DEPARTMENT OF ELECTRICAL
AND ELECTRONICS
ENGINEERING

EE451 PROJECT REPORT – P11

Real Time Data Detection & Analyze with Sound & Distance Sensors via Arduino & MATLAB GUI

Ahmet Yasin Bulut

250206001

Barbaros İnak

250206004

Emre Nedim Hepsağ

250206012

Advisor: Mehmet Çalı

DATE: 29/12/2021

Table of Contents:

I. SUMMARY 2

A. Purpose 2

B. Designed System 2

C. Project Schedule 3

II. INTRODUCTION 4

A. Communication Process 4

B. Hardware 5

C. Software 6

D. Collecting Real Time Data 6

III. SYSTEM DESIGN 8

A. Transmission of Data 8

B. Real Time Plotting 9

C. Sound Detection Sensor Applications 10

D. GUI Design 12

E. Ultrasonic Distance Sensor Applications 14

IV. RESULTS 15

V. CONCLUSION 20

I. SUMMARY

A. Purpose

In conventional methods, the data is collected then the processing is done on all data to get usable information. In real-time processing, we can save time for getting results, also we can get data and analyze it on one machine and cost can be decreased. We can do the real-time processing on software like MATLAB, or hardware like FPGA or integrated circuits. There are some drawbacks of software processing, such as it requires another machine other than data collecting hardware, also it can be slow & expensive. We wanted to design a real-time processing system, analyze its limits and application potentials. For an introductive project like we wanted to build, we chose software processing on MATLAB because of its easier and flexible design despite the drawbacks.

In this project, we built a system that is able to visualize a real-time sensor signal on a user interface. The signal is generated on Arduino and with a Bluetooth connection between, it can be processed and visualized on MATLAB. In addition to these, we wanted to use some technics we learn on EE451, like sampling, quantization, encoding within a pulse code modulation. We used pulse code modulation in our Arduino and demodulation on MATLAB. Note that the Bluetooth transmission is handled with a sensor and PC, so we can't get any noise or intersymbol interference reduction from any modulation technic.

B. Designed System

Our design is focusing both on the hardware and the software of the system. On the hardware side, we selected a sensor to obtain changeable data, coded a consistent sampling method, and used an optimized PCM for Arduino. On the software side, besides the encoding, we implemented various operations to get different visualizations. Then we put the processing results on a GUI to easily control and effectively observe.

C. Project Schedule

We divided our work to six weeks as can be seen on the Figure 1. In the first week, we created an environment to build our system on around. We use only Bluetooth module, and create a connection between the Arduino and the MATLAB. We test the two-way communication. We used the read and write function of Bluetooth connection with built-in LED of Arduino and the command line of MATLAB.

Next week we used basic data to test real-time transmission quality. We used a potentiometer reading, which gives analog data controlled by hand. In addition to this, we generate a real-time animated plot on MATLAB, which adds new Bluetooth readings to the current plot. In the third and fourth weeks, we tested our sound sensor with sampling adjustments on Arduino. Then we added the encoding and decoding to both ends of the communication to generate a PCM modulation and demodulation. We analyzed the sensor readings on different frequencies, created ideas for different plots, and tried to solve the inconsistent readings caused by the sound sensor itself.

In the fifth week, at first, we generated our sound processing operations as well as the GUI of sound detection. But the inconsistency of the sound readings is increased and became more apparent as we progress. So, we changed our sensor to an ultrasonic sensor and recoded the system to use it for distance detection. In the last week, we complete the GUI design and test it for different measurements. We added a servo motor to make the detection more dynamic and add different results to our project.

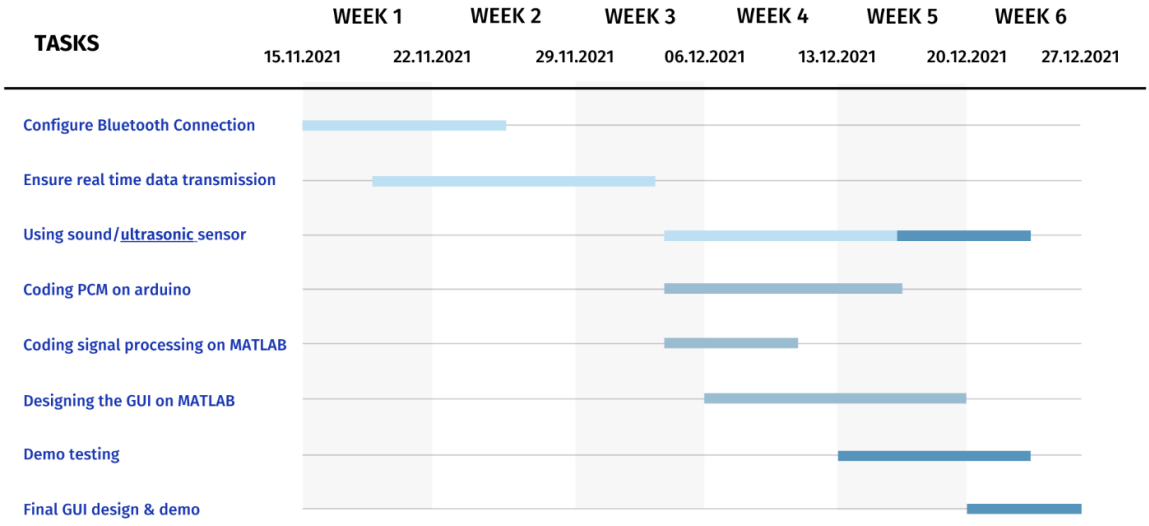

Figure 1 – Gantt Chart of the Schedule

II. INTRODUCTION

A. Communication Process

This project is based on Pulse-Code Modulation via Bluetooth communication. There are several steps that are supposed to be applied to do PCM, which are Low pass filtering to reduce noise, sampling to make the signal discrete in time, quantize to digitize signal amplitudes, and encoding. After those steps, our signal became ready to send with a significant rate of efficiency and accuracy. At the receiver side, the signal is supposed to be decoded according to the encoding method, then applied Low-pass filter as a reconstruction filter. After these steps which are shown in Figure 2, we can observe the original signal with a relative amount of error according to parameters of modulation.

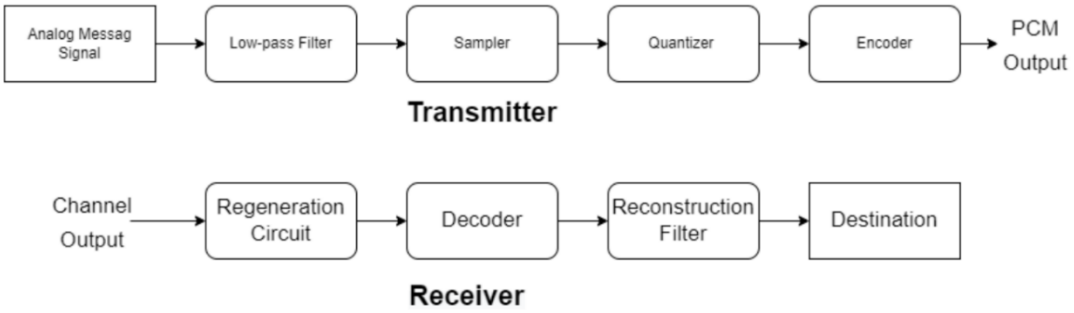

Figure 2 – Pulse Code Modulation Process

HC-06: HC-06 is a common Bluetooth module among electronics projects which has 2.4GHz built in antenna on it and 6 pins to send/receive data, supply, and ground as seen in the Figure 3. It is capable of working at different baud rates from 1200 to 1382400 and its default baud rate is 9600. It is sufficient for most of the applications because it is relatively fast and has -80dBm bit error rate [1].

