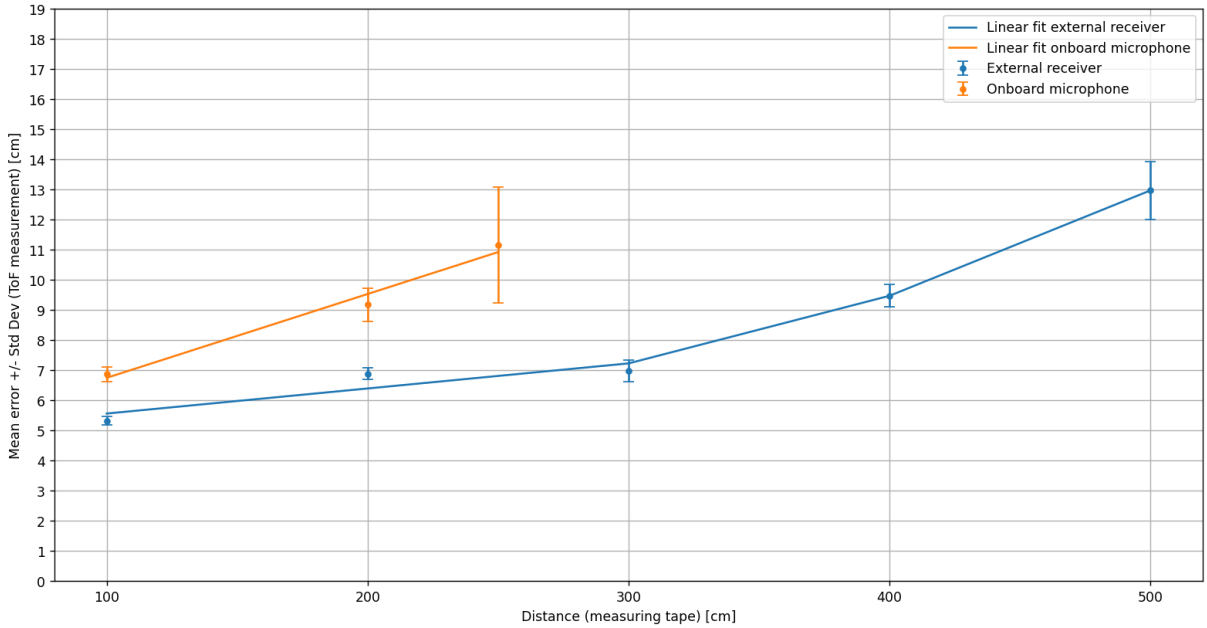



**Figure 24:** Mean error of the measured distances using ToF (with external and onboard receiver) vs the true distance. The error bars show one standard deviation. 500 measurements were taken at each distance and the temperature was 20.7° C. Linear regression of the mean error was used to compensate for the measured bias.

Table 3 and Table 4 shows the 90th percentile of the absolute error, the maximum absolute error and the standard deviation of the estimated distances using the external and internal receivers. The absolute error is found by subtracting the true distance from the calibrated ToF distance measurements, and at least 500 measurements were performed at each distance. The absolute error and the standard deviation is roughly the same up to 4 m using the external receiver, but are much worse at 5 m. For the internal receiver these metrics get much worse with increasing distance, and the SNR of the microphone was too poor to be used beyond 2.5 m.

The data indicates sub-centimeter accuracy and precision up to 4 m is achieved most of the time using the external receiver. While the onboard microphone can only detect ultrasound up to distances of 2.5 m and has significantly worse performance compared to the external receiver.

**Table 3:** Standard deviation and absolute error of the estimated distances using ToF (external receiver) after calibration using the bias model from Figure 24.

| **Distance [cm]** | **90% percentile of absolute error [cm]** | **Maximum absolute error [cm]** | **Standard deviation $\sigma$ [cm]** |
|---|---|---|---|
| **100** | 0.24 | 1.10 | 0.14 |
| **200** | 0.64 | 1.32 | 0.19 |
| **300** | 0.72 | 1.23 | 0.35 |
| **400** | 0.64 | 1.94 | 0.38 |
| **500** | 1.44 | 4.35 | 0.97 |

**Table 4:** Standard deviation and absolute error of the estimated distances using ToF (onboard microphone) after calibration using the bias model from Figure 24.

| **Distance [cm]** | **90% percentile of absolute error [cm]** | **Maximum absolute error [cm]** | **Standard deviation $\sigma$ [cm]** |
|---|---|---|---|
| **100** | 0.46 | 0.98 | 0.25 |
| **200** | 1.12 | 2.84 | 0.55 |
| **250** | 3.03 | 7.96 | 1.91 |

### 4.2.2 Static positioning

Static positioning experiments were performed to evaluate the accuracy and precision of the ultrasonic positioning system. To do this, a 3D reference coordinate system was constructed using a right-angle ruler and a folding meter stick in order to know the true positions of the anchor nodes and the receiver. The ultrasonic transducer casing on the transmitters and the microphone package on the receiver is used as the reference coordinates. Figure 25 shows the experiment setup and Figure 26 shows the coordinates of the anchor nodes and the receiver. Positions 1-5 are used for the position estimation using the external receiver, and position 6 is used for the internal receiver, and at least 300 position estimations were performed at each position. Only one position is used for the internal receiver since the ultrasonic pulse response at this distance and angle is not always enough to trigger detection. All anchor node transducers are roughly at the same height and are pointed at the origin of the reference coordinate system. The receiver's microphone is pointed towards the ceiling to ensure direct line-of-sight ultrasound detection from all four transmitters.

**Figure 25:** Setup of the ultrasonic positioning system. Four anchor nodes (transmitters) are placed at a height pointing their transducers at the center of the positioning area, and the receiver points its microphone towards the ceiling. The anchor nodes form a 3D space where the direct line-of-sight ultrasonic pulse can be detected and used for position estimation of the receiving node.

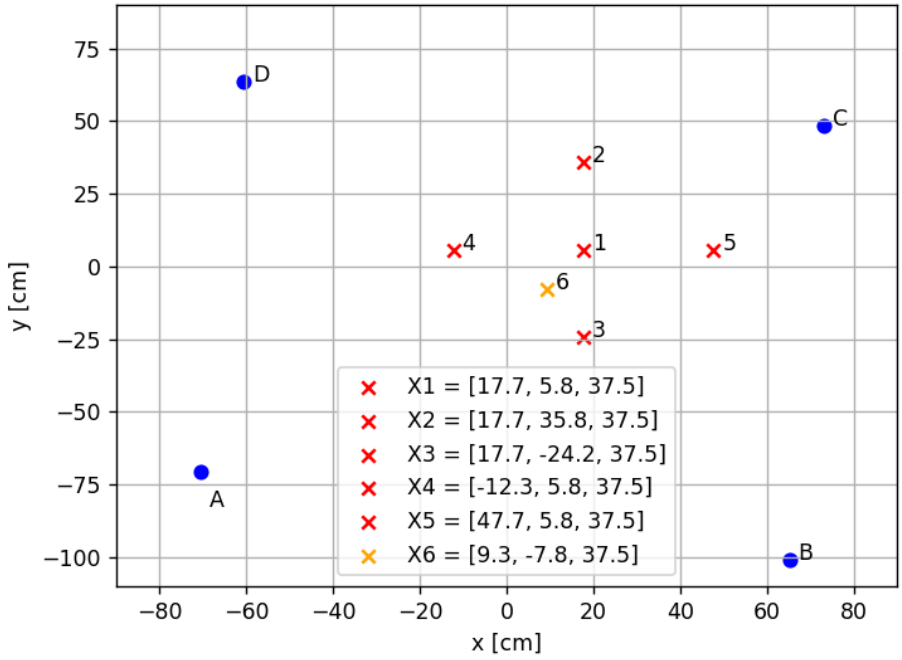



**Figure 26:** Positions of anchor nodes A, B, C and D, and six different receiver positons in the xy-plane. Positions 1-5 are used for the position estimation using the external receiver, and position 6 is used for the internal receiver.

