Sonar Detection System

Sonar and Radar Processing

Prepared by Chris Toner and Kirk Jungles

ECE 435/535 Winter 2020

17-Mar-2020

Table of Contents

Abstract

We have produced a distance detection system using a Raspberry Pi and Matrix OneVoice. We did so using the theory of sonar and radar as taught in our ECE 435/535 class and implemented the system in Python.

Hypothesis

Our hypothesis was that by using the principles we learned in class, we would be able to emulate a sonar system using the tools at hand. We began by writing a python script that would generate a .wav file that would produce an fm chirped signal at a designated center frequency of 2 kHz with a pulse width of 5 ms and PRI of 1 second. We would then generate the pulses from a speaker at the approximate location of the receiver, which is the Matrix OneVoice, thereby emulating a monostatic sonar apparatus.

From the known formulas in class, and our given radar parameters, we have that:

$$Range_{unambiguous} = \frac{V_p * T_{PRI}}{2} = 171.5\ m$$

$$Resolution_{envelope} = V_p * T_{PW} = 1.715\ m$$

We note that we are using a correlated FM Chirp rather than a simple envelope detection scheme, which should yield us a higher resolution.

Another important formula that we will use later is the Time Distance of Arrival formula, which will give us the distance of an object as a function of the time difference between the initial recorded pulse and the correlated return signal:

$$Distance = \frac{V_p * \Delta t}{2}$$

We then predicted that we would be able to apply the theoretical knowledge from the term to determine the distance of objects from our sonar unit. The Matrix OneVoice was attached to a Raspberry Pi, which we built Python code in. The Python scripts we wrote utilized Python's built-in audio and signal processing libraries to make the process run quicker and more smoothly. By recording the audio reflected from a surface, our Pi could push the data into arrays and correlate the recorded sound with the generated FM chirp. This analysis can then be plotted onto a graph to determine the distance of the desired object.

Experimental Setup

Our system relied on a Raspberry pi setup attached to the Matrix OneVoice. We utilized Python scripting to run the audio capture and signal processing. A separate laptop used Audacity to play the constructed FM chirp .wav file. A parabolic dish was also set up to help magnify the reflected waves to our Matrix.

In our initial experiment when verifying the system, we determined an outdoor park area with a solid wall and minimal multi-path interference as a suitable test ground. This most closely matches the Ideal Experimental setup as shown in Figure 1.

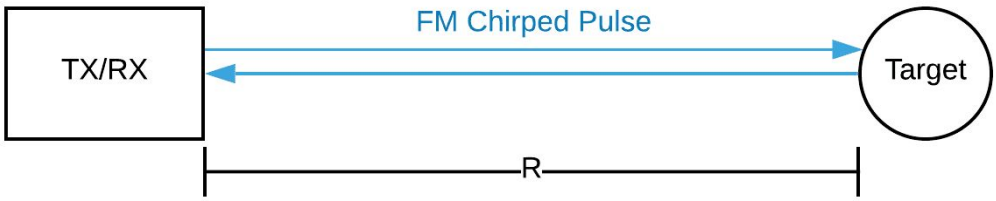

Figure 1: Ideal Experimental Setup

Once set up, we played the audio and ran the Python scripts on the Pi to capture the reflected audio sounds. Once data capture was completed, the Pi then ran our scripts to properly pass the signal, gate, correlate, and finally plot our data in a meaningful display. Our programming algorithm and code is attached in Appendix A.

The outdoor experiment was observed by our venerable professor Brian McGarvey, and checked off as successful. However, we failed to collect that particular set of data, so when It came time to write this report and present our collected data, we had only the Tek Lab to perform our experiment in. The experimental setup for this scenario is shown in Figure 2.

