U N I V E R S I T Y O F C A P E T O W N

Department of Electrical Engineering

EEE3097S – Electrical and Computer Engineering Design Project 2019 Final Report

Group 7

Joshua Eybers EYBJOS001

Lawrence Godfrey GDFLAW001

Matthew Lock LCKMAT002

Shahil Poonsamy PNSSHA003

16 October 2019

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This project report is the work of the team members of my group.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Name 1: **Matthew Lock**

Signature:  **Date:** 16/10/2019

Name 2: **Lawrence Godfrey**

Signature:  **Date:** 16/10/2019

Name 3: **Joshua Eybers**

Signature:  **Date:** 16/10/2019

Name 4: **Shahil Poonsamy**

Signature:  **Date:** 16/10/2019

Contents

1. Introduction

Under the right conditions, a phenomenon can be observed where the sound of one's voice is heard repeated sometime after speaking. Intuitively we understand this phenomenon to be an echo, as the sound you have produced is reflected off surfaces in the nearby environment and travels back to its point of origin. In nature this effect is utilised to enable creatures to map and navigate their surroundings using echoes to determine proximity to nearby objects. Most famously, bats use a sonar system known as 'echolocation' in order to compensate for their weak eyesight by producing ultrasonic sound and listening for echoes. This phenomenon has been adopted by humans in order to create sound navigation and ranging systems (known as sonar) to navigate, communicate with or detect objects on or under the surface of a body of water.

1.1 Goals

The goal of this project is to:

1. Use our understanding of echoes, wave propagation, signal processing, analogue electronics, and embedded systems in order to build and test an imaging sonar capable of detecting and triangulating the horizontal position of objects up to 10 m, operating at 40 kHz in air.

1.2 Objectives

The project will have two different stages of complexity with the first being a one dimensional range profiling sonar capable of detecting the presence and distance of an object in a set space relative to the detector. This means that if more than one object is placed at the same distance within the space, we will not be able to distinguish the different objects. We will be able to distinguish between objects at different ranges. The second degree of complexity will be an angle finding sonar, capable of detecting the presence and distance of objects along a single plane relative to the detector.

Our objectives for this project will be to:

1. Simulate matched and inverse filtering
2. Design, test, and implement hardware for driving and receiving signals from an ultrasonic transducers.
3. Design, test, and implement data acquisition hardware
4. Design, test, and implement signal processing software for matched and inverse filtering
5. Implement a working sonar imaging system within the given time frame and achieve the second degree of complexity such that our system is capable of angle finding.

1.3 Limitations

Going forward there are a number of limitations that were being carefully managed. In order to successfully design and implement both stages of complexity, it is essential that these limitations are carefully considered to avoid going beyond the scope of the given task. These limitations include, but are not limited to :

- Time constraints : In and amongst other course work, it was essential to manage our time as to not spend too much time focusing on the sonar system to the detriment of academic performance. While achieving all deadlines, this limitation meant that adhering to deadlines was a difficult task and that deliverables only met the required specification. Furthermore with no reliable weekend access to laboratories for testing, development of the sonar system could only take place during the week.
- Budget constraints : The budget of R2000 was not an immense constraint but did place pressure on team members designing protectionary circuits. With the main development board costing more than a quarter of the budet, and damage to the development board could not be tolerated. System level testing as a result could only be completed once protectionary circuit had been tested and approved. This placed additional delay on the project.
- Skills : Having never worked on a project of this scope before, group members spent a considerable amount of time troubleshooting and learning new techniques for embedded systems and signal processing. Concepts like DMA, direct interface with registers and circular buffering had to be considered and researched.

1.4 Deliverables

The deliverables contained within this final report report include :

1. A summary of the project, its goals and its objectives.
2. A detailed plan showing how the objectives were achieved, the time frame they were achieved within, and whom was responsible for making sure each objective was met.
3. A list of updated technical and user specifications
4. Updated se case, sequence, activity, and block diagrams for the system 5. An identification of subsystems within the larger system, detailing the constraints, requirements, hardware, and software design of each subsystem.
5. Summary of different design approaches, with a focus on the strengths and weaknesses of each approach
6. Details pertaining to subsystem level testing and integration, as well as system level testing
7. Design of the final product
8. A list of all hardware and software components used
9. Member contribution report

2. Design

2.1 System Requirements

System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. Hence this section of the report will cover both high level specifications of what must be achievable by the system for the user, as well as a low level specification which details how well the system must be able to achieve these design parameters.

2.1. User requirements

The following table shows a list of user requirements for the system

User Requirement Number	Requirement for the System
UR1	Device should be able to connect by a laptop.
UR2	The sonar must not cost a lot of money.
UR3	The sonar should work in air.
UR4	The sonar should be able to calculate range.
UR5	The device should be able to differentiate between two close objects.
UR6	Device has to be able to display objects within a short range.
UR7	The device should be capable of detecting multiple objects.
UR8	The device should have a wide field of view.
UR10	The device should be capable 2D direction finding.

Test cases

1. Plug the device into a laptop. The device passes if it displays an image on the laptop .
2. Sum the cost of all of the components together. The device passes if the total cost is low.
3. Power up the device and plug it into a laptop in a room. Put an object right in front of the device. The device passes it it displays an object in front of it.
4. Power the device and put an object 4m away from the device's sensors. Observe the displayed result. The device passes the test if it displays an object at a range of 4m.
5. place two objects 15cm apart in front the device.Power up the device and aim it at the two objects. Observe the displayed image. The device passes if it correctly displays two different objects.
6. Plug the device into a laptop and put an object 5m in front of the object then Observe the displayed image. The device passing if it displays the object 5m in front of it on the laptop.
7. Place three objects in front of the device, one at 3m one at 6m and the last at 9m. The device passes it correctly displays three distinct objects.
8. Place an object at an angle of 40 degrees from the devices sensor then observe the image it displays. The device passes if it displays an object.
9. Place an object 4m away at an angle of 40 degrees from the devices sensor then observe the image it displays. The device passes if it displays an object at 4m with an angle of 40 degrees.

2.2 technical requirements

The following table shows a list of user requirements for the system

Technical requirement number	Technical requirement
TS1	The device should be able to transfer data to a laptop over a serial port.
TS2	The total cost of the device must not exceed R2000.
TS3	The device needs omit and detect ultrasonic signal in the range of 35 Khz to 45Khz that can travel in air.
TS4	The device requires a minimal field of view of 100 degrees.

TS5	The device needs a minimum object resolution of 10cm.
TS6	The device needs to be able to detect objects within 10m of its sensors
TS7	The device should be capable 2D direction finding with two receiving transducers using phase difference measurements.

Test cases

1. Plug the device into a laptop's usb port. The device passes if it sends data to the laptop via the usb port.
2. Sum the cost of all of the components together. The device passes if the total cost is below R2000.
3. Power up the device and plug it into a laptop in a room. Put a receiving transducer that is connected to an oscilloscope in front of the device.Transmit a chirp signal from the device. The device passes if the oscilloscope displays a waveform between 35Khz and 45 Khz.
4. Place an object at an angle of 50 degrees and another object at an angle of -50 degrees and observe the output. The device passes if it displays two objects, one at an angle of 50 degrees and one at an angle of -50 degrees.
5. place to objects 10cm apart 5m in front of the device. Power up the device and aim it at the two objects. Observe the displayed image. The device passes if it correctly displays two different objects.
6. Plug the device into a laptop and put an object 10m in front of the object then Observe the displayed image. The device passing if it displays the object 10m in front of it on the laptop.
7. Place an object at 4m with an angle of 30 degrees and another object at 7m with an angle of -50 degrees and observe the output. The device passes if it is using two transducers and if it displays two objects, one at 4m with an angle of 30 degrees and one at 7m with an angle of -50 degrees.

2.3. Use Case diagram

A use Case diagram shows the high level dynamic activity of a system. The diagram displays the user requirements of a specific system by displaying how actors and the system interact. It visualizes how you would interact with a system and how the system responds to those interactions.