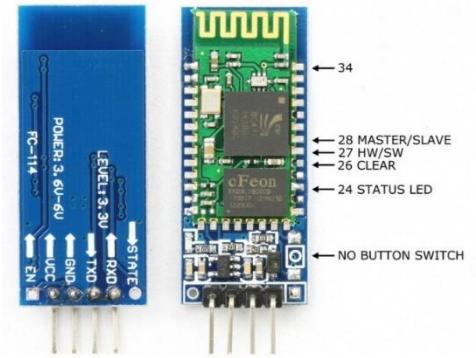

Figure 3 – HC-06 Module & Pins [2]

B. Hardware

To capture analog signal from a sensor and do modulation, a computational hardware is needed. Thus, we used Arduino Uno R3 with its microcontroller of Atmega328p. Atmega328p is a microcontroller from the company of Atmel. It is an 8-bit microcontroller, and Arduino Uno is a development board that bases this microcontroller for its main purposes. In other words, Arduino is an environment that reduces the complexity of programming a microcontroller thanks to its integrated development environment and easy-to-use hardware solutions. Arduino Uno R3 is one of those solutions. It has 14 digital general-purpose input/output pins and 6 analog input pins as seen in Figure 4.

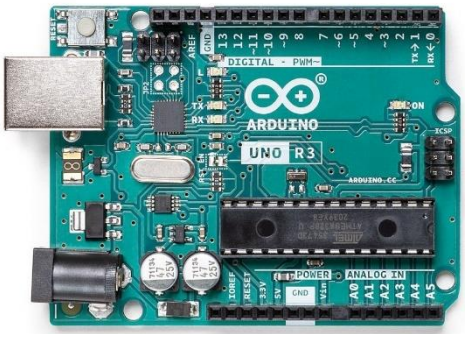

Figure 4 – Arduino Board [3]

Timer & Interrupt: Interrupt is quite fatal feature of any computational hardware. Briefly, when a device raises an interrupt request, which may be from hardware or software, the current program is being stopped, interrupted, and desired or dedicated action is being done. When interrupt routine finishes, the program continues from where it stopped [4]. The switch button of our personal computers is a great example of interrupts. While operating system is running, if we press the button, the PC instantly stops doing what it was doing and shot itself down. This is what we want for our sampling process, but we need one more feature which is the timer.

A timer is a piece of hardware that basically works as a counter. There are different timers in a single chip for multiple and various usages, such as pulse width modulation, delays and so on. Also, we can program our Arduino to interrupt the program when the timer reaches our sampling period. There are different types of parameters to use for our purposes such as modes and Prescaler. In short, the timer ticks at every clock cycle when Prescaler is 1, but when it is 8, it ticks at every 8 cycle. So, we need to adjust our parameters to reach our desired sampling period without coming across counter overflow. When it reaches that level, we can make it to call an interrupt request.

C. Software

In the software side, we need a flexible tool to try different things on real time signal processing. We don't need too much memory or computational speed, so we can use MATLAB. MATLAB is a high-level programming language which is specialized for matrix calculations, and it is full of libraries for various areas such as communication, signal processing, artificial intelligence. Since it is easy to use, trying different implementations can be rapid for Bluetooth communication and the demodulation process. Furthermore, it has another feature called App Designer which is a simple application to build a graphical user interface with drag and drop method [5]. It provides an easy environment to convert MATLAB programs into visual interfaces. This GUI can fulfill our real time visualization need, combined with the MATLAB signal processing.

D. Collecting Real Time Data

We tried different components to collect real time data, as well as sound detector that gives us more dynamic and functional data and distance detector which gives us more consistent data. The components we used for data generation are KY-037 sound detector, HC-SR04 Ultrasonic Distance Sensor and SG90 9G Servo Motor and of course the HC-06 Bluetooth module we explained at first subsection.

KY-037 Sound Detector: KY-037 is a sound detection sensor that detects sound with its microphone and amplifies it with an analog circuit printed on the module. It has 4 pins as seen in Figure 5, they are analog, digital, ground, and supply pins. It has a potentiometer on it to control its sensitivity and threshold for digital output.

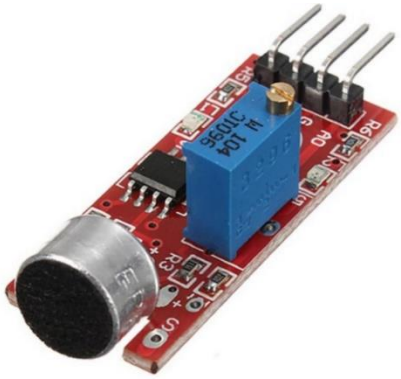

Figure 5 – KY-037 Board [6]

HC-SR04 Ultrasonic Distance Sensor: HC-SR04 sensor module contains an ultrasonic transmitter, an ultrasonic receiver and a control circuit as seen in the Figure 3. Its range is 2cm to 400cm with 15 degree measuring angle. Basically, the transmitter sends 40kHz audio 8 cycle sonic burst, and the receiver is being triggered when it captures the signal back [7]. According to the time passed and speed of sound in air, we can calculate the distance thanks to the formula in the Figure 6.

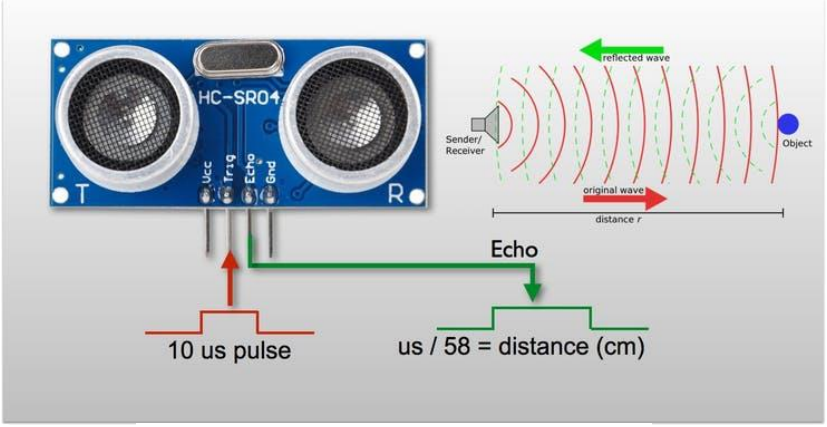

Figure 6 – HC-SR04 module and its pins [8]

SG90 9G Servo Motor: Servo motor is a motor which used for precise rotation movements. They mostly detect their positions with their feedback mechanism, and they are capable of 180 degree rotation. SG90 9G is cheap and relatively week servo motor with its 1.8kgf/cm stall torque [9]. It is controlled with pulse width modulated voltage signals to set its positions with speed of 0.10 sec/60° as seen in the Figure 7. It is worst drawback is that they are very fragile and poor quality, but other than that they are sufficient for small sized applications.

