Ultrasonic Radar
Final Report

Group: 22
Advisor: Jiming Song

Team Members:
Kevin Czerwinski - Electrical Engineer
Derek Thomas - Computer Engineer
Ryan Foster - Electrical Engineer
Samuel Rosette - Electrical Engineer
Jack Riley - Electrical Engineer
Abubaker Abdelrahman - Electrical Engineer

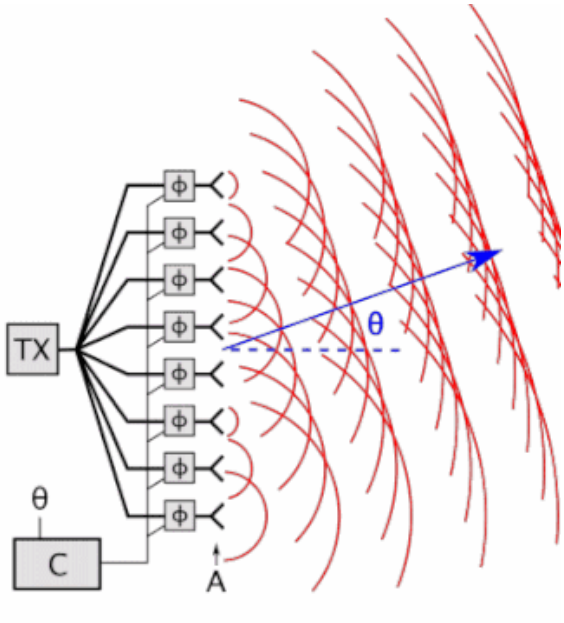
Team Email: sdmay23-22@iastate.edu
Team Website: https://sdmay23-22.sd.ece.iastate.edu
12/02/2022

Table of Contents

1 Introduction

1.1 Problem Statement

The purpose of this project was to create an object detection device using ultrasonic sound waves. This device will consist of a phase array of ultrasonic transmitters to allow for scanning at different angles without using any mechanical movement. This project will be able to detect distances and angles of multiple objects.



1.2 Intended Users and Uses

This product will be used by anyone that needs to be able to detect insects or small animals in a vicinity. As of right now, we are completing this project for our Advisor and Iowa State University with not much knowledge about who will be using it and what for. This kind of radar could be used by many different people for a range or purposes. If it works well enough with high precision it could be used by professors or students in college to learn more about ultrasonic detection systems. This user is hard-working and eager to learn. The only thing they need is to understand the material in order to teach it or learn more about it. It could also be used by scientists that study the environment and keep track of how healthy an ecosystem is by being

able to count the amount of a certain type of insect present in an area. These scientists are trying to find life-changing research which gives their ideas credibility. By using our tool, they will be able to obtain quicker and trustworthy data. An ultrasonic radar could prove useful in many applications.

2 Plans from semester 1

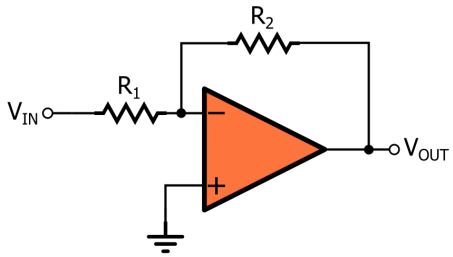
2.1 Work from Semester 1:

For this semester, we tested a lot of components and ideas. The tasks that were completed are shown below.

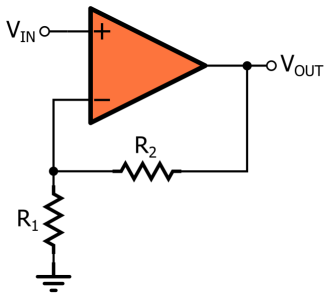
- Tested the transmitter and amplified the transmitted signal.
- Tested receiving signal and amplified it.
- Brainstormed ideas of what the amplifying circuit will be composed of.
- Researched for the best and affordable transducers.

2.2 Changes after Semester 1:

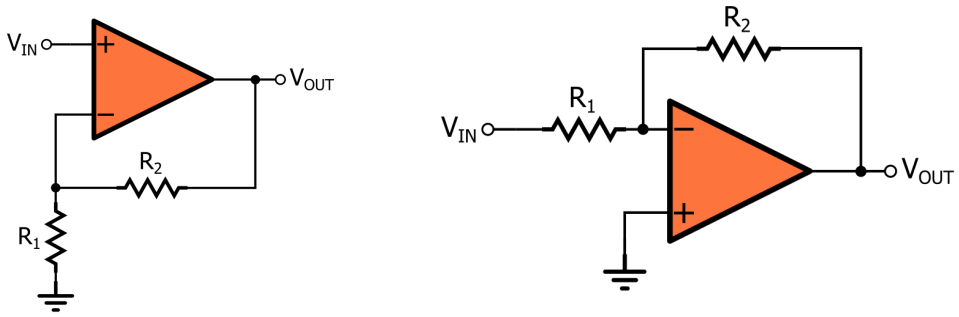
At the end of semester 1, we planned to use a simple inverting amplifier to amplify the input, output, etc. This simple amplification circuit is shown below.



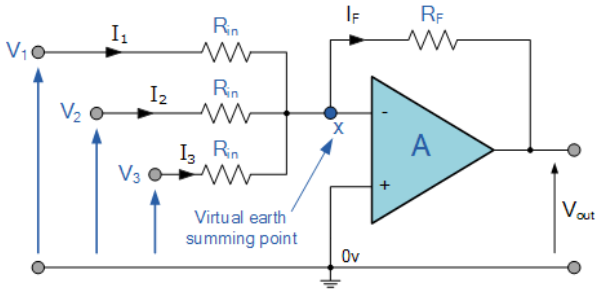
This plan changed as we worked with the circuit to create the ultrasonic radar. We learned that the inverting amplifier was not working quite well on its own. We ended up using a non-inverting amplifier right at the start to to amplify the input pulse. The simple non-inverting amplifier circuit is shown below.