A scatter plot of the estimated x and y coordinates and the true coordinates at each position is shown in Figure 27. The figure illustrates the spread of the estimated positions even though the receiver was static, with a maximum spread of about 2 cm along each axis. All groups of estimated positions have a deviation from their respective true position. The internal receiver shows accuracy comparable to the external receivers, but with a bigger spread in position estimation.

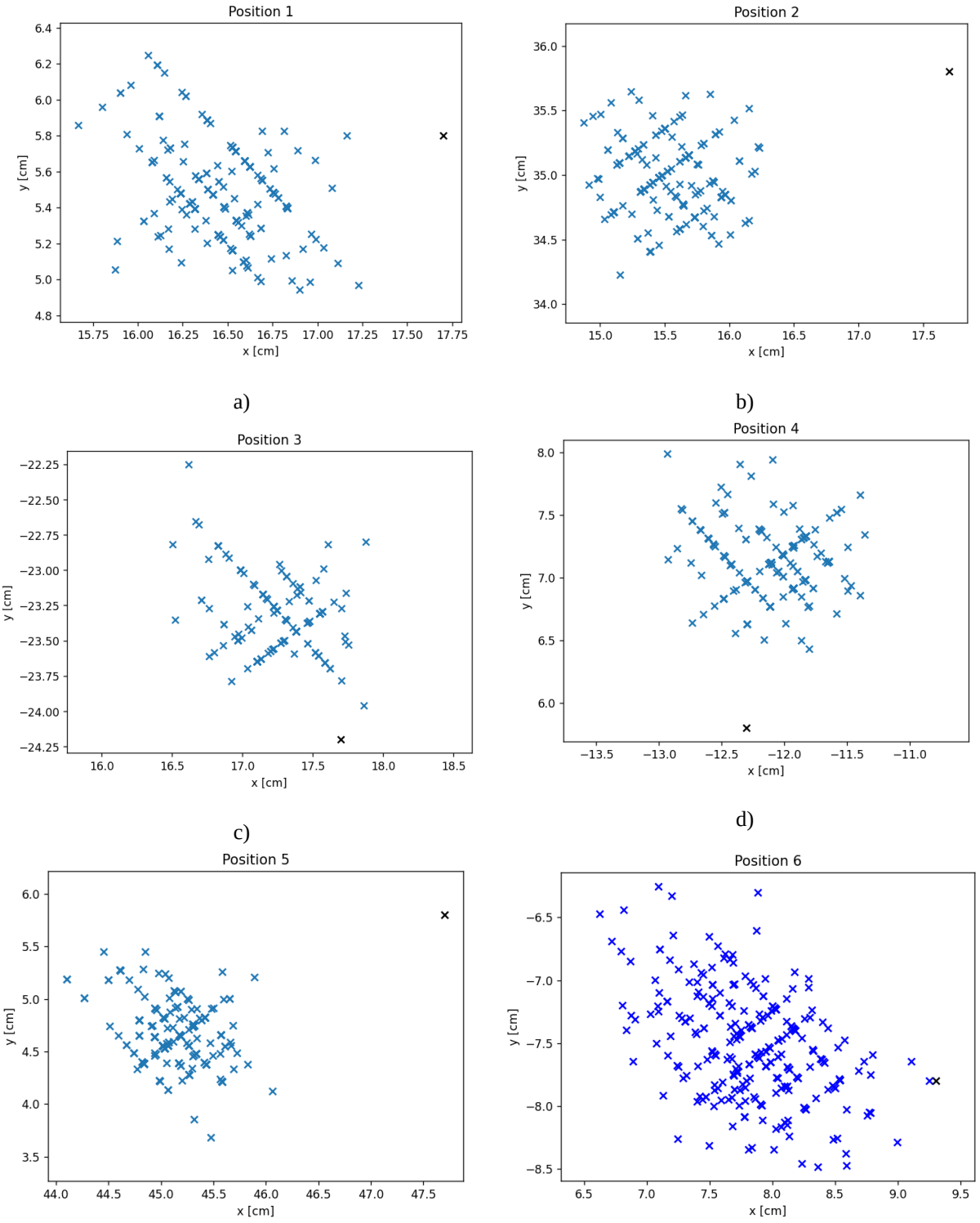



**Figure 27:** Scatter plot of the xy-plane of 300 estimated positions at each position. Positions 1-5 are estimated using the external receiver, and position 6 is estimated using the internal receiver. The black cross shows the true position. Note that the axis ranges are not normalized.

Table 5 contains the standard deviation and the mean position error of the estimated coordinates, where the positioning error is obtained by subtracting the true coordinates from the average estimated coordinates for each position. Figure 28 shows the percentile of absolute error in the Euclidean distance between the estimated positions and the true position. Both the external and internal receiver show the same level of accuracy, but the internal receiver has worse precision as shown by the gentler slope of the CDF plot and the increased standard deviation. Both Table 5 and Figure 28 show a stable level of precision but a big variance in accuracy across the six different positons.

**Table 5:** The mean position error in the x, y and z coordinates and the standard deviation for each estimated coordinate. At least 300 position estimations were performed for each position.

| **Position** | $e_x$ **[cm]** | $e_y$ **[cm]** | $e_y$ **[cm]** | $\sigma_x$ **[cm]** | $\sigma_y$ **[cm]** | $\sigma_z$ **[cm]** |
|---|---|---|---|---|---|---|
| **1** | 1.22 | 0.35 | 1.18 | 0.24 | 0.23 | 0.36 |
| **2** | 2.16 | 0.81 | 1.07 | 0.25 | 0.25 | 0.33 |
| **3** | 0.42 | 0.83 | 0.23 | 0.22 | 0.21 | 0.30 |
| **4** | 0.29 | 1.29 | 1.03 | 0.29 | 0.23 | 0.36 |
| **5** | 2.57 | 1.15 | 0.80 | 0.28 | 0.25 | 0.41 |
| **6** | 1.46 | 0.40 | 1.66 | 0.44 | 0.41 | 0.41 |

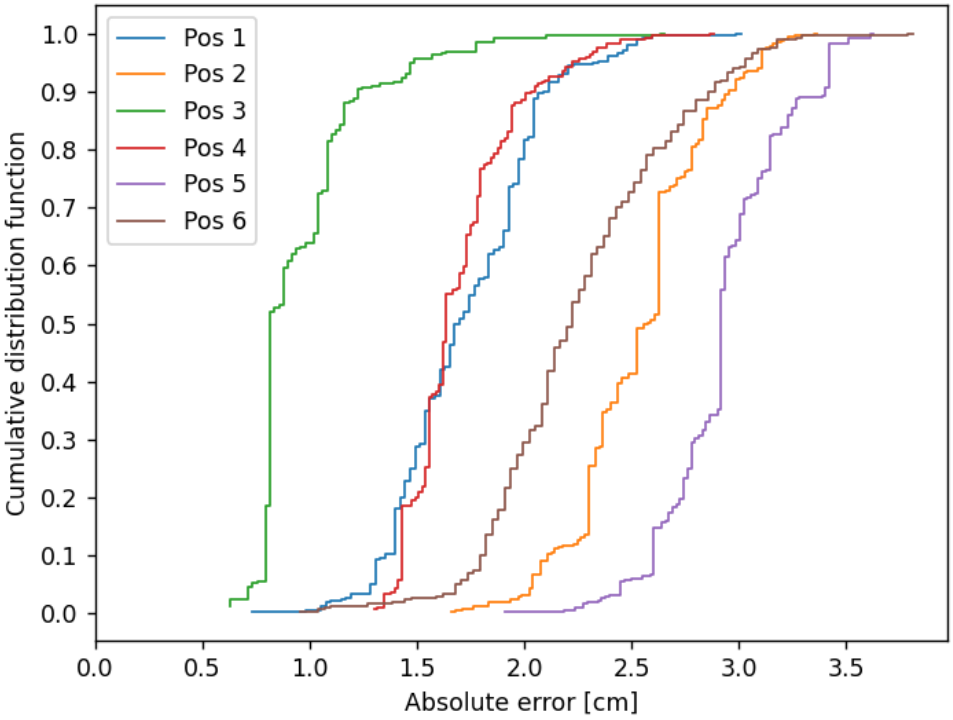



**Figure 28:** Cumulative distribution function of the absolute error in the Euclidean distance between the estimated position and the true position.

