UNIVERSITY OF SOUTHERN DENMARK

ROBOT SYSTEMS ENGINEERING

START DATE: 22. SEPTEMBER 2023

DUE DATE: 22. DECEMBER 2023

GROUP NUMBER: 5

INSTRUCTOR: LEON BODENHAGEN

---

# Mobile Robotic System

## The Turtlebot Burger

---

*Submitted By:*

ALAN RASHID, 07-04-1998, *alras22@student.sdu.dk*

ANAS BUTT HUSSAIN, 15-12-2003, *anhus22@student.sdu.dk*

JAKUB HUBERT RUDOWSKI, 01-02-2001, *jarud22@student.sdu.dk*

PATRICK ØRSTED POVEY ANDERSEN, 10-01-2000,
*paand22@student.sdu.dk*

ROBIN HANSEN, 08-11-1995, *rohan22@student.sdu.dk*

THOMAS KORSGAARD VILHOLM, 18-08-1997,
*thvil22@student.sdu.dk*

# Contents

**1 Introduction** **1**

**2 Project Description** **3**

**3 Fundamentals of the Audio processing protocol** **4**

**4 Audio processing** **5**

4.1 Applying FFT . . . . . 5

4.2 Combating noise . . . . . 7

4.2.1 Control Array . . . . . 7

**5 The Protocol** **8**

**6 Robot movement** **12**

**7 Optimization** **13**

7.1 Audio Optimization . . . . . 13

**8 Experiment** **15**

8.1 Range Limitation Assessment . . . . . 15

**9 Discussion** **17**

9.1 Challenges in Tone Reception . . . . . 17

9.2 Challenges in Robot Movement . . . . . 17

9.3 Battery Level Monitoring . . . . . 17

**10 Conclusion** **18**

**11 Appendix** **19**

**12 Bibliography** **20**

# 1 Introduction

The vision for the project was to create a tracking robot equipped with a camera, aimed at monitoring the elderly. The robot was designed to detect a person's hand which it would follow. Upon initiating tracking, the robot first transmits its current location to the computer server, establishing this point as its origin. The system employs DTMF sounds as a means of communication. The computer server would receive the DTMF sounds and the algorithm would then establish the polar coordinates which the robot was trying to send. The process was overall made by incorporating a working communication protocol, an algorithm for the robot movement and an algorithm for sending and receiving the data through DTMF sounds. These steps are written in this report.

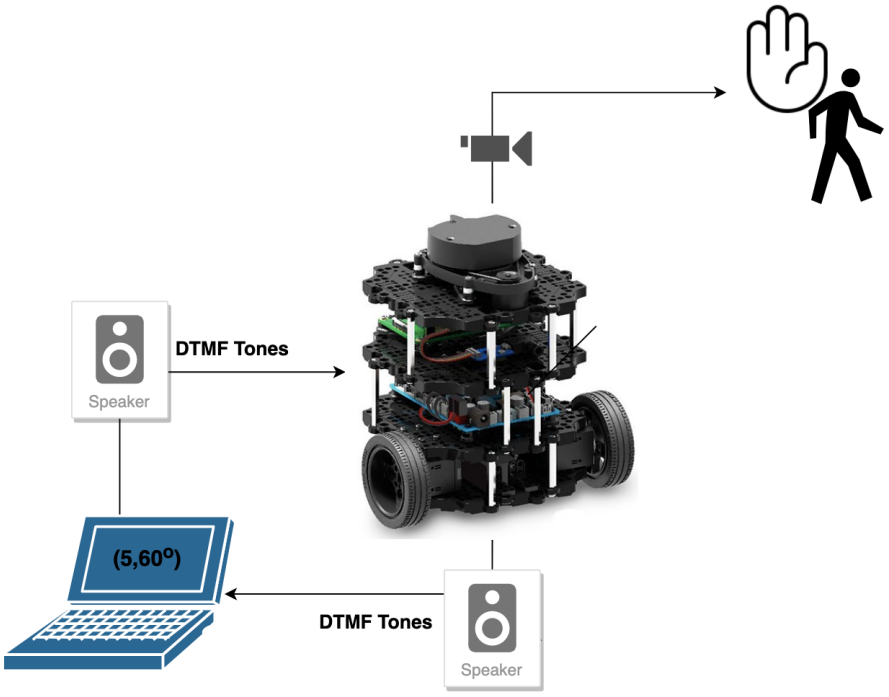


Figure 1: The diagram shows a simplistic view of how the whole system is supposed to work

Below are the required tasks to complete the goal:

- Task 1: Signal processing
- Task 2: Robot movement
- Task 3: Communication Protocol
- Task 4: Experiments

The tasks were distributed as follows:

- Task 1: Robin, Thomas, Patrick
- Task 2: Jakub, Alan, Anas
- Task 3: Robin, Patrick
- Task 4: Alan, Thomas, Jakub

## 2 Project Description

The primary objective of this project is to create a mobile robot, that is able to follow and monitor a person, additionally it is able to send its polar coordinates via DTMF tones to another PC.

