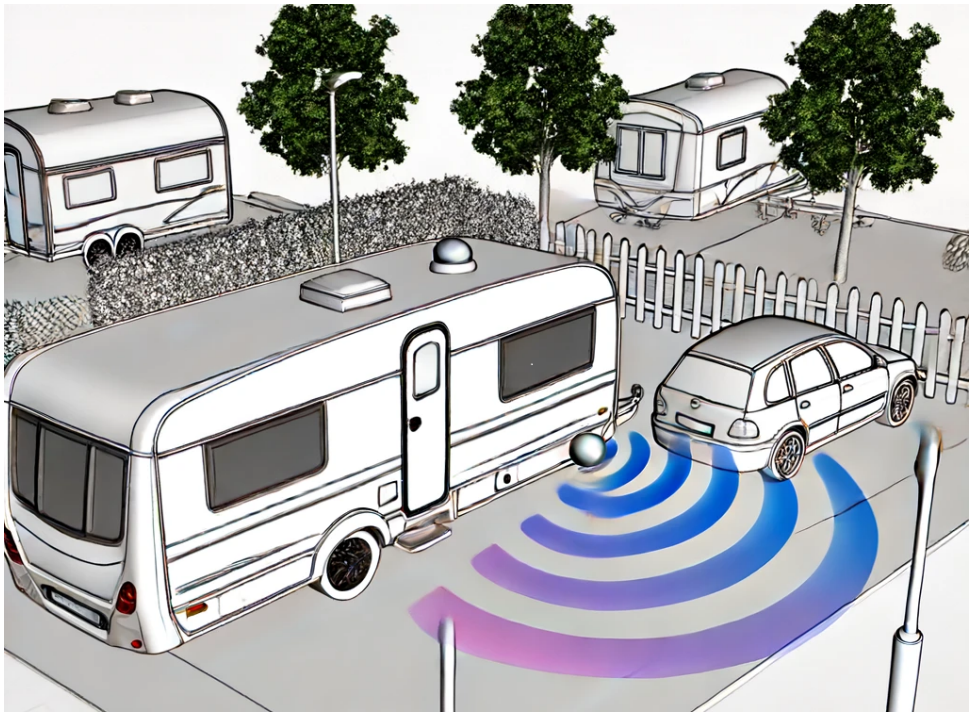

Proximity sensor-based parking aid solution for mobile homes and trailers

Bachelor thesis

Yash Dwarkasing
Batuhan Yildiz

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Project Supervisor: Jeroen Bastemeijer

June 14, 2024

Abstract

This report describes the design and implementation of a sensor-based proximity sensing system implemented to assist the camera-based system. The goal of the sensor-based system is to provide a potential alternative to the camera-based system, and also to compensate for any of the downsides of this system, such as relatively worse performance in low visibility conditions, gauging distances accurately and implementing a collision warning mechanism. This will be done through the use of ultrasonic sensors. The sensor considerations and sensor network will be described in depth. Furthermore, analysis of numerous signal processing techniques will be performed. These techniques will differentiate objects based on their respective echoes, which allows for accurate mapping onto the user interface, once the locations of the objects have been obtained through 3D trilateration. The prototype implementation along with the system integration with the camera-based system will be discussed once all of the previous work is done. Afterwards, the results of the project will be discussed and the work will be concluded with some future recommendations.

Preface and Acknowledgements

Over the course of a few months, a proximity sensor-based system has been designed and implemented to aid parking for mobile homes and trailers. The goal of this system is to provide a potential alternative to camera-based systems and to compensate for their downsides, such as poor performance in low visibility conditions and less accurate distance gauging. This project leverages ultrasonic sensors to overcome these challenges, offering precise distance measurement and collision warning mechanisms. Throughout this project, various aspects have been considered, such as sensor selection, signal processing, and user interface design.

We would like to thank our supervisor, Jeroen Bastemeijer, for their guidance during the course of this project. Furthermore, we want to thank Rick Lenssen and everyone else at E-Trailer for their willingness to receive us at their office every week, and also for providing helpful insight when needed. Lastly, we want to thank our teammates: Stevan Pavlovic, Hashim Karim and Dennis Landman for good teamwork and an enjoyable and productive collaboration.

Contents

Chapter 1

Introduction

1.1 Problem Definition

Camping with caravans has become increasingly popular. In the Netherlands, more than 1 million people are going camping annually using a leisure vehicle such as a camper or caravan. As this is a seasonal activity, the drivers of these large vehicles are not used to maneuvering and parking them. It is harder for the drivers to obtain complete environmental awareness around the leisure vehicle compared to a standard car, which facilitates overlooking and hitting objects, resulting in damage to both the leisure vehicle and the struck object. According to Caravan Guard [1], the parts of the caravan most susceptible to damage are depicted in Figure 1.1.

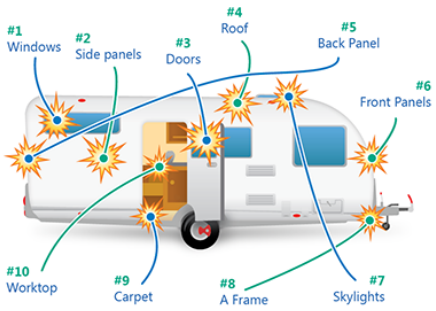

Figure 1.1: Most accident prone parts of a caravan [1]

That same web article also provides the common causes of damage to the specific caravan parts such as:

- **Side panels** (second most common type of damage): Regularly damaged while reversing, parking, or maneuvering, particularly on the driveway.
- **Doors** (third most common type of damage): Often damaged in conjunction with more serious incidents like road traffic accidents.
- **Back panels** (fifth most common type of damage): Typically includes instances where customers accidentally hit something while reversing or when a third party runs into them.
- **Front panels** (sixth most common type of damage): Collisions with structures such as lamp posts, railings, fences, and wheel bins, mostly when driving in tight spaces.

This once more shows the challenge drivers encounter when driving leisure vehicles they are not accustomed to, particularly while navigating tight spaces and executing precise maneuvers. Caravans are also generally much taller than personal vehicles, which complicates the driver's ability to spot high objects. Therefore, the driver needs to be informed about the location of obstacles and alerted to potential collisions in low-speed environments. To address this issue, the client, E-Trailer B.V., has suggested implementing a 360-degree surround view system. This system would provide a bird's eye view of the caravan and incorporate object detection for automatic collision warnings and obstacle mapping in conditions of low visibility. This report will focus on the object detection and mapping components of the system.

1.2 Background

The driver assist parking systems industry is a large segment in the automotive industry. Virtually every new passenger car consists of some type of rear-view camera along with proximity sensors to provide the driver with surrounding objects and visualization [2][3][4]. Proximity detection in vehicle applications is extensively researched due to its use in autonomous driving. Modern vehicles usually contain a fusion of multiple sensors combined with a smart algorithm capable of interpreting this sensor data and constructing a comprehensive map of the environment [5]. These types of sensors range from cameras, radio detection and ranging data (RADAR), light detection and ranging (LiDAR) to ultrasonic and infrared sensors. Interpreting the sensor data and environment mapping is often implemented (in autonomous driving) with deep learning, due to its capability of sensor data integration, feature extraction from raw data and object classification [6]. In [7], a system capable of obstacle detection and depth map generation is developed using stereo-vision cameras with low latency in fast robot applications. [8] shows the advantages and properties of mmWave RADAR and ultrasonic sensors for blind-spot detection in vehicles. [9] mentions an implementation of a full 360 degree surround view ultrasonic-based sensor network, capable of detecting any object within 2 meters of the car. This paper emphasizes on sensor placement and orientation, based on a working principle of no overlap, and individual sensor detection. [10] considers an array of ultrasonic sensors which may achieve a high obstacle detection rate at a decent range. Most proximity detection system rely on the Time-of-Flight method, which calculates the distance using a known speed and a measured time.

1.3 Structure

The second chapter will state the programme of requirements for this project, most formulated by the client. The third chapter will discuss the general design considerations for this project, taking the programme of requirements into account. From there on out, the following chapters will delve in to the design process, systematically iterating from each sub-module of the chosen system. In Chapter 7, an implementation of a prototype will be presented and discussed, conforming to the design considerations and choices. After the prototype construction has been discussed, the system integrated with the camera-based system will be elaborated upon in Chapter 8. The results and interpretation of these results will be presented in Chapter 9. The discussion will provide some additional insight into the overall design process, obtained results and further recommendations or considerations on how to improve the design.

1.4 System subdivision

The chosen system implementation will consist of 3 main parts:

- Sensor network
- Signal processing
- Trilateration algorithm, Wireless communication and Obstacle mapping

Chapter 4 will discuss the sensor network, of which the main function is retrieving signals from the environment. In this chapter, the choice and consideration of sensor type, layout, orientation and position of sensors will be discussed. Chapter 5 will delve in to the signal processing aspect of the the incoming signals from the sensor network, touching upon subjects such as multiple object detection and recognition, noise reduction techniques, performance and power considerations. Chapter 6 will predominantly focus on the design process of object detection, localization along with the design of the user interface. This part also touches upon the wireless transmission design between the signal processing unit and the user interface. Figure 1.2 also shows the general structure of the system.

Figure 1.2: General structure of the system

Chapter 2

Programme of Requirements

This is a Program of Requirements (PoR) for the entire project. The purpose of the PoR is to define and document the minimum specifications that the system must meet. The PoR introduced will be utilized in the following sections for evaluating the design and outcomes throughout this thesis. The requirements are divided into two categories: functional requirements and non-functional requirements. Functional requirements specify what the system should do, whereas non-functional requirements specify how a system should behave. Thus, while the functional requirements focus on the functionality provided by the system, the non-functional requirements focus on the system's characteristics rather than specific functions.

2.1 Requirements for the sensor system

2.1.1 Functional requirements

The functional requirements for the system are:

- **Audible collision warning mechanism**: There should be an audible collision warning mechanism.
- **Object detection and visualization**: All objects within a certain range of the vehicle (including height) are to be detected, localized and visualized.

2.1.2 Non-functional requirements

The non-functional requirements for the system are:

- **Power limit**: The power limit for the sensor system is 1 A at 12 V.
- **Budget limit**: The budget limit for the sensor system is €50 cost price.
- **Calibration and installation time**: The calibration and installation for the end-user should be done within 1-2 hours.
- **Speed constraint**: The entire system should work at 20 km/h.
- **Wireless connection**: There should be no wired connection between the car and caravan.
- **Aesthetics**: The system should be aesthetically pleasing for the end-users.

Chapter 3

Design Overview

Proximity detection systems typically operate by the following principle: Firstly, sensors are used to capture signals from the environment. These are then progressed by a microcontroller, and visualized through a user interface for the end-users. In the sections below, first the various existing implementations will be discussed. Then, the typed of sensors used, the nature of the data collected, the signal processing techniques and the method of visualization will be discussed for each section. Furthermore, each implementation will include an analysis of its pros and cons, from which the suitability for our project will be determined. All of this will then result as the rationale for the final system design.

3.1 Existing implementations

3.1.1 Simple non-overlapping ultrasonic sensor parking system

The most popular parking sensor system in automotive applications today is the ultrasonic parking sensor system, in particular those with the proximity indicator bars. The general working principle of this implementation involves using ultrasonic sensors to emit sound waves, which bounce off objects and return to the sensors. This enables the system to calculate the distance to objects based on the duration of the echo, and it enables the system to provide feedback to the driver through a user interface using the so-called proximity indicator bars. Figure 3.1 shows an example of using three sensors to cover one side of the vehicle. Since the visualization relies on proximity indicator bars, there is no need for sensor overlap, making the configuration efficient with a minimal number of sensors.

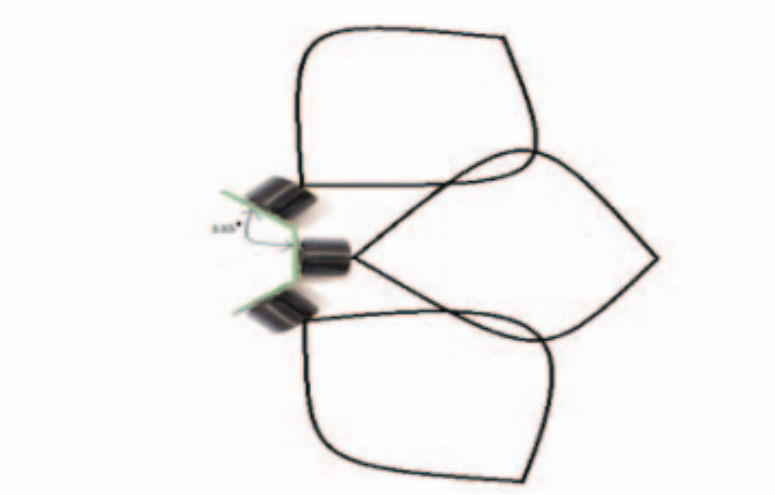

Figure 3.1: Area coverage of three sensors when they are kept 135° apart [11]

This implementation has several advantages. Firstly, it is cost-effective due to low amount of sensors used in such a system. The power consumption would therefore also be low. Moreover, the calibration for the end-user will be easier, as they have to install fewer sensors in comparison to other implementations. Lastly the system

uses a user-friendly interface which provides the driver with an intuitive way to approximate the distance to any nearby objects.

Nevertheless, there are some caveats to this design. First, there is the limited detection capabilities of this system; The implementation is not able to differentiate objects based on heights, and thus it would not be able to detect low-lying or overhanging obstacles as well as other implementations. It is also incapable of estimating the size of the detected objects; it can only detect their presence. Furthermore, it also has reduced accuracy due to the no-overlap-factor. This factor may cause blind spots, which makes it possible to miss smaller or narrowly positioned objects. The proximity indicator bars provide basic feedback, but they lack detailed information about the nature of the detected object. There is also the potential for false alarms, as simple systems such as this are relatively more prone to false positives, such as detecting objects that are not actual threats. Last but not least, there is also the problem of scalability: Expanding upon the system to account for more complex scenarios would require a thorough redesign and additional sensors.

3.1.2 Occupancy grid mapping with sensor fusion

The second existing implementation consists of occupancy grid mapping. In essence, occupancy grid mapping is a method used to create a representation of the complete environment based on sensor data. It divides the environment into a 2D/3D grid of cells. Each of these cells then has a probability value, which indicates the possibility of that area being occupied by an obstacle or being empty. Firstly, the initial state of the grid is that all cells are considered unknown or unoccupied. Then, as sensor data becomes available, the occupancy status of these cells are updated based on the sensor data. Cells which are observed to be clear of obstacles are marked as free, whereas others will be marked as occupied. This process will then be repeated based on the measurements from the sensors. Instead of using a single or two types of sensors, this implementation uses multiple. This process is also known as sensor fusion.

Sensor fusion algorithms merge data from various sensors, like ultrasonic sensors, to enhance object detection accuracy by overcoming individual sensor limitations such as blind spots or measurement errors. Techniques range from basic averaging to advanced methods like Kalman filters or particle filters [12], [13]. Fusion exists at different levels, which ultimately depends on the type of sensors in the detection system. For relatively simple sensor systems, low level fusion (combining raw data from multiple sensor sources to provide a more accurate representation of the sensed area) is adequate, while more complex systems involving RADAR and camera systems require high level fusion. This type of fusion is capable of accurately mapping obstacles to a 3D Map by combining different types of visual and measured data, as illustrated by Figure 3.2. For a two-dimensional system, it is possible to create a grid of obstacles using sensor low level sensor fusion and triangulation algorithms to achieve relatively accurate coordinates of obstacles with respect to the vehicle, and visualize an obstacle grid [14].