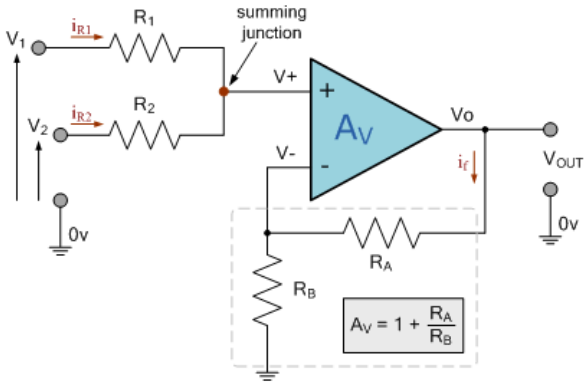
This non-inverting amplifier was good, but not good enough. We experimented with inserting an inverting and non-inverting amplifier after the first non-inverting amplifier, and we concluded that the inverting amplifier worked better.



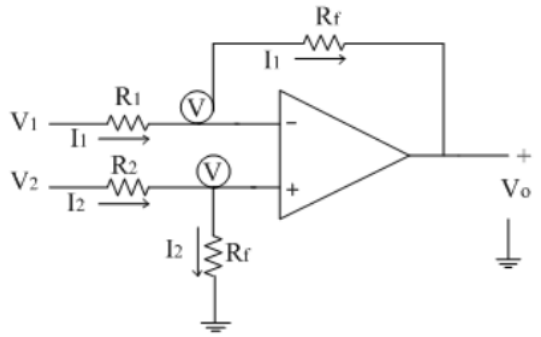
This resulted in a higher amplification but we ran into another issue. We were obtaining high negative voltages as the minimum which was not good because the lowest voltage that the arduino can take is -0.5V. We then inserted a summing circuit after the inverting-amplifier circuit to eliminate the negative voltages by adding 1V to the pulse. The summing circuit is shown below.



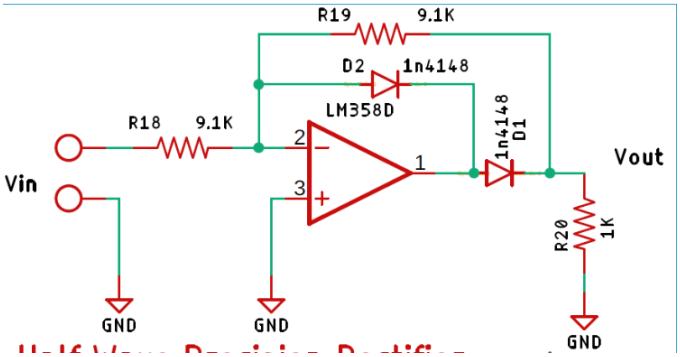
In our case, we only have two inputs, but this didn't work because it was ruining our pulse. The pulse was turning into a constant zero after the summing amplifier. We then decided to use a non-inverting summing amplifier which is shown below.



This had the same result of ruining our input pulse. We then decided to insert a differentiator to subtract -1V which means that we're adding 1V. The differentiator circuit is shown below.



Again, this resulted in ruining our input pulse. Next, we decided to use a half-wave rectifier to eliminate the negative voltages. The rectifier circuit is shown below.



This helped out a lot with the negative minimum voltage. However, it wasn't good enough still. Finally, we decided to implement another half-wave rectifier to eliminate more of the negative voltage. This resulted in around -0.1V for the minimum voltage which the arduino can take. This ended up decreasing our amplification, so we decided to change the ratio of the circuits to increase the amplification. We also added a capacitor to strengthen the input pulse and get rid of some of the unnecessary noise.

Another important development we made had to do with the op amp chips we were using. At first we used LM324 op amp chips because those are the regular ones that are used in most of our classes and labs and they were readily available for us. Unfortunately, with these we were having many problems with amplification of any kind. Peering through the LM324 data sheet, we realized that they are not very optimized for functioning at that high of a frequency. The graph pictured on the right shows how much the maximum output voltage drops off when reaching a frequency around the 40kHz that we are operating at. Many other graphs on the data sheet showed stunted performance at high frequencies for many of the op amps

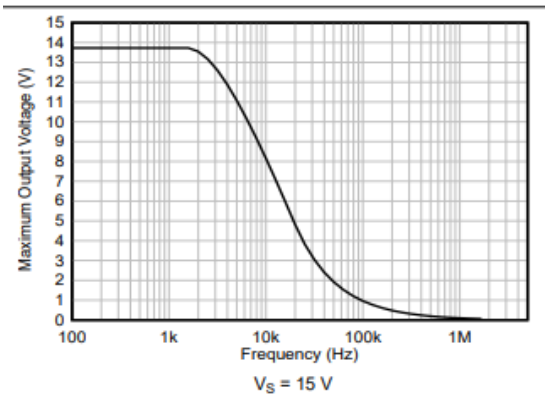
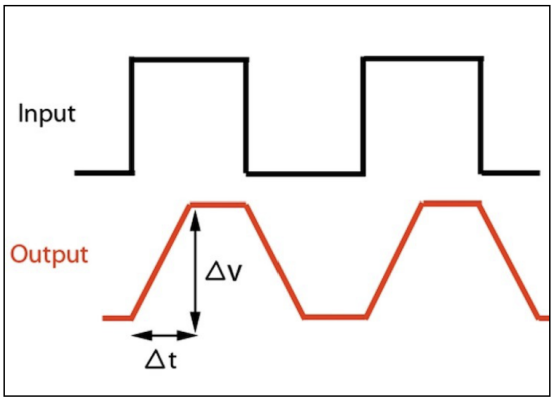


Figure 6-32. Maximum Output Voltage vs Frequency

characteristics. The 324 also has a low slew rate, 0.4 V/µs, which reduces the clarity of the square waves going through the amplifier circuit.



The image above shows how a lower slew rate can cause the square wave to become more triangular. A lower rate increases the delta T value which is undesirable. To combat this we switched to LMC660s. These op amps have a slew rate of 1.1 V/µs which is over twice the 324, due to this we believed it would work much better. Despite the higher slew rate though, the op amp did not work as well as we had hoped. There were still issues in the signal after amplification, it was not clear enough to work with. After doing more research on op amps and looking through op amps that are more optimized for high frequency applications, we decided to move forward with the TLO82 chip. This chip has a slew rate of 13 V/µs which is much more usable for this application and allows us to output a much cleaner receiver output signal.