The solution encompasses the following key components:

1. A person-tracking algorithm: Utilizes the MediaPipe library and webcam to detect and follow a hand in real time.
2. Communication protocol: Creates a base for understanding how the robot and PC were communicating through DTMF sounds.
3. A signal processing algorithm: Allows the robot to encode and transmit its coordinates to a receiving PC using Dual-tone Multi-frequency signaling.

By addressing these objectives, the project aims to develop a sophisticated robotic companion that can assist in personal monitoring.

## 3 Fundamentals of the Audio processing protocol

DTMF frequencies rounded to the closest number divisible by 10, are used to send positional information about the robot. But they are not really ”DTMF” tones, we call them ”MTMF” (”Multi-Tone Multi-Frequency”). The main difference is the absence of original frequencies due to rounding, and the use of a distinct four-by-four matrix with different frequencies. This is to allow higher payloads to be achieved. The reasoning behind this decision can be appreciated in section 5.

**Why are the frequencies rounded to the closest number divisible by 10?**
Because these frequencies exactly map onto the FFT frequency resolution bins when using a sample rate of 5120, and a frame size of 512. (Frequency resolution = $\frac{\text{Sample Rate}}{\text{Frame Size}}$)

**Why this specific sample rate and frame size?**
The frame size was chosen because of the ”Cooley Tukey” based FFT algorithm that the numpy fft library makes use of.*”FFT (Fast Fourier Transform) is an efficient method for computing the discrete Fourier Transform (DFT), by using symmetries in the calculated terms. The symmetry is highest when n is a power of 2, and the transform is therefore most efficient for these sizes.”* [2]

This means that the chosen frame size should optimally be equal to some power of 2. And, the highest frequency that the FFT can reliably measure, is half the sampling rate (also called the ”Nyquist Frequency”). The range of frequencies to capture in this project goes from 400 to 1910 Hz. The chosen sample rate should therefore be at least 3820 per second. However, to fit the target frequencies (all divisible by 10) into the output bins of the FFT, a frame size of 512 is chosen. This size, the first power of 2 that, when multiplied by 10, exceeds the minimum sample rate of 3820. Hence, to efficiently bin the FFT output for the desired frequencies while ensuring computational efficiency, selecting a frame size of 512 and a sample rate of 5120 is the most suitable option.

# 4 Audio processing

To expand on Audio Processing, we'll explore the following key points:

- Applying the FFT (Fast Fourier Transform) on the input signal.
- Removing frequencies outside the DTMF range.
- Finding dominant frequencies.
- Further noise control.

## 4.1 Applying FFT

To understand the Fast Fourier Transform, one first needs to understand the Discrete Fourier Transform:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn} \tag{1}$$

Where:

- N = total amount of samples.
- n = current sample index.
- k = frequency domain index.
- (NOTE FOR LATER: $f_k = \frac{k \cdot f_s}{N}$, would be the actual frequency at index $k$, while $f_s$ is the sampling rate)

The Discrete Fourier Transform shifts signals from time to frequency domain, emphasizing dominant frequencies. When $k > \frac{N}{2}$ and evaluating real-valued signals, which audio signals are, "negative frequencies" which mirror the positive values arise. These negative frequencies can be discarded without information loss. This also ensures staying at or under the Nyquist frequency $f_q$, which is the highest frequency that can be reliably measured given some sample rate $f_s$:

$$f_q = \frac{f_s}{2}$$

For the scope of this paper, this level of understanding is sufficient. If you want to investigate this further, you can take a look at [1] for a beautifully explained introduc-

tion to the Fourier transform and how the frequency domain is reached through it. You can also take a look at figure 2 and figure 3 below.