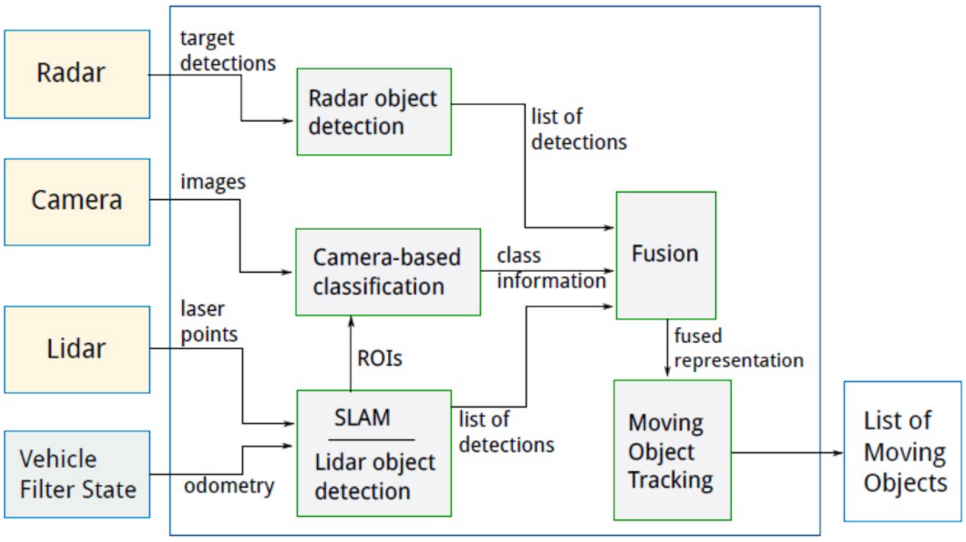

Figure 3.2: High-level sensor fusion [13]

In comparison to the previous design, this one has some other benefits. Firstly, the sensor fusion allows for more accurate detection by compensating for the limitations of the individual sensors. Moreover, the system is also more robust against noise and the environment due to it having combined data from multiple sensors.

Occupancy grid mapping also allows for a comprehensive representation of the environment, which ends up providing the user with more detailed information on the user interface. Lastly, this implementation is very adaptive, as the sensor fusion techniques can be applied to various sensor types.

However, there are also drawbacks to consider. The design becomes significantly more complex with the implementation of sensor fusion algorithms, demanding additional computational resources compared to simpler approaches. Additionally, integrating data from various sensor types and ensuring their compatibility presents considerable challenges. Moreover, the use of multiple sensors along with sensor fusion algorithms increases both the cost and power consumption of the system. Lastly, the calibration process becomes more challenging for end-users, and maintenance requirements are higher compared to other implementations.

3.1.3 Stereo-vision based proximity detection

Another interesting implementation would be the stereo-vision based system using stereo cameras. These cameras, equipped with two or more image sensors, mimic human binocular vision, granting them the ability to perceive depth. This method typically uses triangulation and epipolar geometry to calculate distances, requiring highly synchronized frames for subsequent processing. Epipolar geometry aids in rectifying aberrations within the stereo rig and streamlines the search for corresponding features to one dimension [15]. The former refers to understanding the geometric relationship between the corresponding points in stereo images helps correct any distortions or discrepancies that may arise within such a stereo camera system. This correction ensures that the images captures by the cameras are properly aligned, from which accurate depth perception and object detection is possible. The latter refers to the simplification of the process of finding matching points between the images captured by each camera. By leveraging epipolar geometry, the search for corresponding features is confined to one dimension along the epipolar lines, significantly reducing the computational complexity of stereo matching algorithms. This streamlined approach makes the matching process more efficient and facilitates faster and more accurate depth estimation and object recognition.

This method, through the use of triangulation and epipolar geometry, allows for the computation of the distances to any surrounding objects in the case of both parking and low-speed driving scenarios. Highly synchronized frames ensure accurate depth estimation, whereas epipolar geometry aids in rectifying any distortions within the stereo camera system, which ensures proper alignment of images for accurate depth perception and object detection.

Similarly to the previous implementation, stereo-vision systems offer accurate depth perception, which allows for precise detection of any nearby object in the vicinity of the vehicle. Furthermore, the system provides a detailed representation of the environment, which enables the drivers to obtain detailed information about their surroundings. Furthermore, like the sensor fusion techniques, stereo-vision systems can be adapted to the various sensor types, which makes them versatile for different applications and environments.

As for the disadvantages, they are similar to the previous implementation: The system implementation is complex, as it requires additional computational resources and advanced algorithms in comparison to simpler sensor-based approaches. Moreover, the integration of the multiple image sensors and ensuring their synchronization is also challenging. The cost and power consumption is also relatively higher due to the need of processing the feed. There is also need for alignment, thus calibration is generally more difficult in comparison to the first implementation described. Lastly, this system will struggle in low-visibility environments, such as night, heavy mist or fog conditions.

3.1.4 Multilateration-based object localization

The last implementation uses three or more sensors to precisely determine the position of every object in the vicinity with respect to the sensors through the use of multilateration [16], [17]. As the positions of all the sensors are known, the objects can be localized and also visualized onto a user interface. Multilateration-based object localization can be done with both ultrasonic sensors and other sensor combinations. In the case of ultrasonic sensors, a degree of overlap is needed in order to precisely determine the location of the object.

The main advantages of this in comparison to the simple system is that the precise location of every object in the vicinity would be known. This would leave the 'guess' work that the proximity indicator bars show. This precision allows for accurate spatial mapping and localization, providing users with detailed information about the objects' positions relative to the sensors and the vehicle. By precisely knowing the location of these objects, the users gain an enhanced environmental awareness, enabling them to make informed decisions and maneuver the vehicle more efficiently in congested or tight spaces in comparison to simpler systems. This implementation also has less false alarms due to the precise localization. Lastly, as multilateration-based object localization can

be implemented using various sensor combinations, the implementation has a lot of flexibiliy in sensor selection based on factors such as cost, accuracy and the environmental conditions.

The implementation, similarly to the previous two is more complex in comparison to the simple implementation. Furthermore, ensuring accurate sensor calibration and maintaining their alignment can be challenging and requires regular maintenance to ensure correct placement. There are also environmental conditions which affect the accuracy of the localization, such as interference, reflections or variations in sensor performance. Lastly, depending on what type of sensor configuration is used for multilateration-based object localization, the cost and power consumption can be either relatively low or high in comparison to other implementations.

3.2 System choice and outline

When selecting the final system configuration from the available implementations, it is essential to refer back to the PoR from Chapter 2. The need for detailed object detection and visualization led to the exclusion of the first implementation, which only provides approximate object locations in comparison to the exact localisation which was needed. Furthermore, the power and budget constraints mentioned in the PoR, along with the calibration and sensor overlap requirements, as discussed in the previous section, rendered the subsequent two implementations excluded.

The final chosen implementation utilizes multilateration-based object localization. Depending on sensor configuration, this method offers a relatively straightforward implementation and is cost-effective in terms of power and budget. This approach also allows for complete detection and visualization of all objects and meets the sensor overlap requirements essential for accurate trilateration. Thus, this implementation aligns closely with the PoR specifications established in Chapter 2 and was selected as the foundational system for this project.

The system architecture is as follows: Firstly, data is acquired via a sensor network. This raw data is then subjected to signal processing to ensure it is appropriately conditioned and ready to be put to use. After signal processing, the data is transmitted wirelessly to the user interface, where the detected objects are visualized on the vehicle's display. An overview of the entire system can be found in Figure 3.3. The specifics of each part of this system will be elaborated on further in this report.

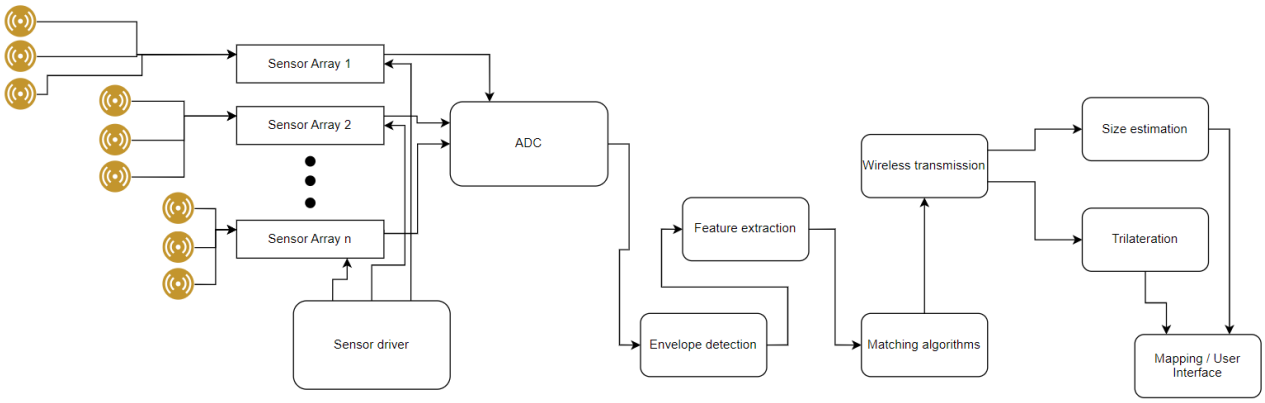

Figure 3.3: Schematic overview of the full system

Chapter 4

Sensor Network

4.1 Requirements for the sensor network

The goal of the sensor network is to detect any object within a certain range of the vicinity of the vehicle. This has to be done with a low sensor count due to the budget and power requirements from the PoR. Because of this, maximum sensor coverage has to be ensured without sacrificing too much of the other aspects of these sensors (such as the amount of deadzone (sensor blindspots) around the vehicle). The output of this module will be sent to the signal processing module, where the interpretation of the data from the sensor network will be done. As for the limitations of the module, not only do the budget and power limit need to be kept in mind, the aesthetics and ease of installation of the sensor network should also be kept into account.

4.2 Sensor type considerations

The first objective in obtaining information is by retrieving signals from the nearby environment, which is carried out by the proximity sensors, which come in 4 main types (for vehicles) [18]:

- LiDAR: Laser reflection detection (wavelengths and delay) to create a 3D image
- RADAR: electromagnetic (radio) wave detection, transmits EM waves and measures the properties of the reflected wave
- Ultrasonic sensor: sound (ultrasound) wave detection, transmits ultrasound pulses and measures delay of echo (Time-of-Flight principle)
- Infrared sensor: light (infrared) detection: transmits infrared pulses and measures reflected intensity

LiDAR [19] offers proper long range detection and good mid-range detection [18], but is relatively expensive compared to other sensor types (for vehicles) [20]. RADAR offers good proximity detection in all environmental circumstances, but has inferior very short range detection in comparison to infrared (IR) or Ultrasonic (US) sensors [18]. The RADAR antennas should also be configured according to the specific application, which could benefit from ODS or ISK layout [21]. Ultrasonic sensors provide very accurate short range sensing (around 5cm), but have a longer delay (due to the intrinsic low speed of sound in air), are at risk of malicious attacks [22] and could be very sensitive to different materials, depending on their reflection and absorption properties [23]. The following table provides some insight into the different sensor types and properties:

Table 4.1: Comparison between functionalities of major sensor technologies [18], [24], [25]

Criteria	**RADAR**	**LiDAR**	**Ultrasound**	**Infrared**
Very short-range detection (up to 1 m)	Ok (2)	Poor (1)	Very good (5)	Good (4)
Short-range detection (1-30 m)	Very good (5)	Very good (5)	Poor (1)	Medium (3)
Long-range detection (30-100 m)	Very good (2.5)	Medium (1.5)	No (0)	No (0)
Angular resolution	Good (4)	Very good (5)	Poor (1)	Poor (1)
Velocity measurement	Yes (5)	No (0)	No (0)	No (0)
Poor weather conditions	Very good (5)	Poor (1)	Good (4)	Poor (1)
Night	Very Good (5)	Very Good (5)	Very Good (5)	Very good (5)
Cost	Medium (12)	High (4)	Low (20)	Low (20)
Power consumption	Medium (12)	High (4)	Low (20)	Low (20)
Total Points	52.5	26.5	56	54

The table employs a point-based evaluation system to determine the most suitable sensor type from the available options. The point assignments are as follows: Very Good (5 points), Good (4 points), Medium (3 points), Ok (2 points), Poor (1 point), Yes (5 points), and No (0 points). Additionally, a weighted scoring system is applied, where the cost and power consumption categories are assigned a weight of 4x to reflect their importance as specified in the PoR. Moreover, the long-range detection category is assigned a weight of 0.5x due to its relatively lower significance for the intended application. Given these considerations, the sensor types are evaluated and ranked accordingly. From Table 4.1 it can therefore be concluded that for close-proximity sensing the best choice of sensors would be either ultrasonic or infrared. However, most automotive applications involve ultrasonic sensors [26] along with LiDAR or RADAR (or both). RADAR is especially suitable for applications with harsh environmental conditions, and is also useful for detecting objects in motion. Hence, obstacle detection for a moving vehicle is best implemented with the use of a RADAR system. Furthermore, the low cost and power usage of ultrasonic sensors allow for an easy integration of an array of these sensors, potentially offering the possibility of a 360-degree lateral obstacle detection mechanism around the vehicle [27].

Infrared seems like a good choice due to its low cost and power consumption and also similar results in the criteria mentioned in Table 4.1. Still, it has some downsides. While infrared sensors can be used for obstacle detection, they require knowledge of the properties of the objects' surface due to the non-linear characteristics and their dependence on reflectance properties. Every surface material reflect and absorbs the IR energy from the infrared sensors differently, and thus a target material identification method would be needed for an accurate distance measurement.

As for RADAR and LiDAR, while they show better results with regards to detection due to their respective properties, their power consumption is higher when compared to ultrasonic (and infrared) sensors. LiDAR also performs relatively worse in poor weather conditions. Furthermore, due to their high complexity, the systems implemented with them are generally more expensive in comparison to systems implemented with ultrasonic sensors [28]. When comparing the different sensor technologies, a decision was made regarding to the types of sensor used. Given that there is a constraint on the budget, the ultrasonic sensors were decided to be used as the sole sensors for this project. These sensors also are able to adhere to multiple of the requirements in the PoR, such as the power limit and the audible collision warning mechanism.

4.3 Ultrasonic sensor model

In essence, ultrasonic sensors detect objects by transmitting and receiving an ultrasonic echo. It works according to the Time-of-Flight principle [28]. A short burst of ultrasonic pulses is transmitted by a piezoelectric transducer, after which said transducer goes into "listening mode". The ultrasonic pulses are reflected by objects within the field of view of the sensor, and this is what the sensor captures. To clarify, a piezoelectric transducer is a transducer which can convert an electrical signal into mechanical vibrations and vice versa. Since it is able to do both of the conversions, it would be able to operate as both the transmitter and the receiver of its ultrasonic echoes. It is important to note that the piezoelectric crystals operate around a specific resonant frequency, which is also the only frequency these sensors are sensitive to. Most models on the market have this resonant peak at 40 kHz and have a current draw between 15 and 30 mA at an operating voltage of 5V [29]. Since the speed of sound is a known variable, and the time difference between the time when the echo was

sent out to the time that the sensor picked up the reflected ultrasonic echo is known, the distance between the sensor and the object can be computed. Thus, using the speed of sound and the captured round-trip time, the distance can be computed by using the following equation:

$$d = \frac{t_{roundtrip} \cdot V_{sound}}{2} \tag{4.1}$$

With this, the distance is computed. It has to be mentioned however that the speed of sound is dependent on the temperature, the humidity of the environment and (potentially) the air pressure. The potential need for speed calibration to handle any differences in the speed of sound will be discussed later on in this chapter.