3 Fundamentals

3.1 Requirements & Constraints

Functional requirements:

- Designing circuits to change phases to each transducer to control the scanning directions.
- The mechanical wave is reflected back from any objects and detected by the transducer.
- The time delay in the pulse-echo can be used to calculate the distance of the objects.

Resource requirements:

- Should be fairly affordable.
- Parts from online or ETG that are available.

Physical requirements:

- Should be very durable because it will be used outdoors to detect insects.

Aesthetic requirements:

- Should be appealing or at least look concise and nothing unneeded.

User experiential requirements:

- Be able to keep track of errors
- Have an idea of what is working and controlled experiments.
- Can see how far the object is at and at what angle.

Economic/market requirements:

- Should be environmentally friendly because this item will be used outdoors. There are a lot of ways this item can mess up due to environmental factors.

Environmental requirements:

- Make sure there is no harm to atmosphere
- Debris shouldn't prohibit the wave from being sent and received.
- Non-invasive detection of objects and creatures

UI requirements:

- Make a radar-like output to show the distance and angle of objects.
- Return information about the object (distance and general shape)

Time Constraint:

- We needed to build this in two semesters
- Had to work on it in between school, work, and life.

3.2 Standards

The IEEE code of ethics will apply to this project due to our responsibilities as engineers.

This project uses ultrasonic sound wave emitters that can possibly interfere with an environment depending on the circumstances of where and when it is used. To follow the code of ethics we must make sure that any kind of usage of our device has a minimal impact. By keeping the pulse durations short and having a significant delay between pulses, there should be no unintended ramifications. The device also operates at low power which further guarantees that the environment it is used in will remain unchanged.

3.3 Security Concerns and Countermeasures

This device has minimal security concerns as it is more a tool to be used for research purposes than other intentions. The programs do not connect to any kind of wireless network so cybersecurity is unnecessary.

4 Testing

For our project, testing will be heavily focused on the accuracy of the scanning capabilities of our radar. Since our ultrasonic radar will need to be able to scan a range, we will also need to test its accuracy on objects that are not placed directly in front of the array. So for our testing we will accurately measure our distances in front of the radar and place objects that can be noticeable detected by it. We will then run the scan and see how well the radar works with picking up an object in front of it vs at an angle. A test will also be completed using smaller and smaller objects to see how small of an object can be detected by our radar.

4.1 Unit Testing

We have been focusing on finding an amplifier that will work with our high frequencies. We started off using a LM394 and an LMC660, we found that these were made for too low of a frequency to work with our 40 kHz signals. We then settled on the TL082 which was much better rated at the 40 kHz range that we are working with.

4.2 Interface Testing

There are a few interfaces in our design. Primarily we have the radar and then we have the software that is separate, but reliant on the radar. We think these two interfaces will need to both work fully for the overall project to succeed and for us to reach our goals. In order to test both, we will have input and output comparisons. So, we will see the mechanical output from the radar, and when connecting it to the software, we will cross-check to see that the input going into the software is the same, to make sure there are no discrepancies in the passage of I/O. Some tools we will use include measurement tools that are functional without mechanical pieces, and then a display and perhaps a code script to run on the software.

4.3 Integration Testing

There are some key critical integration paths, some smaller than others, but all important. We know that the transmitters, whether transmitting or receiving, have to integrate with each other flawlessly for us to get the feedback we desire. In order to test this, we will have to generate sound waves in specific patterns. When we visualize the output, we should see a trend given the sound waves we generate and then we can see how far off or how close we are to the expected results. Another critical integration path is then when translating what the radar detects to a computer software. In order to get this translated and tested, we will create a testing software script to check our integration of software alone and the software communicating with the hardware.

4.4 System Testing

In order to test our device, we must take into consideration a few parameters. We must define accuracy, precision, range, and additional capabilities. An example of an additional capability is the ability to detect objects behind other objects; which has been emphasized that this is a rather difficult task. The tools that can be used to determine the accuracy and range could involve a testing rig. This rig could use a set of objects to be detected in a field. These objects must have known dimensions and would be placed a known distance from our sensor. Precision in our case would be defined by the smallest spatial resolution that our device can detect. We could use the previously mentioned range rig for this test as well. Using objects of incrementally decreasing length and width, we can determine the resolution of our sensor at different distances; which corresponds to the precision of our device. Finally the additional capabilities can be measured by simply placing two small objects (such as two pens) in front of one another. If we can detect the more distant object, we can confirm the additional capability parameter of our project.

4.5 Regression Testing

In order to avoid going backwards and breaking anything that we knew were working from testing, we will have checkpoints. Just like a video game has checkpoints, when our code fails or the test fails, we will have it saved to the last point as to which all tests were passing or majority was. This way we will not lose any work done and we will be able to see how far we have gotten without errors and we can move up from there. In order so it will not break, we have to make sure our system is rigorous and the testing is multifaceted. The requirements will be specific and based on what the software and hardware needs and we are still figuring out what specific tools will be used, but we have the overall idea.