### 4.2.3 Dynamic tracking

Dynamic positioning was performed to prove that the positioning system can be used to track a moving target in real-time. The position update rate is 1.25 Hz and this presents a challenge when positioning a dynamic target since it takes 800 ms to estimate the targets static position when using the system of equations (5). This means if the target has moved during one static position estimation cycle, the $x$, $y$ and $z$ coordinates in (5) are not the same for all distances $d_k$. This will result in inaccurate position estimation for fast moving targets.

For this experiment the same setup from Figure 25 and anchor node positions from Figure 26 were used. The receiving node used the external ultrasonic receiver and was placed at start coordinates [3.0, 30, 2.0]. It was then pulled with an USB-cable along the y-axis along a folding meter stick to the end coordinates [3.0, -30, 2.0]. Effort was done to keep the pull slow and with constant velocity, but some jerky movement was inevitable. The goal of this experiment was to look at the stability of the estimated $x$ and $z$ coordinates and the precision

in distance traveled in the $y$ coordinate. Time was measured using the same laptop that receives the ToF measurements and calculates the estimated positions.

Figure 29 shows a plot of the estimated $x$, $y$ and $z$ coordinates vs time, and Table 1 shows the maximum deviation in the $x$, $y$ and $z$ coordinates and the standard deviation of the estimated $x$ and $z$. The figure shows a near constant movement in the $y$-direction while the $x$ and $z$ stays the same. The standard deviation in $z$ coordinates is about the same as in the case for static positioning in Table 5, while for the $x$ coordinate it is slightly larger, but this could be explained by slight movements in the $x$-direction because of the slightly jerky pull. We can also see that some estimated positions are missing, and this is because for whatever reason not all four ToF measurements were successful within 800 ms, the measurement cycle is restarted.

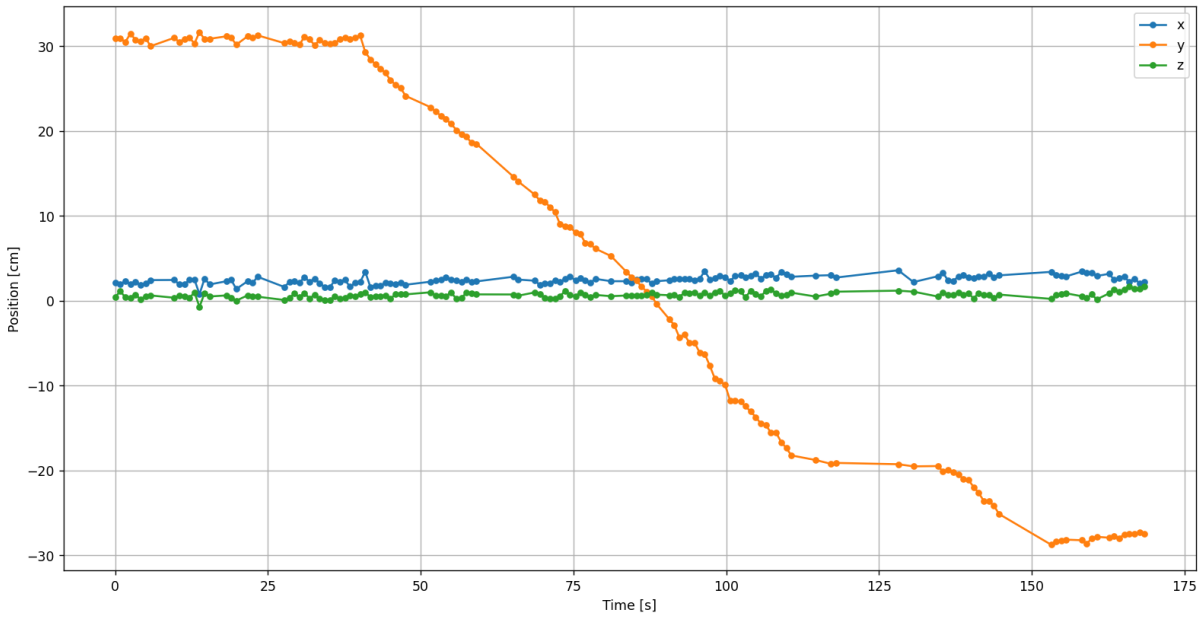



**Figure 29:** Position estimation of a target moving slowly along the y-axis.

**Table 6:** Maximum deviation in the $x$, $y$ and $z$ coordinates, and the standard deviation of the estimated positions of a dynamic target.

| $\mathbf{max}(x) - \mathbf{min}(x)$ **[cm]** | $\mathbf{max}(y) - \mathbf{min}(y)$ **[cm]** | $\mathbf{max}(z) - \mathbf{min}(z)$ **[cm]** | $\sigma_x$ **[cm]** | $\sigma_z$ **[cm]** |
|---|---|---|---|---|
| 2.83 | 60.39 | 2.38 | 0.48 | 0.34 |

# 5 Discussion

## 5.1 Synchronization

The variability in the radio packet delay is quite low and is near constant most of the time, and the maximum measured delay difference was 20 µs which corresponds to a distance error of 0.68 cm. It was expected that the wired synchronization method is more precise than the wireless method, but since the variability of the radio packet delay is so low, the precision in distance measuring is still limited by the wavelength of the 40 kHz ultrasonic signal. Even though the propagation time of radio signals is negligible compared to the ToF of ultrasonic signals, the receive time delay of radio packets is significant. However, since the delay is near constant, it can be calibrated for which results in time synchronization with microsecond precision between the transmitting and receiving nodes.

Still, sending and receiving a synchronization radio packet for every measurement is not energy efficient, and another synchronization protocol such as the Reference Broadcast Synchronization would be better. Or perhaps using a TDoA measuring technique which will eliminate the need for synchronization between transmitters and receivers, and requiring only the transmitters to be synchronized.

More accurate and precise synchronization methods are possible and Medina et al. (2013a) presents solutions to the synchronization problems in their ultrasonic positioning system, and Qi & Liu (2017) managed to reduce the maximum radio packet delay to 1 µs using Reference Broadcast Synchronization method in their ultrasonic positioning system.

## 5.2 Accuracy

A constant bias was expected at all distances, but the results show almost linear increase with distance. A contributor to this could be that the “true distance” which was measured with a measuring tape is not correct. The true distance was measured in 1 m intervals, and the measurement error in each interval adds up with distance. The measurement tape was kept

tight and straight between the transducer casing and the microphone package, but the actual sound producing and receiving membrane is positioned an unknown distance inside the package, and that still limits the measurement accuracy. It is possible that each measured interval was bit longer than 1 m, but best effort was done to try to keep the true distance accuracy smaller than the 9 mm wavelength of the ultrasound. Even though the accuracy of the "true distance" has its limitations, it is unlikely that this measurement error had any significant contribution to the bias that was observed.