4.4 Sensor placement and orientation

The sensor layout has to satisfy the system requirements. Hence, one of the key points to take in to consideration is to aim for sensor overlap, as this is required for multi-object trilateration. The sensor placement and orientation needs to adhere to two requirements:

- The coverage area within a certain range of the vehicle has to be overlapped by at least 3 sensors
- The deadzones should be less than 0.3 m

The sensor placement is highly dependent on the beam angle and the range of detection. Furthermore, there are some other considerations taken in to account when specifying the sensor coverage area:

- Objects lower than the clearance of the vehicle are irrelevant
- Objects at a greater height than the vehicle are irrelevant
- The front side of the vehicle has be to exempt from object detection (excluding the area at the height of the caravan and the corners)

These considerations decrease the coverage area, which allows for the use of less sensors. Figure 4.1 provides a visual approximation of the coverage requirement for the sensor network, as seen from the top, side and back.

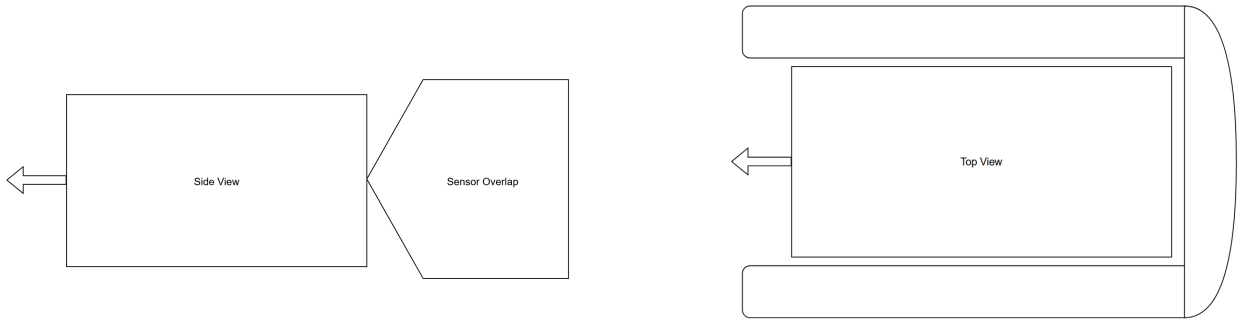

Figure 4.1: Sensor coverage requirement

The layout will also be determined by the dimensions of the leisure vehicles, as caravans have a large variance in length. The physical vehicle dimension determine the required area of coverage. Due to the ambiguous sizes in caravans, the best approach would be to design a modular sensor layout, which is, taking in to account the requirement of a quick installation time, easy to adjust according to specific caravan models.

Furthermore, the spacing of the sensors depends on the detection mechanism, especially the echo-matching, as is further discussed in Chapter 5. Considering the echo-matching principles, with emphasis on the distance matching, the sensors need to be placed accordingly in order to minimize the ambiguity which arises when one measured object is relatively closer to a sensor compared to the spacing between the sensors. For this implementation, the accuracy of the echo-matching algorithm depends largely on the spacing between the sensors. However, when sensors are too close to each other, the triangulation algorithm will suffer the consequences, due a larger set of possible estimations.

The deadzones relate to the height of installation (relative to the clearance of the vehicle), orientation and beam angle in the following way:

$$d = (H - D_2) \cdot \tan(\phi_2 - \frac{\beta}{2}) \tag{4.2}$$

The deadzone can be reduced by either decreasing the height relative to the bounds of the vehicle, or minimizing the tan term, of which the angle represents the field of the view of the sensor, accounted for its orientation. One design consideration would be to focus on traditional caravans (excluding teardrop and fifth-wheel caravans). This allows for treating the height dimensions of the caravans as generally the same, as the typical variance in height for these caravans is between 2.5 meters and 2.7 meters. It is therefore possible to design a single height-adjusted solution for these caravans. As the importance of 3 sensor overlap is discussed, a sensor triangle array is proposed, of which two sensors reside on the same level, and the third at a greater height. Looking at this system in 2D from the side, the layout model is illustrated in Figure 4.2.

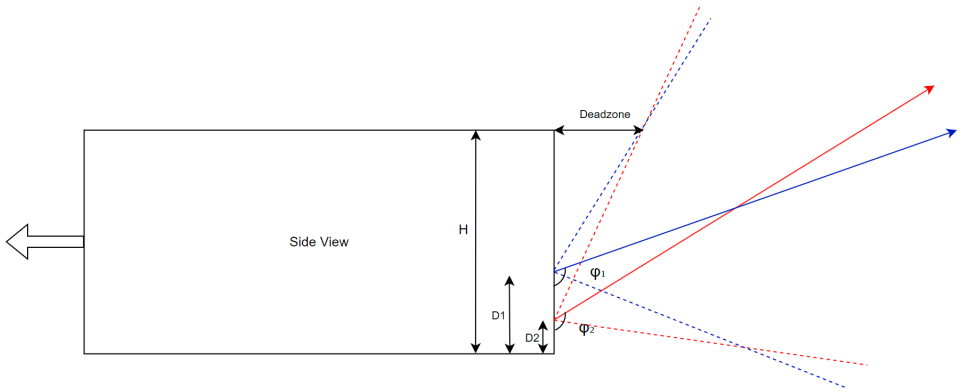

Figure 4.2: Schematic overview of the side view

Furthermore, one of the key methods implemented in the signal processing is the distance matching algorithm. This algorithm assigns echoes with similar distances to each other. Hence, the sensors need to be sufficiently close to each other for this to work. The distance discrepancy between sensors is illustrated in Figure 4.3 and is related to the spacing between sensors in Equation 4.3, assuming the maximum deviation.

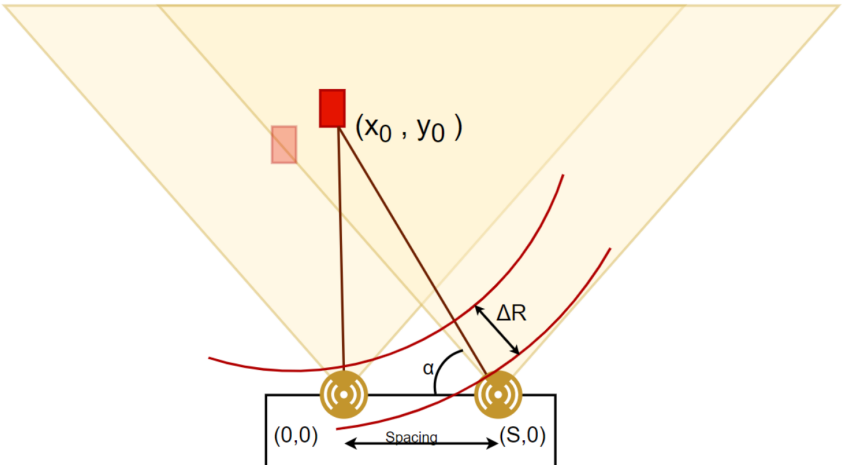

Figure 4.3: The effect of inter-sensor spacing on the range deviation

$$\Delta R = \sqrt{x_0^2(1 + \tan^2(\alpha)) - 2S \cdot x_0 + S^2} - x_0\sqrt{1 + \tan^2(\alpha)} \tag{4.3}$$

The angle α depends on the orientation and directivity of the sensor. In the case of this deviation calculation, the angle corresponds to a vertical orientation and a beam angle of 65°, resulting in a value of 57.5° for α. For

the maximum deviation, the location of a possible object along the edge of the field of view of the sensor closest to the other sensor is chosen (displayed as the low opacity object within Figure 4.3). Figure 4.4 illustrates how the absolute deviation relates to the spacing between sensors, for hypothetical objects placed at different locations, assuming $\alpha = 57.5°$. From this graph, the error is for most distances within a relatively close margin between 0.3 and 0.7 meters. Hence, the sensors should be installed within these margins in a single array.

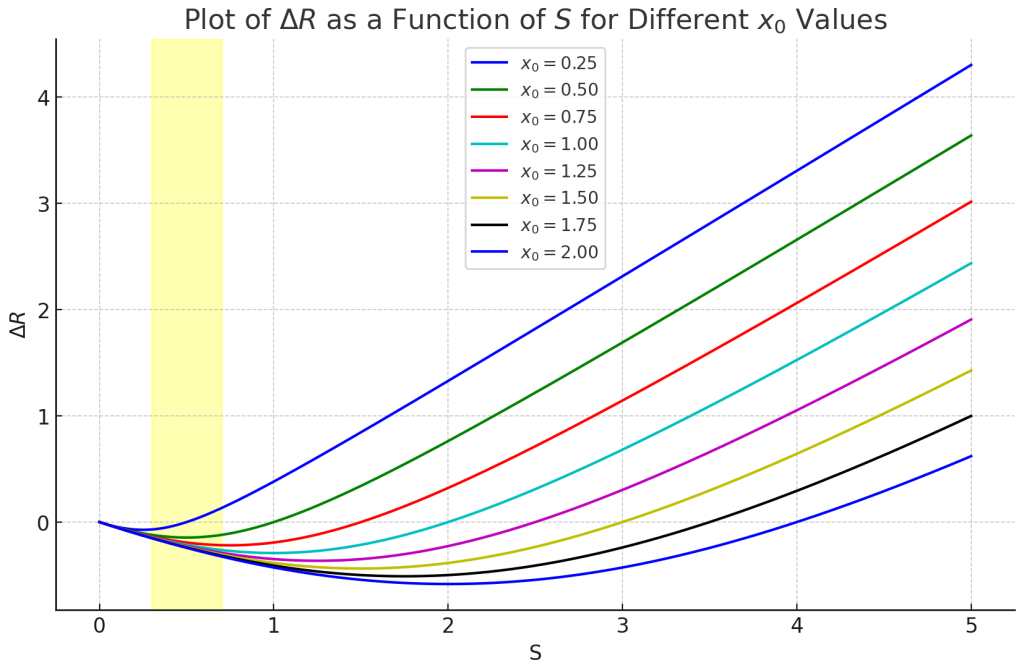

Figure 4.4: Range deviation as a function of the inter-sensor spacing

The coverage area each of sensor array depends on the range (R) of the individual sensors, the maximum spacing between the sensors (S), and most importantly, the beam angle (β). The following equations (derived from the schematic provided in Figure A.1) provide some insight in to the relation between the deadzone and width of the coverage area and these variables:

$$\textbf{Deadzone} = R \cdot \sin\left(180° - \cos^{-1}\left(\frac{S}{R}\right) - \beta\right) \tag{4.4}$$

$$\textbf{Width} = S + R \cdot \cos\left(180° - \cos^{-1}\left(\frac{S}{R}\right) - \beta\right) \tag{4.5}$$

The first equation relates how much the deadzone will be when the farthest sensor is angled toward the first sensor. This deadzone becomes smaller as the beam angle increases, and is virtually zero when the beam angle becomes larger than 90 degrees. This is schematically illustrated in Figure A.3. From the plot of the second equation (Figure A.4), it is seen that the conceptual width of the coverage area does not surpass the range of the ultrasonic sensor, also in combination with the beam angle of 90 degrees, which is expected. Hence, in this implementation, the range of the ultrasonic sensor is used to sense partly alongside the vehicle to obtain the optimal coverage area. Assuming the circumference of a caravan as looked at from a top down view to be 17 m (excluding the front, assuming 7 m in length, 3 m in width, and a sensor range of 4 m), one would only need 5 sensor arrays, translating to about 15 sensors, to cover the circumference of the leisure vehicle.

4.4.1 Sensor multiplexing

Due to the Time-of-Flight principle of the ultrasonic sensor, these can be quite susceptible to cross-talk, or interference between sensors. In order to mitigate the effect of cross-talk, neighboring sensors are sequentially activated, ensuring that only one sensor per array is active at the same time. For the chosen design, where one sensor array consists of 3 sensors, these sensors can be activated sequentially from each other. However, this will impact the overall refresh rate of the system, as each sensor operates on average at a third of their sensing

speed. Overlapping sensors from different arrays will also be operating out-of-phase, in order to mitigate the effect of cross-talk. Figure 4.5 schematically demonstrates how this multiplexing system iterates between the sensors.

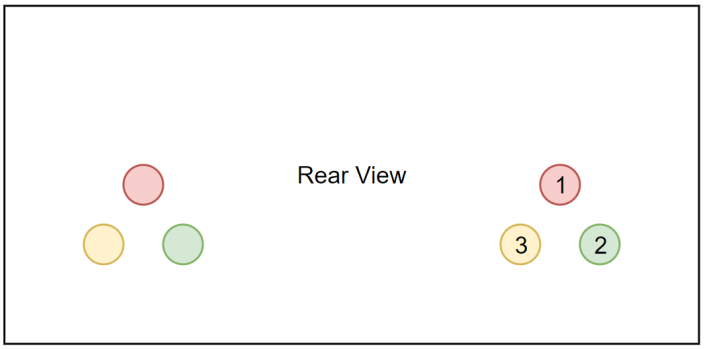

Figure 4.5: Sensor multiplexing

4.5 Calibration

4.5.1 Installation

As mentioned before, leisure vehicles come in various shapes and sizes. It is therefore important to emphasize that there is no fully optimized out-of-the-box system for each caravan. The sensor arrays need to be calibrated to function properly on the leisure vehicles. The variables related to this calibration are:

- Length
- Height
- Width
- Mounting surface properties

The variance in length should be accounted for by creating modular sensor arrays, applicable to specific units of length. The smaller variance in height of the caravans allows for a single height-adjusted design that is capable of covering the entire height of various caravans, considering the maximum height detection requirement equals the upper bound of the the variance in standard caravans. Given the variation of width of standard travel caravans (2-2.55 m), the sensor network should be adjusted accordingly. For the rear side of the caravan, a central sensor array module should be able to cover this full range, and the stitching of the sensor arrays on the side will depend on the degree of overlap and the distance between the sensor modules. All sensor modules will be labeled and assigned to a location within the full layout, which eliminates the need to calibrate the mutual locations of the sensor locations. The proposed method for the installation and calibration regarding the width requirements are:

- The user measures the width and length of the caravan and inputs this data in the user interface.
- The user installs the rear sensor module in the middle of the caravan. The height of installation should be measured and fed to the User interface.
- The side sensor modules are placed starting from the edge, and installed at predetermined distances from each other. This predetermined distance is realized using sensor strips. The height of this installation should be measured and fed to the user interface.

The last factor which needs to be accounted for when designing the sensor mounts for the leisure vehicles is especially related to the mounting surface. Especially the rear bumper has quite some variation in overall shape, and possible mounting surfaces. These are usually quite irregular, or at an angle different than the ideal straight vertical plane. The proposed solution to this problem is to offer modular angle sensor mounting strips, which are able to be oriented towards the "ideal" sensor orientation. This sensor orientation calibration need not be perfect if the beam angle of the sensors is wide enough, in order to provide sufficient overlap.

4.5.2 Speed of sound calibration

The speed of sound, the principle on which ultrasonic sensors perform distance calculations, affects the accuracy of the system. This section will explore whether some calibration is required in order to make up for the discrepancy caused by the speed of sound. First, the impact of these environmental factors need to be recorded in order to justify the creation of the speed of sound calibration for this system. The effect of temperature on the speed of sound is shown in Equation 4.6:

$$v = 331.3 + 0.606 \cdot T \tag{4.6}$$

Where T represents the temperature in degrees Celsius. Assuming a standard room temperature of 20 $°C$, the speed of sound is 343.4 m/s. A deviation of about 30 degrees Celsius relates to a change of 18.2 m/s, which is an error term of about 5%. The humidity of air affects the speed of sound at most by 0.5 m/s, or at an error term of 0.15%, as this is the observed difference between 0 % and 100 % ambient air humidity. The air pressure, given a constant temperature, does not impact the speed of sound directly. Hence, the total error term for environmental factors regarding the speed of sound is approximately 5.15%. This error term is for the purposes of object localization around leisure vehicles regarded as insignificant, as a few centimeters of discrepancy will not negatively affect the overall working principle of the system. Hence, there will be no speed calibration module implemented.