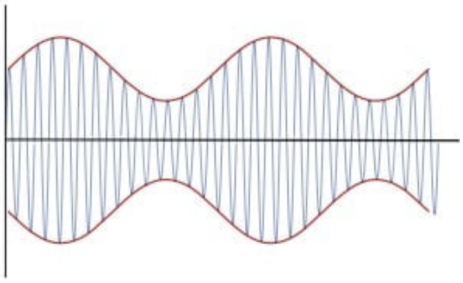
5 Implementation

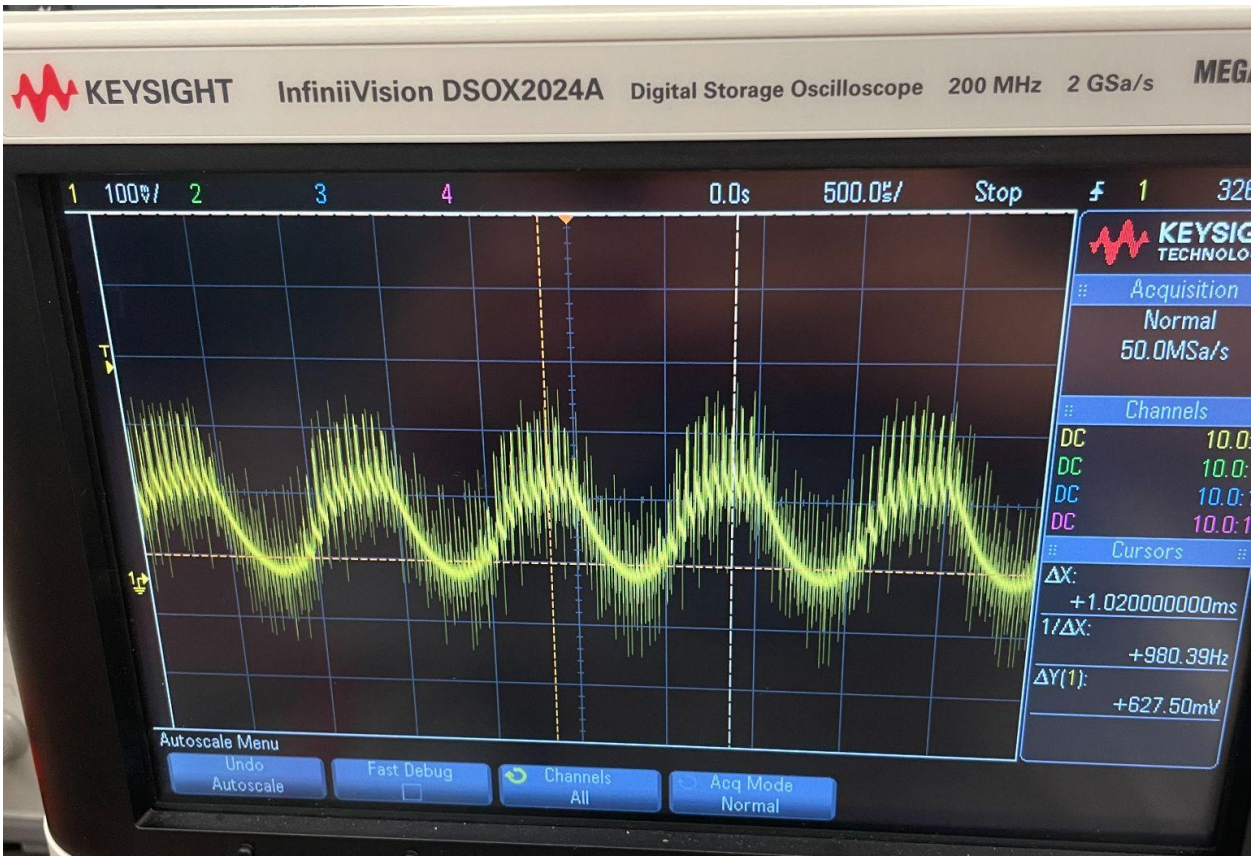
5.1 Circuit

The transmitters originally had inverting op amp circuits to amplify the square wave pulses from the arduino but after testing, these circuits were removed to improve the clarity of the pulses. The transmitters are now connected directly to the arduino.

The receiver circuit has gone through many stages of changes and improvements. This is the most important circuit in our project because without receiving a detectable signal that can be read in by the Arduino, the device would not work. The output of the receiving circuit also cannot jump higher than 5V because this voltage is the Arduino input limit, anything above it could risk the board getting fried. We began with simple inverting and non-inverting amplifier configurations but quickly realized that the required circuit would need to be much more complex. After a non-inverting amplifier stage we realized that a lot of noise was distorting the

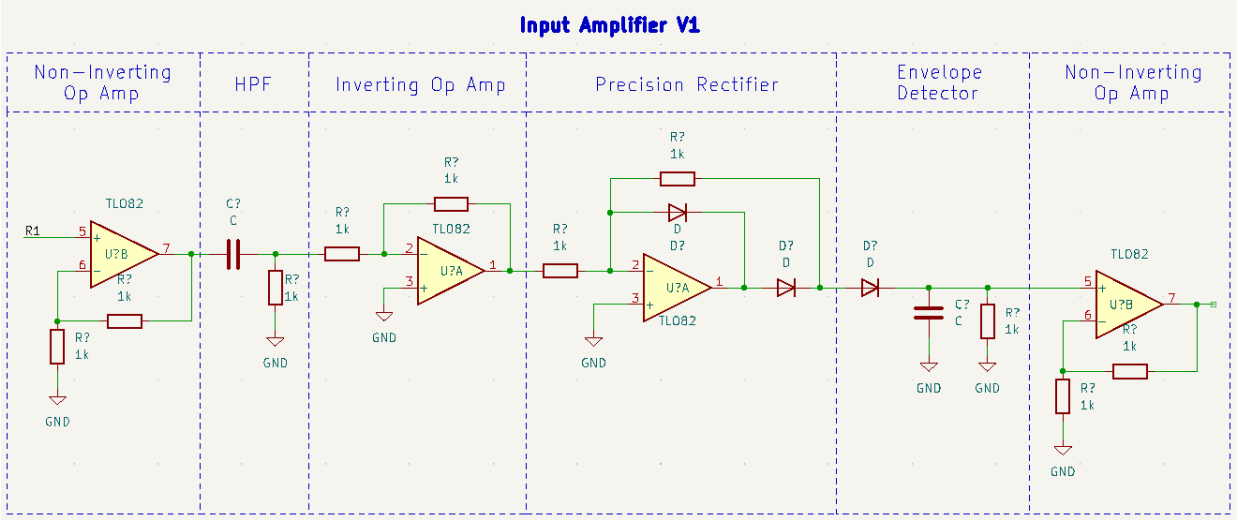
output so we included a high pass filter to get rid of any low frequency noise. This improved the signal and allowed for another op amp stage, this time an inverting op amp. The next part of the circuit was a precision rectifier to try and filter out the negative voltages and get the signal in a range that could be detected by the arduino. The output signal of this rectifier was still very hard to work with. The output looked very similar to an amplitude modulated signal (an image of this is shown below). This type of signal has been studied in the communication systems courses here at ISU, and the message signal (the red sine wave) can be recovered using an envelope detector. Using this lower frequency, DC shifted sine wave would work ideally for our analog input. To design this envelope detector we needed the carrier frequency, which we know to be 40kHz, and the message frequency which we did not know. This value was not difficult to find using an oscilloscope. By measuring the time difference between peaks we realized that the message signal had a frequency of about 1kHz. Using both of these frequencies, we were able to create an envelope detector that functioned properly.