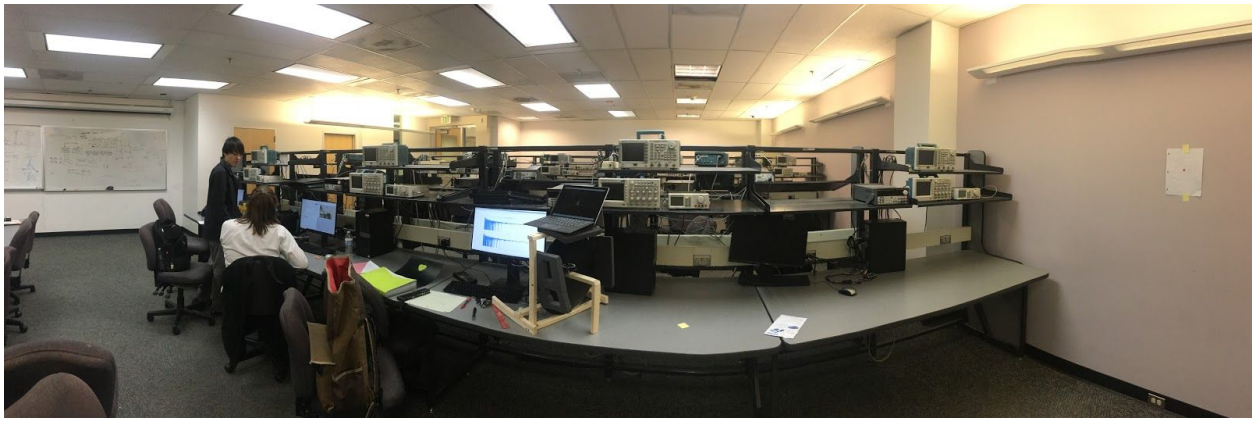

Figure 2: Actual Experimental Setup

As you can see, this setup makes verification of our system more difficult, and is better modeled by something approximated by the setup shown in Figure 3.

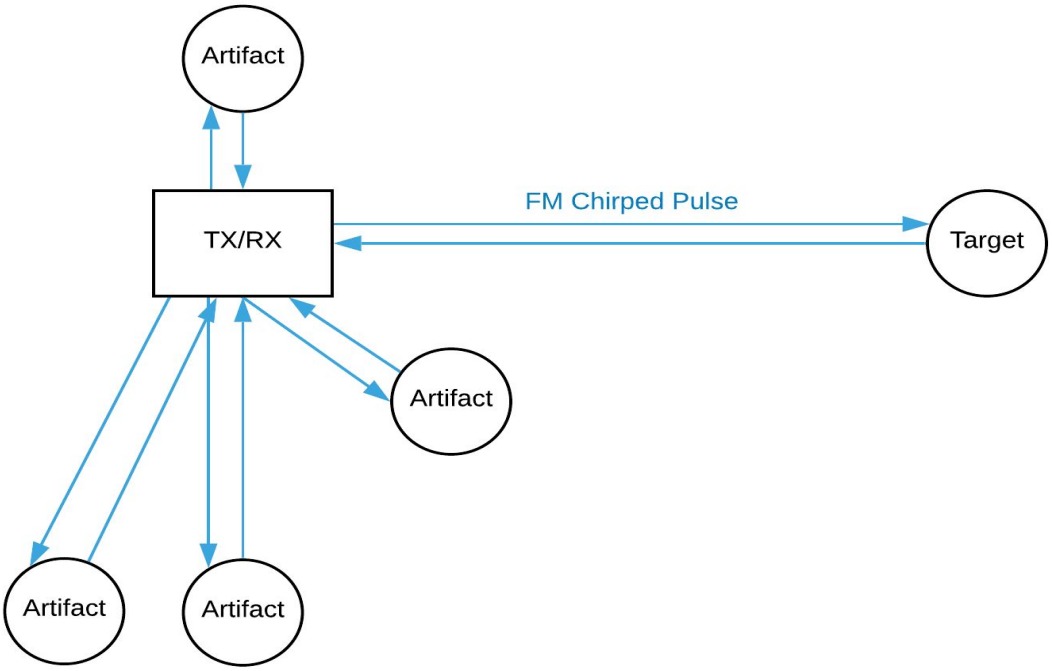

Figure 3: Model of Actual Experimental Setup in Tek Lab

We can see that the many artifacts in the Tek lab will produce undesired correlated return signals, and it is nearly impossible to discern the target from other artifacts. However, we can still check to see if there is a correlated return signal at the expected distance of our Target.

We then measured that the distance from the TX/RX unit to the northern wall is ~8.5 feet, the distance to the ceiling is ~6 feet, and the distance to the southern wall is . We then expected to find correlated returns at these corresponding distances. We then ran the experiment, and the collected data and resulting discussion is shown in the next sections.