4.6 Design considerations regarding aesthetics

As stated in the PoR, the sensor network should be aesthetically in line with the vehicle on which it is to be installed. In order to compose the design which fits this requirement, there first needs to be some understanding as to what exactly aesthetically pleasing means in this context, as there is no fixed quantification for this. This section aims to quantify the aesthetic requirements, and impose limitations as to what is possible and what not. Some general requirements are:

- There should be little contrast between the sensor strips and the color of the vehicle.
- The footprint of the sensor network housing should not include empty spaces, each covered element should contain a functional part.
- The sensor network should not be instantly visible upon a glance from 20 meters away of the vehicle.

The proposed solution to fit these requirements is to create low-profile transducer mounting strips, with a thin (< 5 mm) cable tray running across the entire length of the strip, tasked with communication with the driving circuit and ADC. The width of the transducer mounting strip will not exceed 80 mm, including the cable tray. Two levels of these mounting strips are required to be installed, due to the requirement of the triangular sensor array structure. With this, the end-result should be aesthetically pleasing for the users, and therefore the requirement is covered.

Chapter 5

Signal Processing

5.1 Requirements and limitations

The signal processing module retrieves the raw sensor data from the sensor network, and is responsible for interpretation. The input for this system is the acoustic analog signal provided by the ultrasonic sensor arrays. The output yields a list of objects (within the field of view) complete with their respective distances to each sensor with a known location, as the final module (trilateration and user interface) is able to use this data to create a comprehensive map of obstacles around the vehicle. The limitations to this part are the budget and power restrictions as this micro controller will reside inside the caravan, which reduces the processing power and speed.

5.2 Object detection method

5.2.1 Envelope detection

In order to accomplish multiple object detection with single ultrasonic sensors, a single sensor should yield a full list of possible range measurements to objects within its field of view (not accounting for obstructed objects). Furthermore, as will be discussed later in this chapter, the gathered echoes must be able to be analysed, in terms of their amplitude, shape and the width of the echo (for size estimation). The signal envelope provides this required information, hence why an envelope detection method has been chosen for the design.

5.2.2 A review on the distinctiveness in envelope shape

According to [30], different materials inhibit unique ultrasound reflection properties. The testing in this paper has been done in the 0.5 to 1 MHz band, but this is beyond that of the sampling capability of the microcontrollers used in this project. [31] discusses and illustrates how the ultrasonic reflection properties are affected by the material type, which will be the key premise when applying the echo matching algorithm.

5.2.3 Threshold detection and obstacle distinction

As per the working principle of the ultrasonic transducer, a pulse is sent out at t = 0. When the sound wave reflects from an object back to the sensor, the transducer will start to oscillate at the instance of arrival. This oscillation is measured, and as soon as the amplitude crosses a threshold, one can infer that an obstacle resides at a distance corresponding with the time delay of the returned echo. There are two functions associated with this threshold detection method. The first is the indication of the presence of an object and the second is the making the system less sensitive to noise and random fluctuations. The design considerations related to setting this threshold are:

- Setting the threshold low: The likelihood of false positives is increased, because the system is more sensitive to random fluctuations and noise.
- Setting the threshold high: Softer objects with a lower reflection coefficient might not trigger the system due to the amplitude of reception being lower, increasing the likelihood of false negatives.

The value of this threshold has been determined experimentally, taking into account the sampling range of the

ADC of the SoC, the average noise level, along with reflection measurements to get an indication about the true object detection level at multiple distances. In order to distinguish between multiple objects, there needs to be an minimum delay between one echo and another. This acts as a buffer for small fluctuations, reducing the likelihood of incorrectly classifying one object as multiple. Increasing this delay threshold will however reduce the resolution of the object detection system, as a larger discrepancy of the range between objects will be ignored. This will also obscure the size estimation of the system, as the width of the envelope is related to the estimated object size.

5.3 Object detection matching between sensors

The goal of the distance measurements from each sensor is to localize objects using a trilateration algorithm. However, in the context of multiple distance measurements per sensor each corresponding to a single object, the echoes measurements across sensors need to be matched in order to prevent false positives and misinformation regarding object locations. Echo matching distinguishes objects and assigns sensor distance measurements to each detected object. There are multiple methods to use the echo information from each sensor in order to match the distances from the objects together. These include:

- Distance matching
- RSS (Received Signal Strength) matching
- Echo Cross-correlation
- Matching algorithms using multiple criteria

The working principle of the sensor network provides the following information about the received signal:

- Delay to the start of echoes
- Signal strength
- The shape of the signal

5.3.1 Distance matching

This method is essentially measuring and matching the delays to echoes using a threshold in order to establish the start, middle and end of a echo corresponding to an object, across multiple sensors. Distance measurements of multiple sensors within a certain tolerance of each other are classified as the same object. However, there are multiple drawbacks to using this matching algorithm on its own:

- This model assumes all sensors to see the same amount of objects.
- When objects are significantly closer to one sensor, the distance measurements to that object from the other sensors may not be classified as the same object.
- The sensors are to be placed relatively close to each other in order to reduce ambiguity when distinguishing between multiple objects.
- This model can not distinguish between different objects at approximately the same distance of the sensors, due to sole reliance on threshold detection.

5.3.2 RSS matching

Another method of matching echoes to each other is to analyze the signal strength from the reflections and matching echoes with similar signal strength, but still applying threshold detection to obtain the delay of the echo. This method exploits the different acoustic impedances of different materials, which could be useful in distinguishing multiple objects (from different materials from each other). The known attenuation factor of air can be implemented to correct for the attenuation over a distance in free space, in order to normalize measured RSS. This work on acoustics [32] discusses the signal strength properties of ultrasound, and concludes alongside [33] that the downsides in the case of ultrasonic sensors are:

- This models assumes objects in the vicinity to have different acoustic reflection properties

- Signal strength is more susceptible to be affected by noise and other disturbances, due to the intrinsic susceptibility of ultrasonic sensors to environmental factors
- RSS-based feature extraction is better suited to RF applications, not great for ultrasound applications

5.3.3 Cross-correlation

Cross-correlation is a method to estimate the degree to which two signals are correlated [34]. This essentially calculates the delay for which a signal is to be found in another according to Equation 5.1. For this method, a reference sensor is chosen, from which its echoes are detected and distinguished. These echoes are isolated, and each is matched to the other sensor input streams, which yields a delay as to where these echoes are most likely to be found. The cross-correlation technique is able to check for the most fitting similar shape in a signal compared to the reference. The premise behind this method is that different objects have different acoustic reflection properties, which also results in unique envelopes [30]. This allows (when echoes are treated as separate objects) for placing the sensors at greater distances from each other, as the ambiguity in distance (compared to distance matching) is compensated by the shape of the signal analysis.

$$(x \star y)[k] = \sum_{n=-\infty}^{\infty} x[n] \cdot y[n+k] \tag{5.1}$$

The cross-correlation algorithm correlates a reference signal to another sensor data stream and peaks at the instance where the reference signal matches the a similar instance of itself within the sensor data stream. The intensity of the peak (at a certain time delay) determines the degree to which the datastream recognizes the echo provided by the original sensor, and could be useful when selecting the correct peak in the case where there are multiple objects present. In order to accurately match the shape of echoes to each other, the shape must be normalized, due to the obscurity of the cross correlation coefficient when the signals have largely varying amplitudes. In order to account for this, normalized cross correlation can be applied, where the correlation coefficient is normalized by the geometric mean of the power of the two signals [35]:

$$R_{xy}[k] = \frac{\sum_{n=-\infty}^{\infty} x[n] \cdot y[n+k]}{\sqrt{\sum_{n=-\infty}^{\infty} x[n]^2 \cdot \sum_{n=-\infty}^{\infty} y[n]^2}} \tag{5.2}$$

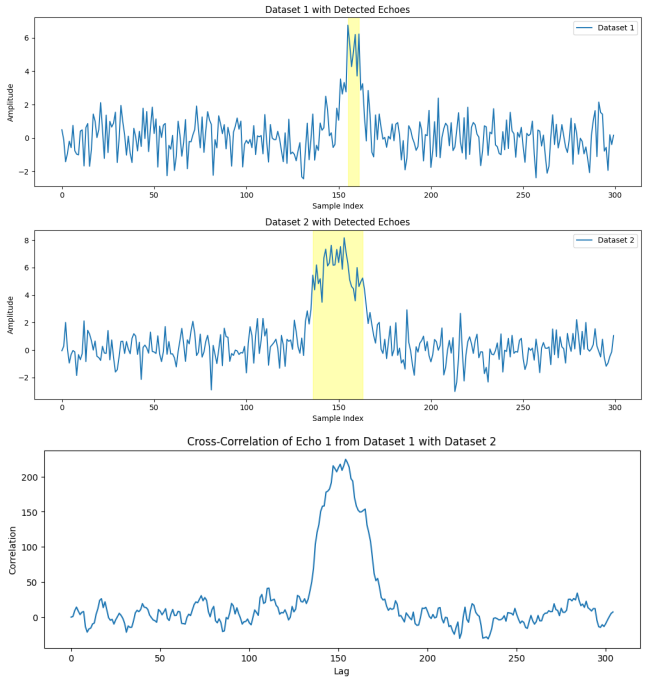

Figure 5.1: Example sensor data with cross-correlated delay estimation

5.3.4 Matching algorithms using multiple criteria

The last option regarding echo matching lies in the implementation of a decision algorithm based on multiple properties of incoming echoes. The signal feature extraction described above are yield the degree of similarity

between echoes, and are able to describe the signal across multiple dimensions. In order to match the echoes to each other using this selection algorithm, one must take the following considerations into account:

- Feature normalization: Ensures that no specific feature becomes dominant (except for when that is intended).
- Distance metric choice: The distance metric determines how the feature vectors of signals quantify the degree of similarity. The appropriate method to choose this is to train different models on echo data and observe the relative accuracy of each method.
- The matching process: This part could be either thresholding (checking whether signals are alike when their degree of similarity is above a certain threshold) or clustering, which involves comparing groups of signals to each other based on a certain characteristic and deciding upon whether they belong to the same group.

There are some limitations to this approach:

- High complexity of algorithm
- More computational resources required
- Training the algorithm is not feasible within the scope of this project due to time constraints

Due to the poor reliability of the RSS, this method will be disregarded. The decision algorithm will solely focus on the distance matching and cross-correlation intensity. The first task at hand, feature normalization will be applied to the distance matching technique. The distance matching technique uses as reference the threshold which acts as the maximum amount of spacing between echoes to be registered as some coefficient of matching. The closer the echoes are, the higher this matching coefficient is, and at complete overlap, the matching coefficient will be 1. For the cross-correlation principle, the signals are normalized to their power, which does not allow the maximum absolute value of the correlation coefficient to exceed one. The next step is calculating a weighted sum of these two variables for each echo pair, which serves as an indicator towards how much echo pairs match with each other. This weighted sum will then be compared to a certain threshold, which serves as the final decision condition for matching echoes.

The values for the weights of the weighted sum, the decision threshold among others are to be selected experimentally. The performance will be evaluated using some statistical measures, such as the precision and recall to check how well different sets of parameters may work. The score of matching on which the echoes will be evaluated according to Equation 5.3, where S is the matching score, α is the distance weight coefficient and Δ is the distance between the starts of compared echoes.

$$S = R_{xy} - \alpha \cdot \Delta \tag{5.3}$$

Chapter 6

Trilateration and mapping interface

6.1 Requirements

The trilateration and mapping interface is tasked with extracting the object locations using the distance measurements obtained from the previous modules. The distance measurements are obtained through the wireless communication module. Once this has been achieved, this module visualizes the locations of every detected object in the vicinity of the vehicle.

6.2 Wireless communication module

The wireless communication module is responsible for bringing the data from the microcontroller in the trailer to the microcontroller which is in the vehicle's screen. The module has to ensure that the whole data is transmitted correctly, but also received correctly with little to no errors. There are numerous types of wireless technologies which can be used for this module, only four were considered: Bluetooth, Wi-Fi, Zigbee and other Radio Frequency (RF) modules. Table 6.1 shows some comparisons in these technologies

Table 6.1: Comparative Analysis of Wireless Communication Technologies [36]

Criteria	**Bluetooth**	**Wi-Fi**	**Zigbee**	**RF Modules**
Bandwidth	1-3 Mbps	54 Mbps - 1 Gbps	250 Kbps	<1 Mbps
Range and Freq. band	1-100m, 2.4 GHz	10-200m, 2.4/5 GHz	1-100m, 2.4 GHz, 868 MHz	50-1000m, freq. dep.
Power Consumption	Low	High	Very Low	Low to Medium
Cost	Low	High	Low	Low to Medium
Overall Rating (1-5)	**3**	**2**	**5**	**5**

It can be seen from the results that Bluetooth has moderate bandwidth and a range of up to 100m at low power consumption and cost. On the other hand, Wi-Fi has a higher bandwidth and also a higher range. However, it does have a relatively higher power consumption and cost. Besides this, Zigbee is known for its very low power consumption and cost, along with having the same range as Bluetooth and a frequency band of 2.4 GHz or 868 MHz. Lastly, RF modules are pretty versatile in terms of range and frequency band. They have low to medium cost and power consumption, which makes them suitable for many different applications. As cost and power consumption are important requirements (PoR), it can be concluded that Wi-Fi performs worse in comparison to the other technologies.

Furthermore, there can also be some interference from other systems, especially in the 2.4 GHz band due to the frequency band being crowded with many devices. While Bluetooth can circumvent this issue through frequency hopping, it is still affected by this band. the 5 GHz band of Wi-Fi is relatively more reliable in comparison to its 2.4 GHz band, but it is still not as robust as Zigbee or the RF modules. Zigbee has some methods to handle interference, and is also designed for low power and reliable communication in noisy environments. As for the RF modules, the susceptibility to interference depends on the application and design. It can operate in less

crowded bands such as the 433 MHz band, which is less susceptible to interference. It also performs relatively well in noisy environments.

It should also be noted that, for cases where there is no line-of-sight propagation between the wireless communication modules, the performance of high frequencies is worse in comparison to low frequencies due to the positive relation between the frequency and the rate of attenuation [37]. Therefore when penetrating obstacles, the waves with a high frequency will decrease in amplitude relatively more than waves with a low frequency. Considering that the wireless transmission has to occur between a microcontroller in a trailer surrounded by metals and other similar materials, and a microcontroller in the car's screen, a wireless transmission with low frequency is preferred. This makes Bluetooth, Wi-Fi and the higher band usage of Zigbee also undesirable. In contrast, this makes RF Modules more attractive. Therefore, for the final design, either Zigbee or RF Modules will be used for wireless communication.