Wrong sound speed model can also contribute to increasing bias at further distances, because wrong sound speed results in increasing errors in the estimated ToF distance with increasing distance as shown in equation (4). Temperature is the most significant factor that influences the sound speed and the sensitivity to temperature variation is $1.7\ \mathrm{mm/K/m}$, meaning 1° C error gives an error of 8.5 mm at 5 m. Sound speed modeling is a source of error, but it cannot make up for the big increase in bias with distance. Ideally continuous ambient temperature measurements should be integrated into the positioning protocol, that way an accurate sound speed can be modelled when temperature is changing, but in indoor situations the temperature usually stays constant within a few degrees Celsius.

The most significant contributor to the increasing bias is likely the detection method, because the signal strength decreases with distance and the same pulse is detected later in its arrival (see Figure 14 for an illustration of the pulse detection using a threshold value). Further, Figure 21 shows the actual pulse response using the external receiver, and it shows that the received pulse maximum is about $250\ \mu\mathrm{s}$ after the start of the pulse response. This means with decreasing signal amplitude at further distances only the pulse maximum is over the threshold, while at closer distances the start of the pulse is over the threshold. A $250\ \mu\mathrm{s}$ difference in detection equals to about 8.6 cm, which is close to the difference in 1 m and 5 m distance estimations in Figure 24. The reduced SNR also explains the increasing variance in ToF distance estimation with increasing distance. A solution could be to create an envelope of the received pulse and estimate its maximum, as this maximum would always be at the same position in the received pulse. Another option could be to use a cross correlation detector which maximized the SNR, but this would increase the computational complexity of the detector.

Instead of changing the detection method, this thesis presented how the limits of the detection method can be compensated for by modeling the bias. The bias can be characterized at different distances and modelled using linear regression which can then be used for calibration, but then the accuracy is as good as the bias model. The accuracy requirement is application dependent, and the sub-centimeter 90th percentile accuracy achieved here is good for positioning the Micro Bit boards, since the board length and width are several centimeter.

Both Table 5 and Figure 28 show a stable level of precision but a big variance in accuracy across the six different positons. Based on the accuracy achieved after calibration from Table 3 and Table 4 we should expect better accuracy and not such large difference in minimum absolute error between the estimated positions. This is most likely due to inaccurate "true positions" of the receiver and the anchor nodes, which were also estimations. That is also supported by the standard deviation which is about the same for all positions, only slightly worse than the ToF distance measurements from Table 3, and indicates sub-centimeter precision.

It was not easy to construct a three-dimensional orthogonal space with good accuracy using a folding meter stick, and it was even more difficult to find where the anchor node transducers projected into the xy-plane. It is therefore hard to evaluate the positioning accuracy, but the precision was good and consistent, and it is likely that with more accurate "true positions" the positioning accuracy would be comparable to the accuracy of the ToF distance estimations.

Using the external receiver provided much better accuracy and precision than the internal receiver. Since the same microphone is used on both receivers, it shows that a simple analog front-end can significantly increase the performance. The main downside of both receivers is a wide bandwidth, which makes it more susceptible to out-of-band noise. The external receiver has a second-order high pass filter that dampens audible frequencies, but the onboard microphone has no filter and noise such as loud music could possibly trigger the pulse detection threshold. A sharp analog or digital filter that dampens out-of-band frequencies would make the receivers more robust to noise. Another option could be to use the ultrasound transmitting transducer as a receiver since it has a narrow bandwidth at the operating frequency.

This thesis showed that an accurate and precise ultrasonic positioning system based on a wireless sensor network is possible with COTS devices such as the Micro Bit V2, which has everything but an ultrasonic transducer for transmission onboard. Using the onboard microphone has a limited range and precision, but simple external receivers can be made with COTS components, and there are some COTS ultrasonic receivers on the market such as the HC-SR04 40 kHz range finder made for Arduino.

## 5.3 Update rate

The update rate is affected by the number of nodes and is limited by the time it takes for the reflections die out. The estimated position is updated every 800 ms, but could be decreased when using short pulses and making sure that the line-of-sight pulse does not contain any reflections. The update rate could be doubled for this positioning system, which would give 100 ms time slots to estimate the ToF distance to each anchor node using the time-division multiple access (TDMA) protocol. Sound travels about 34 m during 100 ms, giving enough time for it to be absorbed by the air and walls.

Fast moving targets will experience some latency in their estimated position if the system should be used for live tracking. Positioning accuracy of a moving target is also affected since the target's coordinates would have changed during a measurement cycle. Other multiple access protocols such as frequency-division multiple access (FDMA) or code-division multiple access (CDMA) can increase the positioning update rate by estimating the required ToF distances concurrently. In FDMA this is achieved by assigning different transmitters their own transmission frequency, and CDMA uses coded transmission pulses. Both methods can increase the update rate, as well as robustness to noise, but at the cost of receiver complexity.

## 5.4 Coverage area

The range and coverage area is limited because of the decreasing accuracy and precision beyond 4 m with the external receiver and 2 m with the internal receiver, thus the coverage

area should be limited to those distances. Perhaps even less since the ToF distance estimations were performed with direct line-of-sight between the transducer and the microphone, while for positioning the pulse will come at an angle with less SNR. Coverage area could be improved with a more accurate detection technique such as receiver envelope maximum estimation. It is also dependent on the sound pressure the transmitter can generate and the receiver sensitivity, but with a better detection method these requirements can be reduced. The external receiver has a signal amplifier, but not the transmitter, and the Murata transducers could generate a higher SPL if the 3 V PWM pulse is pre-amplified. If the coverage is limited by the SNR, amplification of the transmitted signal should be considered.

The coverage area of the experiments was limited to about 1.5 m x 1.5 m, and room level coverage area was not tested. The system's deployment area requires no obstacles in the positioning area such that the line-of-sight pulse can be detected. A way to work around this issue could be to install more transmitters in the room, such that at least four transmitters have a line-of-sight to the target, while ensuring low dilution of precision (DOP).

## 5.5 Privacy

Privacy concerns arise in location-based services if the user's position is calculated at a central unit, or if the location is shared or collected in some other way at a central server. The knowledge of a user's position may contribute to the identification of their personal information. A positioning system can be made privacy-oriented by allowing users to calculate their position without sharing it. In positioning systems for tracking robots or objects in a warehouse, maintaining privacy is not a concern, and tracking at a central server is a benefit.

This positioning system is a centralized system since all the ToF estimations are sent to a laptop where the receiver node position is computed, but we need the data on a central device anyway in order to present it. It is also possible for the receiver to calculate its own position without sharing it to make the positioning system privacy-oriented. The issue of privacy has not been a focus of this positioning system, but should be considered in future work.

## 5.6 Deployment effort

It would be easier if the anchor nodes can be deployed near the walls of a room or suspended from the ceiling, because it will make it easier to estimate the xyz-coordinates. Using the room itself as a reference coordinate system is much simpler than constructing a new coordinate system inside the room. The experiments in this thesis were performed in a lab room occupied by workbenches and other inventory so the room itself could not be used as a reference system.

This positioning network's deployment effort is kept low by using small, battery-powered wireless nodes. This makes the system much easier and less costly to install than systems that require connection to wall power, because additional power outlets or signal wires may be required to install, which must be done by certified electricians. Additionally, this positioning system is built using COTS devices, reducing the time and effort to setup this system at the end user. One needs to ensure that for this positioning system to work, the nodes require line-of-sight and proper geometric placement of anchor nodes to ensure low dilution of precision. With some knowledge of embedded programming this system can be easily adapted for different applications.