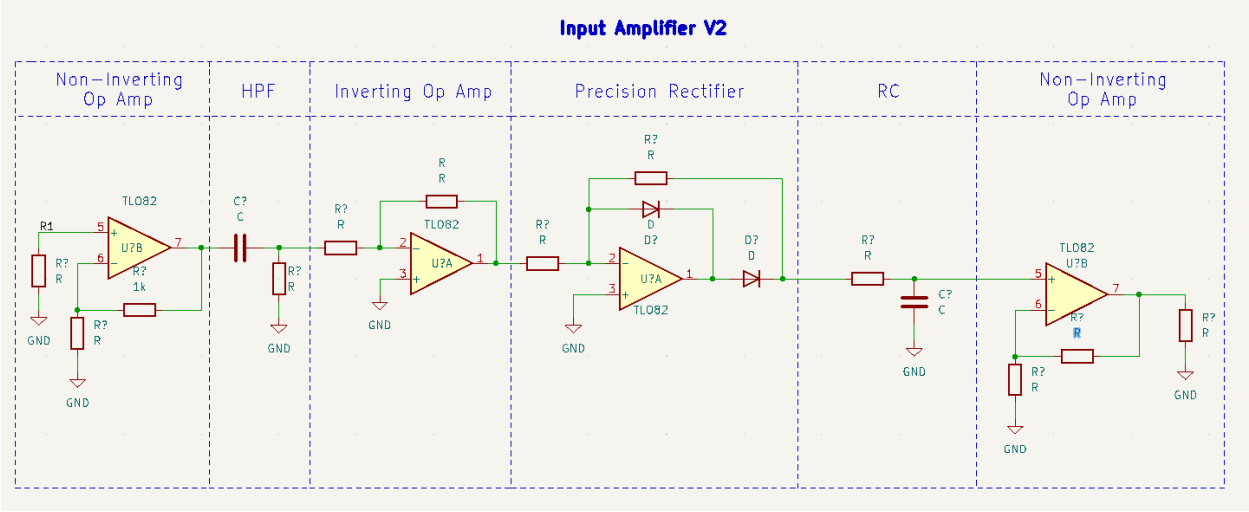


The image above shows the signal at the envelope detector output. The cursors on the screen show the time difference between the peaks of the wave and the calculated frequency is shown

on the screen. This wave was still quite noisy but it was much cleaner now and would work much better for the Arduino input. Unfortunately, despite how well the detector worked and the cleanliness of the output signal, the envelope detector after a few minutes of functioning well, all of the sudden stopped working. No matter what we tried we could not get the circuit as well as the first few minutes. This was very frustrating as it was working as intended and we are not sure what caused it to fail. The schematic below shows the envelope detection circuit.



After a couple hours of trying to figure out the problem, we decided to scrap it and try something else. The next iteration just had an RC circuit after the rectifier output and it kept the non-inverting op amp at the end. It also contained a couple of extra pull down resistors to stable out the output. The second circuit we developed which ended up being our final design is pictured below.



5.2 Code

Arduino Source Code:

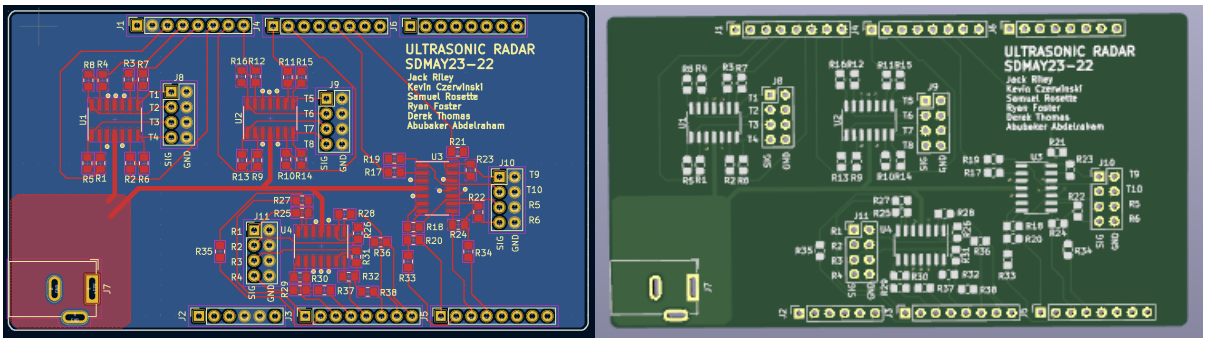
The Arduino code is posted at the end of this document. It consists of a pulse function to send out a volley of 10 pulses that bounce off of objects in the environment. A timer is then started until a pulse is received from the A0 analog input. Using the speed of sound and the time recorded allows a distance to be calculated. The code then iterates through all of the angles. The distance and angle are then sent to the display software.