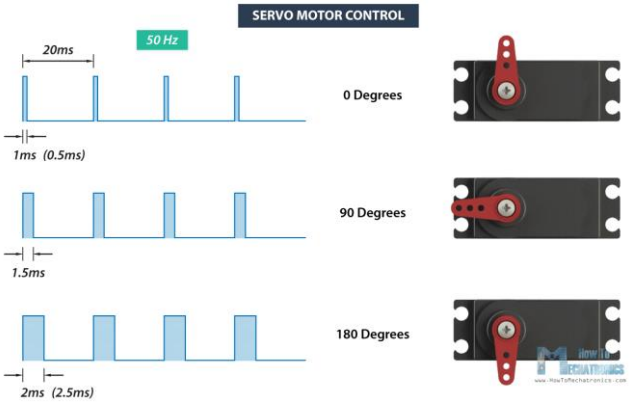

Figure 7 – Driving Servo Motor with PWM signal [10]

III. SYSTEM DESIGN

A. Transmission of Data

In this project, we wanted to send our real time data via pulse code modulation. To do pulse code modulation, we have to follow some distinct steps, first one is sampling. Although, Arduino samples data while reading data from its GPIO pins, it is not periodic and predictable. Therefore, to sample our data precisely, we used interrupt and timers. If we used built-in delay function of Arduino, we would encounter some problems such as inconsistency because even if we add some delay to a capturing mechanism, we cannot measure how long does it take to read, send data and so on. Thanks to timer feature of Arduino, our board is requesting interrupt at every 200ms. For this sampling period, we used CTC timer mode to reset when timer reaches tick number for 200ms.

We calculated this number just like equation below:

$$Speed\ of\ Arduino\ Board = 16MHz$$

$$Ordinary\ tick\ period = \frac{1}{16}us$$

$$Timer\ tick\ number = \frac{200ms}{\frac{1}{16}us * precaler}$$

$$With\ 256\ prescaler => Timer\ tick\ number = \ 12500$$

So, when 12500 tick occurs in timer2 of Arduino Uno, which is only 16-bit timer on the board, it resets itself and requests timer interrupt. Inside the interrupt service routine, we read the data from the dedicated pin. Due to this process, we precisely sample our data according to our order.

For the second step, we need to quantize analog signal, coming from the sensor. Actually, Arduino Uno has 10-bit analog to digital converter, which means that every analog data is quantized to 10-bit digital signal inside the Arduino hardware. Therefore, the only applications can be done are reduction the quantization level or interpolation. We prepared a code block to change the quantization level, but we set it to 10-bit, so it leaves the signal how it is coming from the ADC.

Last step for the transmitter side is encoding. We preferred to use binary encoding for simplicity. Sampled and 10-bit quantized signal is turning into 10-bit unsigned binary number. Thanks to this encoding we are reducing the error rate although Bluetooth module has pretty low error rate.

For the receiver side, which is being done by MATLAB, we are decoding the signal according to modulation parameters. We built our system flexible for different parameters. Bluetooth module is bombarding the computer with data so, if we enter right encoding parameters, which is 10 bit binary encoding in our application, we are able to read correct values. Otherwise, with wrong encoding parameters MATLAB is unable to decode the data correctly.

Bluetooth Connection: For the wireless transmission, we connected HC-05 module to Arduino and use it for mainly transmitter. Arduino has serial communication library that helps us to define our communication type of module and use it with directly write and read commands. We send our signal after encoding operations in the end of interrupt function. On the other end, we used computer's Bluetooth on MATLAB to operate as a receiver. We used the built-in functions of MATLAB that helps to find Bluetooth devices, create & end connections, clearing the buffers and reading & writing on Bluetooth connection. We also use the same functions on App designer and they work fine there.

B. Real Time Plotting

To visualize anything on this project, we need a real time plot. At first, we used plot function with drawnow function, in a for loop. This seems working, but we realized this method overwrites all the data on old data plus the new sample, which means a huge redundant computational work. To eliminate this case, we use a different plotting method, which adds points to figure one by one. To use it, we define an animatedline object, then at every sample, we add points to this object, then using drawnow function, we could plot it fast and efficiently. This plotting method uses animation of change in the data in plotting, but in our case, we can't see this animation because of the fast and tight plot of new samples.

C. Sound Detection Sensor Applications

To detect sound, we used KY-037, which is both analog and digital sound detection sensor. We first built our circuit as seen in the Figure 8, connected Bluetooth module and fed KY-037 with reading its analog output by analog pins of the Arduino board.

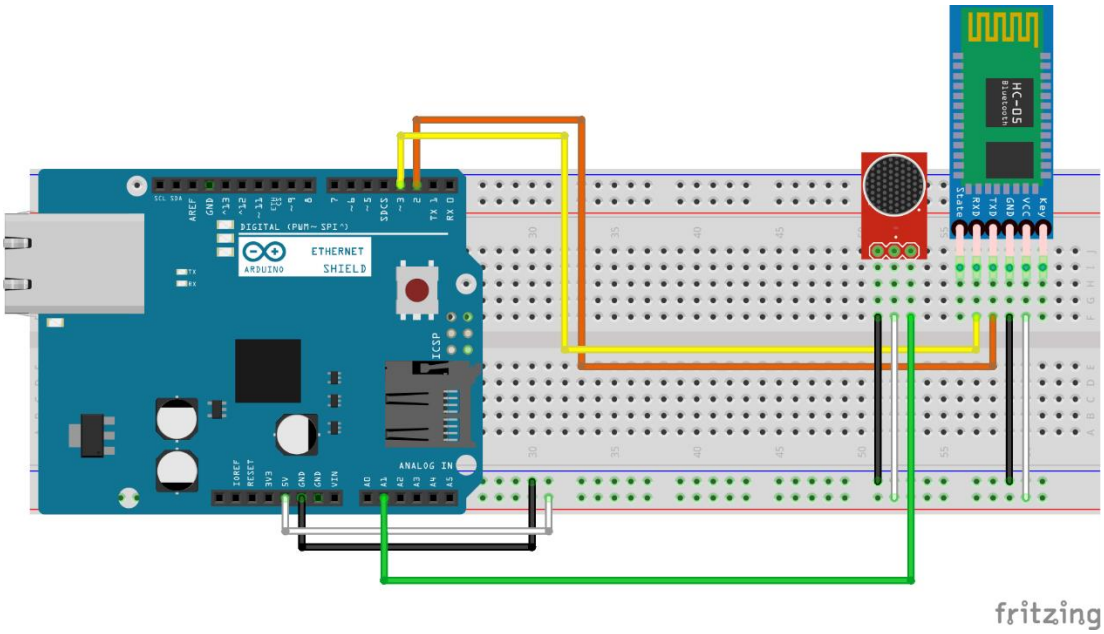

Figure 8 – Schematic of the Sound Detection Circuit

The first thing we have done was setting the sampling frequency to detect at least 1500Hz sinusoidal wave. Hence, we set the sampling frequency of the Arduino greater than Nyquist frequency which is 3000Hz. After that, we started to do our test if the sensor works properly. Most of the time we see noise like visuals at our plot, only for distinct pitches we were able to see some meaningful figures. Also, there was another problem, at some frequencies, we observed ordinary sinusoidal wave like figure but for others we saw envelope like signals with a carrier frequency as seen in the Figure 9, which is not logical.