6.3 Trilateration algorithm

The localization of the objects is done through the use of the trilateration method. In essence, trilateration is a method which is primarily used to determine the location of an unknown point, based on its distance to multiple known points. In the case of 3D space, there is a need for at least four positions with known coordinates and four distances from these positions to the object which has yet to be localised. The distances in this case define spheres around each known point. The unknown object lies somewhere on the surface of each sphere. In most cases, four positions are needed as the spheres can have intersections in at least two positions if three points are used. This would then create some ambiguity, making it difficult to differentiate between invalid and valid coordinates. This can be circumvented though, through the use of some boundary conditions. Assuming that the three points represent the sensors, if the positions of these sensors are known, then for each sensor array it is possible to make boundary condition(s) where one of the found solutions is rejected, and the other is accepted. With these given constraints, it is feasible to determine the unique position of the unknown points using only three points as the constraints help to eliminate the potential ambiguity which would otherwise require a fourth point to eliminate. With the amount of known points now finalised, the distance between these points and the unknown point can be utilised to compute the coordinates of the unknown point. The mathematical model which is used to compute the location of this object is represented by the following equation:

$$\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} = r_i \tag{6.1}$$

Here (x_i, y_i, z_i) are the coordinates of the known point, and r_i is the radius of the sphere(the distance to the unknown point). In this case, the coordinates to the center of the object are computed by taking the distance measurement at the middle of every echo, the use of this will be explained later. Since there are three of such equations, the coordinates of the unknown object can be found through the point at which these three spheres intersect with each other. The corresponding equations for the three sensors can then be formulated as follows:

$$\sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} = r_1 \tag{6.2}$$

$$\sqrt{(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2} = r_2 \tag{6.3}$$

$$\sqrt{(x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2} = r_3 \tag{6.4}$$

As mentioned before, the coordinate system is taken in such a way that y represents the horizontal distance between the objects and the sensors, z represents the height, and x represents the line on which the sensors are on. Furthermore, given that (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3) are known, solving this set of nonlinear equations is feasible, allowing the location of the object to be determined. An example of trilateration using the formula's mentioned can be seen in Figure 6.1. For simplicity, the example is in 2D. The figure illustrates the method to determine the correct position of the detected object. Based on the boundary conditions, one of the two possible intersections will be rejected, and the correct position will be chosen.

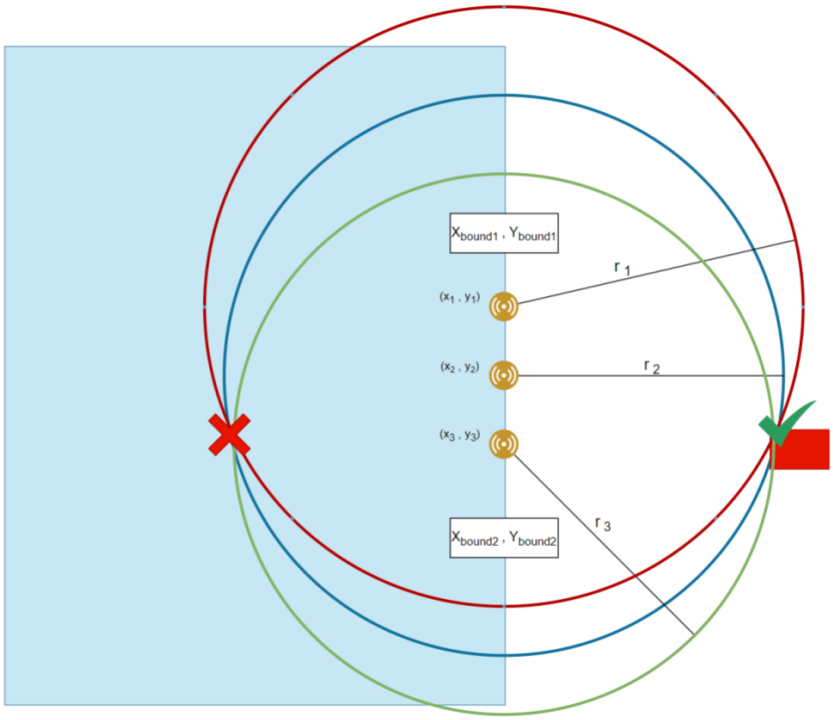

Figure 6.1: Trilateration in 2D with boundary conditions

In practice, while this allows the location of the object to be determined, due to measurement errors, the spheres might not intersect perfectly at a single point. Due to this, the system of nonlinear equations may not have an exact solution. To address this problem, an optimization approach is used. With an optimization approach, an objective function is defined to quantify the difference between the computed radii r_i and the distances calculated from a candidate position. These differences are known as residuals. The optimization algorithm then iteratively adjusts the candidate position to minimize the objective function used. In this case, the least squares method is used, which finds the position that minimizes the sum of the squared residuals. The function used with this method can be represented by the following formula:

$$S(\theta) = \sum_{i=1}^{n} (r_i - \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2})^2 \tag{6.5}$$

$S(\theta)$ is the objective function where θ represent the parameters of the model. In this case, θ represents the coordinates (x, y, z) of the unknown object. The function $S(\theta)$ in this case represents the sum of squared residuals, where r_i are the computed radii(distances to the unknown object) and (x_i, y_i, z_i) are the coordinates of the known points. The goal is to minimize the function (find the global minimum) by adjusting the values for the parameters x, y, z to find the best-fit position of the unknown object. Furthermore, an initial guess needs to be provided for the position of the unknown point in order for the optimization algorithm to converge faster and more accurately. The initial guess also assists with finding the global minimum of the optimization algorithm and preventing it from converging to a local minimum.

6.4 Size estimation

When planning to visualize the estimation of the locations onto the user interface, it would be nice to know the size of the object. Because without this information, the visualization would treat different sizes of the objects as the same size. Thus the method of size estimation is needed. In essence, this method uses the 'echo duration' from the matched echoes, as mentioned in Chapter 5, to approximate the size of the object. The basic principle works as follows: if the beginning and the end of the echo duration are known, then, through some calculation, the front surface area of the object can be obtained. As mentioned in the previous section, the coordinates to the middle of the echo are known. Thus, once the area is known, half of this area is put to the left of this point and the other half is put to the right. With this, an approximation of the size is achieved. This is possible due to the echo duration representing how much of the object every sensor 'sees'. To illustrate this idea, take a look

at Figure 6.2:

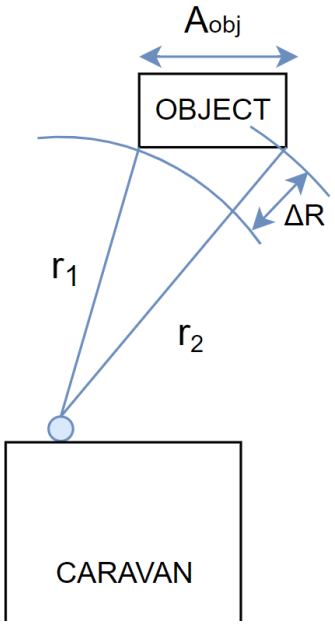

Figure 6.2: Size estimation using echo duration

In Figure 6.2 the area A_{obj} is what needs to be obtained through ΔR, where ΔR represents the difference in distance measurements from the beginning of the echo and the end of the echo. The area A_{obj} can then be computed through the use of the following formula:

$$A_{obj} = k \cdot \Delta R \tag{6.6}$$

Here k is a proportionality constant which will be determined empirically by testing out values for k, and then using the object's real area with its computed value. Once a value for k has been obtained such that the relation between A_{obj} and ΔR is clear, it is possible to compute A_{obj} for one sensor. Since there are three sensors doing measurements, the area can be obtained more accurately by taking the average of the individually obtained areas. This method can be used as the sensors are placed relatively close to each other, and thus the assumption can be made that they approximately 'see' the same object and object area (the former is also confirmed through echo matching as described in Chapter 5). Once a size estimation is obtained for all objects, the module will visualise it on the user interface for the end-users.

6.5 User interface

The requirements from the client regarding the user interface are the following: firstly, the camera-based system is responsible for visualizing the surroundings from multiple angles. It has to display these surroundings via a display in the car with sufficiently small latency in a 360 degree top-view fashion of which an example was provided in the form of Figure 6.3.

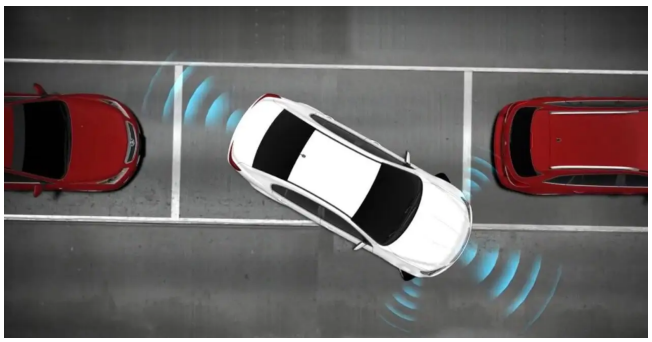

Figure 6.3: Proposed solution for the camera-based system

As for the proximity sensing system, the system should provide an audible collision warning mechanism as mentioned in the PoR. Furthermore, the system should have some visual module showing the locations of the detected objects. From these requirements, it can be concluded that there is no need for a complex user interface for the sensing system. Showing the detected objects on the user interface, through the use of lines or something similar while keeping the respective coordinates of both the vehicle, the sensors and the objects in mind, will be sufficient. Moreover, an audible sound warning mechanism will be implemented using simple threshold detection.

Figure 6.4 illustrates an example of the threshold detection method. when the car's sensor encounters an object, it will arrive to the Arduino. The microcontroller then makes a decision based on the remaining distance to the object. This decision determines whether the LCD display shows a green, yellow, or red color, indicating the proximity of the car to the object. As for a real-case scenario, the distance measurements would arrive to the microcontroller in the screen, and instead of LCD's, something such as beeping would be used which gets more rapid as the vehicle approaches the object.

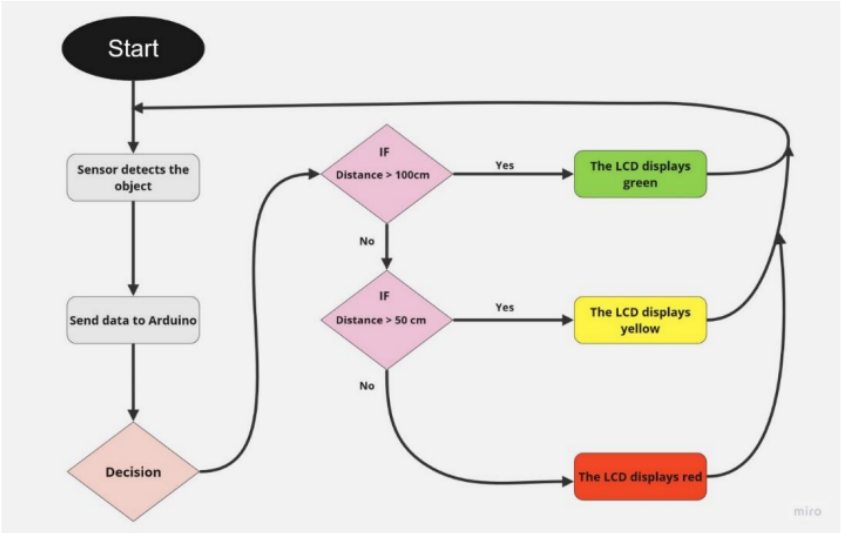

Figure 6.4: Example of a threshold detection system [12]

Chapter 7

Prototype implementation

The system for the prototype follows the design process which was mentioned in Figure 3.3. First three sensors are placed close to each other, forming the respective sensor array, then the microcontroller drives these sensors to transmit and receive ultrasonic echoes. The acoustic measurement are then fed to the signal processing microcontroller in order to distinguish echoes and apply feature extraction to these echoes. Then, some echo matching methods are used to correctly correlate echoes from different sensors to each other. Once this is done, this data along with the distance to each object is sent, through wireless communication, to the last module where the size of the object is estimated, and the location of the objects with respect to the vehicle is determined through the use of trilateration. Lastly, these objects are then mapped onto the user interface for the end-user.

7.1 Sensor network

For the prototype's sensor network, the decision was made to focus solely on the back of the trailer. This is because the principles applied to the back are the same as those for the sides. Therefore, if the system is successful for the back, it should also work for the sides. Implementing the system at the back of the vehicle will serve as the proof-of-concept for this project.

Due to the time constraints, it was not feasible to design and print custom sensor mounting strips, so a simple model using a cardboard box, along with the sensors connected to breadboards were implemented. Because of the narrow field of view of the chosen sensors, the spacing between the sensors has been chosen to be about 25 cm, which is closer than the ideal situation, and they are oriented slightly (about 5 degrees) towards each other to have an optimized overlap area. The main objective for this prototype implementation has been to accurately test the object localization, while the area of coverage was not as greatly emphasized. For now, the prototype is implemented with the relatively simple HC-SR04 ultrasonic sensors [29]. The working operation of these sensors are the same as mentioned in Chapter 4. An example of this sensor can be found in Appendix A.2.

The most important aspect of the project, the detection of multiple objects with a relatively cheap ultrasonic sensor, is done through analog output processing of the HCSR04 sensors [38]. This is done by reading the analog output of the sensor's comparator. With this achieved, the different objects are then able to be differentiated and identified through the use of advanced signal processing methods as mentioned in Chapter 5.

7.2 Signal processing and wireless transmission

The signal processing is divided in two parts, and run on separate microcontrollers. The first relates to extracting and sampling from the sensors, creating a list of the signal envelope data. This is realized on the Arduino Mega 2560 R3. The analog sensor data, once sampled, is formatted to a list for the different sensors. These lists of sensor data are then uploaded to **'Serial1'** on the Mega, which is connected to the RX pin on the HC12 RF wireless transmission module, of which the properties are also correctly programmed (baud rate). The HC-12 library available for Arduino IDE facilitates interfacing this communication module on the Mega. The next HC12 module is interfaced on the Arduino which is connected to a laptop. The second part of the signal processing will be implemented in Python, due to the vast amount of signal processing libraries available and the relative speed of development compared to a lower level language like C or C++. The first implemented function was that of normalizing the acoustic measurement with respect to the signal mean due to the inherent

offset when measuring from the HC-SR04. Furthermore, the signal has been squared to obtain its power. Then, echo detection and distinction using the threshold conditions has been realized. Furthermore, once the echoes are isolated from the reference sensor, they are cross-correlated to the other sensor data, in order to obtain the peaks at the overlapping delays. The distance matching has also been done alongside this cross correlation, and together, they are fused to match the echoes from sensors and provide a matching score. The features of the echo pairs are classified in to objects, of which this distance information is then ready for the trilateration calculation.

7.3 Trilateration and mapping interface

The localization of the objects is done through the algorithm described in Chapter 6. The inputs for the system is a list with the sensor coordinates along with distances to the unknown object. This list is then used in the localization algorithm. The coordinates of every detected object is computed and their sizes are estimated through the size estimation algorithm. Once the variables are known, the coordinates of the objects are shown on the display along with their estimated sizes. A simple example of the described user interface can be seen in Figure 7.1.

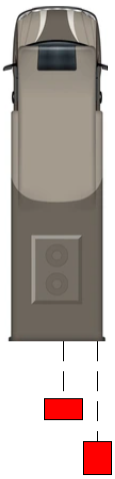

Figure 7.1: Simple prototype of the user interface

There is also a collision warning mechanism which will sound a 'beep' depending on the distance of an object to the vehicle. The beeping will get more frequent and louder as the object comes closer to the vehicle. Furthermore, the system will also include a malfunction detection mechanism that verifies the functionality of the audible collision warning system. This mechanism emits a 'beep' sound during the system's startup (when the vehicle is started) and shutdown (when the vehicle is turned off). If either of these 'beeps' is not heard, it signals a malfunction in the system.

Chapter 8

System integration

This chapter is dedicated to integrating the camera-based system [39] and the sensor-based system. The sensor and camera network will be combined alongside the user interface of both respective systems.

8.1 Sensor network and Camera network

The cameras will be mounted separately from the sensor strip due to their placement requirement near the top of the vehicle, whereas two sensors will be positioned near the bottom and one sensor at a mid-level position (not at the height of the cameras). Therefore, distinct strips will house each network of cameras and sensors.