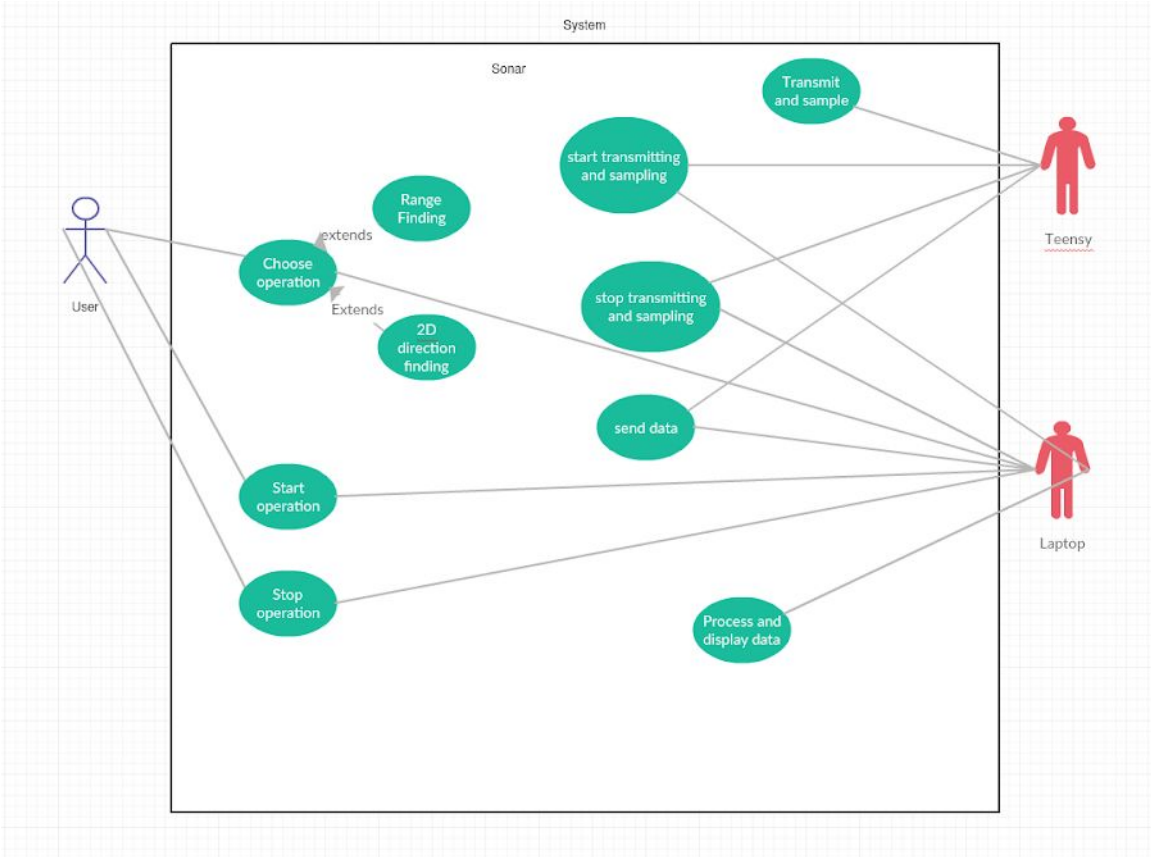


Figure 2.1: Use case diagram

In the above diagram there are three actors; the user, the laptop and the teensy board. The user can pick an operation type by interacting with the laptop which then passing on the information to the teesny board. The teensy board updates the data and sends the data to the laptop. The sent data is processed by the laptop before it is displayed on the laptop.The user can also start or stop the operation at anytime.

2.4 Sequence diagram

A sequence diagram shows interactive behavior of a system. It shows the interactions and the sequence/timing of the interactions between actors in the system. It achieves this by visually portraying the interactions in sequential order. Sequence diagrams also display the messages that one actor sends to another when they interact with one another. Sequence diagrams enable people to see how and when interactions in the system take place.

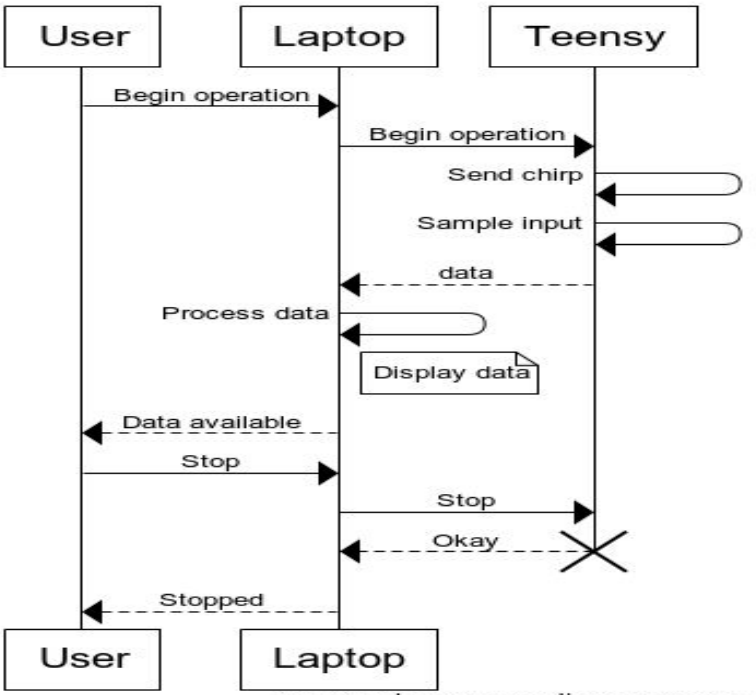


Figure 2.2: Sequence diagram

In the above sequence diagram The dotted lines represent return calls while the solid lines illustrate asynchronous messages.The return the operation begins with the user sending a message to the laptop. The laptop then sends a begin operation message to the Teensy board. The Teensy board then initiates the transmitting transducer with a message and then starts the receiving transducer with another message. The Teensy board then sends the data to the laptop. The Laptop processing the data and sends a message to the user to indicate that the data is available. The user sends a stop message to the laptop that then sends a stop message to the Teensy board that terminates after sending an okay message. The laptop then sends the user a message to indicate that operations have ceased.

2.5 Activity diagram

An activity diagram is a type of flowchart that represents the movement from one event to another within a system. These events describe how the system will operate. An activity diagram captures the dynamic operation of a system by displaying the sequential,branched or concurrent flow of events. The activity diagram differs from the sequential diagram by not displaying the messages sent between the different actors in the system.

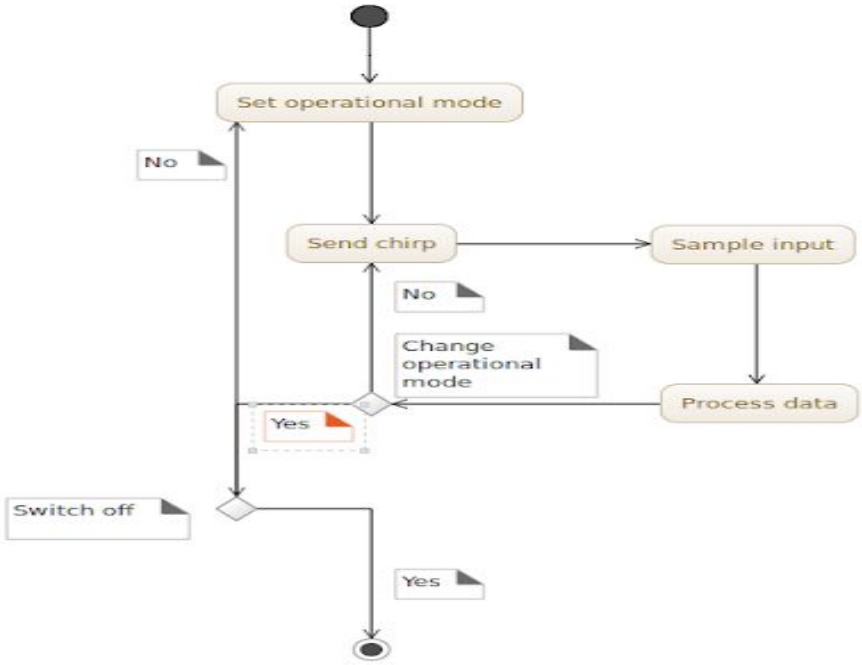


Figure 2.3:Activity Diagram

In the above diagram the activity of the system can be seen. The system is started and an operational mode is chosen,either 1D range finding or 2D direction finding . The sonar then emits a pulse, samples the response, process and displays the signal before checking if the operational mode has changed. If it hasn't received a message to change its operation then the system repeats the previous steps from sending the chirp pulse. If the answer is no, a check to see if the sonar has been switched off or if the operational mode is being changed occurs. If the latter occurs all the previous steps are repeated. If the former occurs the sonar switches off and operation is suspended.

3. Circuit Diagram

The following circuit diagram shows the receiving circuit used on both of the receiving circuits. The input from the receiving transducer produces a voltage. There is a resistor over the input terminals of the transducer to reduce noise. The amplification circuitry then amplifies the voltage with a gain of 55. A DC component is added and voltage is limited to 0 and 3.3V. A LPF is also used to reduce noise. The output is fed into one of the ADCs on the Teensy board.

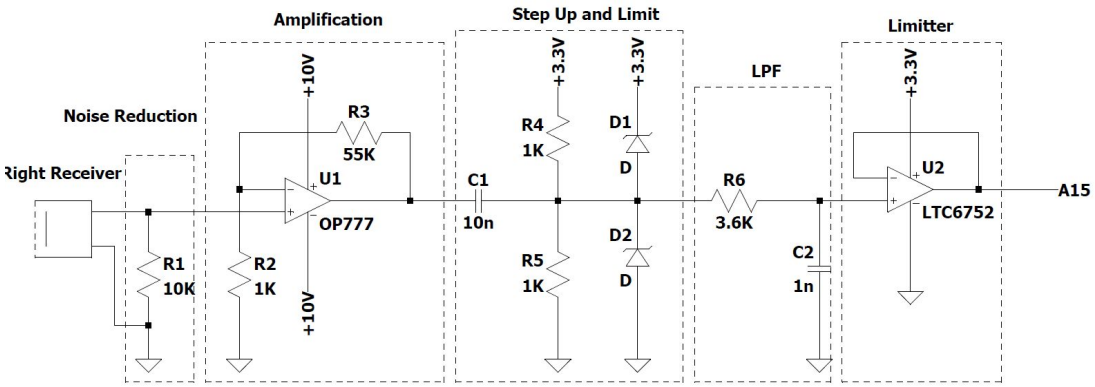


Figure 2.5: Circuit Diagram of Receiving Circuit

The following circuit diagram shows how the DAC output of the Teensy Board is converted to a voltage which can be used by the transmitter. DC blocking is implemented to remove the DC offset from the DAC output. The 3.3V pk-pk voltage is then amplified with a gain of 5.6 to get a pk-pk voltage around 19 V.