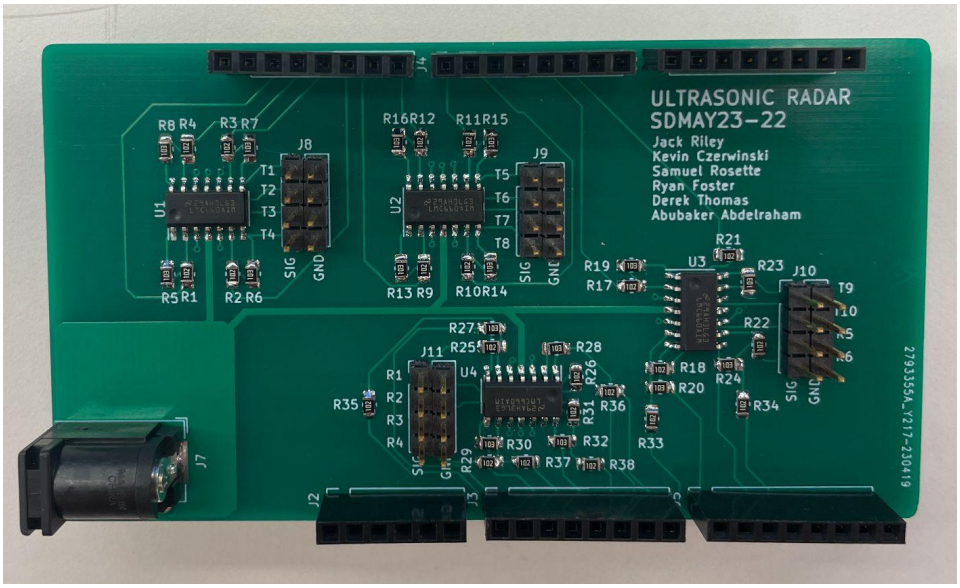
Display Code:

In order to output our design, we used an application called Processing. The Processing IDE allows us to plot and create a radar to accurately detect and display the results that our design collects. The software is very similar to the Arduino code, and we were able to reference some open source projects to come up with a way to draw out a grid from 0 to 180 degrees. What the code does is communicate with the Arduino's serial port and takes in an angle and a distance. We take in the angle and the distance and use this to draw the lines to detect the objects and where they are. From there, we are able to plot the radar given the inputs. Based on the Arduino source code, we can either make a for loop to constantly scan and detect or only scan when there is an object detected. The Processing IDE allows us to output our results and make it visually appealing.

5.4 PCB

The PCB was designed to plug straight into the top of the arduino. It amplified the output transmission signal and the input signal from the receiver. There was a 12V power jack giving the Op-amps power. Header pins were included so that we would be able to use cables to connect the transmitters and receivers allowing for more freedom in the case.





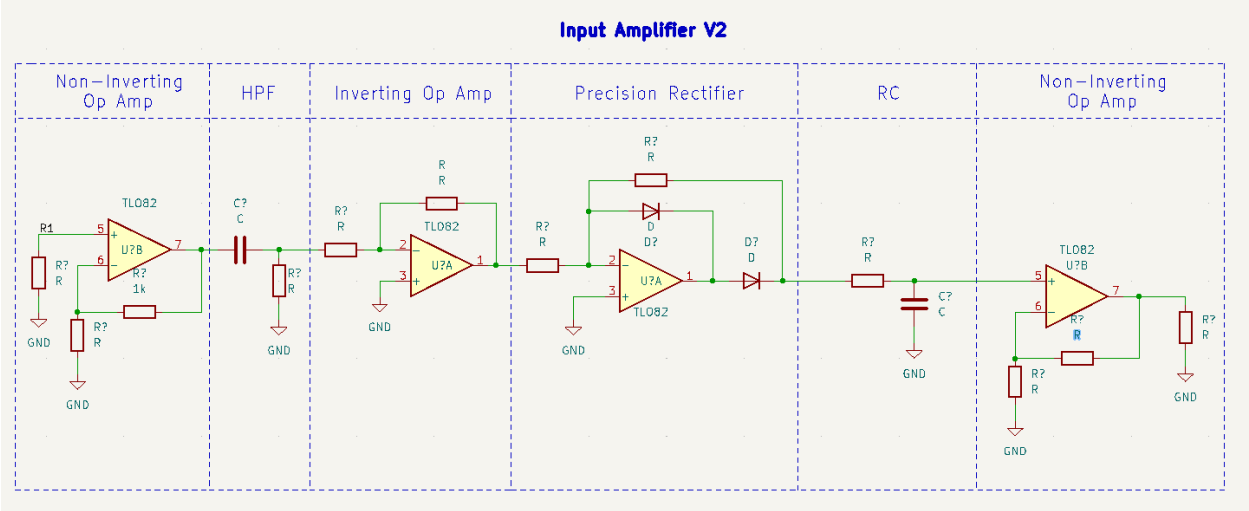
Through testing, we realized that this circuit did not perform as well as it did during the initial testing. The receiver circuit was not able to output a clean signal that could be read in by the arduino and the transmitter amplifiers had minimal effect on the square wave being sent to the transmitters. We attempted to change some of the resistor ratios as well as the chips we used but the amount of changes we made to get the circuit to work better ended up making it more efficient for us to just create a new circuit board and make this one on perfboard. Given more time, we would be able to order a new PCB and have a much cleaner final project. Instead of using a PCB we will be using a perf board for the circuit. While this isn't as ideal as the PCB once everything is secured in the case it will not have any visual effect on the final product.