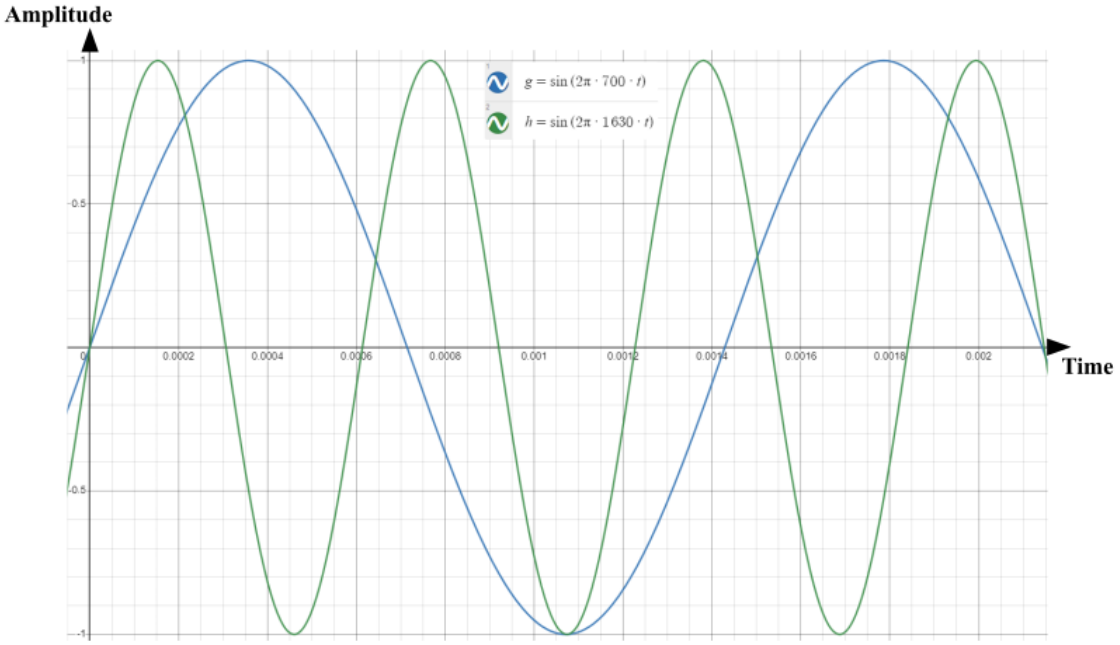


Figure 2: Audio Input

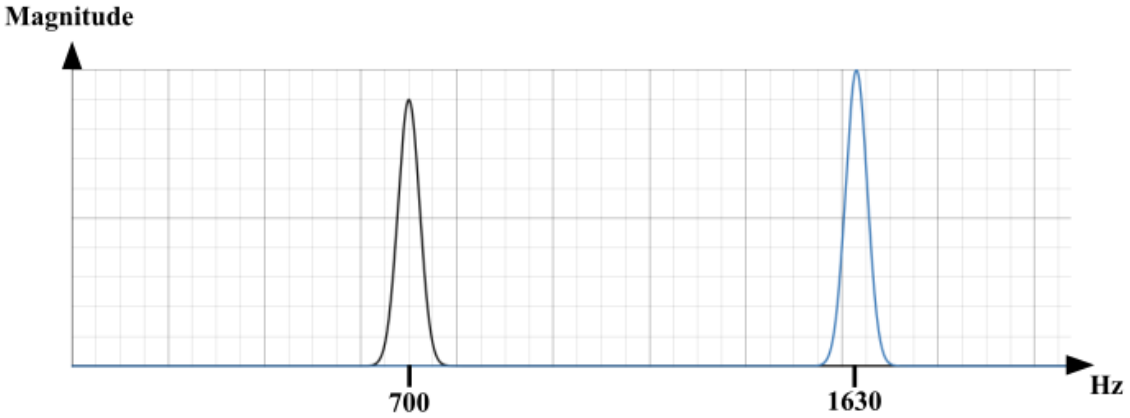


Figure 3: FFT Output

The Fast Fourier Transform does exactly the same as the DFT when it comes to output. The FFT however, is not a function, but an algorithm that makes use of the DFT, and exploits the symmetry and repeating patterns found in the DFT (the details of which are beyond the scope of this paper). This exploit reduces the required computations from $O(N^2)$ to $O(N \log N)$.

For example, for a 512 frame size to be computed with DFT 262144 computations are required, whereas FFT would only require 4608 computations. Understandably,

this makes FFT quite popular for applications requiring audio editing, and much more.

After computing the frequency domain of the input signal, the output gives a clear picture of the dominant frequencies in the signal (as shown by figure 3 above). These dominant frequencies can then be analyzed to see if they match any MTMF tones. From this point on, some approaches to combat noise can be taken.

### 4.2 Combating noise

One of the ways to combat noise, would be to simply remove all frequency bins that do not map to any MTMF frequency.This method underwent light testing, and while it significantly reduced false positives, relatively low levels of noise were required to yield true positives. Unfortunately, a log of this testing was not maintained.

Another way to combat noise would be to allow for frequencies unrelated to the desired frequencies, but magnify the magnitude of relevant frequencies, this way, despite noise, a signal may be read despite noise frequencies having greater amplitudes. Although this method was better than the first at capturing MTMF signals despite noise, it was more prone to false positives. Unfortunately, this testing was not logged.

In any case, both scenarios could trigger false positives. To control for this, a ”Control Array” was implemented.

#### 4.2.1 Control Array

The ”Control Array” is simply a ”sliding window system” that holds the last few registered values. The array is of some length that is manually specified, in our case the length was set to 5. When an MTMF tone is caught, the tone value will be added to the control array, when the same value appears thrice, that value will be registered as valid, this way false positives largely become a non-problem, and even if a false positive were to slip through, it will be checked further down the line in the communication protocol (look at the ”Error Handling” point in section 5). Also, when no matching MTMF Tone is caught in a frame, a ”None” value is added to the array to keep it cleared of values that are no longer current. With the information from both the current and the previous sections in mind, it is now possible to proceed to the actual protocol.

## 5 The Protocol

The communication protocol of this project employs a unique approach to transmit and interpret data using ”MTMF” signals. The process is as follows:

Each four-by-four carries distinct tone combinations, each of which can be represented with a single hexadecimal symbol. With two four-by-four matrices, 255 unique values can be sent, offering extensive flexibility, as each value also can correspond to a potential command. Below in figure 4, a map of the DTMF frequencies rounded to the nearest number divisible by 10. Figure 5 shows a new 4x4 matrix of frequencies that are used in combination with the rounded DTMF sounds.