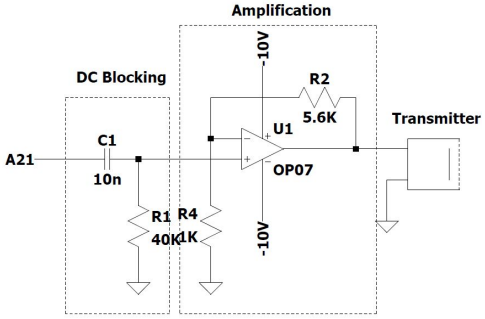


Figure 2.6: Circuit Diagram of Transmitting Circuit

4. Project plan

Task	Responsible	Target Date	Status
Test and document transducer characteristics	Lawrence, Shahil	Monday 5th August	Completed
Simulate matched and inverse filtering	Joshua, Lawrence, Matthew, Shahil	Monday 12th August	Completed
Build test circuits for driving transmit transducers	Lawrence, Matthew	Friday 16th August	Completed
Investigate and report on general working of DAC and ADC of teensy	Joshua	Friday 16th August	Completed
Output chirp pulse using DAC	Lawrence, Matthew	Wednesday 21st August	Completed
Build test circuits for receiving transducers	Joshua, Matthew	Friday 6th September	Completed
Sample receiver transducers using ADC	Lawrence, Matthew	Friday 6th September	Completed
Initiate serial transfer of data between microcontroller and Julia workbox	Matthew	Wednesday 11th September	Completed
Signal processing of one dimensional sonar imaging	Shahil	Friday 13th September	Completed
Build and test one dimensional sonar imaging system	Joshua, Lawrence, Matthew, Shahil	Friday 13th September	Completed
Investigate and report on the use of digital pins to produce chirp pulse	Joshua	Monday 16th September	Completed
Investigate and report on the use of simultaneously using two DMA channels	Joshua	Wednesday 18th September	Completed
Implement reception of chirp through two transducers	Lawrence, Shahil	Friday 20th September	Completed

Implement reception of chirp through two transducers using software	Joshua, Matthew	Friday 20th September	Completed
Implement Signal Processing for extraction of phase difference	Joshua, Matthew	Friday 4th October	Completed
CAD design and laser cutting of simple enclosure for the system	Matthew, Lawrence	Tuesday 8th October	Completed
Build and test one dimensional sonar imaging system with angle finding capabilities	Joshua, Lawrence, Matthew, Shahil	Thursday 10th October	Completed

From the above project plan, it can be seen that the original project plan has changed over time since the scope of the project has changed. Tasks relating to the implementation of a two dimensional sonar imaging system were scrapped and replaced with a one dimensional system with angle finding capabilities. The project has been split into smaller tasks that have been assigned to different group members. Since the most essential tasks will be performed sequentially, tasks have been split between members as modestly as possible. This has been done not only to draw on individual strengths of each member, but also to evenly split the workload and allow members breaks between deliverables to focus on other tasks. Hence the timeline for each task began on the target date of the previous task. This can be found in ***Appendix A*** which shows a more intuitive workflow by means of a gantt chart.

Furthermore it can be seen that all tasks were completed completed according to schedule. These tasks do not include the disregarded tasks that were deemed out of scope. A break down of these completed tasks can be found in ***Appendix B*** which includes segments of the report on transducer characteristics, simulated matched and inverse filtering, and extracts of code used for angle finding.

5. Block diagram of the system

This block diagram will use schematics to show the general high-level arrangement and interconnection of subsystems within this system.

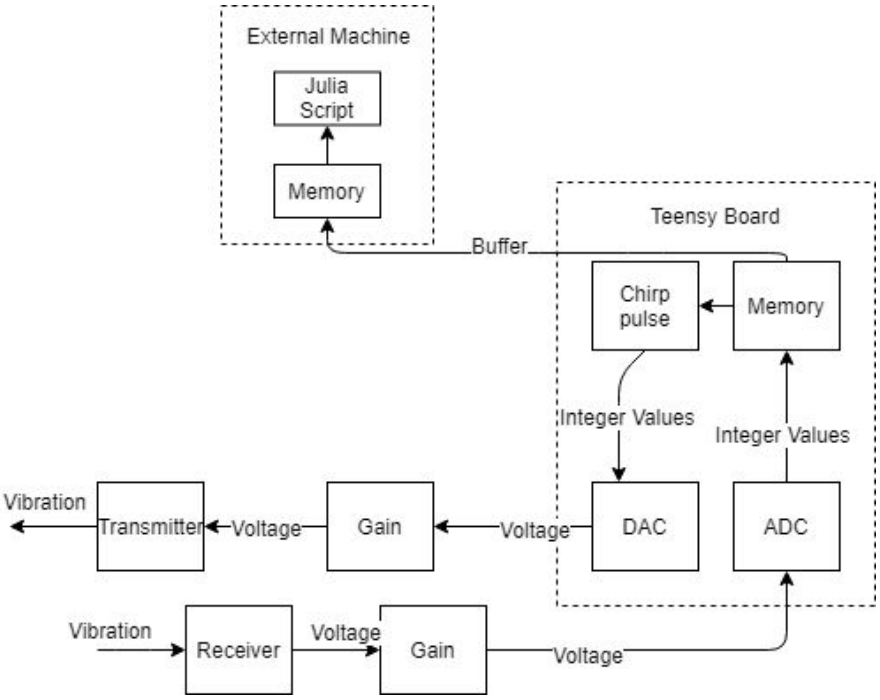


Figure 5.1: Block diagram of system.

This diagram shows the parts of the teensy board we will be using, namely, the code which produces the chirp pulse, the DAC which converts the integer values into a voltage, the ADC which converts a voltage to an integer value and the memory which holds the received waveform which is sent to an external machine as a buffer.

The voltage sent out by the DAC is amplified and then transmitted. The received waveform is also amplified and fed into the ADC.

6. Design approaches

We could have decided to do signal processing on the Teensy board itself and then sending the processed signal to a computer for plotting instead of using Julia for all the post processing. This could have made the processing faster since it would be in C++ and on a fairly fast microcontroller, however, doing signal processing in C++ is harder due to its low-level nature and lack of signal processing functions. We decided to use Julia since it is fast enough and much easier to troubleshoot and modify than C++ code.

7. Sub-system requirements

1. We will be using a Teensy 3.6 board to produce the 40 kHz pulse. This board requires a USB connection to supply power to it. A simple C++ program is required to generate a chirp pulse at the correct frequency. A table of values for the chirp pulse is also used to increase speed.
2. The voltage produced by the Teensy board is low and requires amplification with a gain of 5.6. The DC offset also needs to be removed. The amplification circuitry requires its own ± 15 V DC supply voltage in order to supply the amplified voltage to the transducer.
3. To transmit the chirp pulse we will use a piezoelectric transducer.
4. We are using a piezoelectric transducer to receive waveforms.
5. We need amplification circuitry to amplify the received voltage, as well as level-shifting circuitry to create a DC offset since the Teensy board can only take in positive voltages. We will also use clamping circuitry to ensure the voltage can't exceed 3.3V or go below 0V.
6. The Teensy board has an on-board ADC to convert the voltage into an analogue waveform.
7. A program on the Teensy board will sample the signal at a certain rate, save a certain number of samples to a buffer and send the buffer to another computer through a serial bus.
8. Some of the steps required for signal processing are outlined below:
    - Fourier transform received waveform.
    - Filter the received waveform using a matched filter which is matched to the waveform we produced.
    - Create an analytic signal.
    - Compensate for range dependance.
    - Calculate the angle of an object based on the time difference between the two received signals
    - Plot the angles with ambiguities

8 Logical Integration of Subsystems and Testing

In order to realise a fully working system, it was essential to implement the sonar imaging system by integrating the subsystems incrementally. This approach was taken as it promised to reduce time spent troubleshooting. Had the subsystems been integrated wholly in one step, troubleshooting and narrowing the scope of an error might have been a tedious and unnecessary tasks. With this in mind, all systems were designed and implemented with full awareness of the capabilities of other subsystems. This was done to ensure that means of communicating between subsystems was predetiremed, that connects could be made seemingly between the subsystems, and that all inputs and outputs were valid between systems.

The three main subsystems chosen for integration testing were:

1. Transmission circuitry and transducer
2. Receiving circuitry and transducers
3. Teensy 3.6 chirp transmission, Reception, and serial data output

It was observed that there was no direct connection or communication between transmission and receiving circuit. Rather these systems worked independently of one another, instead interfacing directly with the Teensy 3.6. For this reason it was decided to only run three integration tests. These tests would validate the operation of the subsystems, as well as validate successful integration with the final sonar imaging system

8.1 Chirp Pulse Transmission Testing

In order to validate that the system was successfully able to generate a suitable chirp pulse via one the DAC output pins, a validation test was run. This test involved probing of the Teensy 3.6 output pin and probing of the final output after amplification. The test procedure, as well as expected outcomes and final results can be found below Note tests were carried out sequentially and could not advance until test results proved successful.

Procedure for testing chirp output:

1. Create serial connection between Teensy and the Julia environment using SerialPorts
2. Serial write the value “t” to the teensy and capture the output of pin A21 using an oscilloscope. Save a screenshot of this recorded data for later analysis.
3. Connect the common ground between the Teensy 3.6 and the chirp amplification circuitry.
4. Connect pin A21 of the Teensy to the input of the amplification circuitry.
5. Once again serial write the value “t” to the teensy and capture the output of pin A21 using an oscilloscope. Save a screenshot of this recorded data for later analysis.

