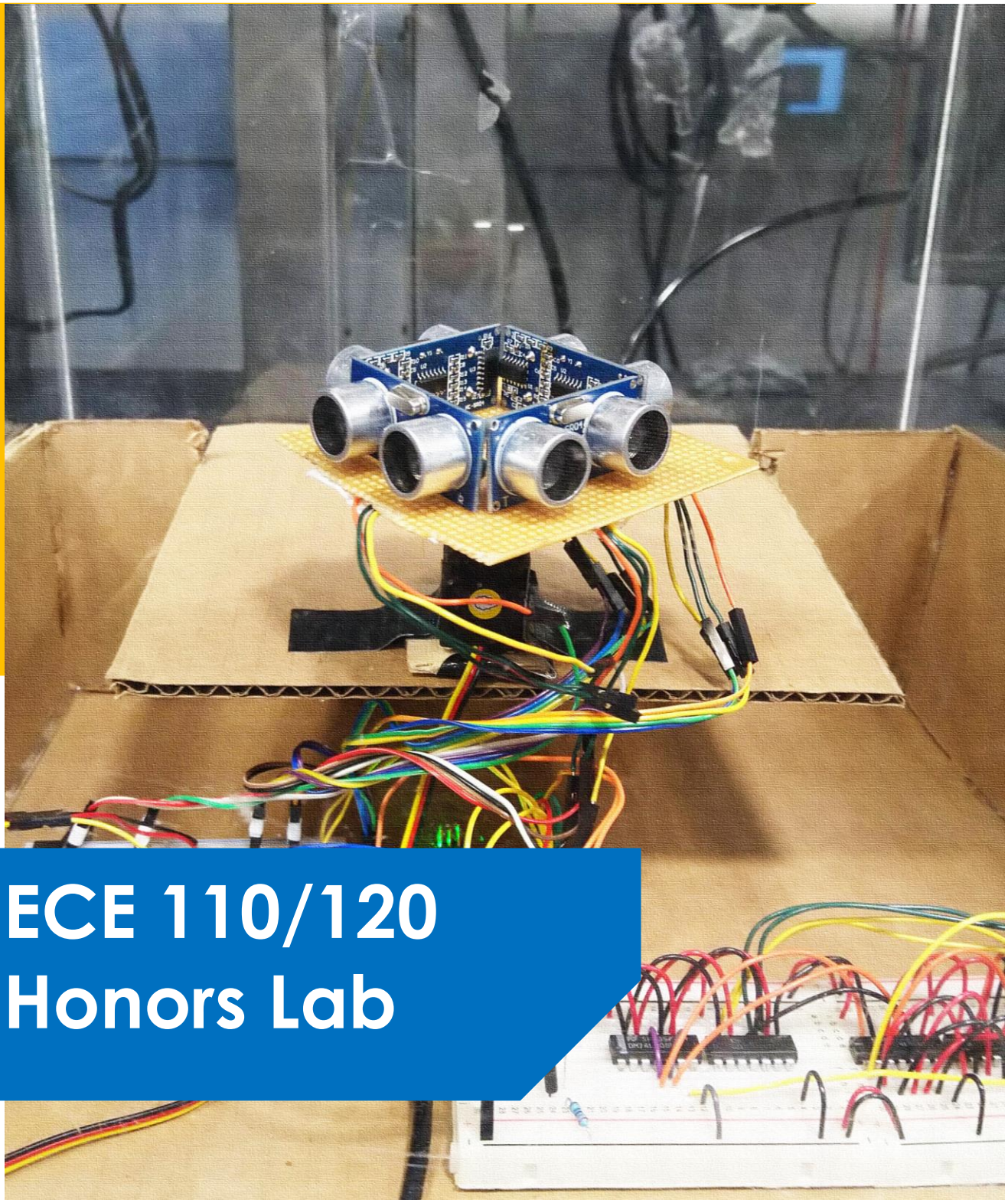


ECE 110/120 Honors Lab

Project Report

Xinglong Sun (xs15)

Aryan Prinjha (prinjha2)

Tonglong Liu (tl20)

TABLE OF CONTENTS

Introduction ____ 3

Problem Description ____ 3

Design Concept ____ 3

Analysis of Components ____ 4

Sensor Characterization ____ 4

Sensor and Data-Stream Health Detection ____ 5

Design Description ____ 6

Physical Construction ____ 6

Design Diagrams ____ 8

Arduino Functioning - Block Diagram ____ 8

Sensor – Data Stream Error-Detection ____ 9

Python Code Analysis ____ 10

Results & Project Demo ____ 12

Working Demo of Our Project ____ 12

Scan the Barcode to view the project in motion ____ 12

Conclusion and Self-Assessment ____ 13

Future Improvements and Applications ____ 13

Lessons Learnt ____ 13

Self-Assessment ____ 13

INTRODUCTION

> "The innovation point is the pivotal moment when talented and motivated people seek opportunity to act on their ideas & dreams." – W. Arthur Porter