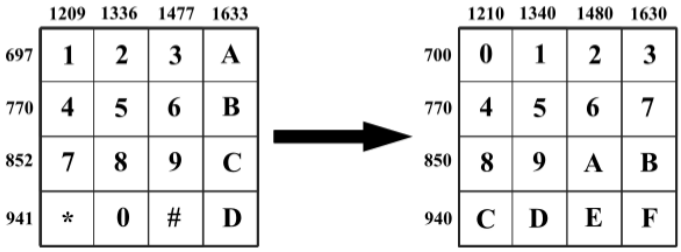


Figure 4: Rounding DTMF frequencies

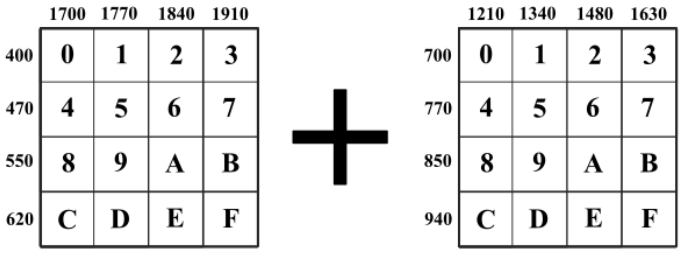


Figure 5: Combining two four-by-four frequency matrices. Capturing the frequencies 1700, 550, 1480 and 700 all together, would for example mean that the hexadecimal value 92 was received

There are two aspects to this protocol, the ”sender”, and the ”receiver”. Below, a point-by-point walk-through:

- **Sending signals (sender):** To send a signal, first a condition has to be met; The robot has had to be in motion, and then stop. Now that the Robot is no longer in motion, it will send its current position in polar coordinates relative to its initial position; This process repeats everytime the robot is set in motion. There are two steps to sending a signal, the first step is sending a raw value for the distance

and/or angle, and the second step is accepting/declining the echo the receiver sends back, while clarifying if the raw value corresponds to a radial or angular value

- **Echoing signals (receiver):** When the receiver detects a MTMF tone, it will echo it back, but with a caveat. The matrices that make up the hexadecimal map in figure 5, will have the first and fourth quadrant switch places along with the second and third quadrants (see figure 6). The reasoning behind this, is that when the sender sends a signal, and simultaneously listens for an echo, the echo will not share any frequencies with the original signal, making communication more efficient. For example, if the value A5 is sent, which makes use of the frequencies 1840, 550, 1340, and 770, the echo will also be A5, but consist of the frequencies 1700, 400, 1630, and 940.

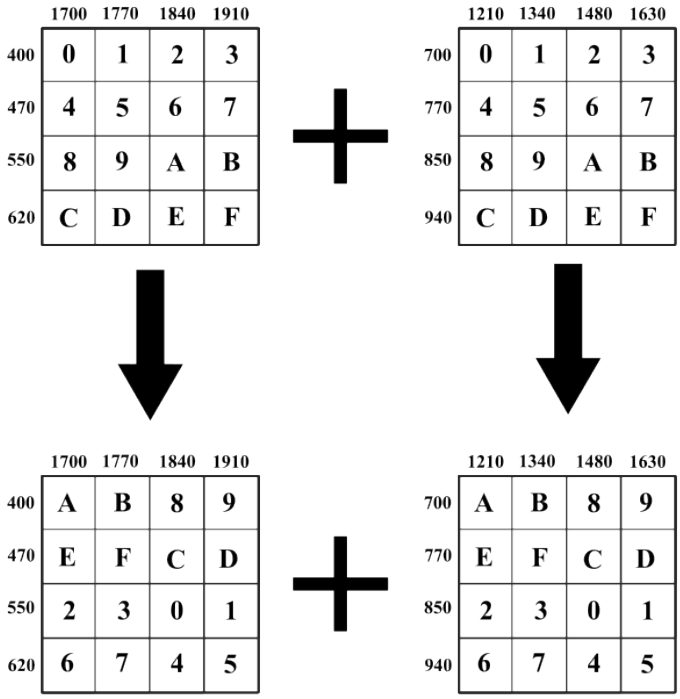


Figure 6: The upper two matrices, is a mapping of frequencies to hexadecimal values in the senders perspective. In the lower two matrices, the first and fourth quadrant switch places along with the second and third quadrant, this represents the mapping of frequencies to hexadecimal values in the receivers perspective.