Condition Tested	Expected Results	Final Outcome
Teensy 3.6 chirp output	● Chirp pulse output to pin A21 of Teensy 3.6 ● Offset of 1.6 V ● Peak to peak amplitude of 3.3 V ● Bandwidth of 2 kHz ● Centre frequency of 41 kHz ● Chirp length of 4 ms	The Teensy 3.6 was able to produce a chirp pulse on the appropriate pin. The chirp pulse was found to have the correct offset, amplitude, bandwidth, centre frequency and chirp length
Amplification of chirp output	● Replication of the chirp pulse produced by the Teensy, but with a gain such that the peak amplitude of the output signal is 20 V	The amplification successfully managed to amplify the chirp pulse to achieve a peak to peak amplitude of 19.8 V

Procedure for transducer sampling:

1. Connect the common ground between the Teensy 3.6 and the receiving transducer circuit.
2. Connect pin A14 of the Teensy to the output of one of the receivers
3. Connect pin A16 of the Teensy to the output to the second receiver
4. Create serial connection between Teensy and the Julia environment using SerialPorts
5. Serial write the value “t” to the teensy and capture the output of both receivers using an oscilloscope. Save a screenshot of this recorded data for later analysis.
6. Serial write the value “p” to print the values stored within the first buffer to the Julia environment. Extract the data points and create an array for the received voltage waveform.
7. Serial write the value “o” to print the values stored within the second buffer to the Julia environment. Extract the data points and create an array for the received voltage waveform.

Condition Tested	Expected Results	Final Outcome
Dual channel transducer sampling	● Reception of two voltage waveform, identical to those sampled using the oscilloscope. The oscilloscope can be used to verify that there is a phase shift in the received signal.	The Teensy 3.6 was able to successfully sample two receiving transducers using DMA. These signals were shown to be relatively noise free and almost identical to the recorded results.

8.2 Testing Circuit

When implementing the circuits shown in Figures 2.5 and 2.6, we first built each sub-system indicated by the dotted boxes. We then fed in an expected input (eg. a sine wave at 40kHz from the signal generator) and checked that the output of the subsystem was as expected using an oscilloscope. After testing all the subsystem we connected them together to create the final circuits, and then tested these circuits.

We tested that our system could detect multiple objects at different angles by using corner reflectors (three reflective surfaces which are mounted crosswise [Figure 10.3]) at known distances and angles and checking that our plot output was as expected.

9. System level testing

Once all the subsystems were tested, validated and integrated into the final system, system level testing would commence to validate the functioning of the system as a whole. Test failures at this point would prove extremely expensive and would require revision of earlier project steps. The procedure for system level testing involved the use of the following components:

- Teensy board with accompanying circuitry for receiving and transmitting transducers
- Power supply (Used two 9v batteries in the final implementation)
- Laptop for image processing
- Corner reflectors

Once everything was connected, the testing consisted of verifying that the image produced matched expectations of the reality of the environment. Where subsystem testing was very technical and involved calculation and a strict validation loop of test-correct-test, the system level tests were largely speculative; if the result or final image produced did not appear as expected, very little could be done to identify and correct at which subsystem a problem had occurred or whether the problem lies in the premise of the system as a whole. To mitigate this ‘black-box’ effect, waveforms received at varying subsystem steps were shown alongside the final 2D image, in order to identify issues between receiving the signal and processing an image, as well as further validate that the final output not only matched expectations of the environment layout and position of the corner reflector, but also correctly translated the received transducer pulses into base-banded, filtered signals. The final interface for this stage is shown below.

The filtered signal (second waveform from the bottom) shows peaks which indicate where objects are detected, and a hard-coded threshold displays the peaks which cross this it in the bottom waveform. Two main tests were conducted, which both had the same set up and validation procedure. The first test was to move a corner reflector, placed directly in front of the transducers, further back until a maximum distance of 10m. The test would be passed if the 2D

image correctly displayed the location of the detected object. In testing, it was seen that objects stopped appearing in the 2D image beyond a distance of around 7m, but would still appear as peaks in the matched filter waveform, lending to the idea the inference that the peak threshold value was too low. However, lowering the threshold to a low enough value would cause noise to appear as detected objects, which is incorrect, resulting in the conclusion that the received signal was too weak to be properly detected beyond this point, and a possible revision of the amplification process would be required. In the end, the test can still be considered to be passed, because the system correctly located the distance of objects up to an effective range. A deviation of this test was to place the detector in front of a wall, to pick up hard boundaries, which was also passed.

The second test consisted of moving the corner reflector horizontally at different ranges from the transducers, and would be passed if the 2D image correctly showed the object moving off-center. This test was passed with the acknowledgement that only two receiving transducers were used instead of an array, thus making the system capable of horizontal location detection, but not effective at it.

The third test consisted of two objects being used. The test would pass if the system could detect both objects independently.

The images below show examples of the procedures used for testing (with a ceramic cup in place of where a corner reflector would normally be used):