For the data processing part, the sensors and cameras on the leisure vehicle will be implemented on separate single-board computers and separate transmission protocols. On the driver side, the algorithms of the camera system and the sensor system will be merged, and implemented on one single-board computer.

8.2 User interface

As for the user interface, as mentioned before, the display primarily consists of the bird-eye view from the camera system, with the proximity sensing of the ultrasonic sensor on top, as this was the requirement from the client. The localization of the detected objects will be done with the shared microcontroller in the screen. Object localization is handled by a shared microcontroller integrated into the screen. Once all object locations are determined, the proximity sensing system will display the distance to the objects using arcs: more arcs indicate the object is closer, while fewer arcs suggest it is farther from the vehicle. Size estimation of objects is omitted to maintain clarity, as the end-user should not be confused with two different showings of objects. The display on the screen will look like Figure 8.1. The figure illustrates a detected object located ahead of the vehicle. If the object gets too close, an audible warning will be transmitted, indicating that the object is too close to the vehicle.

Figure 8.1: Camsense view with an object detected in front of the vehicle

Chapter 9

Results and Discussion

9.1 Validation results

9.1.1 Individual sensor performance

The first result section will provide some insight into the sensor performance of the HCSR04. The first setup has been constructed with two objects in the field of view of the sensor, the first at 90 cm and the second at 180cm from the sensor. As seen in Figure 9.1, the relative distance is consistent with the locations of the echo starts in the waveforms. Hence, one could assume the sensors to work according to expectation, as it is able to detect multiple objects.

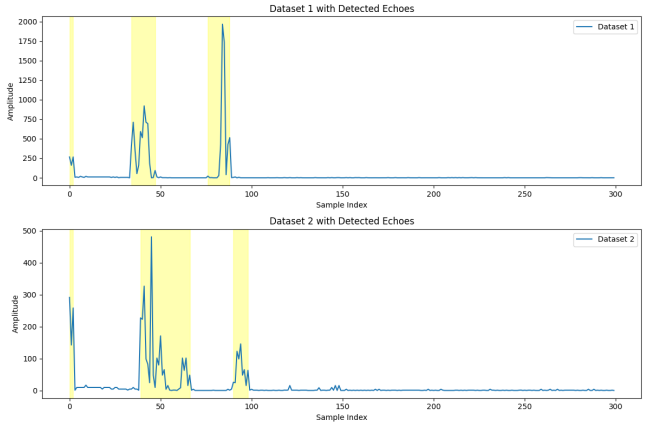

Figure 9.1: Sensor test with two objects placed at 90cm and 180 cm

9.1.2 Matching algorithm performance for a single sensor array

The algorithm has successfully matched the echoes of a sensor pair to each other. For the example mentioned in Figure 9.1, the program yielded the correct distance matching. However, the cross-correlation matching does not yield indicative results, especially with the distance matching combination algorithm. This could be the result of unoptimized weights. The cross correlation is likely flawed due to use of normalized cross-correlation combined with the lack of noise attenuation. Hence, it would be more logical to implement distance matching. The output in Figure 9.2 shows how distance matching was achieved for the previous example.

```
Matches based on distance:
Echo at index 0 in Dataset 1 matches with Echo at index 0 in Dataset 2
Echo at index 34 in Dataset 1 matches with Echo at index 39 in Dataset 2
Echo at index 76 in Dataset 1 matches with Echo at index 90 in Dataset 2
```

Figure 9.2: Distance matching output for example Figure 9.1

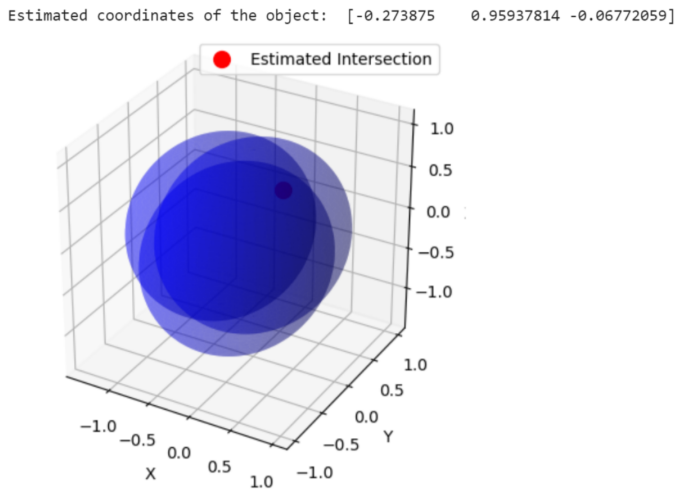

Figure 9.3: 3D trilateration result

9.1.3 Localization algorithm

The localization algorithm has been tested using a single object at the location of (-0.3, 0.9, 0.2). The sensors are placed at the coordinates (0, 0, 0), (-0.4, 0, 0) and (-0.2, 0, -0.34). The measured distances to the object from the sensors are 1.00, 0.97 and 1.00 m respectively. When feeding this information, along with the locations of the sensors to the python script, the following result was printed in Figure 9.3.

The only major discrepancy with this result lies in the calculation of the z component, which deviates with about 26 centimeters from the actual object. The deviation from the x and y values is less than 10 cm. Nevertheless, the object can be said to be detected successfully and rather accurately.

9.2 Discussion

Although the overall project has been developed into a largely functional proof of concept, there are still a lot of limitations and further possible improvements to implement. One of the biggest limitations with regard to the design of this device was the scarcity of time. This prevented the extensive exploration and experimentation with sensor fusion and alternative sensor choices, which potentially missed out on a better solution to the overall problem. Another aspect of this project, which is both an opportunity and limitation, is the wide-ranging scope of developing a proximity sensing solution. There are many ways to achieve this, but carefully evaluating each method comes at the cost of sacrificing the quality of the chosen implementation. Even for the chosen system, there were a lot of aspects which were to be developed and looked in to, many of which are presented in this thesis, but also quite a lot which have not made an appearance, such as noise filtering, outlier detection, optimal mounting options in terms of adhesives and custom transducer driver circuitry to name a few. However, of the implemented parts, the overall goal of the system, which is a novel method to accurately detect obstacles has been proven to work. Sadly, due to the 2-Dimensional nature of the camera system and the joint decision to implement the obstacle detection as an overlay, some key functionalities of the designed sensor system are redundant. For instance, using the multi-object trilateration, it would be possible to create a full 3D surround map of the vehicle with approximations for objects to scale. As the main request was a 360° topview fashion camera-based display as shown in Figure 6.3, the sensor system was 'forced' to scale down the complexity and only show the distance to the objects.

Another important part of the project was the capability to detect multiple objects with a ultrasonic sensor, as often only expensive ones are capable of this. Cheap ultrasonic sensors such as the HC-SR04 or even the JSN-SR04T are generally designed for single-object detection. While advanced techniques and algorithms can be used to distinguish multiple objects, these require a more sophisticated sensor system and processing capabilities. Hence why the cost for these sensors would be high. Nevertheless, with this project the ability to measure multiple objects using analog envelope detection with cheap ultrasonic sensors is achieved, and therefore it can be said that this is an innovative way of detecting multiple objects.

Furthermore, while the user interface is primarily based on the camera-based system, there is also a possibility of having a user interface which only shows the ultrasonic sensors. This option would also be there for the end-users. An example of this interface can be found in Figure 9.4.

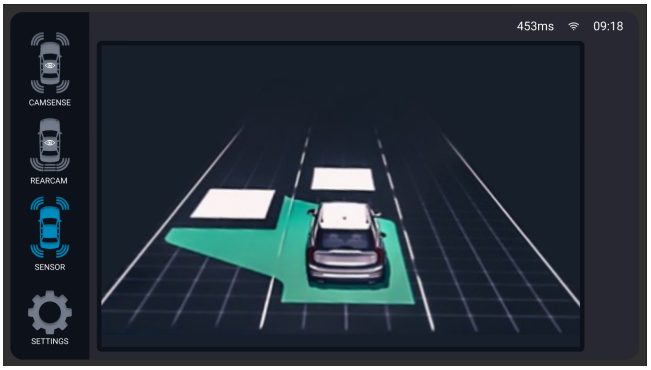

Figure 9.4: Sensor-only view

This figure is an example of Volvo's IntelliSafe display. Something similar is also possible for this particular sensor-based system as there is a method for size estimation already made, and the objects are able to be localized relatively accurately. Therefore, by adjusting the camera's to identify only the road markings, the sensor system should also be able to deliver a comprehensive display.

Lastly, the main goal of the sensor proximity sensing system in general was to assist the camera-based system by making up for its downsides of having relatively worse performance in low-visibility cases, having difficulty gauging exact distances and also having no mechanism to alert the users of any potential objects. Through the use of this system, each of these cases are handled; there is a collision avoidance warning mechanism, the objects are able to be localized accurately (and thus the distances are measured), and the sensors are able to detect objects even in low visibility conditions. Therefore, the project can be concluded to be successful.

Chapter 10

Conclusion

10.1 Conclusion

In this BAP-project, a sensor-based proximity sensing system has been developed to cover the shortcomings of the camera-based system for leisure vehicles. The main requirements for the sensor-based system were to include an audible collision warning mechanism which emits some sound when the vehicle gets too close to an object, and also to detect and visualise these objects. Furthermore, sensor overlap was required all around the vehicle for effective use of the trilateration algorithm. The system has successfully covered all main requirements. Moreover, the sensor-based system is able to detect multiple objects through advanced signal processing techniques using relatively cheap ultrasonic sensors (which are, under normal circumstances, not made for multi-object detection), and show them onto the user interface. As for the user interface displaying the detected objects in the vicinity of the vehicle, for simplicity reasons, only the back of the caravan was considered as a proof-of-concept. The system integration incorporates the measured distances to the object and shows it on the display with the camera feed.

10.2 Future work

While the presented results comply with the main requirements, there are still a few improvements which can be made in future projects:

- The developed system only applies to the back and the sides of the caravan. The front corners and caravan from the front are still to be evaluated.
- The user interface consisting of mainly the ultrasonic sensors, as described in Chapter 9, could be made with the addition of some other sensor which could cover the weaknesses of ultrasonic sensors.
- The sensor properties still has room to be optimized. Due to time constraints, this was not able to be done. It can be achieved by constructing a custom PCB for the ultrasonic sensors to obtain the desired characteristics for maximum coverage.
- As mentioned in the discussion, the lack of time caused some lost opportunities with regards to the optimal development of the sensor-based system. If some more time is taken to delve deeper into each topic, a result of higher quality could be achieved. The final product could then also be an unique solution where the camera-based system could be disregarded.

Bibliography

[1] Caravan Guard. "Top 10 most accident prone parts of a caravan revealed!" [Online]. Available: https://www.caravanguard.co.uk/news/accident-prone-part-caravan-7252/.

[2] Toyota Motor Corporation, *Toyota to launch enhanced parking support systems*, 2023. [Online]. Available: https://global.toyota/en/detail/4201181.

[3] Bosch Mobility Solutions, *Rear view system*, 2023. [Online]. Available: https://www.bosch-mobility.com/en/solutions/assistance-systems/rear-view-system/.

[4] CarAndBike, *Reverse parking sensors will be soon made mandatory on new cars*, Accessed: 2023-06-01, 2023. [Online]. Available: https://www.carandbike.com/news/reverse-parking-sensors-will-be-soon-made-mandatory-on-new-cars-1456634.

[5] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep learning sensor fusion for autonomous vehicle perception and localization: A review," *Sensors*, vol. 20, no. 15, 2020, ISSN: 1424-8220. DOI: 10.3390/s20154220. [Online]. Available: https://www.mdpi.com/1424-8220/20/15/4220.

[6] Y. Huang and Y. Chen, *Autonomous driving with deep learning: A survey of state-of-art technologies*, 2020. arXiv: 2006.06091 [cs.CV].

[7] Y.-C. Tsai, K.-H. Chen, Y. Chen, and J.-H. Cheng, "Accurate and fast obstacle detection method for automotive applications based on stereo vision," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2018, pp. 1–4. DOI: 10.1109/VLSI-DAT.2018.8373249.

[8] BlindSpotMonitor, *Radar blind spot monitoring systems*, Accessed: 2023-06-01, 2023. [Online]. Available: https://blindspotmonitor.com/radar-blind-spot-monitoring-systems/#:~:text=Blind%20spot%20monitor%20systems%20detect,sides%20of%20the%20rear%20bumpers.

[9] P. Hosur, R. B. Shettar, and M. Potdar, "Environmental awareness around vehicle using ultrasonic sensors," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 1154–1159. DOI: 10.1109/ICACCI.2016.7732200.

[10] H. Hatano, T. Yamazato, and M. Katayama, "Automotive ultrasonic array emitter for short-range targets detection," in *2007 4th International Symposium on Wireless Communication Systems*, 2007, pp. 355–359. DOI: 10.1109/ISWCS.2007.4392361.

[11] P. H. et al, *Environmental Awareness Around Vehicle Using Ultrasonic Sensors*. Intl. Conference, 2016.

[12] A. Ebrahimi and E. Akbari, *Design and implementation of an affordable reversing camera system with object detection and OBD-2 integration for commercial vehicles*. KTH Vetenskap och konst, 2023.

[13] J. Kocić, N. Jovičić, and V. Drndarević, "Sensors and sensor fusion in autonomous vehicles," in *2018 26th Telecommunications Forum (TELFOR)*, 2018, pp. 420–425. DOI: 10.1109/TELFOR.2018.8612054.

[14] H. Cui, J. Song, and D. Liu, "Ultrasonic array based obstacle detection in automatic parking," in *Proceedings of 2013 Chinese Intelligent Automation Conference*, Z. Sun and Z. Deng, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 129–140, ISBN: 978-3-642-38460-8.

[15] B. Bhowmick, S. Bhadra, and A. Sinharay, "Stereo vision based pedestrians detection and distance measurement for automotive application," *Intelligent Systems, Modelling and Simulation, International Conference on*, vol. 0, pp. 25–29, Jan. 2011. DOI: 10.1109/ISMS.2011.14.

[16] R. Kapoor, S. Ramasamy, A. Gardi, C. Bieber, L. Silverberg, and R. Sabatini, "A novel 3d multilateration sensor using distributed ultrasonic beacons for indoor navigation," *Sensors*, vol. 16, no. 10, 2016. DOI: 10.3390/s16101637. [Online]. Available: https://www.mdpi.com/1424-8220/16/10/1637.

[17] T. W. Moleski and J. Wilhelm, "Trilateration positioning using hybrid camera-lidar system," in *AIAA Scitech 2020 Forum*. DOI: 10.2514/6.2020-0393.

[18] A. Rahimi and Y. He, *A review of essential technologies for autonomous and semiautonomous articulated heavy vehicles*. CSME, 2020.

[19] X. Z. et al, *Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications*. IEEE, 2020.

[20] Neuvition, "LiDAR Price for Cars," 2022. [Online]. Available: https://cdn.neuvition.com/media/blog/lidar-price.html.

[21] Texas Instruments, *Best practices for placement and angle of mmwave radar devices*, SWRA758, Application Brief, Texas Instruments Incorporated, Jan. 2023. [Online]. Available: https://www.ti.com/lit/ab/swra758/swra758.pdf?ts=1714242625381.

[22] W. X. et al, *Analyzing and Enhancing the Security of Ultrasonic Sensors for Autonomous Vehicles*. IEEE, 2018.

[23] S. A. et al, *Performance comparison of Infrared and Ultrasonic sensors for obstacles of different materials in vehicle/ robot navigation applications*. IOP Publishing, 2016.