- **Confirming signals (sender):** When the receiver detects an echo, it will compare it to the value it originally sent. If the echo matches the original value, it will send a confirmation. There are three types of confirmations:

- FF: Set distance, stay on pointer
- FE: Set distance, angle=0
- FD: Set angle times 2 (and if distance not already set, set distance=0)

When the receiver detects a confirmation, it will store the raw values according to figure 7

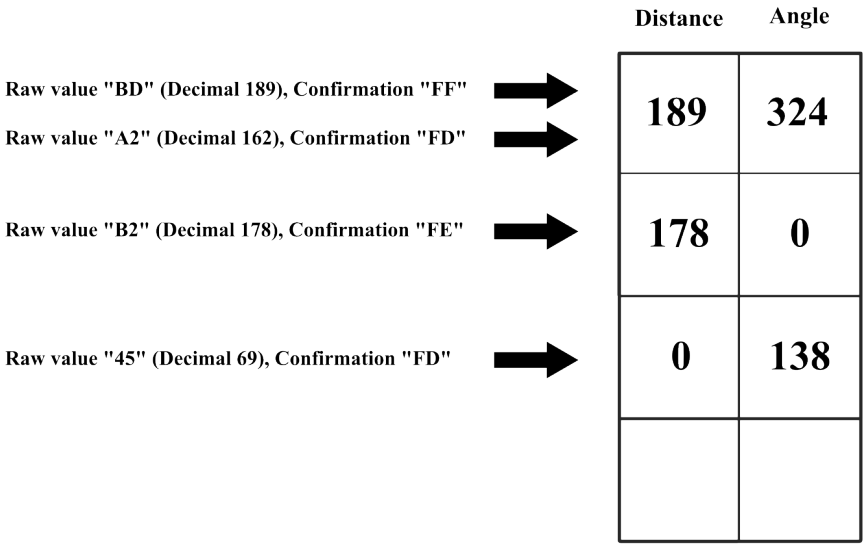


Figure 7: Storing raw values according to confirmations.

- **Echoing confirmations (receiver):** For good measure, the receiver will echo back confirmations, this gives the sender an opportunity to resend a confirmation signal if the echo does not match the original confirmation. If the receiver detects a confirmation tone before detecting a new raw value, it will take the confirmation tone as a command to overwrite the previously stored value according to the new confirmation signal. Below is a chart (figure 8) showing the complete communication process.

In summary, the project's communication protocol proves to be efficient and resilient to noise. Key features include the use of hexadecimal data units for data transmission and a robust error handling mechanism. The protocol's quick tone transmission and confirmation, despite occasional misinterpretations, align well with the application's needs, as demonstrated in fig. 8