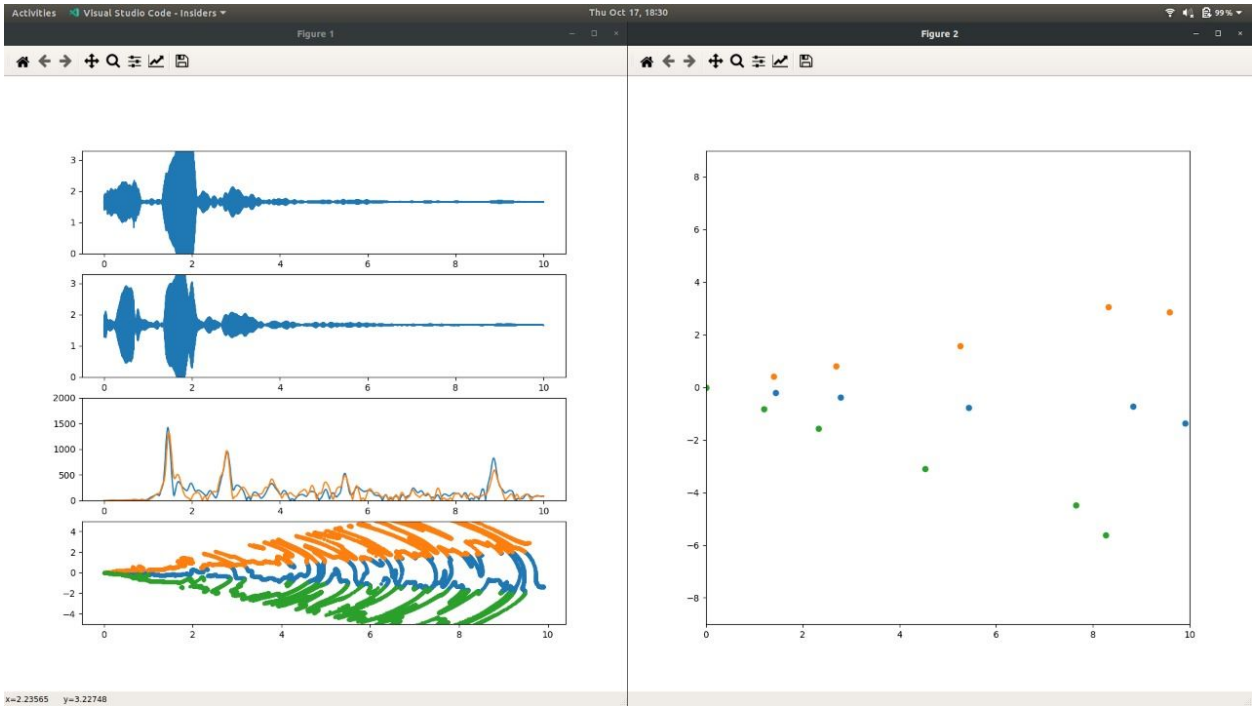


Figure 9.1: Test 1A - Placed in front of a wall

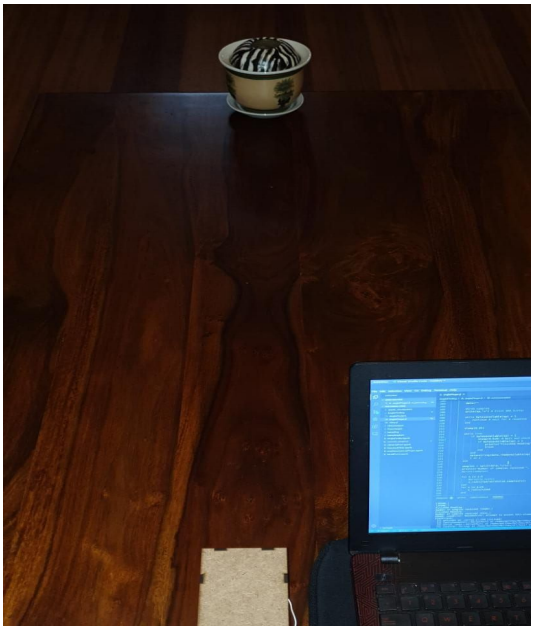


Figure 9.2: Setup of Test 1B-Object placed in front

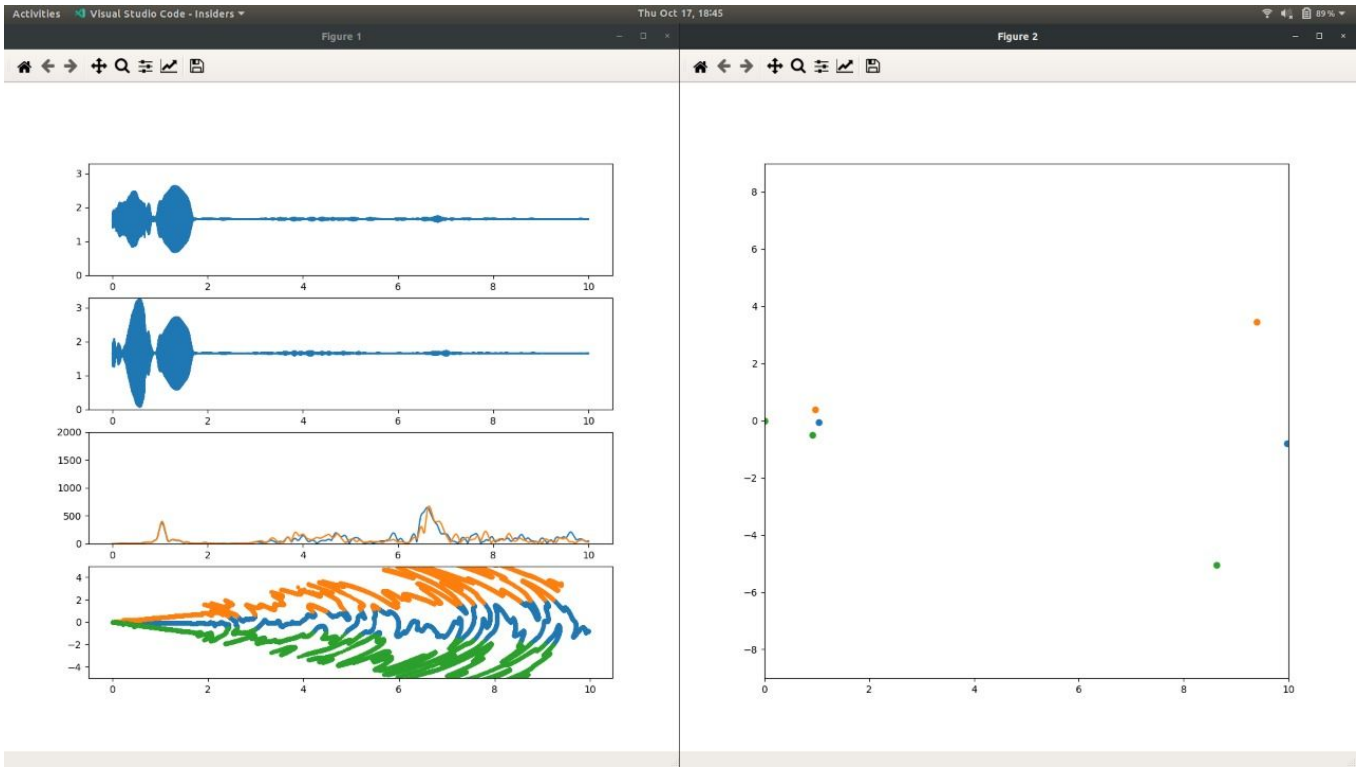


Figure 9.3: Results of Test 1B-Object placed in front

For test 1B, only targets at a range of less than 3m were considered. Therefore any targets above the threshold but outside this range were disregarded. An example is shown in figure 9.3 were the reflection from the wall was disregarded while the object was acquired at 1 m. This can be seen again in tests 2 and 3.

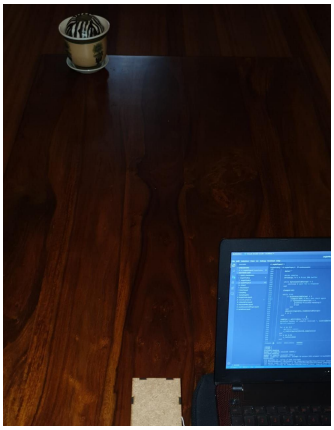


Figure 9.4: Setup of Test 2-Object placed at an angle

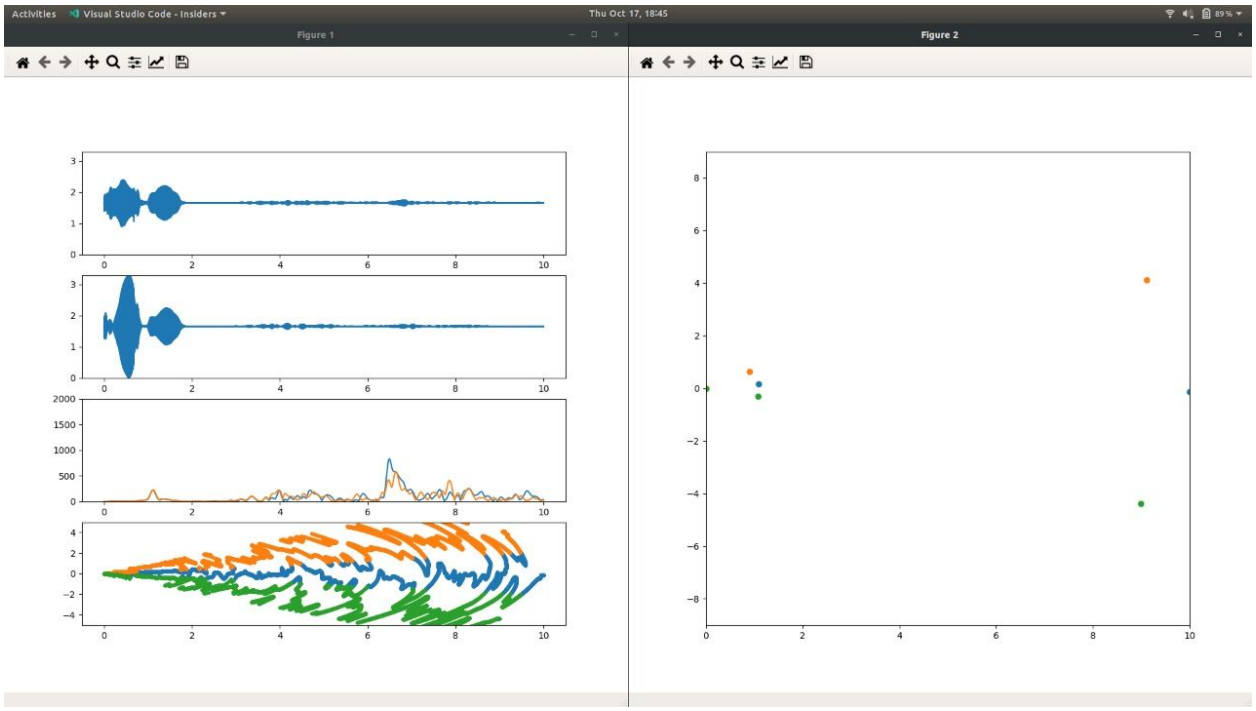


Figure 9.5: Results of Test 2-Object placed at an angle

From figures 9.4 and 9.5 it can be seen that initial system level tests were successful. Figure 9.5 shows that as the object moved from its original position in test 1B, the appropriate location in space was acquired. Target acquisition and angle finding was this concluded to be successful.

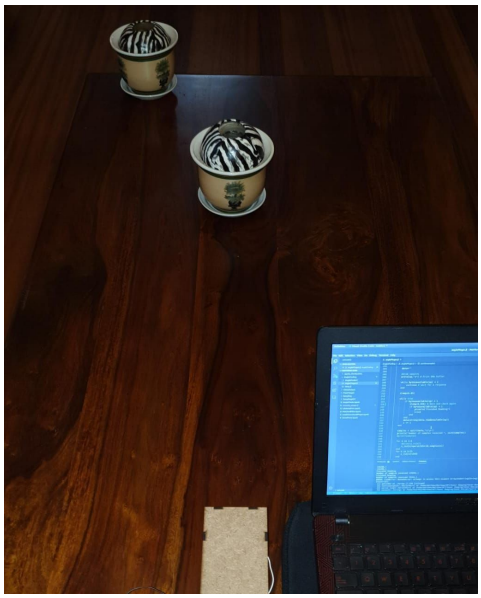


Figure 9.6: Setup of Test 3- Two objects

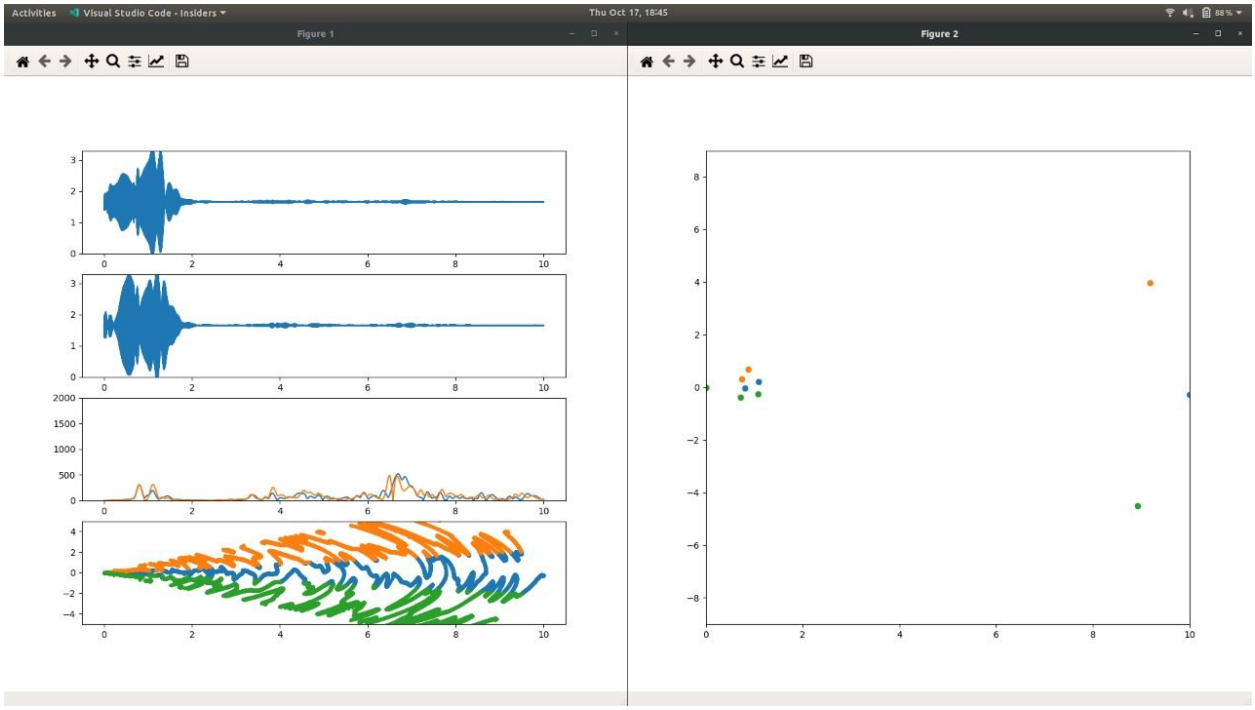


Figure 9.7: Results of Test 3- Two objects

Figure 9.7 shows that system level testing as a whole was successful. The system was able to perform target acquisition, angle dinding, and do so for multiple targets with a resolution of 10 cm.