Recorded Data

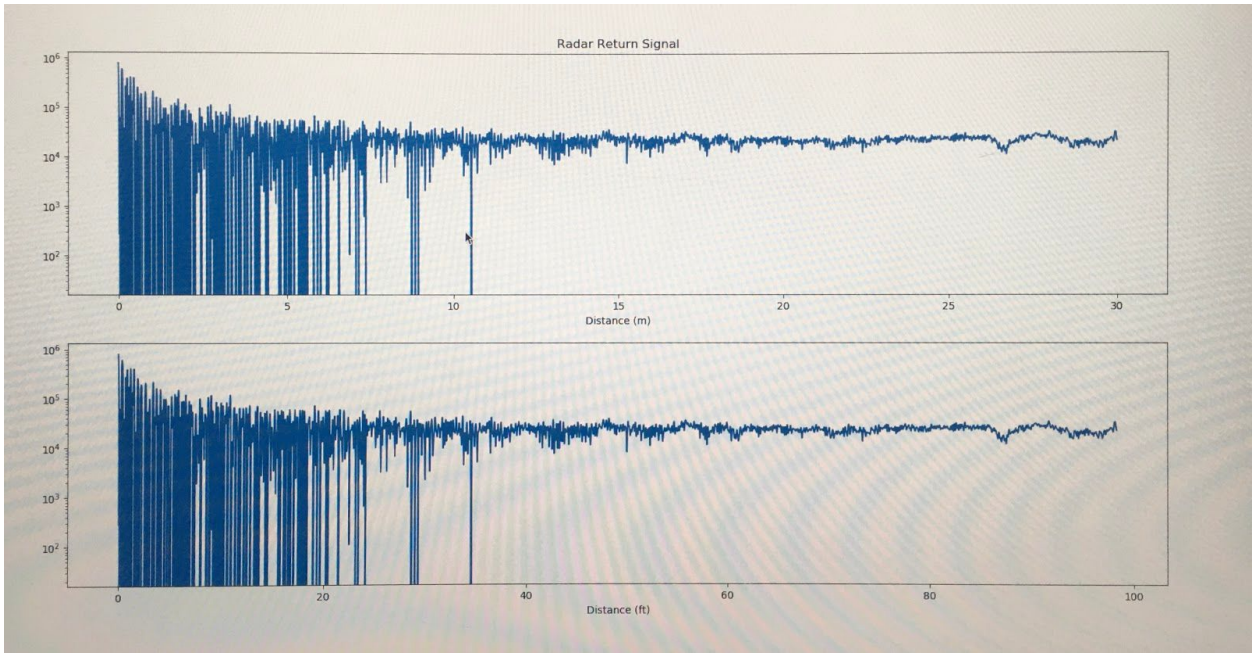

Figure 4: Recorded Signals within 100 foot range

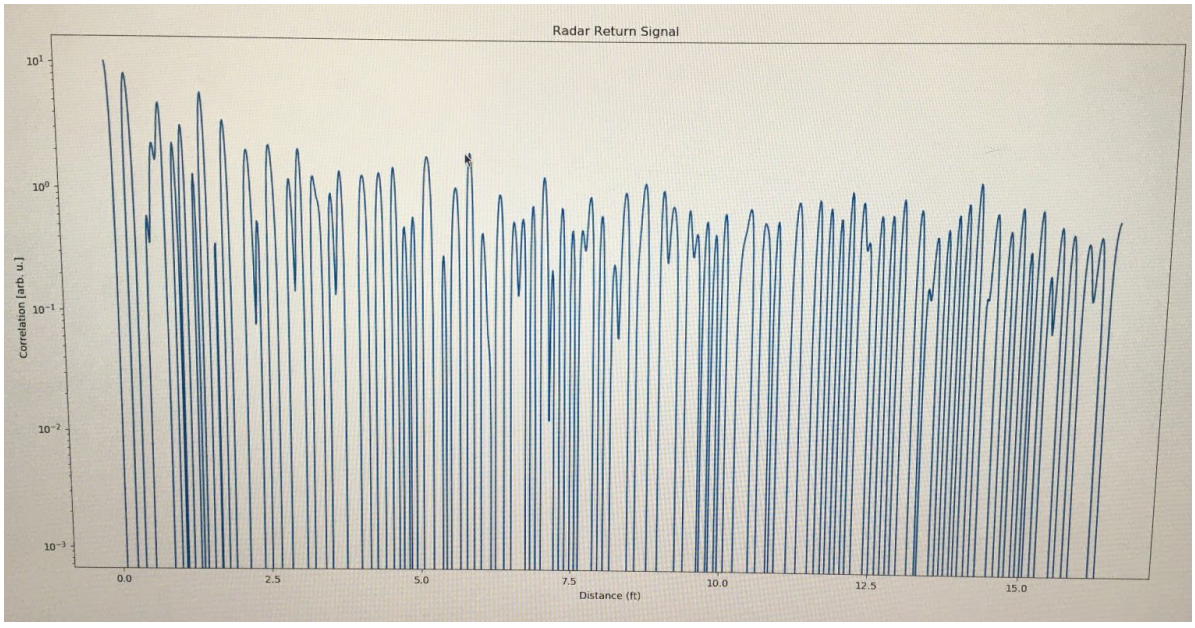

Figure 5: Recorded Signals within 15 foot range

Results

In our initial outdoor experiment, we were able to observe very prominent peaks at the approximate distance from the receiver to the large wall, our target. We then compared these results to those of our classmates, noting that they recorded peaks at approximately the same distances. We then concluded that our system is an effective distance measurement device.