## 5.7 Multi-user capability

Multi-device positioning was not tested due to not having more than five Micro Bit V2 boards at the time of testing, but the system should support positioning of multiple receivers with only slight modifications to the anchor node software. The ToF measurement cycle is setup such that additional targets can use the same synchronization signals and ultrasonic pulses to estimate their ToF, and will not influence the positioning of other targets. As the software is setup now, targets cannot freely enter or leave the positioning area, and the software needs to be modified according to the number of targets that will be positioned.

## 5.8 Scalability

Scalability of the positioning system was not a prioritized design requirement, and no improvements to the scalability issue have been made here. Since ultrasonic waves do not propagate through walls, the coverage area is limited to the room area, and the same positioning system has to be setup in additional rooms where positioning is required. With at least four anchor nodes required for positioning, scaling can be costly.

# 6 Conclusions

The aim of this thesis was to design a wireless sensor network based ultrasonic indoor positioning system that is built using commercial off-the-shelf devices. The system presented here estimates the ultrasonic ToF between transmitting and receiving wireless nodes using a Micro Bit V2 microcontroller board with an embedded microphone for ultrasound reception, and a Murata MA40S4S ultrasonic transducer for transmission. Because of the poor performance of the embedded receiver, it was replaced with an external receiver which more than doubled the reception range, accuracy and precision. By using four transmitter nodes with known positions and the estimated ToF distances to the receiving node, the receiver was positioned by applying multilateration.

Accurate time synchronization between the wireless nodes is important because ToF estimation requires the precise time of transmission and reception of an ultrasonic pulse. The method of synchronization employed here involves sending a radio signal right before the ultrasonic pulse is transmitted, and the ToF is measured as the time difference between the reception of the radio signal and the ultrasonic pulse. This method inevitably results in an additional bias to the estimated ToF mainly because of the radio packet receive time delay. The experimental results showed that the variance in radio packet delay is less than 4 µs 95 % of the time, meaning as long as the bias is characterized and calibrated for, this method of synchronization works well and does not significantly contribute to the variance in the estimated ToF.

Experimental results of the ToF distance estimations showed that the bias increased with distance, which is due to the limits of the ultrasonic pulse detection method. Instead of changing the detection method, the bias was modeled using linear regression which was used to calibrate the estimated ToF distances. After calibration the estimated ToF distances showed a 90th percentile sub-centimeter accuracy and precision up to 4 m range using the external receiver, with much worse performance using the embedded receiver. The 3D positioning accuracy was difficult to evaluate because the “true” 3D coordinates were estimated using a folding meter stick and were inaccurate. The maximum variance in the estimated 3D coordinates was about 2 cm, and it is likely that the real positioning accuracy of this system is

also below 2 cm. The positioning area was limited to about 1.5 m x 1.5 m in the experiments, but the system's maximum coverage area is limited by the range at which accurate distance estimations can be achieved, which was 4 m for this system.

This thesis demonstrated that an accurate wireless ultrasonic positioning system is possible with off-the-shelf devices, which is easily adaptable for different applications that require accurate positioning with room-level coverage and low deployment effort. With the decreasing cost, increasing processing power of commercial microcontroller boards and easier programming tools, making such systems is increasingly available for the consumers.

## 6.1 Future work

This thesis presented important design considerations and techniques that are used in wireless sensor network based ultrasonic positioning systems, along with the challenges of building such a system. Despite achieving good performance metrics, there are some suggestions for improvements and possible applications.

The most important limiting factors of accuracy and precision of this positioning network are the ultrasonic pulse detection method and bias modeling. This could be improved by implementing another detection technique such as an envelope detector, which would improve accuracy, precision and coverage area. The synchronization method could also be replaced with a more accurate time synchronization method, but mostly to save power from not sending a radio packet for each ToF measurement.

More experiments should be performed with larger coverage area and with better knowledge of the true coordinates, and perhaps with multiple receivers and more anchor nodes for better coverage.

Some possible applications of this system could be to replace ultrasound echo based distance measuring in self-driving robots, which can only measure the relative distance to the object that the transducer is pointed at, or positioning of drones that are built using the Micro Bit. The Micro Bits could even be used as wearable "tags" that can provide accurate human positioning, although this requires modifications to parts of the positioning system.

# Bibliography


Berber, J. (n.d.). *Build a Bat Detector*. Nuts and Volts Magazine. Retrieved May 4, 2022, from https://www.nutsvolts.com/magazine/article/june2011_berber

Devine, J., Finney, J., de Halleux, P., Moskal, M., Ball, T., & Hodges, S. (2019). MakeCode and CODAL: Intuitive and efficient embedded systems programming for education. *Journal of Systems Architecture*, *98*, 468–483. https://doi.org/10.1016/j.sysarc.2019.05.005

Fang, B. T. (1986). Trilateration and extension to Global Positioning System navigation. *Journal of Guidance, Control, and Dynamics*, *9*(6), 715–717. https://doi.org/10.2514/3.20169

Finney, J. (2016). *Architecture* [Illustration]. GitHub. https://github.com/lancaster-university/microbit-docs/blob/master/docs/resources/examples/concepts/architecture.png

Finney, J., Devine, J., Williams, M., Moskal, M., Gault, R., Kent, S., & Auston, J. (2023). *Microbit-v2-samples*. GitHub. https://github.com/lancaster-university/microbit-v2-samples

Holm, S. (2019). *Waves with Power-Law Attenuation*. Springer International Publishing. https://doi.org/10.1007/978-3-030-14927-7

Huang, H., & Gartner, G. (2018). Current Trends and Challenges in Location-Based Services. *ISPRS International Journal of Geo-Information*, *7*(6), Article 6. https://doi.org/10.3390/ijgi7060199

Järve, M. (2022). *Making a Arduino stackable bat detector* [Unpublished work]. Department of Physics. University of Oslo.

Knowles Electronics. (2013). *Knowles SPU0410LR5H-QB-7* (Revision H).

Langley, R. B. (1999). Dilution of precision. *GPS World*, *10*(5), 52–59.

Liu, M., Cheng, L., Qian, K., Wang, J., Wang, J., & Liu, Y. (2020). Indoor acoustic localization: A survey. *Human-Centric Computing and Information Sciences*, *10*(1), 2. https://doi.org/10.1186/s13673-019-0207-4

Medina, C., Segura, J. C., & de la Torre, Á. (2013a). Accurate time synchronization of ultrasonic TOF measurements in IEEE 802.15.4 based wireless sensor networks. *Ad Hoc Networks*, *11*(1), 442–452. https://doi.org/10.1016/j.adhoc.2012.07.005

Medina, C., Segura, J. C., & de la Torre, Á. (2013b). Ultrasound Indoor Positioning System Based on a Low-Power Wireless Sensor Network Providing Sub-Centimeter Accuracy. *Sensors*, *13*(3), Article 3. https://doi.org/10.3390/s130303501

Meyer, T. H., & Elaksher, A. F. (2021). Solving the Multilateration Problem without Iteration. *Geomatics*, *1*(3), Article 3. https://doi.org/10.3390/geomatics1030018

microbit-mark. (2022). *Microbit-overview-2-2* [Illustration]. GitHub. https://github.com/microbit-foundation/dev-docs/blob/3bbf45929105ba08b6f850c5e890cc51d2ac1663/hardware/assets/microbit-overview-2-2.png

Moutinho, J. N., Araújo, R. E., & Freitas, D. (2016). Indoor localization with audible sound—Towards practical implementation. *Pervasive and Mobile Computing*, *29*, 1–16. https://doi.org/10.1016/j.pmcj.2015.10.016

Murata. (n.d.-a). *Application Note MA40S4S/MA40S4R* [Illustration]. https://www.murata.com/-/media/webrenewal/products/sensor/ultrasonic/open/applinote_maopn.ashx

Murata. (n.d.-b). *Murata MA40S4S Ultrasonic Sensors*. Retrieved April 9, 2023, from https://www.murata.com/products/productdetail?partno=MA40S4S

Pierce, A. D. (2019). *Acoustics: An Introduction to Its Physical Principles and Applications*. Springer International Publishing. https://doi.org/10.1007/978-3-030-11214-1

Puricer, P., & Kovar, P. (2007). Technical Limitations of GNSS Receivers in Indoor Positioning. *2007 17th International Conference Radioelektronika*, 1–5. https://doi.org/10.1109/RADIOELEK.2007.371487

Qi, J., & Liu, G.-P. (2017). A Robust High-Accuracy Ultrasound Indoor Positioning System Based on a Wireless Sensor Network. *Sensors*, *17*(11), Article 11. https://doi.org/10.3390/s17112554

Roche, M. (2006). Time synchronization in wireless networks. *CSE574S: Advanced Topics in Networking: Wireless and Mobile Networking (Spring 2006), Washington University in St. Louis*, *23*.