[24] C. Ilas, "A review of essential technologies for autonomous and semiautonomous articulated heavy vehicles," in *8th International Symposium on Advanced Topics in Electrical Engineering (ATEE 2013)*, 2013, pp. 0–5.

[25] N. E. Brown *et al.*, "Development of an energy efficient and cost effective autonomous vehicle research platform," vol. 22, no. 16, p. 5999, 2022, ISSN: 1424-8220. DOI: 10.3390/s22165999. [Online]. Available: https://doi.org/10.3390/s22165999.

[26] R. Ayala and T. K. Mohd, "Sensors in autonomous vehicles: A survey," *ASME Journal of Autonomous Vehicle Systems*, vol. 1, no. 3, p. 031 003, Jul. 2021. DOI: 10.1115/1.4052991. [Online]. Available: https://doi.org/10.1115/1.4052991.

[27] Texas Instruments, *TI Designs: TIDA-01597 Automotive Ultrasonic Sensing Module Reference Design for Park Assist*, TIDA-01597, Texas Instruments Incorporated, 2018. [Online]. Available: https://www.ti.com/tool/TIDA-01597.

[28] T. Instruments, "Ultrasonic sensing basics," Texas Instruments, Tech. Rep. SLAA907D, 2021. [Online]. Available: https://www.ti.com/lit/an/slaa907d/slaa907d.pdf?ts=1716615127795.

[29] Handson Technology, *Hc-sr04 ultrasonic sensor module*, 2019. [Online]. Available: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf.

[30] M. Agrawal, A. Prasad, J. R. Bellare, and A. A. Seshia, "Characterization of mechanical properties of materials using ultrasound broadband spectroscopy," *Ultrasonics*, vol. 64, pp. 186–195, 2016, ISSN: 0041-624X. DOI: https://doi.org/10.1016/j.ultras.2015.09.001.

[31] J. Krautkramer and H. Krautkramer, *Ultrasonic Testing of Materials*, 2nd ed. Berlin, Heidelberg: Springer-Verlag, 1977, Translated from the Third Revised German Edition, ISBN: 978-3-662-02296-2. DOI: 10.1007/978-3-662-02296-2.

[32] L. E. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders, *Fundamentals of Acoustics*, 4th ed. John Wiley Sons, 2000.

[33] J.-S. Lee and Y.-K. Kim, "Advances in ultrasonic sensors for environmental monitoring," *Sensors and Actuators A: Physical*, vol. 162, no. 2, pp. 248–259, 2010.

[34] P. Bourke, *Cross correlation and autocorrelation*, Online, Available online: http://www.paulbourke.net/miscellaneous/correlate/, 1996.

[35] J. G. Proakis and D. K. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed. Prentice Hall, 2007.

[36] T. Trainer, *Comparison between different wireless technologies*, https://www.telecomtrainer.com/comparison-between-different-wireless-technologies/.

[37] M. Bahjat, T. Rahman, and O. Abdul Aziz, "Propagation path loss modeling and outdoor coverage measurements review in millimeter wave bands for 5g cellular communications," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 2254–2260, Aug. 2018. DOI: 10.11591/ijece.v8i4.pp2254-2260.

[38] Hackster.io, *Multiple object detection with a single hc-sr04 sensor*, 2019. [Online]. Available: https://www.hackster.io/news/multiple-object-detection-with-a-single-hc-sr04-sensor-a30aa9659ded.

[39] H. Karim, D. Landman, and S. Pavlović, *Wireless 360 degree surround view camera system for leisure vehicles*, Faculty of Electrical Engineering, Mathematics and Computer Science, 2024.

Appendix A

Additional figures

A.1 A.1 Additional figures for the sensor network

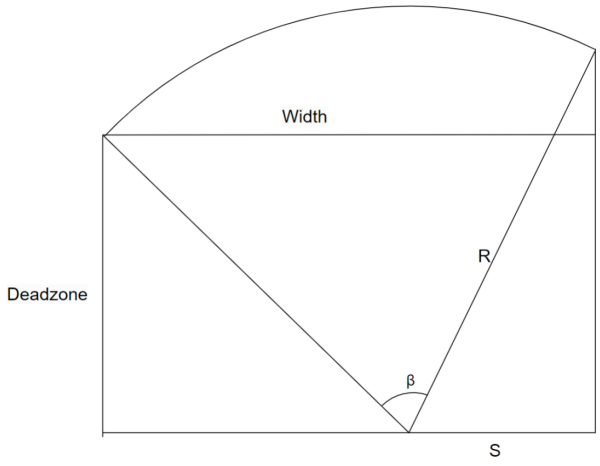

Figure A.1: Schematic view of coverage between first and farthest sensor

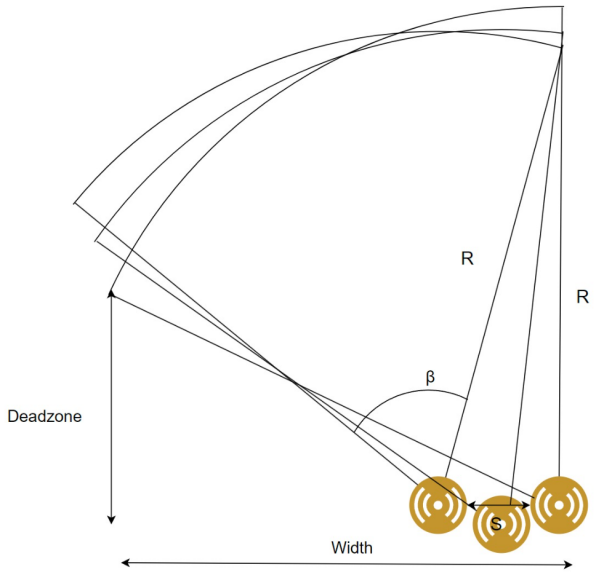

Figure A.2: Estimated coverage area for a sensor array

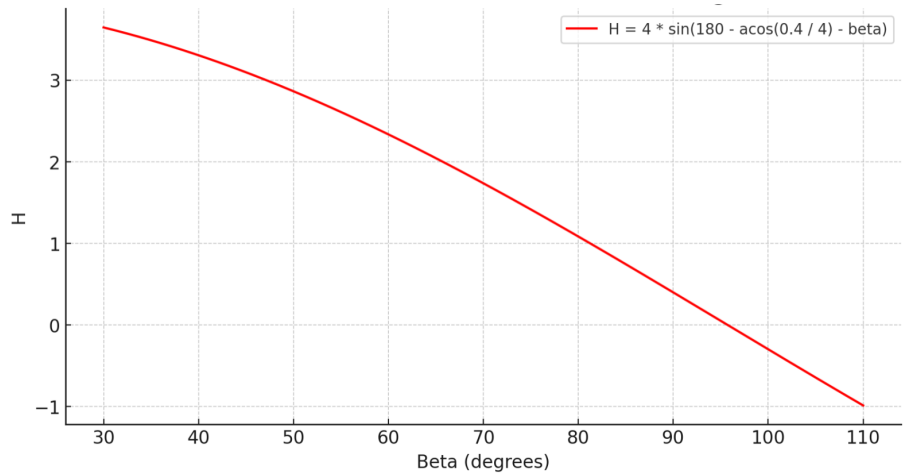

Figure A.3: Plot of Deadzone as a function of the beam angle (30 to 110 degrees)

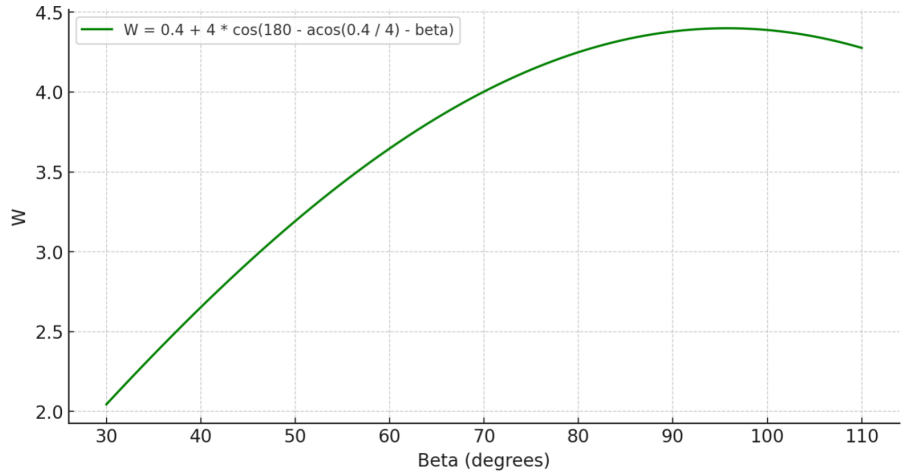

Figure A.4: Plot of Width as a function of the beam angle (30 to 110 degrees)

A.2 A.2 Additional figures for the prototype

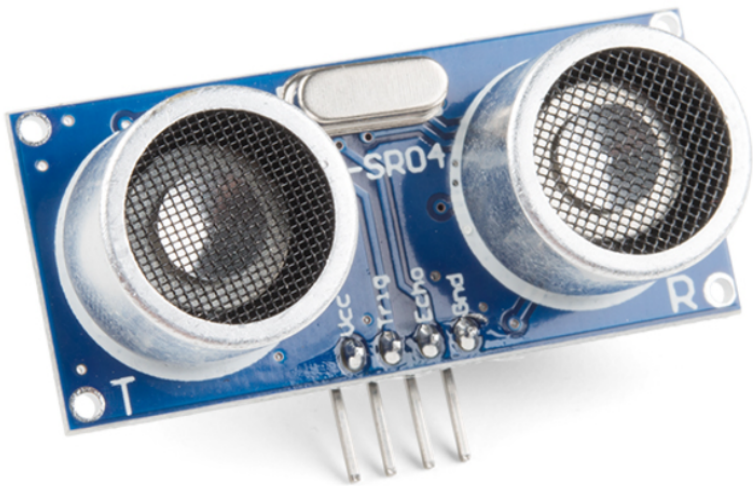

Figure A.5: HC-SR04 Sensor

Appendix B

Python code

B.1 B.1 Simple threshold detection

```
import numpy as np
import matplotlib.pyplot as plt

def detect_envelope(echo_data, threshold):
    start_index = None
    peak_index = None
    peak_value = -np.inf
    end_index = None
    below_threshold_count = 0

    for i, value in enumerate(echo_data):
        # Detect start of the envelope
        if start_index is None and value > threshold:
            start_index = i

        # Continue if start has been detected
        if start_index is not None:
            # Find peak
            if value > peak_value:
                peak_value = value
                peak_index = i

            # Detect end of the envelope
            if value < threshold:
                below_threshold_count += 1
            else:
                below_threshold_count = 0

            if below_threshold_count > 4:
                end_index = i - below_threshold_count
                break

    return start_index, peak_index, end_index, peak_value

# Simulate the echo data
np.random.seed(0) # For reproducibility
echo_data = np.random.normal(0, 1, 300) # Random data simulating echo signal
echo_data[50:150] += 10 # Simulate an echo envelope between indices 50 and 150

threshold = 5 # Set a threshold
start, peak, end, peak_val = detect_envelope(echo_data, threshold)
```

```
# Isolate the echo segment for frequency analysis
echo_segment = echo_data[start:end] if start and end else None

# Perform FFT on the isolated echo segment
if echo_segment is not None:
    fft_result = np.fft.fft(echo_segment)
    frequencies = np.fft.fftfreq(len(echo_segment), d=1/10000) # Sample rate 10 kHz
    magnitude = np.abs(fft_result)

    # Plot the frequency spectrum
    plt.figure(figsize=(12, 6))
    plt.subplot(121)
    plt.plot(echo_data, label='Echo Signal')
    plt.axhline(y=threshold, color='r', linestyle='--', label='Threshold')
    plt.scatter(start, echo_data[start], color='green', s=100, label='Start')
    plt.scatter(peak, echo_data[peak], color='orange', s=100, label='Peak')
    plt.scatter(end, echo_data[end], color='red', s=100, label='End')
    plt.title('Echo Signal Detection')
    plt.xlabel('Sample Index')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.grid(True)

    plt.subplot(122)
    plt.plot(frequencies[:len(frequencies) // 2], magnitude[:len(magnitude) // 2])
    plt.title('Frequency Spectrum of the Detected Echo')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude')
    plt.grid(True)

    plt.tight_layout()
    plt.show()
else:
    print("No echo detected or echo segment is too short for FFT.")

import numpy as np
import matplotlib.pyplot as plt

def detect_multiple_echoes(echo_data, threshold):
    echo_starts = []
    echo_ends = []
    below_threshold_count = 0
    in_echo = False

    for i, value in enumerate(echo_data):
        if value > threshold and not in_echo:
            in_echo = True
            echo_starts.append(i)

        if in_echo and value < threshold:
            below_threshold_count += 1
        else:
            below_threshold_count = 0

        if in_echo and below_threshold_count > 4:
            echo_ends.append(i - below_threshold_count)
            in_echo = False
            below_threshold_count = 0

    # Check if the last echo does not end properly
    if in_echo:
```

```
        echo_ends.append(len(echo_data) - 1) # Assume echo ends at the last sample

    return echo_starts, echo_ends, len(echo_starts)

def plot_frequency_spectrum(data, sample_rate, title):
    fft_result = np.fft.fft(data)
    frequencies = np.fft.fftfreq(len(data), d=1/sample_rate)
    magnitude = np.abs(fft_result)

    plt.figure(figsize=(12, 6))
    plt.plot(frequencies[:len(frequencies) // 2], magnitude[:len(magnitude) // 2])
    plt.title(title)
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude')
    plt.grid(True)
    plt.show()

# Simulate the echo data with two distinct echoes
np.random.seed(0)
sample_rate = 10000 # Sample rate in Hz
echo_data = np.random.normal(0, 1, 300) # Background noise
echo_data[50:150] += 10 # First simulated echo
echo_data[170:200] += 8 # Second simulated echo

threshold = 5 # Detection threshold
echo_starts, echo_ends, echo_count = detect_multiple_echoes(echo_data, threshold)

# Plot the results - time domain
plt.figure(figsize=(12, 6))
plt.plot(echo_data, label='Echo Signal')
for start, end in zip(echo_starts, echo_ends):
    plt.axvline(x=start, color='green', linestyle='--', label='Start of Echo' if start ==
        ↪ echo_starts[0] else "")
    plt.axvline(x=end, color='red', linestyle=':', label='End of Echo' if end == echo_ends
        ↪ [0] else "")
plt.title(f'Echo Signal Detection - {echo_count} Echoes Detected')
plt.xlabel('Sample Index')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.show()

# Plot the frequency spectrum of the entire signal
plot_frequency_spectrum(echo_data, sample_rate, "Frequency Spectrum of the Entire Echo
    ↪ Signal")

# Optionally, plot the frequency spectrum of individual echoes
for start, end in zip(echo_starts, echo_ends):
    plot_frequency_spectrum(echo_data[start:end], sample_rate, f"Frequency Spectrum of Echo
        ↪ from {start} to {end}")
```