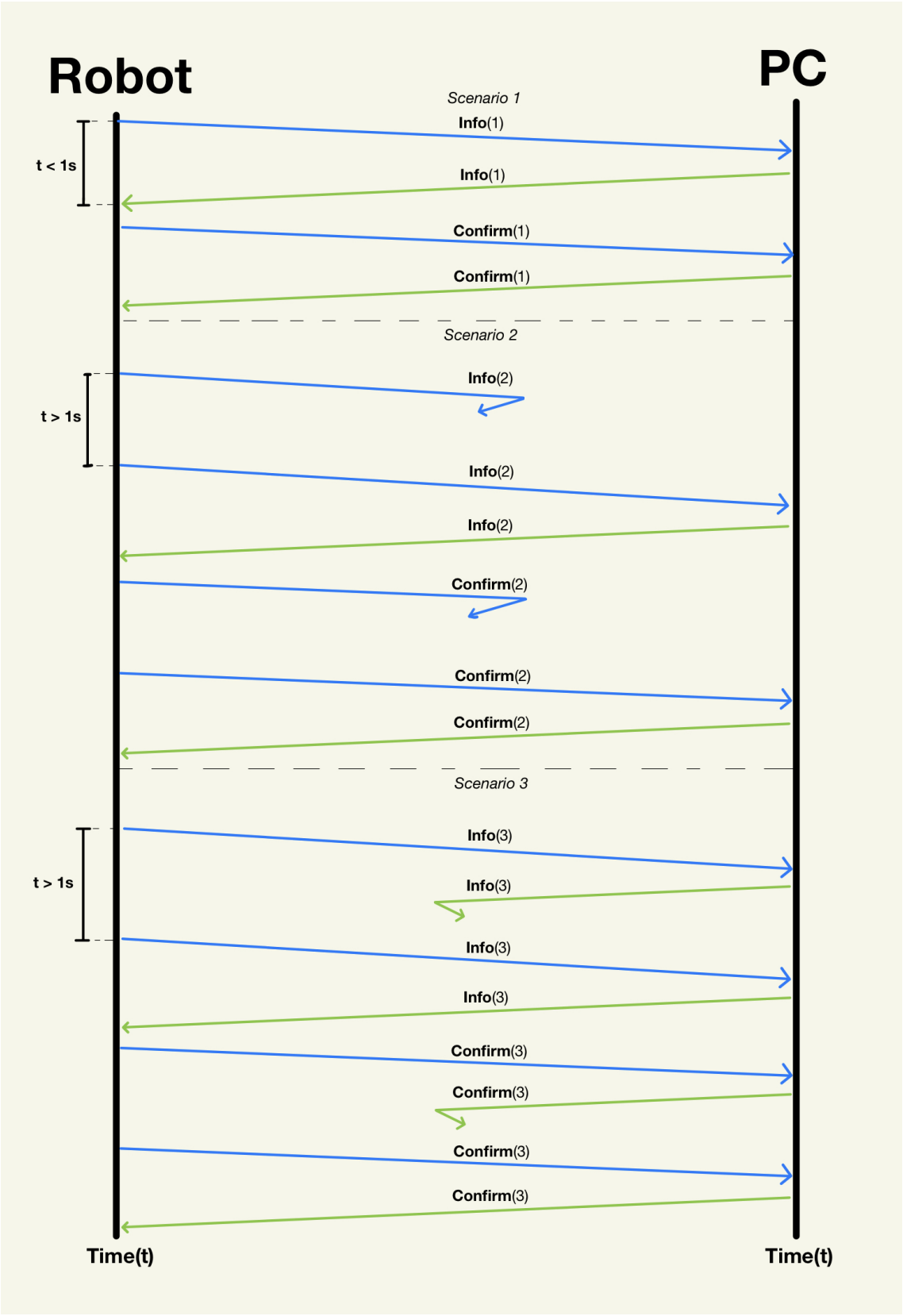


Figure 8: A visualisation of the communications protocol over time, showing different scenarios that can happen in the communication between the robot (sender) and computer (receiver). The blue arrows belong to the sender, the green arrows belong to the receiver. The sender will keep sending the same signal if the echo is either not heard at all, or invalid

## 6 Robot movement

The robot's movement functionality is outlined as follows: It is developed in Python, facilitating hand detection. The hand detection algorithm draws inspiration from Murtaza's Workshop ([3]).

**• Initializing processes needed:**

Mediapipe library is initialized to detect a hand in the video capture frame. The frame from the logitech camera is also inizialied with the usage of the OpenCV library.

**• Python functions needed:**

There were two main functions used update_position_and_angle (l. 402 in TrackingRobot.py) and calculate_distance_of_hand (l. 425 in TrackingRobot.py). The update function creates a X and Y coordinate system where 1 unit is equal to 1 human step. This coordinate system helps the user to navigate to the placement of the robot. The algorithm determines angular displacement and then projects the robot's movement onto the X and Y axes using cosine and sine functions, based on the duration of movement. The X and Y coordinates are converted from Cartesian to polar form.

**• Movement of the robot:**

The robot moves by using the Twist functions, which were implemented from the ROS system library. These functions are responsible for the angular and linear movement of the robot. The robot activates if the hand, as detected by the camera, was over 50 cm away, as specified in the code (lines 447-560 of TrackingRobot.py). If there has not been detected any hands then the robot stops. The hand's distance was calculated with simple equations found in: (l. 458-472 in TrackingRobot.py).

**• Updating and sending coordinates:**

While the robot moves close enough to the hand, the robot would stop and send the coordinates via the communication protocol and the signal processing algorithm.

In conclusion, the robot movement algorithm implements two libraries MediaPipe and OpenCV. The robot moves when the hand was in webcam's frame, the algorithm detected the distance of the hand and the Twist functions send a geometry messages to the robot with usage of the ROS system library.

# 7 Optimization

This section is dedicated to the trials, errors, and improvements made in the project to enhance its efficiency and effectiveness.

## 7.1 Audio Optimization

- **MTMF Tone Duration:**The reason behind wanting to send the MTMF tone for a duration that is neither too long or too short is that the receiving end can sometimes detect it quicker or slower than expected. In the project the optimal duration to play the tone was around 0.5 seconds because the control array needs at least 3 out of 5 values to confirm that a tone has been detected correctly, and each value is detected within 0.1 seconds due to the sample rate and frame size choices
- **Signal Processing Techniques:** A sliding window system, named 'Control Array', was introduced to handle potential false positives. This array, stores the last few detected values. It validates a tone only after it has appeared thrice in the array. It was this limit that was found to be the most optimal, paired with the playtone() functions duration of 0.5 seconds.
- **Turtle-bot and PC Tone Difference Implementation:** To reduce the chance of accidentally detecting a tone senders own tone, the matrix for the tones on the Turtle-bot and PC have been flipped. Meaning that quadrant 1 would be quadrant 4 in the other matrix, and quadrant 2 would be quadrant 3 (fig. 5). This proved effective in reducing the chance of either side detecting something incorrect.
- **Decision Against Serial Bitstream Transmission Method:** Originally, the robot's X and Y coordinates were transmitted by converting them into a binary string and sending an MTMF tone for each bit. The receiving code would then know which bit to set to 1 in the 8-bit string. (the 8bit string in the receiving code was initialized with [0,0,0,0,0,0,0,0]) This way of transmission was slow and not optimal for a moving robot where the position can change rapidly. It was during this process that it was discovered how effective it was to keep sending a tone until the sender is sure that the tone has been correctly received, and only then moving on to send the next tone.

In conclusion, various audio optimizations were made to enhance efficiency. The optimal playtone() duration was set to 0.5 seconds, balancing the need for accurate reception with the avoidance of self-confirmation by the sender. The implementation of distinct tone matrices for the Turtle-bot and PC effectively reduced misinterpretation of signals. Additionally, the shift from a slower serial bitstream method for transmitting coordinates to a faster hexadecimal approach improved the system's responsiveness to rapid position changes, highlighting the importance of continuous tone transmission until correct reception is confirmed.