10. Final Implementation

We first implemented the circuit using multiple breadboards. This allowed us to easily troubleshoot and test the sub-circuits and the system as a whole. Once we were happy with the functionality of the system we moved the circuits from the breadboards onto a veroboard which sits inside a box and uses two 9V batteries as a supply.

This allows for a much less cluttered circuit with a lower chance of cables disconnecting or short-circuits from occurring. The box also held the transducers firmly in place which gave a much more consistent readings.

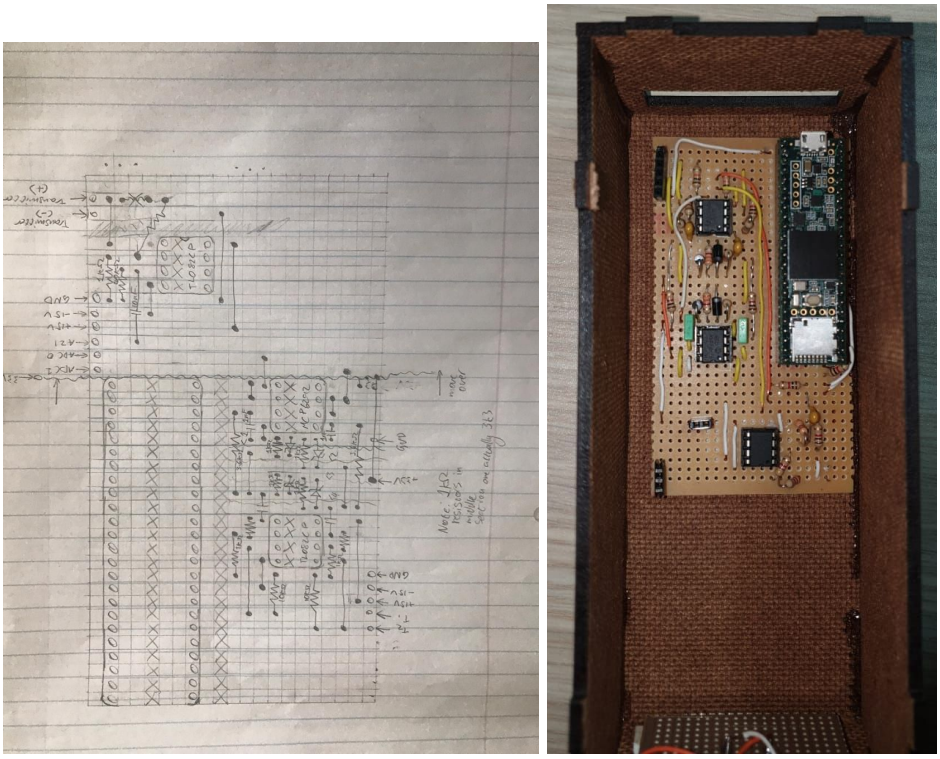


Figure 10.1: Veroboard design and implementation

We soldered the transducers (two receivers and one transmitter) onto a seperate veroboard which attaches to the box. We used a triangle formation with the transducers to try and minimise the distance between the receivers and have the transmitter equidistant from the receivers.

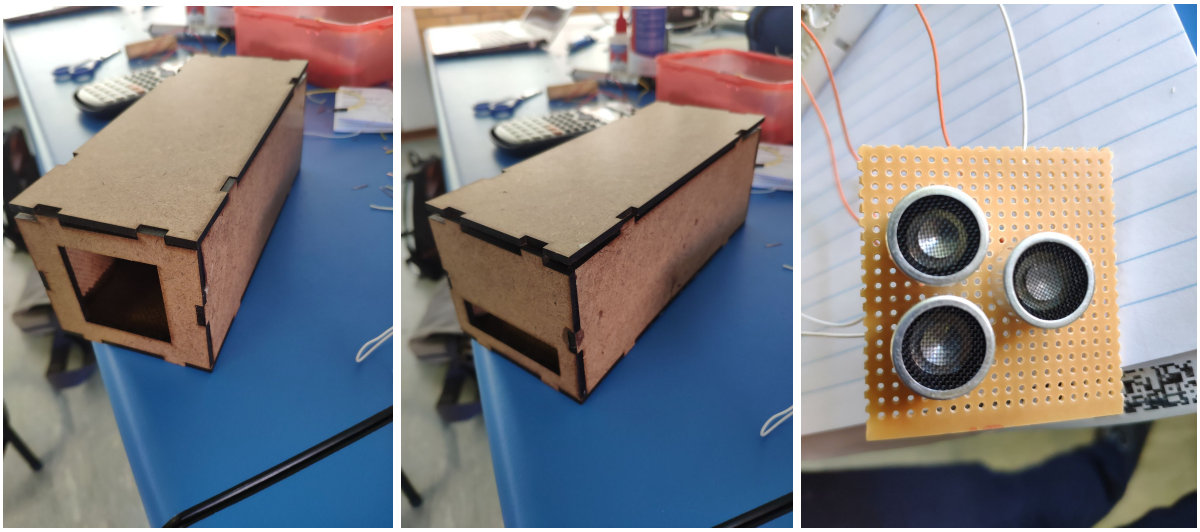


Figure 10.2: Implementation of box and transducer mount

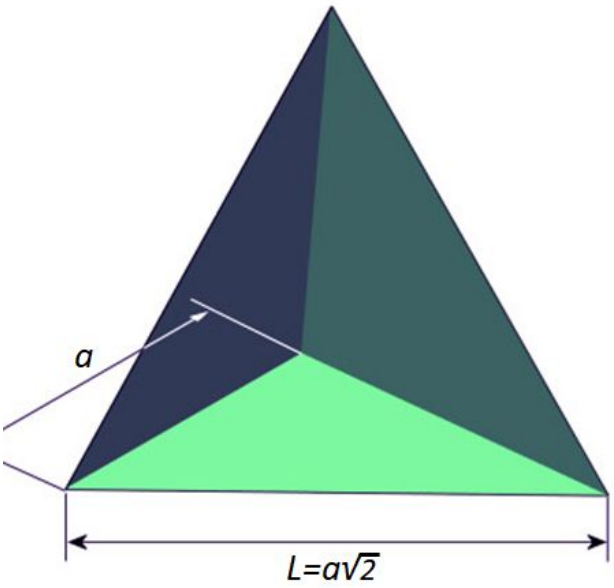


Figure 10.3: Corner Reflector

11. Bill of Materials

The listed bill of materials shows a final list of all raw materials and components needed to realise the final system. This bill of materials does not account for expenses incurred for tools used in the implementation of the system, but rather exclusively focuses on the material found within the final implementation of the product.

System	Component	Price
Transmitter	Transmitting Transducer x1	*R 10.00*
	LF351 x1	*R 80.00*
	10 nF Cape x1	-
	40 kΩ Resistor x1	-
	5 kΩ Resistor x1	-
Receiver	Receiving Transducer x2	R 20.00
	LF351 x2	*R 160.00*
	10 nF Cape x4	-
	1 kΩ Resistor x4	-
	3.6 kΩ Resistor x2	-
	Schottky Diodes x4	-
Power	9V battery x3	*R180.00*
Signal Processing	Teensy 3.6	*R 600.00*
	Laptop with Julia Environment	-
Total Price		*R 1050.00*

12. Summary of member contribution

The tasks in this project were divided up between group members to split up the workload. This proved to be an ineffective strategy and members worked better with the help of the whole team. This resulted in everyone working a little bit on every task. However, certain people were responsible for, or worked more on certain areas. These are summarised as follows:

Member	Responsibility
Matthew	● Image processing ● Julia code ● Subsystem testing
Lawrence	● Julia code ● Documentation ● Subsystem testing
Joshua	● Circuit design and testing ● Julia code
Shahil	● Circuit design and testing ● Documentation

Conclusions and Recommendations

The design process and implementation of the sonar system was deemed a success. From the section describing system level testing, it is seen that the final product managed to meet the final specifications as laid out by Associate Professor A.J Wilkinson. This is as we achieved an angular resolution of roughly 11 degrees which is close to specification, and a range resolution of 6 cm. Although the range resolution was slightly larger than expected for a chirp pulse with a bandwidth of 4 kHz (4 cm), we can attribute this to the passband of the transducers resulting in the full bandwidth not being transmitted. Although the system was successfully run on four 9 V batteries, time constraints prevented the team from placing switches between the circuitry and the batteries. To prevent drain of the batteries they are only connected when in use. This could be annoying for the user.