5.5 Case

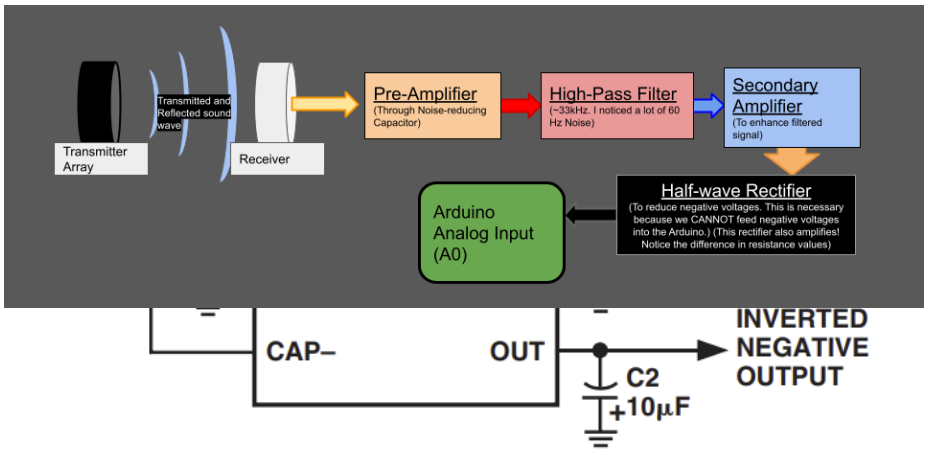
Originally we had designed a 3D printed case but ran into problems when we tried to print it. The case had multiple errors during printing causing the whole thing to be scrapped. We came to the conclusion that it was too large to print on the 3D printers available to us. After realizing this, we decided to use an existing plastic case that we then milled to add the holes where we wanted them. With the help of Boyd labs we were able to make a case that fits everything perfectly. On the front, there are 10 holes for the transmitters and the hole for the receiver. The side allows the user to access all of the ports on the arduino and the circuit board.. The arduino then fits onto a base plate that secures itself to the haunting holes on the case's bottom. In the end we are very happy with how the case turned out and think that it is better than if we had been able to 3D print the whole thing.

6 Results

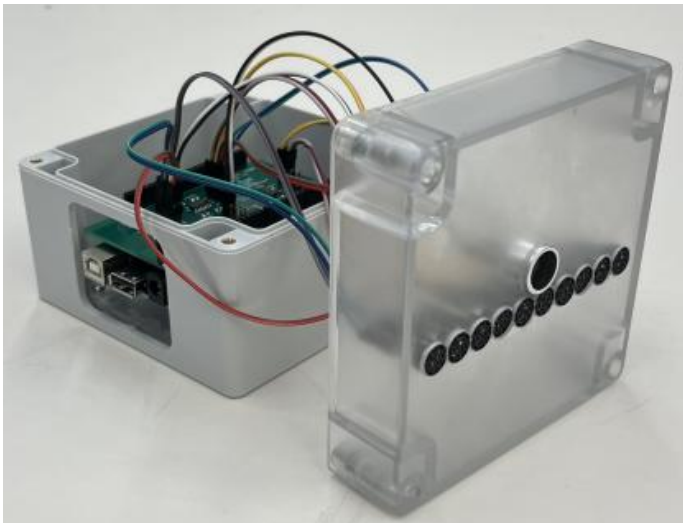
6.1 Circuit



A -5 V voltage converter was also used to provide the negative rail power to the op amps. Below is the schematic for this converter.



6.3 Final Product



6.4 Outcome

Overall we were able to create a functioning distance scanner that can detect objects within a range in front of itself. We struggled with the phase array code due to the complexity of it and the fact that we have not worked with this kind of technology in the past. We ended up making good discoveries and learned a lot throughout this project while also being able to apply many of the fundamentals that we have learned throughout our time here at ISU.

6.5 Future Work

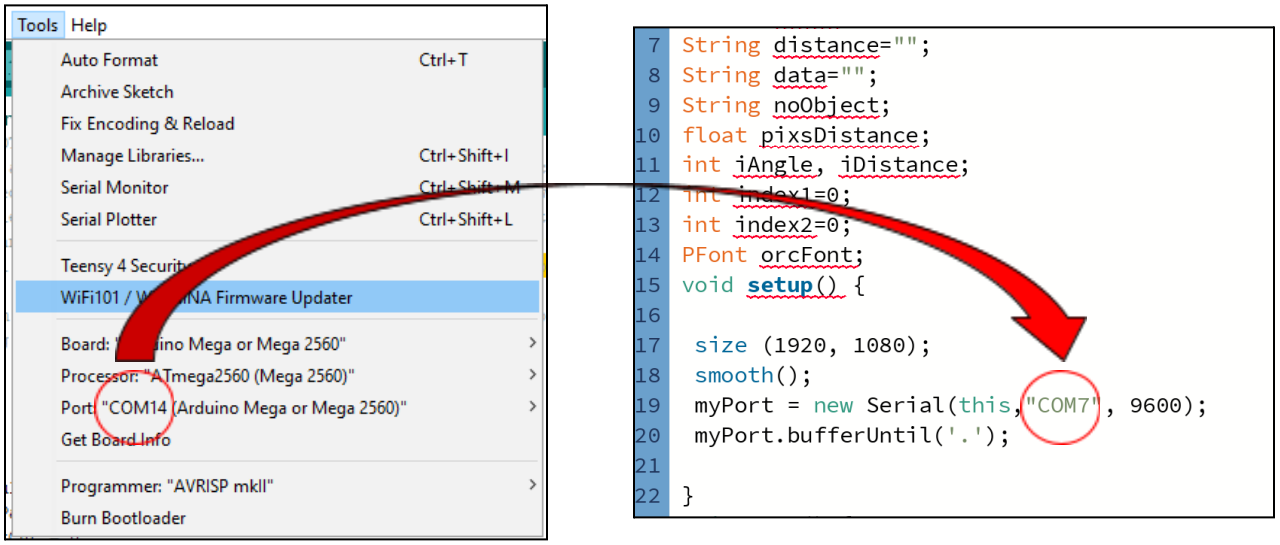
There are many ways that our project can be improved upon in the future. The phase shifting code can be optimized and more receivers can be added to increase accuracy. Unfortunately this also greatly increases the complexity of the software. The microcontroller can be switched to a raspberry pi to boost the processing power and greatly improve the functionality of the device. A circuit can be created to control higher voltage pulses and allow for a much more powerful transmission signal which can increase the scan distance. One last thing that could be improved is the hardware, throughout our research we could not find smaller transmitters, but in the future if one is created or if they become much more readily accessible, then using smaller transmitters can greatly improve the phase array.