Sertatıl, C., Altınkaya, M. A., & Raoof, K. (2012). A novel acoustic indoor localization system employing CDMA. *Digital Signal Processing*, *22*(3), 506–517. https://doi.org/10.1016/j.dsp.2011.12.001

Sonair. (n.d.). *Sonair*. Retrieved April 21, 2023, from https://sonair.com/technology-for-ultrasonic-3d-mapping/

Ureña, J., Hernández, Á., García, J. J., Villadangos, J. M., Carmen Pérez, M., Gualda, D., Álvarez, F. J., & Aguilera, T. (2018). Acoustic Local Positioning With Encoded Emission Beacons. *Proceedings of the IEEE*, *106*(6), 1042–1062. https://doi.org/10.1109/JPROC.2018.2819938

# Appendix A

## Micro Bit V2 receiver code (main.cpp)

```
#include "MicroBit.h"

MessageBus bus;

// led display setup
static const MatrixPoint ledMatrixPositions[5*5] =
{
    {0,0},{0,1},{0,2},{0,3},{0,4},
    {1,0},{1,1},{1,2},{1,3},{1,4},
    {2,0},{2,1},{2,2},{2,3},{2,4},
    {3,0},{3,1},{3,2},{3,3},{3,4},
    {4,0},{4,1},{4,2},{4,3},{4,4}
};
NRF52Pin row1(ID_PIN_ROW1, P0_21, PIN_CAPABILITY_AD);
NRF52Pin row2(ID_PIN_ROW2, P0_22, PIN_CAPABILITY_AD);
NRF52Pin row3(ID_PIN_ROW3, P0_15, PIN_CAPABILITY_AD);
NRF52Pin row4(ID_PIN_ROW4, P0_24, PIN_CAPABILITY_AD);
NRF52Pin row5(ID_PIN_ROW5, P0_19, PIN_CAPABILITY_AD);
NRF52Pin col1(ID_PIN_P4, P0_28, PIN_CAPABILITY_AD);
NRF52Pin col2(ID_PIN_P7, P0_11, PIN_CAPABILITY_AD);
NRF52Pin col3(ID_PIN_P3, P0_31, PIN_CAPABILITY_AD);
NRF52Pin col4(ID_PIN_P6, P1_5, PIN_CAPABILITY_AD);
NRF52Pin col5(ID_PIN_P10, P0_30, PIN_CAPABILITY_AD);
NRF52Pin* ledRowPins[5] = {&row1, &row2, &row3, &row4, &row5};
NRF52Pin* ledColPins[5] = {&col1, &col2, &col3, &col4, &col5};
const MatrixMap ledMatrixMap{ 5, 5, 5, 5, (Pin**)ledRowPins, (Pin**)ledColPins,
ledMatrixPositions};
MicroBitDisplay display(ledMatrixMap);

//serial setup
NRF52Pin usbTx(ID_PIN_USBTX, MICROBIT_PIN_UART_TX, PIN_CAPABILITY_DIGITAL);
NRF52Pin usbRx(ID_PIN_USBRX, MICROBIT_PIN_UART_RX, PIN_CAPABILITY_DIGITAL);
NRF52Serial serial(usbTx, usbRx, NRF_UARTE0);

//system timer setup
NRFLowLevelTimer systemTimer(NRF_TIMER1, TIMER1_IRQn);
Timer timer(systemTimer);

//adc setup
NRFLowLevelTimer adcTimer(NRF_TIMER2, TIMER2_IRQn);
NRF52ADC adc(adcTimer, 5);

// pin setup for the receiver
MicroBitPin P0(MICROBIT_ID_IO_P0, MICROBIT_PIN_P0, PIN_CAPABILITY_ANALOG);
MicroBitPin P1(MICROBIT_ID_IO_P1, MICROBIT_PIN_P1, PIN_CAPABILITY_AD);
NRF52Pin microphone(ID_PIN_MIC, P0_05, PIN_CAPABILITY_AD);
NRF52Pin runmic(ID_PIN_RUNMIC, P0_20, PIN_CAPABILITY_AD);

//radio
MicroBitRadio radio;

static NRF52ADCChannel *mic = NULL;
static LevelDetector *level = NULL;
static StreamNormalizer *processor = NULL;
static int pulse_counter = 0;
static uint32_t start;
```

```
static uint32_t end;
static PacketBuffer from_mbit;


#define THIS_MBIT 1   // node number for receiving node


// When LevelDetector detects pulse, this code is run by the event handler
void onPulse(MicroBitEvent evt)
{
    adc.setSleep(true);

    pulse_counter++;
    if (pulse_counter >= 10)
        pulse_counter = 0;

    int32_t duration = level->windowCount*5;
    PacketBuffer packet(6);
    uint8_t *buf = packet.getBytes();
    memcpy(buf, &duration, sizeof duration);
    packet[4] = from_mbit[0];
    packet[5] = THIS_MBIT;
    radio.datagram.send(packet);
    serial.printf("%d, %d \n", level->windowCount*5, end-start);
    display.printAsync(pulse_counter);
    level->windowCount = 0;
}


void onRadio(MicroBitEvent evt)
{
    adc.setSleep(false);
    start = evt.timestamp;
    level->status &= ~LEVEL_DETECTOR_HIGH_THRESHOLD_PASSED;
    from_mbit = radio.datagram.recv();
}


void rx_init()
{
    if (mic == NULL){
        // Connect external mic output to P0, and choose gain accordingly
        adc.setDmaBufferSize(128);
        mic = adc.getChannel(P0, true);
        mic->setGain(3,1);
        mic->setBufferSize(128);

        // Uncomment code below to use internal mic, and comment code above
        // mic = adc.getChannel(microphone);
        // mic->setGain(7,0);
        // mic->setBufferSize(128);
        //
        // runmic.setDigitalValue(1);
        // runmic.setHighDrive(true);
    }

    if (processor == NULL)
        processor = new StreamNormalizer(mic->output, 1.0f, true,
DATASTREAM_FORMAT_UNKNOWN);

    if (level == NULL)
        level = new LevelDetector(processor->output, 500, 75, MICROBIT_ID_IO_P0);
}


int main()
{
```

```
    scheduler_init(bus);
    rx_init();

    //set priorities
    NVIC_SetPriority(PWM0_IRQn, 6);          // General Purpose PWM on edge
connector (servo, square wave sounds)
    NVIC_SetPriority(TIMER1_IRQn, 4);        // System timer (general purpose)
    NVIC_SetPriority(TIMER2_IRQn, 3);        // ADC timer.
    NVIC_SetPriority(SAADC_IRQn, 3);         // Analogue to Digital Converter
    NVIC_SetPriority(GPIOTE_IRQn, 4);        // Pin interrupt events
    NVIC_SetPriority(UARTE0_UART0_IRQn, 5); // Serial port
    NVIC_SetPriority(RADIO_IRQn, 2);         // Packet radio

    serial.setBaudrate(115200);
    radio.enable();
    radio.setTransmitPower(3);
    bus.listen(MICROBIT_ID_IO_P0, LEVEL_THRESHOLD_HIGH, onPulse,
MESSAGE_BUS_LISTENER_DROP_IF_BUSY);
    bus.listen(MICROBIT_ID_RADIO, MICROBIT_RADIO_EVT_DATAGRAM, onRadio,
MESSAGE_BUS_LISTENER_IMMEDIATE);
    while(1){
        fiber_sleep(1000);
    }
}
```