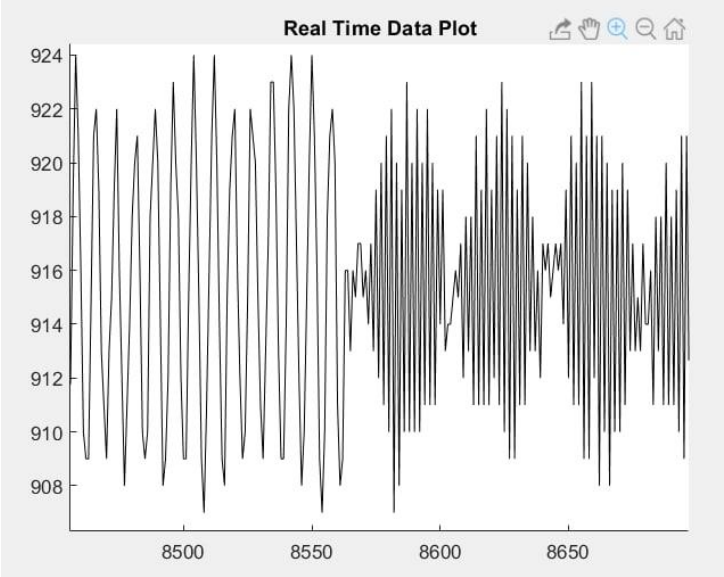

Figure 9 – Inconsistent Result of the Sound Detector

Moreover, at some specific pitches, which are mostly corresponding to a musical note, we are able to observe sinusoidal wave. As seen in the figure 10, 11 and 12 below, we see waves with increasing frequencies. We see spikier figure at lower frequencies because sampling frequency is more than enough and the sensor is not as accurate. When both are combined, we see these kinds of visuals.

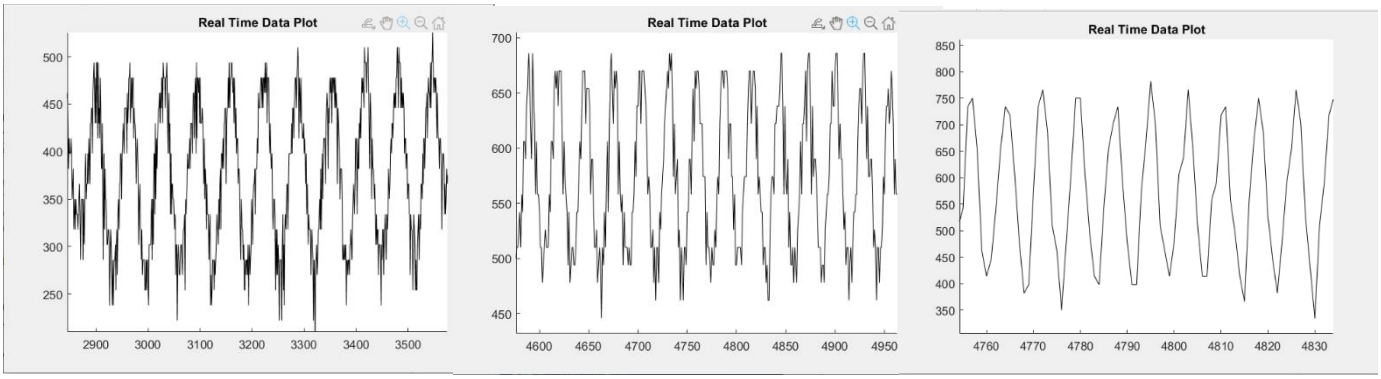

Figure 10,11,12 – Different Frequency Detection Results – Increasingly Ordered

There are various applications we can do with our sound data, but we realized that with this poor quality, we cannot play the sound on the computer. Thus, we focused on other things like finding frequency from captured signal. At first, we had to smooth our data because spikes are creating high frequencies that we do not want so we want them to be eliminated with smoothing. We used different smooth and interpolation techniques such as, moving average filter, Savitzky-

Golay filter, moving median filter, and Gaussian filter, but the real time filtering caused some problems on interface, like slowing the app and plotting with the delay. Additionally, to find the frequency in real time, we tried to apply fast Fourier transform at every 100 bauds of data, and by finding the maximum peak of the FFT, we tried to calculate dominant frequency of the most recent sound signal.

However, both methods were not as accurate as we aimed. These were not just because of techniques, also poor sensor output. Moreover, the row output got worse later on, we could not even see any sinusoidal waves. Thus, at this point, we decided to change our inaccurate sensor, so some GUI applications and the Arduino code as well, maintaining pulse code modulation, demodulation and Bluetooth communication parts.

D. GUI Design

MATLAB has an easy-to-use GUI maker of App Designer. To design our interface, the first things we did are deciding what features we want and how our template is supposed to be. We wanted to see 2 plots at the same time. The one above is demodulated data and the one below is modified or processed data as seen in the Figure 13. With push buttons, we wanted to setup Bluetooth connection and start the data stream. Lastly, with a vertical slider, we wanted to scale our plot window real time. Since MATLAB is high level programming language, it is not designed for fast real time multitasking processing. Therefore, we come across some problems to plot 2 figures at the same time because it obviously slows the whole process down and cause big delay.

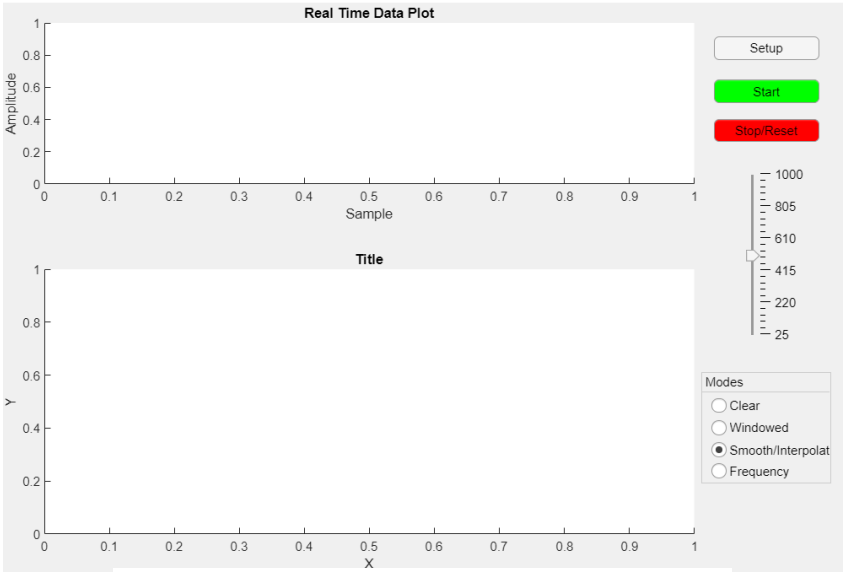

Figure 13 – First Real Time Plotting GUI Design

To overcome this problem, we set only one figure on the interface, and it changes according to mode switch. Since we changed sound detection sensor to distance sensor, we changed modes as well.