# 8 Experiment

## 8.1 Range Limitation Assessment

In the pursuit of enhancing turtle robot capabilities, the first experiment focused on testing its hardware limits, particularly the microphone and speaker connected to the robot for interaction with a computer.

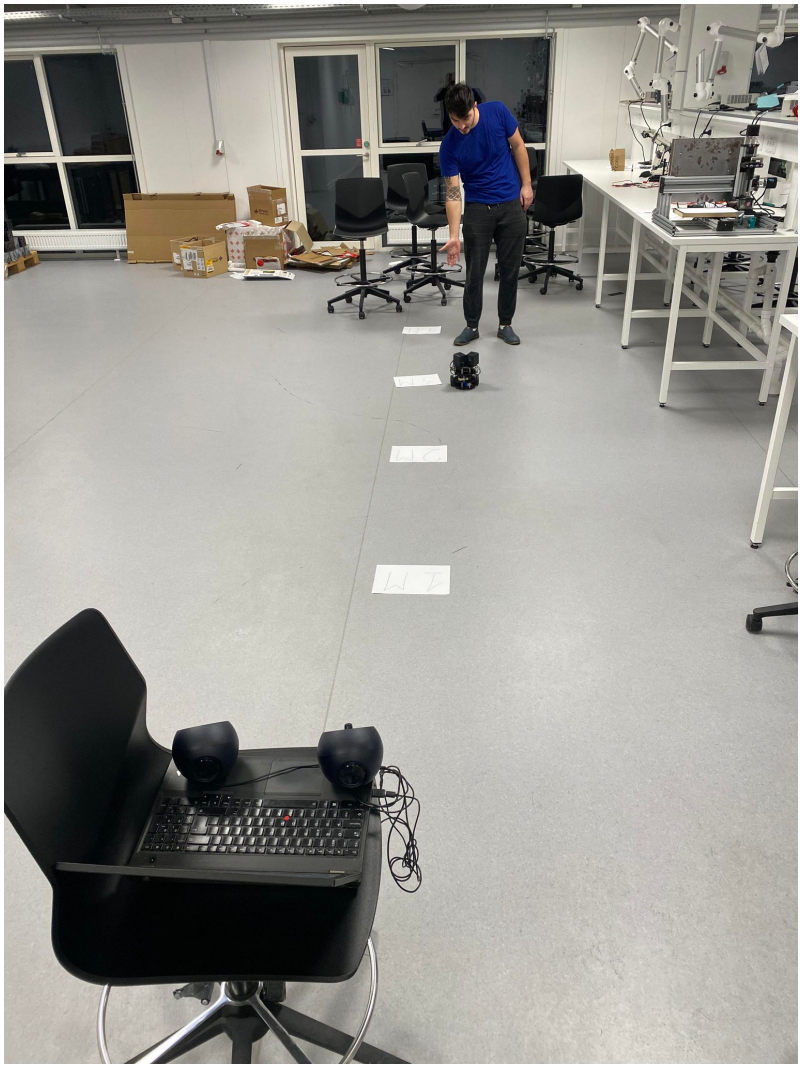

Figure 9: Picture of experiment Range Limitation Assessment

The findings indicated a communication range limit of about 3-4 meters. Precise coordinates could be found as long as the robot stayed within this range. The robot's movement limitations are attributed to the performance of its microphone and speaker. The diagram (fig. 10) below illustrates the effectiveness between the movement and communication. The experiment was done 20 times and the diagram shows how many

times the robot communications were successful. It is shown in the diagram that for the first meter the communication was perfect. The communication remained excellent up to two meters, but at three meters and beyond, a noticeable deterioration can be seen. The robot's limitations are clearly evident. The hardware of the robot is not able to process long distance sounds. This explains why the communication between two computers could reach a higher range than the communication between the robot and the computer.