# Micro Bit V2 transmitter code (main.cpp)

```
#include "MicroBit.h"

MessageBus bus;

// led display setup
static const MatrixPoint ledMatrixPositions[5*5] =
{
    {0,0},{0,1},{0,2},{0,3},{0,4},
    {1,0},{1,1},{1,2},{1,3},{1,4},
    {2,0},{2,1},{2,2},{2,3},{2,4},
    {3,0},{3,1},{3,2},{3,3},{3,4},
    {4,0},{4,1},{4,2},{4,3},{4,4}
};
NRF52Pin row1(ID_PIN_ROW1, P0_21, PIN_CAPABILITY_AD);
NRF52Pin row2(ID_PIN_ROW2, P0_22, PIN_CAPABILITY_AD);
NRF52Pin row3(ID_PIN_ROW3, P0_15, PIN_CAPABILITY_AD);
NRF52Pin row4(ID_PIN_ROW4, P0_24, PIN_CAPABILITY_AD);
NRF52Pin row5(ID_PIN_ROW5, P0_19, PIN_CAPABILITY_AD);
NRF52Pin col1(ID_PIN_P4, P0_28, PIN_CAPABILITY_AD);
NRF52Pin col2(ID_PIN_P7, P0_11, PIN_CAPABILITY_AD);
NRF52Pin col3(ID_PIN_P3, P0_31, PIN_CAPABILITY_AD);
NRF52Pin col4(ID_PIN_P6, P1_5, PIN_CAPABILITY_AD);
NRF52Pin col5(ID_PIN_P10, P0_30, PIN_CAPABILITY_AD);
NRF52Pin* ledRowPins[5] = {&row1, &row2, &row3, &row4, &row5};
NRF52Pin* ledColPins[5] = {&col1, &col2, &col3, &col4, &col5};
const MatrixMap ledMatrixMap{ 5, 5, 5, 5, (Pin**)ledRowPins, (Pin**)ledColPins,
ledMatrixPositions};
MicroBitDisplay display(ledMatrixMap);

//serial setup
NRF52Pin usbTx(ID_PIN_USBTX, MICROBIT_PIN_UART_TX, PIN_CAPABILITY_DIGITAL);
```

```
NRF52Pin usbRx(ID_PIN_USBRX, MICROBIT_PIN_UART_RX, PIN_CAPABILITY_DIGITAL);
NRF52Serial serial(usbTx, usbRx, NRF_UARTE0);

//system timer setup
NRFLowLevelTimer systemTimer(NRF_TIMER1, TIMER1_IRQn);
Timer timer(systemTimer);

// pin setup for the transmitters
MicroBitPin P0(MICROBIT_ID_IO_P0, MICROBIT_PIN_P0, PIN_CAPABILITY_ANALOG);
MicroBitPin P1(MICROBIT_ID_IO_P1, MICROBIT_PIN_P1, PIN_CAPABILITY_AD);
MicroBitPin P5(MICROBIT_ID_IO_P5, MICROBIT_PIN_P5, PIN_CAPABILITY_AD);

//button A is used to node "A" to start positioning
Button buttonA(P5, DEVICE_ID_BUTTON_A, DEVICE_BUTTON_ALL_EVENTS, ACTIVE_LOW);

//radio
MicroBitRadio radio;

static NRF52PWM *pwm = NULL;
static MemorySource *pwmSource = NULL;
static uint16_t square[4];
static uint32_t last_packet;
static int run_sys = 0;

#define THIS_MBIT 'A'       // change for each transmitting node
#define MICROBIT_ID_SEND_PULSE 4000


// Button A is used to turn the positioning system on and off
void run(MicroBitEvent evt){
    if (run_sys){
        run_sys = 0;
    }
    else {
        run_sys = 1;
        Event(MICROBIT_ID_SEND_PULSE, THIS_MBIT);
    }
}


void onRadio(MicroBitEvent evt){
    PacketBuffer p = radio.datagram.recv();
    last_packet = evt.timestamp;

    if (p.length() > 1){
        // if packet length > 1, then we received ToF data
        int32_t *duration = (int32_t *) &p[0];

        if (THIS_MBIT == 'A'){
            //  node "A" prints ToF data to serial
            serial.printf("%d, %c\n ", *duration, p[4]);

            if (p[4] == 'D'){
                // restart the positioning cycle
                fiber_sleep(200);
                Event(MICROBIT_ID_SEND_PULSE, THIS_MBIT);
            }
        }

        if ((THIS_MBIT - p[4]) == 0x01){
            // check which transmitting node is the next to send the ultrasonic
            // pulse according to TDMA protocol
            fiber_sleep(200);
            Event(MICROBIT_ID_SEND_PULSE, THIS_MBIT);
        }
    }
}
```

```
// Send ultrasonic pulse
void sendPulse(MicroBitEvent evt){
    PacketBuffer packet(1);
    packet[0] = THIS_MBIT;
    if (run_sys || (THIS_MBIT != 'A')){
        radio.datagram.send(packet);
        pwmSource->play(square, 4, 10);
    }
}


void tx_init()
{
    if (pwmSource == NULL)
        pwmSource = new MemorySource();

    if (pwm == NULL)
        pwm = new NRF52PWM(NRF_PWM0, pwmSource->output, 40000);

    pwm->connectPin(P1, 0);     // Connect PWM to P1
    pwm->setDecoderMode(PWM_DECODER_LOAD_Common);
    pwm->setStreamingMode(true, false);

    for (int i=0; i<4; i++)
        square[i] = pwm->getSampleRange()/2;
}


int main()
{
    scheduler_init(bus);
    tx_init();

    //set priorities
    NVIC_SetPriority(PWM0_IRQn, 3);         // General Purpose PWM on edge
connector (servo, square wave sounds)
    NVIC_SetPriority(TIMER1_IRQn, 5);       // System timer (general purpose)
    NVIC_SetPriority(TIMER2_IRQn, 6);       // ADC timer.
    NVIC_SetPriority(SAADC_IRQn, 6);        // Analogue to Digital Converter
    NVIC_SetPriority(GPIOTE_IRQn, 5);       // Pin interrupt events
    NVIC_SetPriority(UARTE0_UART0_IRQn, 4); // Serial port
    NVIC_SetPriority(RADIO_IRQn, 2);        // Packet radio

    serial.setBaudrate(115200);
    radio.enable();
    radio.setTransmitPower(3);
    P5.eventOn(MICROBIT_PIN_EVENT_ON_EDGE);
    bus.listen(MICROBIT_ID_IO_P5, MICROBIT_PIN_EVT_RISE, run,
MESSAGE_BUS_LISTENER_IMMEDIATE);
    bus.listen(MICROBIT_ID_SEND_PULSE, THIS_MBIT, sendPulse,
MESSAGE_BUS_LISTENER_QUEUE_IF_BUSY);
    bus.listen(MICROBIT_ID_RADIO, MICROBIT_RADIO_EVT_DATAGRAM, onRadio,
MESSAGE_BUS_LISTENER_QUEUE_IF_BUSY);
    display.printAsync(THIS_MBIT);
    while(1){
        if ((system_timer_current_time_us() -
 last_packet > 1000000) && (THIS_MBIT == 'A')){
            //if for some reason a full measurement cycle was not completed withing
1 second, then restart.
            //this will happen if the receiver does not detect the ultrasonic pulse
            Event(MICROBIT_ID_SEND_PULSE, THIS_MBIT);
        }
        fiber_sleep(500);
    }
}
```