We note first that there is a lot of evident ‘noise’ in the system, but considering that the signal displayed is not so much noise as it is the result of correlation with a very specific signal, that is an fm chirp with a 5 ms pulse width, linearly sweeping from 1 kHz to 3 kHz, we can then regard the collected data as being legitimately returned chirps. It should come as no surprise then, that the beginning of the recording has a lot of returned chirps. Referencing Figure 2, we see that there are an innumerable amount of artifacts in the immediate vicinity of the TX/RX system. We can then take those artifacts to be the cause of all the returned signals indicated as being near the TX/RX unit.

We note also that at larger distances, we seem to be recording a ‘noise floor’ without any significant returns, while in the shorter distance range, the correlated signals actually do approach a value close to zero. I suspect that this is because in the close range, we are only recording direct returns of the FM chirp from nearby artifacts, but as the range increases, the chirped signal has had time to propagate throughout all of the room and off of all of the artifacts present, thereby constantly bombarding the receiver with echoed chirps and producing the appearance of a ‘noise floor’.

We then turn our attention to the distances at which we are expecting more prominent returns, such as from the ceiling at 6 feet and from the wall at ~8.5 feet. In closely inspecting Figure 5, we observe that although the signals seem to be buried amidst the artifact returns, we can still see more prominent peaks at around the distances we were expecting to see them at, those being 6 feet and 8.5 feet.

Conclusion

This experiment was a very thought-provoking exercise in how we can use sound to determine distance of objects. Sonar as an abstract idea seems so simple, but when delved into, is much more complicated than originally thought. Many questions arise: How do you determine the desired signal from other noise? How do you amplify a signal’s response? How long do you make your pulse? Sonar is a constant weighing of less desirable options. Whenever you want to improve some part of your system - say, the resolution of your radar - you inevitably have to give up some quality elsewhere; in this case, your maximum range will be affected.

Ultimately the Matrix OneVoice is a very powerful tool when combined with Python scripting and Python's vast array of DSP libraries available. Having to manually correlate several different signals would be very daunting; with Python, though, it is as simple as feeding the proper arrays into a function and letting the code do the work.

For this project we tried to construct a method of building a simple sonar system that could be easy to follow for any other open-source users. We began the steps of turning this sonar system into a radar system that could determine a more precise location of an object, as opposed to just the distance to the object. With a solid grounding in trigonometry, we believed this to be an easy task. Alas, as we have found out throughout this entire sonar class, what sounds easy is much more difficult in practice. For future classes interested in pursuing this route, we would like to lay out some questions to pursue earlier on in the construction of your system. How do you turn time delay between the different microphones on the Matrix into distances? What are the relationships between these distances? How can you use Python to trigger recording an incoming wave from the first microphone that the wave hits, and how do you determine which arrays from which channels to use to begin doing computational analysis?

We were very determined to figure out the answers to these questions, but unfortunately, there is only so much time in a semester. Our Matrix packages came a bit too late into the class, and we had a lot of troubleshooting to take care of with the initial setup of the Raspberry Pi's (the package deal with the Matrix came at the expense of some basic functionality that had to be addressed). However, getting to be the guinea pigs for a new class that could pave the way to make it easier for future classes to follow our footsteps was very exciting, and the ability to build a basic sonar system from the ground up was a very valuable lesson.

Appendix A

Receiving and processing TDOA high level algorithm:

Record 10 pulses using PyAudio Stream into Data array

Reshape Data array into 10 row, Chunksize columned matrix

Roll each row of Data such that the maximum recorded value(initial pulse) is on index 50(this lines up each row to correspond to the same peak gather)

Sum each row of data to create one gated array

Correlate gated array with expected 1kHz - 3 kHz, 5 ms PW chirp

Roll correlated array such that maximum correlate value(initial pulse) lands on index 0

Convert index vector to time vector(in seconds) using known sampling frequency

Convert time vector to distance vector using known phase velocity of sound (D = t*Vp/2)

Plot correlated array vs distance vector

Code Used:

Since the internet was inaccessible on the RPi, and it couldn't talk to a usb drive, screenshots of the code were taken instead, and are available in the Class Github Repo at https://github.com/richard-romano/Radar435/tree/master/Winter2020/Team%205%20-%20Kirk%20Jungles_Chris%20Toner

An executable .py file can be made upon request.