Appendix I - Manual

The device needs a direct connection to a computer via a USB type AB cable in order to function. Two different software programs, Arduino and Processing, are required to operate and view the Ultrasonic Radar output. The links to the programs are linked here:

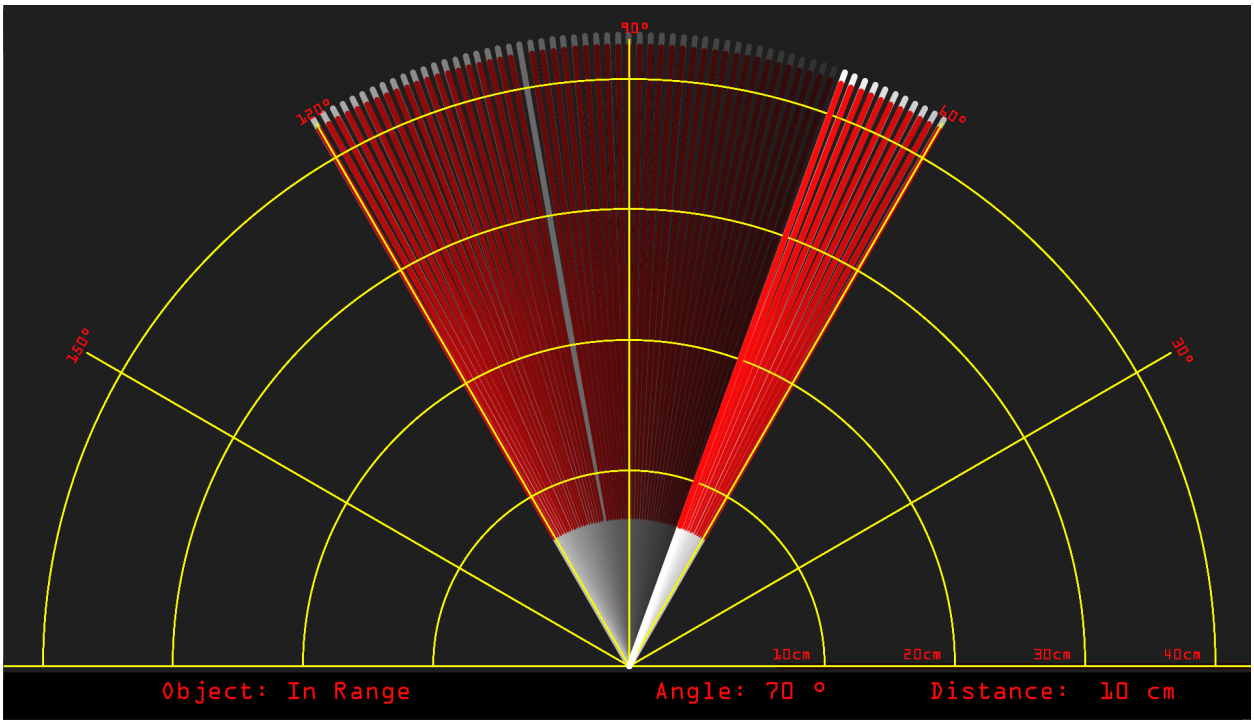
Arduino: LINK
Processing: LINK

With both programs downloaded and installed, Arduino must be opened first to allow the data output from the Arduino Mega to be sent through the serial monitor. This is how the Processing software is able to display the data. IMPORTANT, the COM port that the Arduino Mega is connected to must be noted as this COM port label can vary from device to device. This Can be found in the "Tools" tab below the processor information. With this noted, the Processing software can be opened. There is one important line of code that needs to be changed in order to ensure the device works as intended. The COM port needs to be filled out on line 19 with the port that the Mega is connected to. If this is not changed, the software will not have any data input to display. The images below show where to find and input this information.



These two COM ports need to match up. Once the port is set up and the code is saved there should not be any further need to change it unless the device being used is different or the port is actively changed.

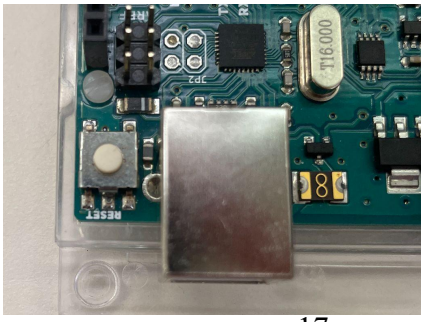
Now that this setup step is complete, the code can be run from the Processing program by pressing the arrow in the top left corner. This should begin the startup process for the radar display. If everything is set up correctly then the program should launch a display window that shows a radar scanning the front of the device where the transmitters and receiver are present.



This window should look like the display above with the line scanning from angles of between 60 to 120 degrees. The angle and distance are displayed on the bottom with a visual indication shown by the length of the red line in the radar scan. Each of the yellow semi circles protruding from the center are indicators for distance ranges incrementing up by 10 cm per step. This gives a rough estimate of the distance of the object being scanned as well as the angle it's being scanned at.

Troubleshooting: If nothing is being displayed then attempt to re-run the display code in Processing. This should fix the issue as long as Arduino is open and the Arduino Mega is connected to the correct COM port. If the distances and angles do not seem to be working properly, you can press the white reset button present on the back of the arduino from the access hole on the side of the device. This will

Reset Button



run the code on the Mega again and hopefully solve any problems. If the issue persists, re-upload the code from Arduino to the Mega to reset the device.

Appendix II - Alternative

Version Considered Before Client's Specifications and Learning More About the Project

Some prior work that we have seen with regards to this project uses basic Arduino ultrasonic sensors to send out and receive data that bounces off of an object. In a previous course, most of us had to use one of these sensors on a Roomba to scan an environment. It's a common sensor that is even posted on the Arduino website as a project. This can be used to create an ultrasonic radar but only contains a single transmitter and receiver. This differs from our project as we need to use an array of sensors to make it much more accurate. Another project we found was a youtube video of a guy that created an array of ultrasonic sensors to view the effect of the interference of all of the emitters. This is much closer to what we want to achieve, although he is not using it as a radar. He is simply observing the phase shift effect. Our project needs to be able to take advantage of this phase shift effect and be able to calculate distances with the pulses that are sent out.

Appendix IV - Code

https://iastate.box.com/s/0gdnnjlsc50ro7iejudlsedd5xzbxci4

The dropbox linked above has both the Arduino and the Processing code.