To tell all features of the app seen in the Figure 14:

- Connection light which gives us feedback about Bluetooth connection.
- Setup button does connection with computer with Bluetooth module.
- Stop button freezes the real time plot to give us change to observe the data.
- Disconnect button unlinks the connection between computer and Bluetooth module.
- Radar on/off button send a signal to Arduino to run servo motor to use radar feature.
- Vertical slider sets the size of x axis to zoom in or out horizontally.
- Speed mode calculates the speed of the object in front of the radar.
- Deviation mode draws the changes happening on the plot.
- Angle mode show us the angle of the servo at that time.
- Radar mode opens polar axis plot to see 180-degree radar like plot.
- Distance mode directly draws the decoded distance data.

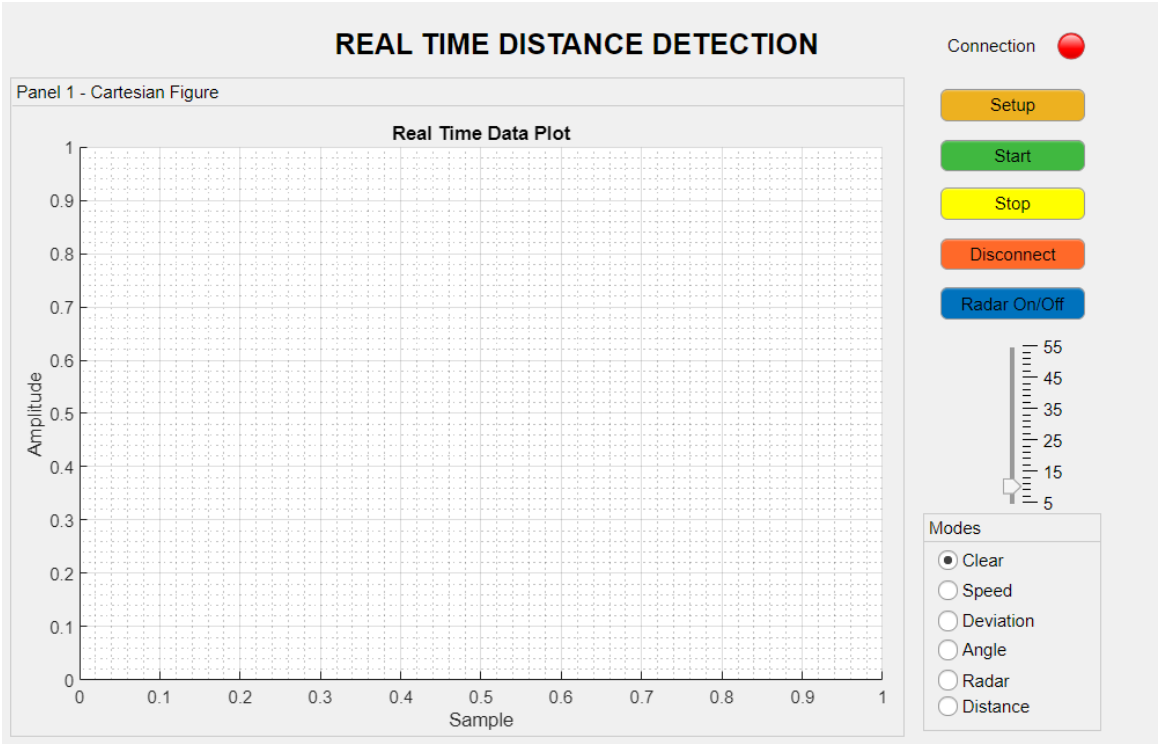

Figure 14 – Final GUI Design

E. Ultrasonic Distance Sensor Applications

With the ultrasonic sensor, we planned to create distance measurement, speed detection and object detection visualizations. We changed our circuit to Figure 15, in this figure we can see we added two components instead of the sound detection sensor. The HC-SR04 sensor gives us more consistent and reliable measurements instead of KY-037, and servo motor gives HC-SR04 a movement ability, which we used for measurements for different angles.

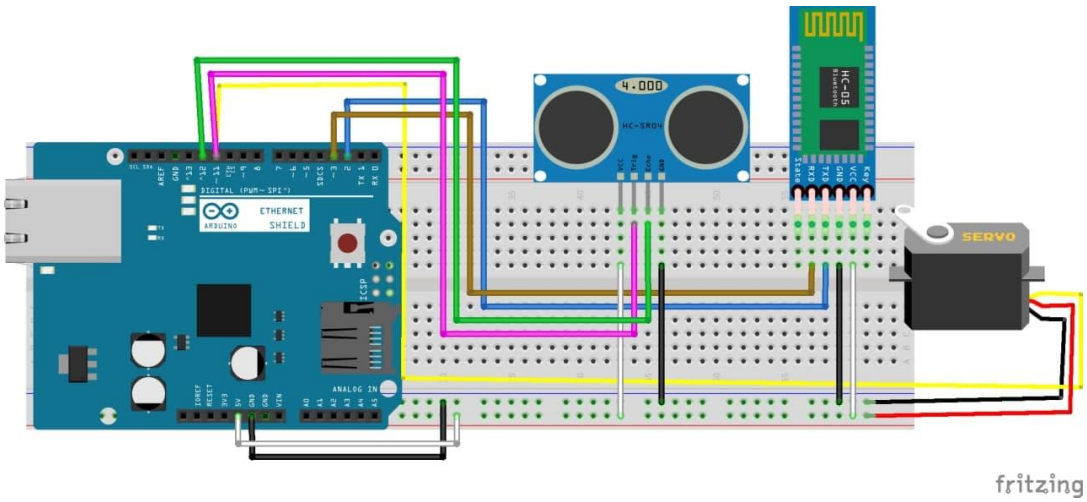

Figure 15 – Ultrasonic Distance Sensor Circuit

The first thing we do is changing the Arduino code. For using HCSR-06 we can directly use two pins for trigger and echo pins and create a distance calculation with parameters of speed of sound. But for using servo motor is more problematic. The interrupt we used at the beginning, can't used with servo control library because of the conflicting. So, we found another library to use timer 2, which can be used with servo motor. Same as the first circuit, we made all the operations in the timer. In addition to first design, this time, we have two data, first is the distance measured by HC-SR04 and second is the angle of the servo motor. We encode them together according to Figure 16, and send both data together. For the receiver side, we use same encoding bit numbers to decode the data, and use the GUI operations to process and visualize them.

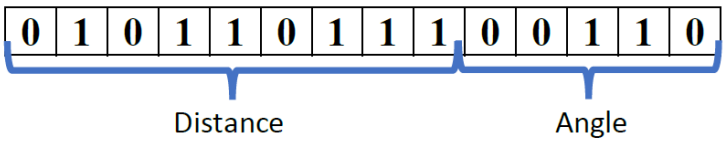

Figure 16 – Encoding for Ultrasonic Sensor Application

IV. RESULTS

At the final design, we put our HC-SR04 on top of the servo motor and glued them together as can be seen on Figure 17. At any part of our design, we tested components and our codes for errors. For example, in the first week, we used a simple blink example to test Arduino & pin functionality. Then we write other basic Arduino codes to test components, and also basic data transmission examples to test our MATLAB operations as well. Thanks to these tasks, in the final design, we are sure for operations of our components and codes individually. So, for testing the circuit, we directly used final GUI and final circuit for different areas and objects.