## Replace the LevelDetector class member function pullRequest with the code below (LevelDetector.cpp).

```
int LevelDetector::pullRequest()
{
    ManagedBuffer b = upstream.pull();
    int16_t *data = (int16_t *) &b[0];
    int samples = b.length() / 2;

    for (int i=0; i < samples; i++)
    {
        if (!(status & LEVEL_DETECTOR_HIGH_THRESHOLD_PASSED)){
            windowCount++;      //repurpose this variable
        }

        if ((!(status & LEVEL_DETECTOR_HIGH_THRESHOLD_PASSED)) && abs(*data) > high
Threshold){
            Event(id, LEVEL_THRESHOLD_HIGH);
            status |=  LEVEL_DETECTOR_HIGH_THRESHOLD_PASSED;
            status &= ~LEVEL_DETECTOR_LOW_THRESHOLD_PASSED;
        }
        data++;
    }
    return DEVICE_OK;
}
```

# Appendix B

## Python code to read serial data from the Micro Bit, and calculate and plot live positions.

```
import threading
import serial  # pyserial: https://pyserial.readthedocs.io/en/latest/
import queue
import time
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation


# Create a command queue for key
cmd_queue = queue.Queue(10)

# Keyboard thread to detect input commands and put them in the command queue
def keyboard(run_app):
    while run_app():
        cmd = input("\r\n> ")
        cmd_queue.put(cmd)
        time.sleep(0.5)


# Keep program running as long as True. Terminate by writing "quit" in terminal
window
run_app = True

bias_model1 = 1.00832649*np.arange(0, 301, 1) + 4.726687024097089
bias_model2 = 1.03499292*np.arange(301, 501, 1) - 4.527503445573814
#bias model for the external receiver:
bias_model = np.concatenate((bias_model1, bias_model2)) - np.arange(0, 501, 1)

#bias model for the internal receiever:
# bias_model = 1.0278285*np.arange(0, 251, 1) + 3.9625963672100966 - np.arange(0,
251, 1)


def t_bias(value):
    # subtract the modeled bias value that is closest to the measured ToF distance
    index = (np.abs(bias_model - value)).argmin()
    return bias_model[index]


temp = 21.5                                 # ambient temperature
c = np.sqrt(1.4*287*(273.16+temp))          # speed of sound model
mbit_list = []                              # used to double check the measurement
order A, B, C, D

A = np.array([-70.5, -82.8, 126.5])         #3D coordinates for anchor node A
B = np.array([65.2, -101., 127.5])          #3D coordinates for anchor node B
C = np.array([73.2, 48.4, 126.5])           #3D coordinates for anchor node C
D = np.array([-60.7, 63.7, 127.5])          #3D coordinates for anchor node D

#generate a matrix
a = 2*np.array([[A[0] - B[0], A[1] - B[1], A[2] - B[2]],
            [A[0] - C[0], A[1] - C[1], A[2] - C[2]],
```

```
            [A[0] - D[0], A[1] - D[1], A[2] - D[2]]])

dist = []                                   # list of estimated ToF distances
x = [[17.7], [5.8], [37.5]]                 # nested list of the estimated 3D
coordinates
Time = []                                   # list of time instances when positioning
was estimated


def multilateration():
    b = np.array([-np.dot(B, B) + np.dot(A, A) - dist[-1][0]**2 + dist[-1][1]**2,
                  -np.dot(C, C) + np.dot(A, A) - dist[-1][0]**2 + dist[-1][2]**2,
                  -np.dot(D, D) + np.dot(A, A) - dist[-1][0]**2 + dist[-1][3]**2])
    lstsq = np.linalg.lstsq(a, b, rcond=-1)

    x[0].append(lstsq[0][0])
    x[1].append(lstsq[0][1])
    # x[2].append(lstsq[0][2])
    # another way to calculate the z-coordinate if lstsq doesn't give a stable
value:
    x[2].append(D[2] - np.sqrt(dist[-1][3]**2 - (D[0] - x[0][-1])**2 - (D[1] -
x[1][-1])**2))
    Time.append(time.perf_counter())

    print(x[0][-1], x[1][-1], x[2][-1], len(x[0]))


# Thread for reading serial data
def serial_data(run_app):
    # Open serial port
    ser = serial.Serial("COM12", 115200, timeout=1)

    while run_app():

        no_bytes = ser.inWaiting()
        if no_bytes >= 9:
            data = ser.read(no_bytes)
            if len(data) == 9:
                ToF = int(data[0:4])
                mbit = chr(data[6])

                if mbit == "A":
                    distA = c*1e-4*ToF
                    distA = distA - t_bias(distA)
                    mbit_list = [1]
                elif mbit == "B":
                    distB = c*1e-4*ToF
                    distB = distB - t_bias(distB)
                    if len(mbit_list) == 1:
                        mbit_list.append(1)
                elif mbit == "C":
                    distC = c*1e-4*ToF
                    distC = distC - t_bias(distC)
                    if len(mbit_list) == 2:
                        mbit_list.append(1)
                elif mbit == "D":
                    distD = c*1e-4*ToF
                    distD = distD - t_bias(distD)
                    if len(mbit_list) == 3:
                        dist.append([distA, distB, distC, distD])
                        multilateration()
                        mbit_list.clear()

        time.sleep(0.010)
    ser.close()
```

```
def update_plot(i):
    ax.clear()
    ax.scatter(A[0], A[1])
    ax.annotate("A", (A[0]+1, A[1]))
    ax.scatter(B[0], B[1])
    ax.annotate("B", (B[0]+1, B[1]))
    ax.scatter(C[0], C[1])
    ax.annotate("C", (C[0]+1, C[1]))
    ax.scatter(D[0], D[1])
    ax.annotate("D", (D[0]+1, D[1]))
    ax.scatter(x[0][-1], x[1][-1])
    ax.annotate("x1", (x[0][-1]+1, x[1][-1]))
    ax.set_xlim([-75, 75])
    ax.set_ylim([-110, 70])
    plt.xlabel("x [cm]")
    plt.ylabel("y [cm]")
    ax.grid()


if __name__ == "__main__":

    # Create threads
    keyboard_thread = threading.Thread(target=keyboard, args=(lambda: run_app,))
    serial_thread = threading.Thread(target=serial_data, args=(lambda: run_app,))

    # Set threads as deamon -- threads are automatically killed if program is
killed
    keyboard_thread.setDaemon(True)
    serial_thread.setDaemon(True)

    # Start threads
    keyboard_thread.start()
    serial_thread.start()
    # scatter_thread.start()

    # plot the estimated positions live
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    ani = animation.FuncAnimation(fig, update_plot)
    plt.show()

    while run_app:
        while not cmd_queue.empty():
            cmd = cmd_queue.get()
            if "quit" == cmd.lower():
                np.savetxt("3Dpositioning_emic0.txt", np.transpose(np.array(x))[1:,
 :], delimiter=',')
                np.savetxt("3Ddistances_emic0.txt", np.array(dist), delimiter=',')
                np.savetxt("3Dtime_emic0.txt", np.array(Time), delimiter=',')
                run_app = False

        time.sleep(0.1)
```