Future improvements that could be made include producing a more accurate chirp pulse by matching the DAC output rate as closely to the simulated sampling rate. This would allow for matched filtering to be centered around a simulated pulse as opposed to one that is directly recorded. Since we were not able to place transducers at a distance smaller than the wavelength of the centre frequency, more could have been done to account for the ambiguity in the angle finding. And finally the speed of serial transfer could have been significantly improved by changing the serial protocol. As opposed to sending out values character by character, sending out values as a series of two bytes could have reduced transfer speed by a factor of 2.5 and allowed for faster updating of plots. This would have realised a more fluid sonar imaging system.

The project as a whole was considered to be a success. Although there was confusion regarding deadlines and the required specifications for final and intermediate deliverables, the project was found to be enjoyable and met with enthusiasm by our members. We learnt a great number of skills in our endeavour to produce a working product and will definitely take these lessons and apply them to future design problems.

List of References

Teacher.scholastic.com. (2019). *Science Explorations: Journey Into Space: Radar and Sonar | Scholastic.com*. [online] Available at: http://teacher.scholastic.com/activities/explorations/bats/libraryarticle.asp?ItemID=234&SubjectID=110&categoryID=3 [Accessed 1 Sep. 2019].

Teacher.scholastic.com. (2019). *Science Explorations: Journey Into Space: Radar and Sonar | Scholastic.com*. [online] Available at: http://teacher.scholastic.com/activities/explorations/bats/libraryarticle.asp?ItemID=234&SubjectID=110&categoryID=3 [Accessed 1 Sep. 2019].

ElectronicsTutorials. (2019). [online] Available at: https://www.electronics-tutorials.ws/diode/diode-clipping-circuits.html [Accessed 2 Sep. 2019].

Whitaker, John. “Johnowhitaker/teensy_adc_tips.” *GitHub*, github.com/johnowhitaker/teensy_adc_tips.

Appendix A

Appendix contains n updateda gantt chart which shows the general flow of work between the two phases (complexity stage one and two) and provides an associated timeline and responsible group member for each task.

GANTT CHART FOR SONAR IMAGING PROJECT

TASK NAME	START DATE (M/D)	END DATE (M/D)	DURATION (WORK DAYS)	TEAM MEMBERS	PERCENT COMPLETE
Level One Complexity Phase					
Test and document transducer characteristics	8/5	8/5	0	Lawrence, Shahil	100%
Simulate matched and inverse filtering	8/5	8/12	5	Joshua, Lawrence, Matthew, Shahil	100%
Investigate and report on general working of DAC and ADC of teensy	8/12	8/16	4	Joshua	100%
Build test circuits for driving transmit transducers	8/12	8/16	4	Lawrence, Matthew	100%
Output chirp pulse using DAC	8/16	8/21	4	Lawrence, Matthew	100%
Build test circuits for receiving transducers	8/21	9/6	13	Joshua, Matthew	100%
Sample receiver transducers using ADC	8/21	9/6	13	Lawrence, Matthew	100%
Initiate serial transfer of data between microcontroller and Julia workbox	9/6	9/11	4	Matthew	100%
Signal processing of one dimensional sonar imaging	9/11	9/13	3	Shahil	100%
Level Two Complexity Phase					
Investigate and report on the use of digital pins to produce chirp pulse	9/13	9/16	2	Joshua	100%
Investigate and report on the use of simultaneously using two DMA channels	9/16	9/18	2	Lawrence, Matthew	100%
Build and test one dimensional sonar imaging system using chirp from digital output pins	9/18	9/18	1	Matthew	0%
Implement reception of chirp through two transducers	9/18	9/20	3	Lawrence, Shahil	100%
Implement reception of chirp through two transducers using software	9/18	9/20	3	Lawrence, Shahil	100%
Implement Signal Processing for extraction of phase difference	9/18	10/7	14	Joshua, Matthew	100%
CAD design and laser cutting of simple enclosure for the system	9/18	10/8	15	Joshua, Matthew	100%
Build and test one dimensional sonar imaging system with angle finding capabilities	9/18	10/10	17	Joshua, Lawrence, Matthew, Shahil	100%

Appendix B

Transducer Characteristics Report

Receiver voltage as a function of transmitter voltage

Voltage into transmitter (40kHz, 1.65m) pk-pk	Voltage at receiver
1V	28 mV
5V	140 mV
10V	252 mV
15V	364 mV
20V	423 mV

Receiver voltage as a function of distance

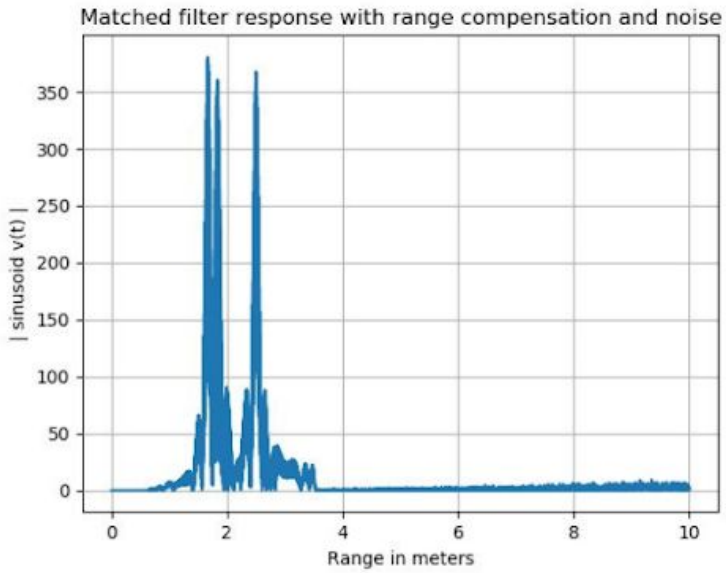
Distance between receivers (40kHz, 10V pk-pk)	Voltage at receiver
0.1 m	3 V
0.2 m (specified minimum)	2 V
0.5 m	700 mV
1 m	300 mV
2 m	282 mV
3 m	50 mV
4 m (Specified Max)	40 mV
5 m	20 mV

Receiver voltage as a function of angle

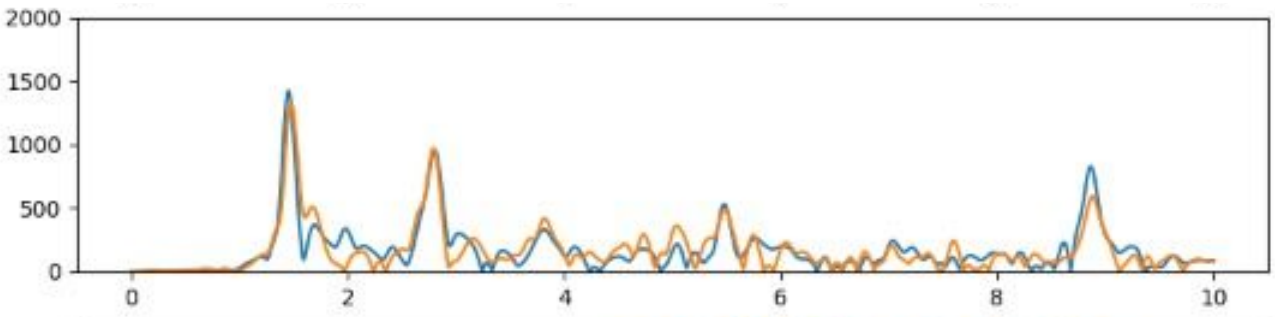
Angle of receiver from line of sight	Voltage at receiver
0 degrees	600 mv pk-pk 40cm
15 degrees	600 mv, 600 mV
60 degrees	200 mv, 150 mV
70 degrees	160mv, 140 mv
0 degrees	1.2 V at table level
23 degrees (half beamwidth)	920 mv

Appendix C

Matched filtering



Simulated matched filter response with three targets



Implementation of matched filter filter

The implemented matched filter was transducer created by using coaxial cables to point a transmitting transducer directly at the receiver. A chirp pulse was then recorded and a matched filter was created around that. The figure showing implementation of the matched filter shows the matched filter response of a chirp pulse bouncing of a wall. The repeated responses at regular intervals could be the signal bouncing back and forth while other responses could be erroneous echoes from a noisy (cluttered) environment.