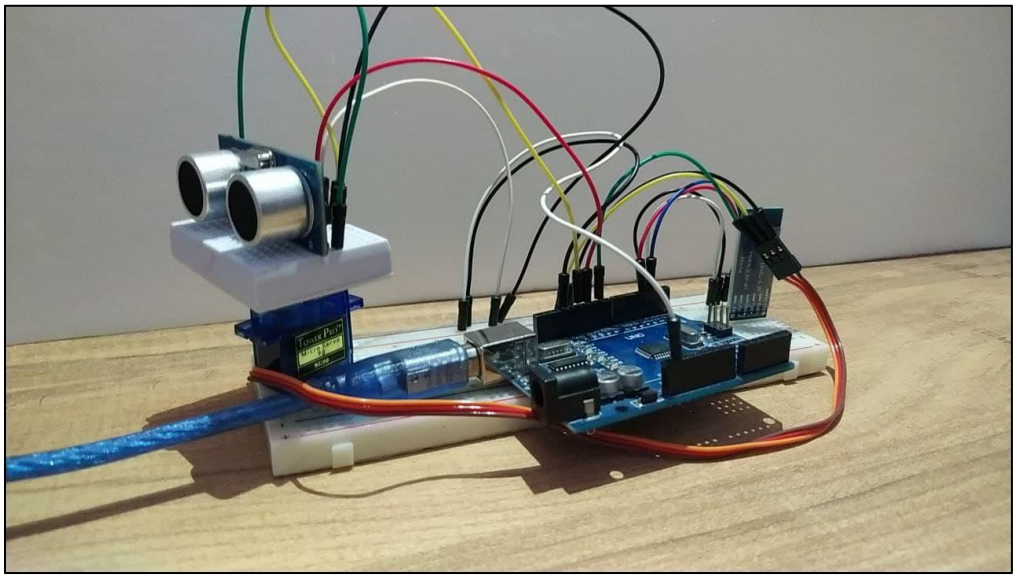

Figure 17 – Final Circuit with All Components Attached Together

We tested our circuit for all the operations of GUI, including the connection, disconnection, start, stop, clear buttons, and the modes that calculates and plots the distance, speed, deviation, of measurement, angle of servo motor.

1. Distance change test: Distance is increased ~10 cm to ~30 cm slowly.

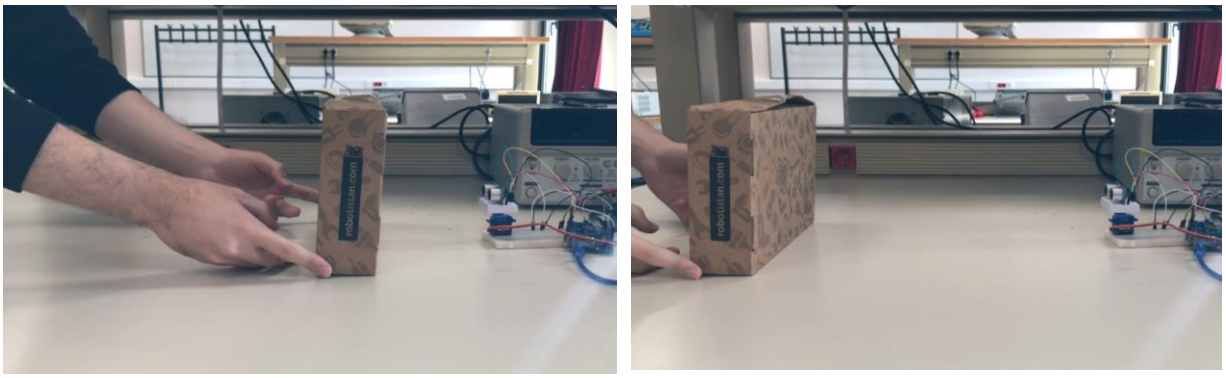

Figure 18 – Distance Test Photos for 10 cm and 30 cm

Figure 19 – Distance Test Result on GUI

We can say that in the Figure 18 and 19 the result bounds are very accurate for 11 and 32 cm, and the change on the distance is very smooth. This smoothness shows the median filtering eliminated the spikes in this short period.

2. Speed measurement test: Distance is increased 60 cm (no object) to 15 cm in one second.

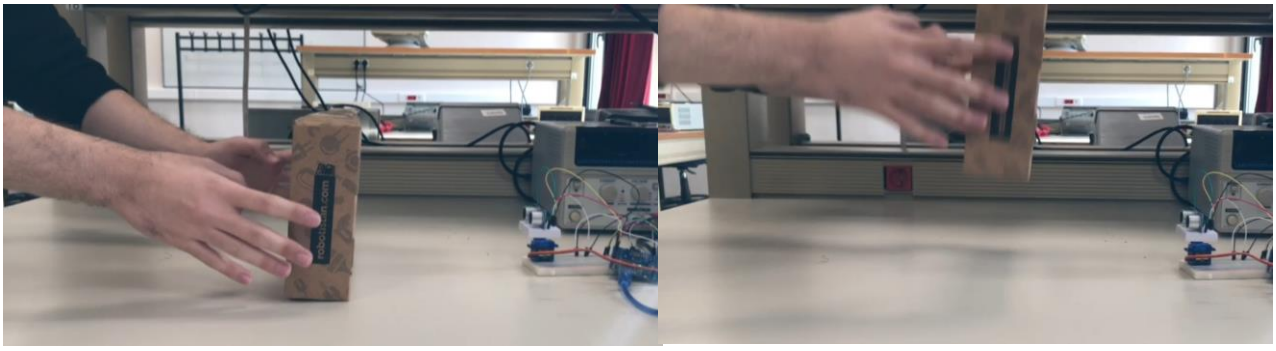

Figure 20 –Speed Test Photos for 15 cm and 60 cm (empty)