B.2 B.2 Trilateration algorithm

```
    import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import least_squares

def plot_sphere(ax, center, radius, color='b', alpha=0.3):
    u = np.linspace(0, 2 * np.pi, 100)
    v = np.linspace(0, np.pi, 100)
    x = center[0] + radius * np.outer(np.cos(u), np.sin(v))
    y = center[1] + radius * np.outer(np.sin(u), np.sin(v))
    z = center[2] + radius * np.outer(np.ones(np.size(u)), np.cos(v))
    ax.plot_surface(x, y, z, color=color, alpha=alpha)

def trilaterate3D(distances):
    positions = np.array([p[:3] for p in distances])
    radii = np.array([p[3] for p in distances])
    def residuals(x):
        return np.linalg.norm(x - positions, axis=1) - radii
    initial_guess = np.mean(positions, axis=0)
    result = least_squares(residuals, initial_guess)
    return result.x

# Coordinates and radii for each sphere
distances = [
    [-0.40, 0.0, 0.0, 0.97], # Sphere 1
    [0, 0.0, 0.0, 1.00], # Sphere 2
    [-0.2, 0, -0.34, 1.00] # Sphere 3
]
# Expected actual location approximately [0.5, 0.0, 0.0]

# Create a 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Plot each sphere
for dist in distances:
    plot_sphere(ax, np.array(dist[:3]), dist[3], 'b')

# Calculate and plot the estimated intersection
intersection = trilaterate3D(distances)
print("Estimated coordinates of the object: ", intersection)
ax.scatter(*intersection, color='r', s=100, label='Estimated Intersection')
ax.legend()

# Set aspect of the plot to equal to ensure spheres are not distorted
ax.set_box_aspect([1,1,1])

plt.show()
```

B.3 B.3 Matching algorithm: cross-correlation

```
import numpy as np
import matplotlib.pyplot as plt

def generate_echo_data(num_samples, num_echoes, min_separation=100):
    """Generates synthetic echo data with varied shapes and ensures echoes are well-
        ↪ separated."""
    data = np.random.normal(0, 1, num_samples)
    last_end = 0
    for _ in range(num_echoes):
        if last_end + min_separation + 150 >= num_samples:
            break # Not enough space left for more echoes
        start = np.random.randint(last_end + min_separation, num_samples - 150)
        length = np.random.randint(30, 100)
        amplitude = np.random.randint(5, 15)
        data[start:start+length] += amplitude * np.hanning(length) # Smooth echo shape
        last_end = start + length
    return data

def detect_echoes(data, threshold):
    """Detects echoes based on a threshold and returns their start and end indices."""
    echo_starts = []
    echo_ends = []
    below_threshold_count = 0
    in_echo = False

    for i, value in enumerate(data):
        if value > threshold and not in_echo:
            in_echo = True
            echo_starts.append(i)

        if in_echo and value < threshold:
            below_threshold_count += 1
        else:
            below_threshold_count = 0

        if in_echo and below_threshold_count > 4:
            echo_ends.append(i - below_threshold_count)
            in_echo = False

    if in_echo:
        echo_ends.append(len(data) - 1)

    return echo_starts, echo_ends

def extract_echo_segments(data, echo_starts, echo_ends):
    """Extracts segments based on detected start and end indices."""
    segments = []
    for start, end in zip(echo_starts, echo_ends):
        if end > start: # Ensure segment is valid
            segments.append(data[start:end])
    return segments

def compute_cross_correlation(segment, full_data):
    """Computes cross-correlation of a segment with the full dataset."""
    correlation = np.correlate(full_data, segment, mode='full')
    return correlation

def plot_datasets_with_echoes(datasets, echo_infos):
    """Plots datasets with highlighted echo segments."""
```

```
    plt.figure(figsize=(12, 8))
    for i, (data, (starts, ends)) in enumerate(zip(datasets, echo_infos)):
        plt.subplot(len(datasets), 1, i + 1)
        plt.plot(data, label=f'Dataset {i+1}')
        for start, end in zip(starts, ends):
            plt.axvspan(start, end, color='yellow', alpha=0.3) # Highlight echo regions
        plt.title(f'Dataset {i+1} with Detected Echoes')
        plt.xlabel('Sample Index')
        plt.ylabel('Amplitude')
        plt.legend()
    plt.tight_layout()
    plt.show()

# Generating datasets
num_samples = 300
dataset1 = generate_echo_data(num_samples, 3)
dataset2 = generate_echo_data(num_samples, 3)
datasets = [dataset1, dataset2]

# Detecting and extracting echoes
echo_info_1 = detect_echoes(dataset1, 5)
echo_info_2 = detect_echoes(dataset2, 5)
segments1 = extract_echo_segments(dataset1, *echo_info_1)

# Plotting datasets with detected echoes
plot_datasets_with_echoes(datasets, [echo_info_1, echo_info_2])

# Performing cross-correlation and analyzing results
for i, segment in enumerate(segments1):
    if segment.size > 0: # Ensure segment is not empty
        correlation = compute_cross_correlation(segment, dataset2)
        plt.figure(figsize=(12, 4))
        plt.plot(correlation)
        plt.title(f'Cross-Correlation of Echo {i+1} from Dataset 1 with Dataset 2')
        plt.xlabel('Lag')
        plt.ylabel('Correlation')
        plt.show()
```

B.4 B.4 Matching algorithm: cross-correlation and distance matching

```
    def detect_echoes(data, threshold, max_below_threshold=15):
    """Detects echoes based on a threshold and returns their start and end indices."""
    echo_starts = []
    echo_ends = []
    below_threshold_count = 0
    in_echo = False

    for i, value in enumerate(data):
        if value > threshold and not in_echo:
            in_echo = True
            echo_starts.append(i)
            print(f"Echo starts at {i}") # Debugging output

        if in_echo and value < threshold:
            below_threshold_count += 1
        else:
            below_threshold_count = 0
        if in_echo and below_threshold_count > max_below_threshold:
            echo_ends.append(i - below_threshold_count)
            in_echo = False
            print(f"Echo ends at {i - below_threshold_count}") # Debugging output
    if in_echo:
        echo_ends.append(len(data) - 1)
        print(f"Echo ends at {len(data) - 1}") # Debugging output
    return echo_starts, echo_ends

def string_to_list(num_string):
    # Split the string by spaces and convert each element to an integer
    return [int(num) for num in num_string.split()]

# Convert the provided string data into lists of integers
dataset1 = string_to_list("477 506 477 496 490 491 489 490 491 489 490 490 490 490 490 490
    ↪ 490 490 490 490 490 490 490 491 490 491 490 492 491 491 491 491 491 493 473 520 512
    ↪ 486 506 469 516 463 520 467 507 494 495 503 490 492 490 492 492 492 493 492 493 493
    ↪ 493 493 493 493 494 493 493 493 493 493 493 493 493 493 493 493 493 493 498 492 495
    ↪ 493 493 494 488 514 449 535 487 514 516 495 491 490 493 496 494 494 493 493 493 493
    ↪ 493 493 493 494 493 493 493 493 493 492 493 493 494 494 492 493 493 493 495 494 493
    ↪ 495 493 494 493 495 493 492 493 493 493 494 494 494 493 495 494 494 493 494 494 493
    ↪ 495 493 495 494 494 495 495 493 494 494 495 493 495 493 493 495 492 495 493 494 493
    ↪ 495 494 493 493 494 494 493 493 494 493 494 494 495 493 494 493 495 493 494 494 494
    ↪ 494 493 495 494 494 494 494 493 495 493 494 493 494 494 493 494 494 494 493 495 493
    ↪ 493 494 493 495 494 495 494 495 493 495 493 495 494 494 494 492 495 493 495 494 493
    ↪ 494 493 494 493 493 494 493 494 493 493 494 494 493 495 493 495 493 493 494 493 494
    ↪ 493 494 494 493 494 493 494 493 494 494 493 493 493 493 493 495 492 493 493 493 494
    ↪ 493 493 494 493 494 493 493 493 493 494 493 493 492 492 494 493 494 494 493 492 493
    ↪ 493 493 493 492 493 493 493 493 494 493 494")
dataset2 = string_to_list("476 505 477 494 490 490 490 490 490 489 490 490 490 490 490 490
    ↪ 490 490 491 490 490 490 490 491 491 490 490 491 491 491 491 491 492 491 491 490 491
    ↪ 491 494 478 508 475 503 484 498 515 500 490 483 502 480 500 485 495 497 492 493 492
    ↪ 492 493 491 496 483 501 483 497 500 494 495 493 493 493 493 493 493 493 493 493 493
    ↪ 494 493 493 493 493 493 493 493 495 494 491 488 498 482 503 481 500 485 497 501 494
    ↪ 495 494 494 494 493 494 494 493 494 493 493 493 494 493 493 494 493 493 492 494 492
    ↪ 497 494 494 494 494 493 494 494 494 494 493 493 493 493 494 494 496 493 494 494 493
    ↪ 494 494 490 494 497 493 497 493 493 493 495 493 494 493 494 493 494 493 494 493 494
    ↪ 493 494 493 494 493 495 493 495 493 494 494 493 494 493 494 494 494 493 494 493 494
    ↪ 494 494 494 494 494 494 494 494 494 493 493 494 493 495 493 494 493 494 493 493 495
    ↪ 494 493 493 493 493 493 493 494 494 493 494 493 494 493 494 494 494 494 493 494 493
    ↪ 493 494 494 493 493 495 494 494 493 494 493 493 494 493 494 493 493 494 493 493 493
```

```
    ↪ 494 492 493 494 493 494 494 493 494 493 493 493 495 493 494 493 494 495 493 494 493
    ↪ 495 493 494 493 494 494 494 494 494 494 493 494 493 493 494 493 493 495 493 493 495
    ↪ 493 494 493 494 493 493 494 493 493 494 493")
# Normalize the datasets by subtracting 490 from each element
mean_dataset1 = sum(dataset1) / len(dataset1)
mean_dataset2 = sum(dataset2) / len(dataset2)

# Normalize the datasets by subtracting the respective means from each element
norm1 = [x - mean_dataset1 for x in dataset1]
norm2 = [x - mean_dataset2 for x in dataset2]
squared_norm1 = [x**2 for x in norm1]
squared_norm2 = [x**2 for x in norm2]

# Replace 'segment.size > 0' with 'len(segment) > 0'
echo_info_1 = detect_echoes(squared_norm1, 20)
echo_info_2 = detect_echoes(squared_norm2, 20)
segments1 = extract_echo_segments(squared_norm1, *echo_info_1)

# Plot the normalized datasets
plot_datasets_with_echoes([squared_norm1, squared_norm2], [echo_info_1, echo_info_2])

# Compute and plot cross-correlation
import matplotlib.pyplot as plt
for i, segment in enumerate(segments1):
    if len(segment) > 0: # Ensure segment is not empty
        correlation = compute_cross_correlation(segment, squared_norm2)
        plt.figure(figsize=(12, 4))
        plt.plot(correlation)
        plt.title(f'Cross-Correlation of Echo {i+1} from Dataset 1 with Dataset 2')
        plt.xlabel('Lag')
        plt.ylabel('Correlation')
        plt.show()
def match_echoes_based_on_distance(echo_indices_1, echo_indices_2, max_distance):
    """Matches echoes based on their distance in samples."""
    matches = []
    for index1 in echo_indices_1:
        # Find indices in the second set of echoes that are within the max_distance from
            ↪ index1
        matched_indices = [index2 for index2 in echo_indices_2 if abs(index1 - index2) <=
            ↪ max_distance]
        for index2 in matched_indices:
            matches.append((index1, index2))
    return matches

# Assume echo_info_1 and echo_info_2 contain the start indices of detected echoes
# echo_info_1 = detect_echoes(dataset1, 5) # Echo detection results for dataset1
# echo_info_2 = detect_echoes(dataset2, 5) # Echo detection results for dataset2

# Assuming you have lists of echo start times for each dataset:
echo_starts_1 = echo_info_1[0]
echo_starts_2 = echo_info_2[0]

# Apply distance matching
max_distance = 30 # Echoes within 30 samples are considered a match
matches = match_echoes_based_on_distance(echo_starts_1, echo_starts_2, max_distance)

# Print matches
print("Matches based on distance:")
for match in matches:
    print(f"Echo at index {match[0]} in Dataset 1 matches with Echo at index {match[1]} in
        ↪ Dataset 2")
```

B.5 B.5 Combined decision algorithm

```
    import numpy as np

def compute_cross_correlation(segment1, segment2):
    """Computes normalized cross-correlation between two segments."""
    correlation = np.correlate(segment1, segment2, "full")
    max_correlation = np.max(correlation)
    return max_correlation / (np.linalg.norm(segment1) * np.linalg.norm(segment2))

def match_and_score_echoes(dataset1, dataset2, echo_info_1, echo_info_2, max_distance):
    matches = []
    for start1 in echo_info_1[0]: # echo starts in dataset1
        for start2 in echo_info_2[0]: # echo starts in dataset2
            distance = abs(start1 - start2)
            if distance <= max_distance:
                # Extract segments based on echo start and some predefined length
                segment_length = 100 # example length
                segment1 = dataset1[start1:start1+segment_length]
                segment2 = dataset2[start2:start2+segment_length]

                # Compute cross-correlation between the two segments
                correlation = compute_cross_correlation(segment1, segment2)

                # Scoring function could be a combination of correlation and inverse of
                    ↪ distance
                score = correlation - 0.1 * distance # example scoring function
                matches.append((start1, start2, correlation, distance, score))

    # Sort matches based on the score in descending order
    matches.sort(key=lambda x: x[-1], reverse=True)

    return matches

# Example of how to use the function
# echo_info_1 = detect_echoes(squared_norm1, threshold=20) # Assume these functions are
    ↪ defined
# echo_info_2 = detect_echoes(squared_norm2, threshold=20)
max_distance = 30
matches = match_and_score_echoes(norm1, norm2, echo_info_1, echo_info_2, max_distance)

# Print the best matches
for match in matches[:10]: # Top 10 matches
    print(f"Match between {match[0]} (Dataset 1) and {match[1]} (Dataset 2) has a score of {
        ↪ match[4]}")
```

Appendix C

Arduino code

```
#define TRIG_PIN_1 9
#define ECHO_PIN_1 10
#define TRIG_PIN_2 11 // Change pin numbers as needed
#define ECHO_PIN_2 12 // Change pin numbers as needed
#define ANALOG_PIN_1 A0
#define ANALOG_PIN_2 A7 // Change analog pin number for the second sensor

#define MAXCOUNT 300

uint16_t adcvals_1[MAXCOUNT];
uint16_t adcvals_2[MAXCOUNT];
unsigned int count = 0;

void setup() {
  Serial.begin(115200);

  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(TRIG_PIN_1, OUTPUT);
  pinMode(ECHO_PIN_1, INPUT);
  pinMode(ANALOG_PIN_1, INPUT);

  pinMode(TRIG_PIN_2, OUTPUT);
  pinMode(ECHO_PIN_2, INPUT);
  pinMode(ANALOG_PIN_2, INPUT);
}

void loop() {
  // Read data from first sensor
  readSensorData(ANALOG_PIN_1, adcvals_1);

  // Send data from first sensor without a delimiter
  sendData(adcvals_1, MAXCOUNT);

  // Read data from second sensor
  readSensorData(ANALOG_PIN_2, adcvals_2);

  // Send data from second sensor without a delimiter
  sendData(adcvals_2, MAXCOUNT);

  // Delay between readings
  delay(1000);
}

void readSensorData(int analogPin, uint16_t adcvals[]) {
```

```
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(TRIG_PIN_1, HIGH);
  delayMicroseconds(500);
  digitalWrite(LED_BUILTIN, LOW);
  digitalWrite(TRIG_PIN_1, LOW);
  delayMicroseconds(1100);

  count = 0;
  while (count < MAXCOUNT) {
    adcvals[count++] = analogRead(analogPin);
  }
}

void sendData(uint16_t adcvals[], unsigned int maxcount) {
  digitalWrite(LED_BUILTIN, HIGH);
  for (int i = 0; i < maxcount; i++) {
    Serial.print(adcvals[i]);
    if (i < maxcount - 1) {
      Serial.print(' ');
    }
  }
  Serial.println();
  digitalWrite(LED_BUILTIN, LOW);
}
```