Appendix D

Chirp pulse transmission and reception

Using the hardware outlined by section six of this report, we were successfully able to transmit a chirp pulse through a transducer. This was achieved by using taking the chirp signal array produced in our match and inverse filter simulations, and outputting that using DAC0 of the teensy board. Small changes were made to this array (such as sample frequency, converting voltage values to bit outputs, making sure peak-to-peak amplitude did not exceed 3.2 V, adding a DC bias of 1.65 V, and removing zeros before and after the chirp) in order to compensate for the speed at which the DAC could output, as well as adjust for 0 - 3.3 V capabilities of teensy. A segmented version of the code used to do this can be seen below

Julia code for producing array of values:

```
using PyPlot
c = 343; # Speed of sound in air in m/s
fs = 342*44100; # This is the sample rate of the sonar.
dt = 1/fs; # This is the sample spacing
r_max = 4; # Maximum range in metres to which to simulate.
t_max = 2*r_max/c; # Time delay to max range

t = collect(0:dt:t_max); # t=0:dt:t_max defines a "range".
r = c*t/2;

f0 = 40000; # Centre frequency is 10 kHz
B = 2000; # Chirp bandwidth
T = 4E-3; # Chirp pulse length
K = B/T; # Chirp rate

rect(t) = (abs.(t) .<= 0.5)*1.0
td = 0.6*T; # Chirp delay

v_tx = sin.( 2*pi*(f0*(t .- td) + 0.5*K*(t .- td).^2) ) .* rect.((t .-
td)/T);
print(round.(Int,v_tx .* 0.94*((2^(12))/2) .+
0.94*((2^(12))/2))[7991:66364])
```

Appendix E

C++ code to generate pulse:

```
#include "chirpout.h"
#define DAC0(a) *(volatile int16_t *)&(DAC0_DAT0L)=a
void setup() {
  analogWriteResolution(12);
  extern volatile uint16_t chirp[58374];

  SIM_SCGC2 |= SIM_SCGC2_DAC0; // enable DAC clock
  DAC0_C0 = DAC_C0_DACEN | DAC_C0_DACRFS; // enable the DAC module, 3.3V
reference

   while (1){
      for (int i = 0; i < 58374; i++){
        DAC0(chirp[i]);
      }
      delay(50);
   }
}
void loop(){
}
```

This chirp pulse was then passed through the subsystems identified in this report to remove the DC bias, and amplify the peak-to-peak voltage to 20 V to drive a transmitting transducer. A rudimentary test was then done to capture the transmission of placing the transmitting receiver on a table pointing upwards, with a receiving transducer facing the same direction next to it. The receiving transducer had 1 kΩ resistor placed across to to account for noise.

Appendix F

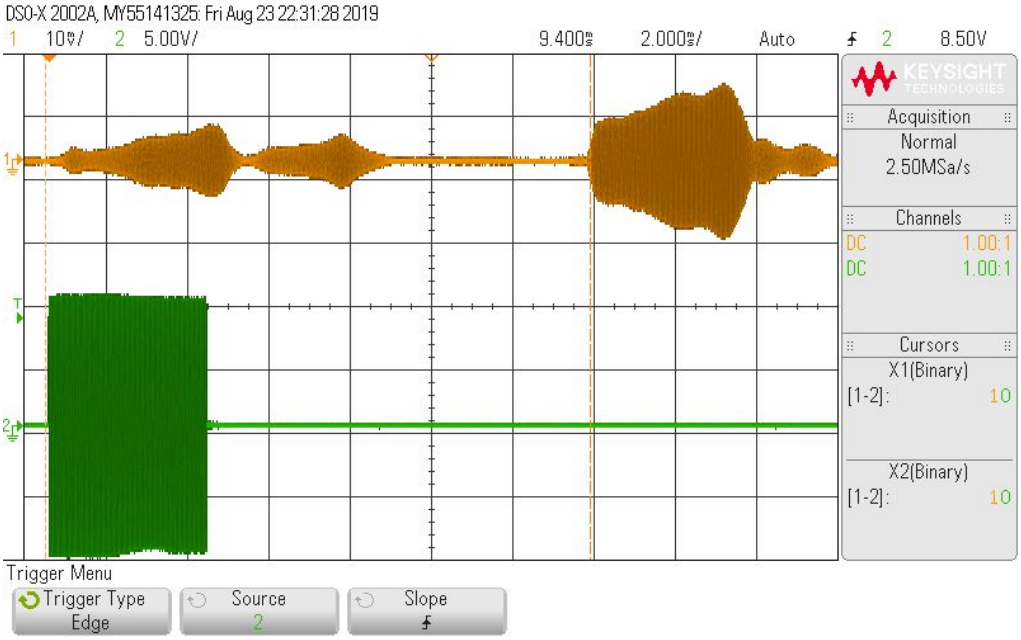


Figure A.1 : Transmitted and received chirp pulse

Figure A.1 is a screenshot of the transmitted and received chirp pulse. The green trace shows the voltage signal across the transmitting transducer while the brown trace shows the voltage signal across the receiving transducer. From this figure we can see that we successfully produced a 20V peak-to-peak chirp signal across the transmitting transducer and were successfully able to capture echoes of the chirp signal. The response on the far right can be recognised as the echo of the chirp signal off the ceiling.

Appendix G

Signal Processing

The following excerpt of code is that of the Julia code that was run in order to simultaneously sample two receiving transducers. This sampling was accplmoplised by sending a serial command to the teensy, which in tern transmitted a chirp pulse, and simultaneously sampled different ADC allocated pins using two direct memory access object. The two samples were stored in two different buffers, for which each could be called to print to serial independently.

```
x_rx1=zeros(Int16,CS) # Samples values received
x_rx2=zeros(Int16,CS) # Samples values received
data=""

# Clear buffer
while (bytesavailable(sp)>0)
    readavailable(sp)
end

# Transmit and Sample command
write(sp,"t")

while bytesavailable(sp) < 1
    continue # wait for a response
end
sleep(0.05) # This extra delay helps with reliability - it gives the micro time to send all
it needs to

timeString=readavailable(sp)
timeArray=split(timeString,"\n")

# Get timing information
time = parse(UInt16,timeArray[1])
timeSeconds = time*10^-6
println("Time taken to sample ",timeSeconds, " s")

timeBetweenTransmitAndRecieve = parse(UInt16,timeArray[2])
println("Time between transmitting and receiving ",timeBetweenTransmitAndRecieve, " us")

# Clear buffer
while (bytesavailable(sp)>0)
    readavailable(sp)
end

#Grab samples
```

```
write(sp,"p") # Print DMA buffer 1

while bytesavailable(sp) < 1
    continue # wait for a response
end

sleep(0.05)

while true
    if bytesavailable(sp) < 1
        sleep(0.010) # Wait and check again
        if bytesavailable(sp) < 1
            println("Finished Reading")
            break
        end
    end
    data=string(data,readavailable(sp))
    i += 1
end

samples = split(data,"\r\n")
println("Number of samples received ", size(samples))

for n in 1:S  # Add extra points to account for wrap around
    x_rx1[n]=parse(UInt16,samples[n])
end
for n in S:CS
    x_rx1[n]=2048
end

# Convert ADC output to voltage
v_rx1=(x_rx1.*(3.3/(2^12)))

data=""
#Grab samples
write(sp,"o") # Print DMA buffer 2

while bytesavailable(sp) < 1
    continue # wait for a response
end

sleep(0.05)

while true
    if bytesavailable(sp) < 1
        sleep(0.010) # Wait and check again
        if bytesavailable(sp) < 1
            println("Finished Reading")
            break
        end
```

```
    end
    data=string(data,readavailable(sp))
    i += 1
end

samples = split(data,"\r\n")
println("Number of samples received ", size(samples))

for n in 1:S # Add extra points to account for wrap around
    x_rx2[n]=parse(UInt16,samples[n])
end
for n in S:CS
    x_rx2[n]=2048
End

v_rx2=(x_rx2.*(3.3/(2^12)))
```

The next excerpt of code of the Jullia code that was run in extract the phase information from the received signal. This is shown for one of the received channels, while it has been assumed that the same process was followed for the secondary channel. Finally the algorithm run to find the angle of the target relative to the receivers is shown.

```
# Baseband signal 1
V_RX1 = fft(v_rx1)
V_RX1 = V_RX1 .* H
V_MF1 = MF[1:CS].*V_RX1[1:CS];
v_mf1 = ifft(V_MF1[1:CS]);
v_mf1=v_mf1.*r[1:CS].*r[1:CS]

V_ANAL1 = 2*V_MF1; # make a copy and double the values
N = length(V_MF1);
if mod(N,2)==0 # case N even
neg_freq_range = Int(N/2):N; # Define range of "neg-freq" components
else # case N odd
neg_freq_range = Int((N+1)/2):N;
end
V_ANAL1[neg_freq_range] .= 0; # Zero out neg components in 2nd half of
array.
v_anal1 = ifft(V_ANAL1);

j=im; # Assign j as sqrt(-1) ("im" in julia)
v_bb1 = v_anal1.*exp.(-j*ω0*t)
v_bb1=v_bb1.*r.*r # Range compensation
```

```
# Get delta_psi
delta_psi = angle.(v_bb1 .* conj(v_bb2))
lambda = (2*pi*c)/ω0;
d=0.016 # Measured distance

k=0; # Accounting for ambiguity
ang0 = asin.((lambda*(delta_psi .+ k*2*pi))/(2*pi*d)) ;

k=1; # Accounting for ambiguity
ang1 = asin.((lambda*(delta_psi .+ k*2*pi))/(2*pi*d));

k=-1;# Accounting for ambiguity
angm1 = asin.((lambda*(delta_psi .+ k*2*pi))/(2*pi*d));
```

The final excerpt of code of the Jullia code that was run in extract the locations of the targets acquired by the sonar system. Since the positions of all received signals had already been calculated, it had been decided to cycle through all values, and move the values of any non-targets off screen so that they would not be shown on display. Target acquisition was done by comparing the current value of the analytic signal and comparing this to the previous and next values. If the value was found to be a local minimum and above a set threshold, a target was considered to be squired.

```
threshold = 200
for n in 2:S-1
    # Find increasing and then decreasing values
    prev0=abs.(v_bb1[n-1])
    current0 = abs.(v_bb1[n])
    next0= abs.(v_bb1[n+1])
    current1 = abs.(v_bb2[n])
    if ((prev0<current0) && (next0<current0) && (current0>threshold ||
current1>threshold)&&(n<5000))
    else
        y0[n]=10
        y1[n]=10
        ymin1[n]=10
    end
```

```
en
```