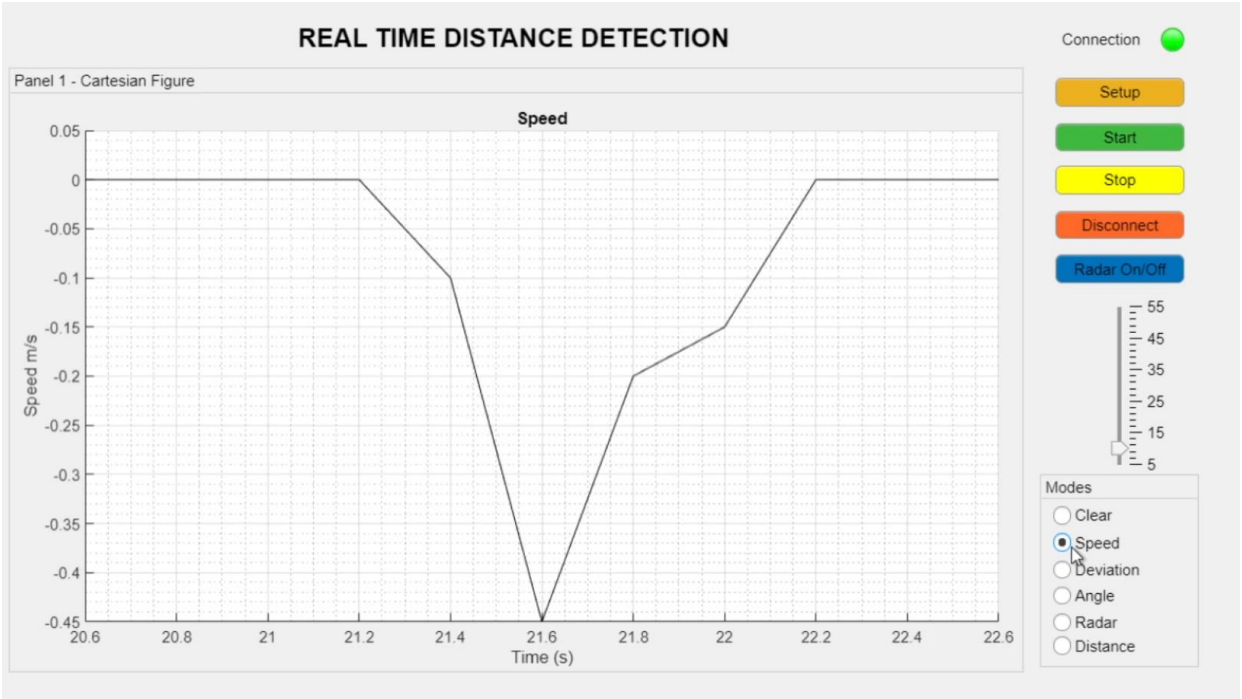

Figure 21 – Speed Test Result on GUI

We can see in the Figure 20 and 21 the result of speed is very accurate for the second sample. We can calculate the speed of the test, which is:

$$15 - 60 = -45\ cm\ over\ 21.2\ to\ 22.2\ seconds \rightarrow \frac{45cm}{1s} = 0.45m/s$$

We can say the speed measurement is limited to 1 second interval, which is 5 samples. We think this limit is comes from the hardware, the ultrasonic sensor we used.

3. Deviation test: Distance is changed positively, stopped, and changed positively again very fast.

Figure 22 – Deviation Test Photos for moving forward, stopping and moving forward again

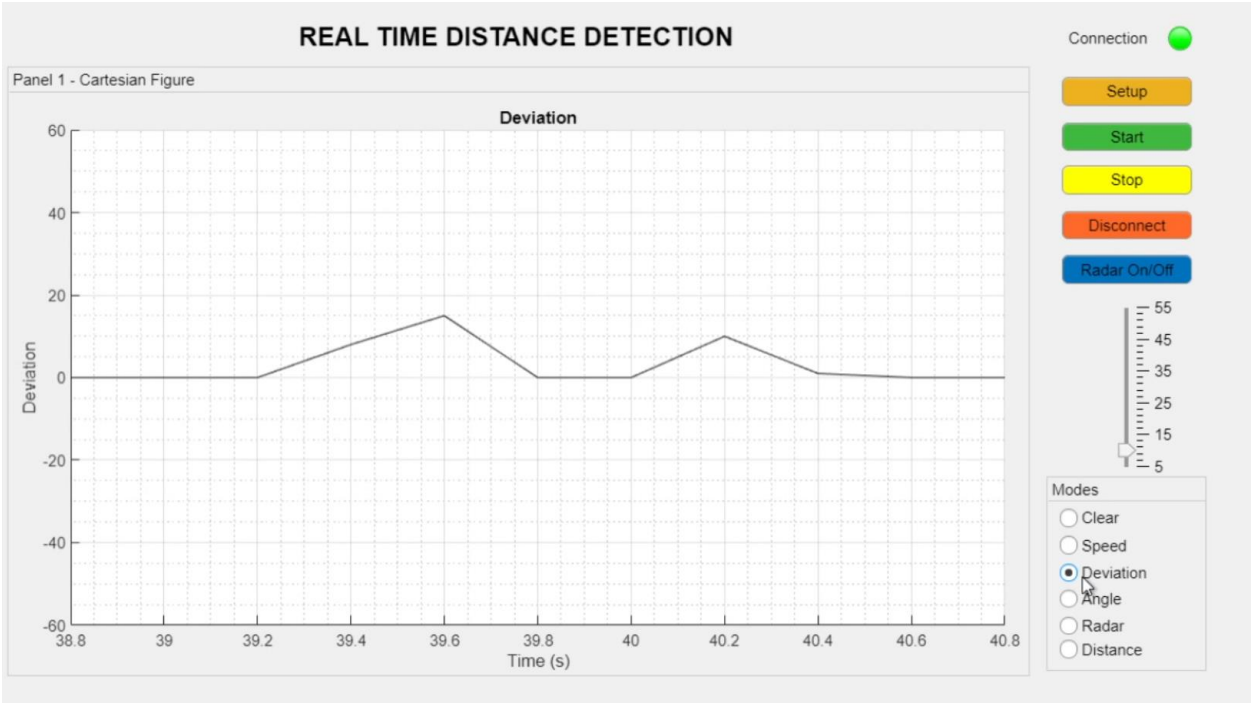

Figure 23 – Deviation Test Photos

We can see in the Figure 22 and 23 the result of change and stop, can accurately detected, and deviation mode doesn't give any result on no change of distance.

4. Radar mode test: Test area with three objects is built. After 180-degree change, the radar plot stopped.

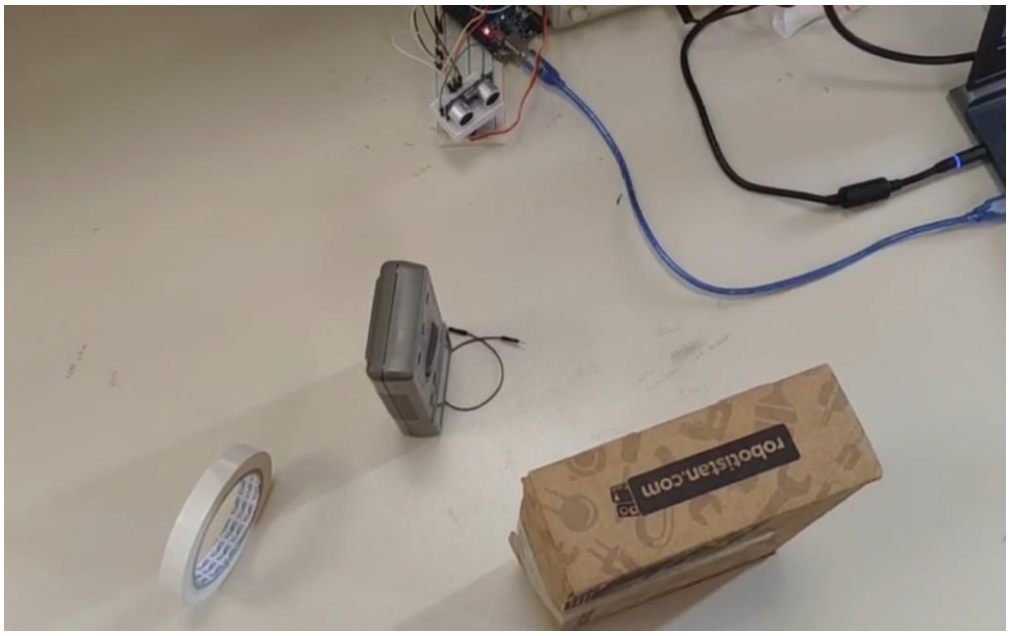

Figure 24 – Radar Mode Test Photo with Three Objects (Angle = 150 degrees at this shot)