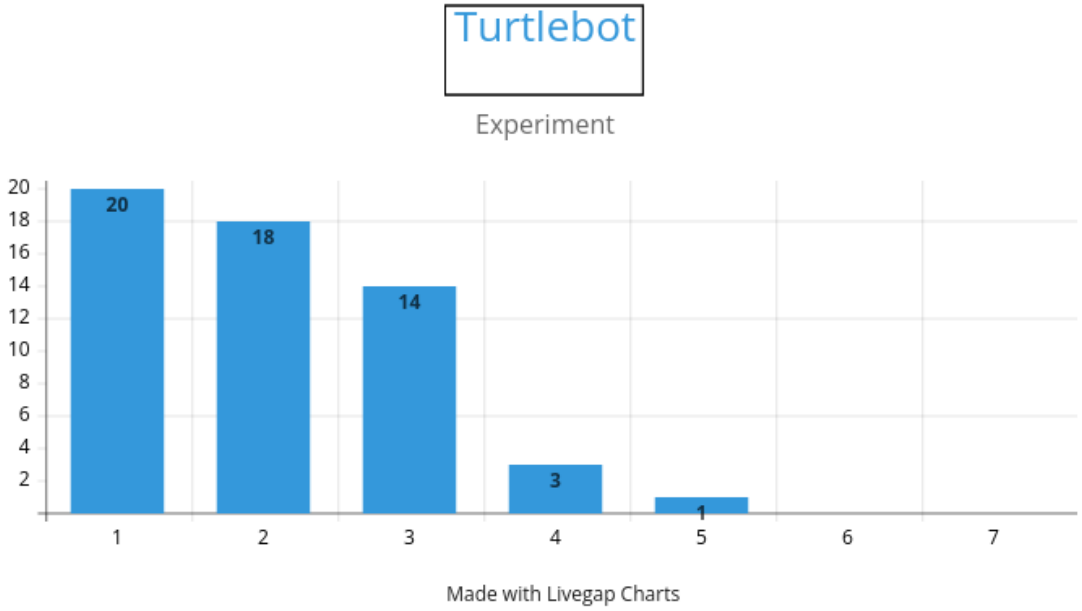


Figure 10: The diagram shows the numbers of success in y-Axis and the length where the robot could receive sounds for each meter in x-axis.

# 9 Discussion

This section addresses the challenges faced, and reflecting on the learnings, and what could be done better. In terms of data processing, alternative methods were considered for handling numerical data, particularly those with decimal points. One approach involved rounding these decimal numbers (doubles) to the nearest integer. This method simplifies the data, potentially making it easier to interpret decimal values.

Another proposed solution was to expand the use of the tones to accommodate decimal values. 10 new modified tones could be created from the existing tones. Each sound being assigned to each decimal value following the comma. This method aims to transmit precise numerical data, thus improving the position accuracy.

## 9.1 Challenges in Tone Reception

While DTMF tones are robust against general noise, it can still be susceptible to interference from specific types of background noise or harmonic distortion, that can result in incorrect tone reception. Even with all the signal processing techniques and matrices used, there were still issues.

## 9.2 Challenges in Robot Movement

While enhancing the robot movement code, few problems occurred. One of the problems were that the robot would not go in a straight line while moving forward. The linear movement of the robot was decreased and solved the problem. Another problem occurred while making the (X,Y) coordinate system. The robot would not meet the exact coordinates, and there was a percentage of coordinates lost due to that fact. The robot could be improved by sending polar coordinates from the robots new origin, that way it could be easier to understand the robots movement and make it more effective.

## 9.3 Battery Level Monitoring

During experimentation and testing, it was found that the robot would quickly run out of battery. A potential solution could be to implement a variable in the code, that checks the battery-level and alerts the user when the charge is low. That would then prevent unexpected interruptions due to battery depletion.

# 10 Conclusion

The goal was to make a mobile tracking robot and have the robot send its current position through DTMF tones. It was quickly realized that DTMF tones are not nearly robust enough, and therefore MTMF tones were implemented. These optimized tones are vastly superior to regular DTMF tones. The unique rounded frequencies and optimized sample rate, drastically improved the audio processing. By aligning the frequencies with the FFT's resolution bins at sample rate 5120 Hz and frame size 512 the frequencies fit precisely within the FFT analysis. The conversion from a number to a hexadecimal value was efficient due to hex being able to represent enough values for the application and doing so with only 2 digits streamlined the transmission process. The protocol's design, with its two different matrices for the sender and receiver as shown in figure 6, reduced the chance of miscommunication. The protocol's focus on signal confirmation, involving multiple confirmation tones as shown in Figure 7, exemplifies the depth of thought put into ensuring data integrity. The echoing of confirmations by the receiver added an additional layer of verification, further protecting the communication against errors.

The robot movement algorithm implements two libraries, MediaPipe and OpenCV, that allows the robot to detect a persons' hand in the webcam frame. The algorithm calculates the distance of the hand relative to the camera. Furthermore, an (X,Y) coordinate system was made, such that the robot could track its absolute position from the start. These coordinates were converted to polar coordinates which were then send to the PC. The movement of the robot was decided by the Twist functions which sent a geometrical messages to the robot; it could either send linear or angular messages. The experiment fig. 10 in Section 8. showed that the audio detection limit as well as the working limit of the robot was 3 to 4 meters from the PC.

# 11 Appendix

This Appendix refers to supplementary files sent together with this document inside a ZIP-file named ”Appendix”. The names of the folders and documents inside the zip file referred to in this document are the following:

- TrackingRobot.py
- Receivercode.py

# 12 Bibliography

[1] 3Blue1Brown. But what is the fourier transform? a visual introduction. https://www.youtube.com/watch?v=spUNpyF58BY&ab_channel=3Blue1Brown, 2018. Youtube.

[2] NumPy Developers. numpy.fft.fft. https://numpy.org/doc/stable/reference/generated/numpy.fft.fft.html, 2023. Documentation.

[3] Murtaza's Workshop - Robotics and AI. Hand distance measurement with normal webcam + game — opencv python. https://www.youtube.com/watch?v=NGQgRH2_kq8&t=244s, 2021. YouTube.