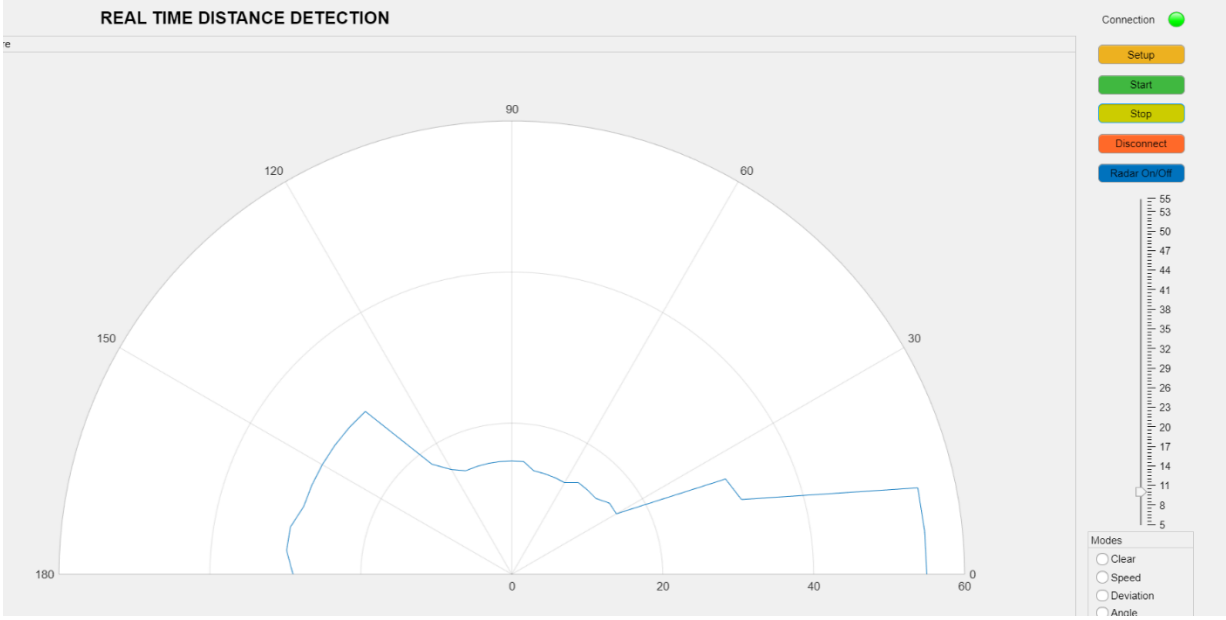

Figure 25 – Radar Mode GUI Result

We can see in the Figure 24 and 25 the radar plot gives four different distance values at different angles. From figure 24 at angle 150, we can say that the big cardboard is 30 cm far, multimeter is ~18 cm far and the tape is again 30 cm far away. And the empty space on the right measured as ~60 cm. We can say that the spaces between objects couldn't detected as an error.

V. CONCLUSION

At the first stage of the project, it was easy to build communication with Bluetooth module and computer because it is a robust communication system, and it is capable of real time communication. Also, we saw that it is simple and reliable way to use pulse code modulation. Thanks to PCM we easily send our real time data consistently with no error and decode it.

Next, we had so many problems with KY-037 sound detection sensor because it is a poor design module. We tried to compensate the drawbacks with software but since the module is inconsistent to sound, we failed to present sound based real time communication project. Moreover, while dealing with real time sound data, we had to apply some filtering mechanism. However, whenever the signal is real time, it is a problem to filter the signal without delay.

Due to the issues of KY-037, we changed our direction from sound to distance with HC-SR04 ultrasonic distance sensor. It is much more consistent device; we had no problem to detect distance. The only drawback is that HC-SR04 occasionally sends low or high peaks due to the misdetections. To figure this out, we used median filter, and it worked quite well although it adds 2 sample delay to our real time plot. Then, while building radar mechanism, we had to solve the problem of sensitivity because sensor is being interfered with any small object. So, even if there is a small object which may cover 5 degree of the radar, it interferes almost 30 degree of the area. It was an existential problem, therefore we leave it as it is because the only solution is changing the sensor.

Lastly, the performance of Arduino Uno board is quite sufficient for real time communication project. The only drawback it has is that its microcontroller Atmega328P has limited timer registers. It causes problems to use different devices using the same timer. Additionally, MATLAB was not great solution for fast and real time data analyzing because it is not capable of refreshing the plot at high frequency. However, simple and rich libraries of it and App Designer solution reduced our algorithmic work down and help us really much.

REFERENCES

[1] T. powerberry: C. A. Perani, "Like a bat with HC SR04," *Hackster.io*, 02-May-2018. [Online]. Available: https://www.hackster.io/powerberry/like-a-bat-with-hc-sr04-829486. [Accessed: 29-Dec-2021].

[2] E. Styger, "Valley of Schwyz," *MCU on Eclipse*, 16-Aug-2014. [Online]. Available: https://mcuoneclipse.com/2013/06/19/using-the-hc-06-bluetooth-module/v. [Accessed: 29-Dec-2021].

[3] "Arduino Uno REV3," *Arduino Official Store*. [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3/. [Accessed: 29-Dec-2021].

[4] "Interrupts," *GeeksforGeeks*, 01-Apr-2021. [Online]. Available: https://www.geeksforgeeks.org/interrupts/. [Accessed: 29-Dec-2021].

[5] "Matlab app designer," *MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/products/matlab/app-designer.html. [Accessed: 29-Dec-2021].

[6] "KY-037 microphone sound detection module," *Opencircuit*. [Online]. Available: https://opencircuit.shop/product/KY-037-Microphone-sound-detection-module. [Accessed: 29-Dec-2021].

[7] "Ultrasonic ranging module HC - SR04 - SparkFun Electronics." [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Accessed: 29-Dec-2021].

[8] T. powerberry: C. A. Perani, "Like a bat with HC SR04," *Hackster.io*, 02-May-2018. [Online]. Available: https://www.hackster.io/powerberry/like-a-bat-with-hc-sr04-829486. [Accessed: 29-Dec-2021].

[9] "SG90 9 G Micro Servo - Ullis roboter Seite." [Online]. Available: https://ullisroboterseite.de/projekte-teedipper/SG90%209%20g%20Micro%20Servo.pdf. [Accessed: 29-Dec-2021].

[10] "How servo motor works & how to control servos using Arduino," *HowToMechatronics*, 08-Oct-2021. [Online]. Available: https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/. [Accessed: 29-Dec-2021].