Problem Description

Some industrial products develop cracks or fissures on their sides due to defective production or time. These cracks if left undiscovered, can cause excruciating problems and consequences. However, it's not easy to spot them early on, especially in finely honed and elaborate products. Therefore, in this honor lab, we intend to make a crack detector robot based on ultrasonic sensors and display a data map on the screen using Python after collecting and processing.

Design Concept

The entire project consists of following components: **Data Collection (Four Ultrasonic sensors), Sensor Base Rotation via Servo, Sensor data validity and Health, Data Plotting and Analysis**. The Arduino controls the sensors rotating part and take in values measured by the data collecting unit consisting of 4 Ultrasonic Sensors mounted on a servo motor. After the data collecting process, Arduino sends those data through serial port to Python which starts to plot graph in polar coordinates accordingly.

Since the data map is created based on the distances measured by Ultrasonic sensors, a decent data map requires very high precision sensors. Due to limited financial and quality resources, we used HC-SR04 Ultrasonic sensors to test our prototype. With high precision sensors, our model will easily show a crack as a spike on the data map generated.

ANALYSIS OF COMPONENTS

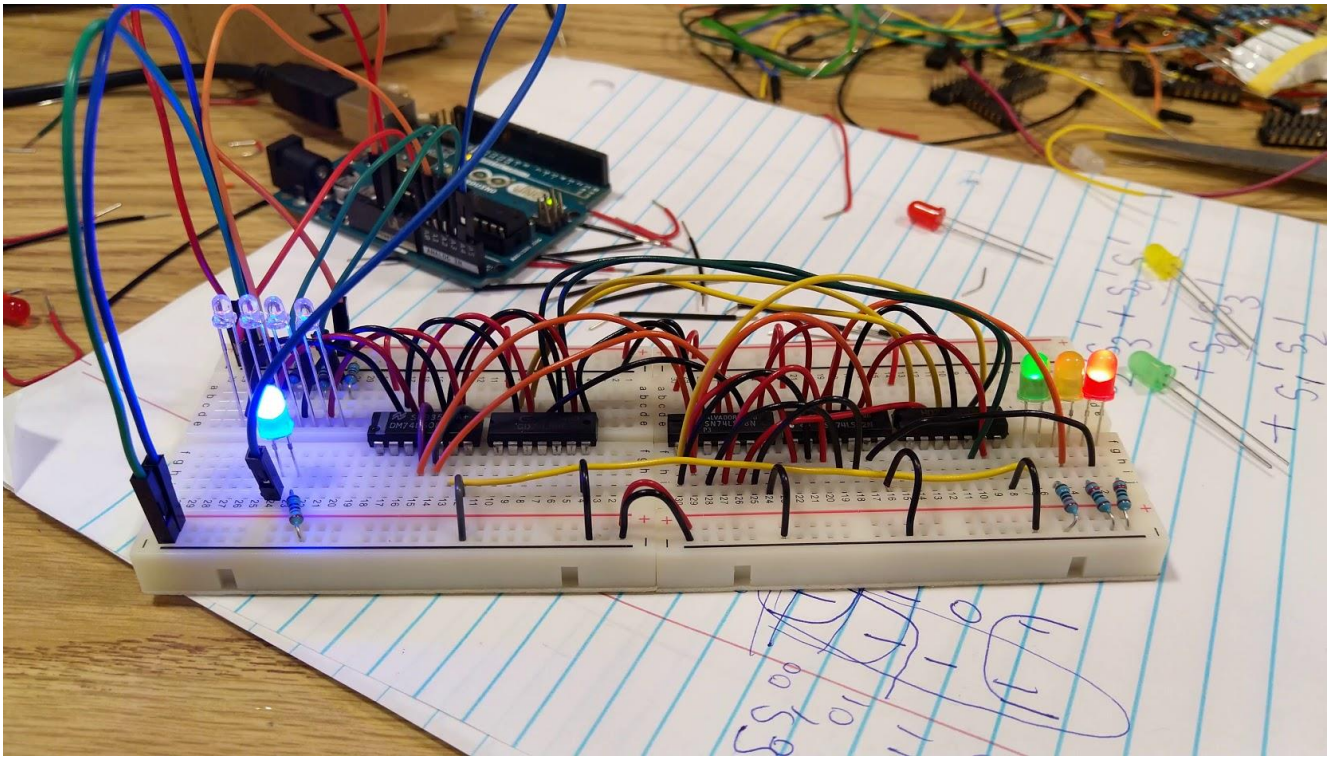
Sensor Characterization

We incorporated four Ultrasonic sensors in our final design. An Ultrasonic sensor is essentially a sensor that measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected from the target. The sensors on this robot are the HC-SRO4 sensors. From the datasheet from the manufacturer, we obtained the following parameters:

Property – HCSR04 Ultrasonic Sensors	REVENUE
Working Voltage	DC 5V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
Measuring Angle	15 degrees
Trigger Input Signal	10us TTL Pulse
Echo Input Signal	TTL level signal
Sensor Physical Dimensions	45*20*15 mm

As our project progressed, we discovered that these sensors are not necessarily the most reliable ones. Frequent abnormal readings were measured during testing. However, a high precision sensor would easily fix these underlying problems and ensure Industry 0 grade reliability of our project.

Sensor and Data-Stream Health Detection



Outside Circuit for Sensor Error Detection

This component is responsible for relaying the accuracy of the data being transmitted by the sensors. It has been constructed by using the properties of K-Maps and Boolean simplification. Based on the Boolean equations the following components were used in its construction –

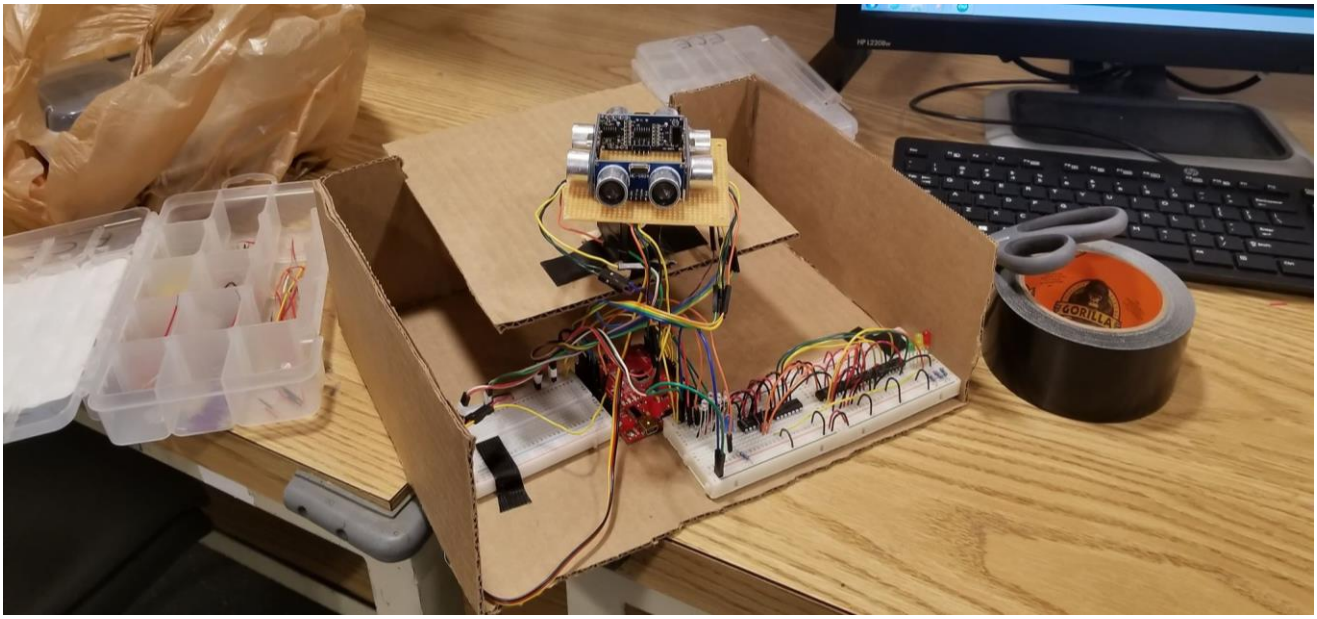
Part Name	Part Number
Quad 2-Input NOR Gate	74LS02
Hex Inverter	74LS04
Quad 2-Input AND Gate	74LS08
Quad 2-Input OR Gate	74LS32
Quad 2-Input Exclusive OR Gate	74LS86

DESIGN DESCRIPTION

Physical Construction

Our project consists of mainly 4 prime sections:

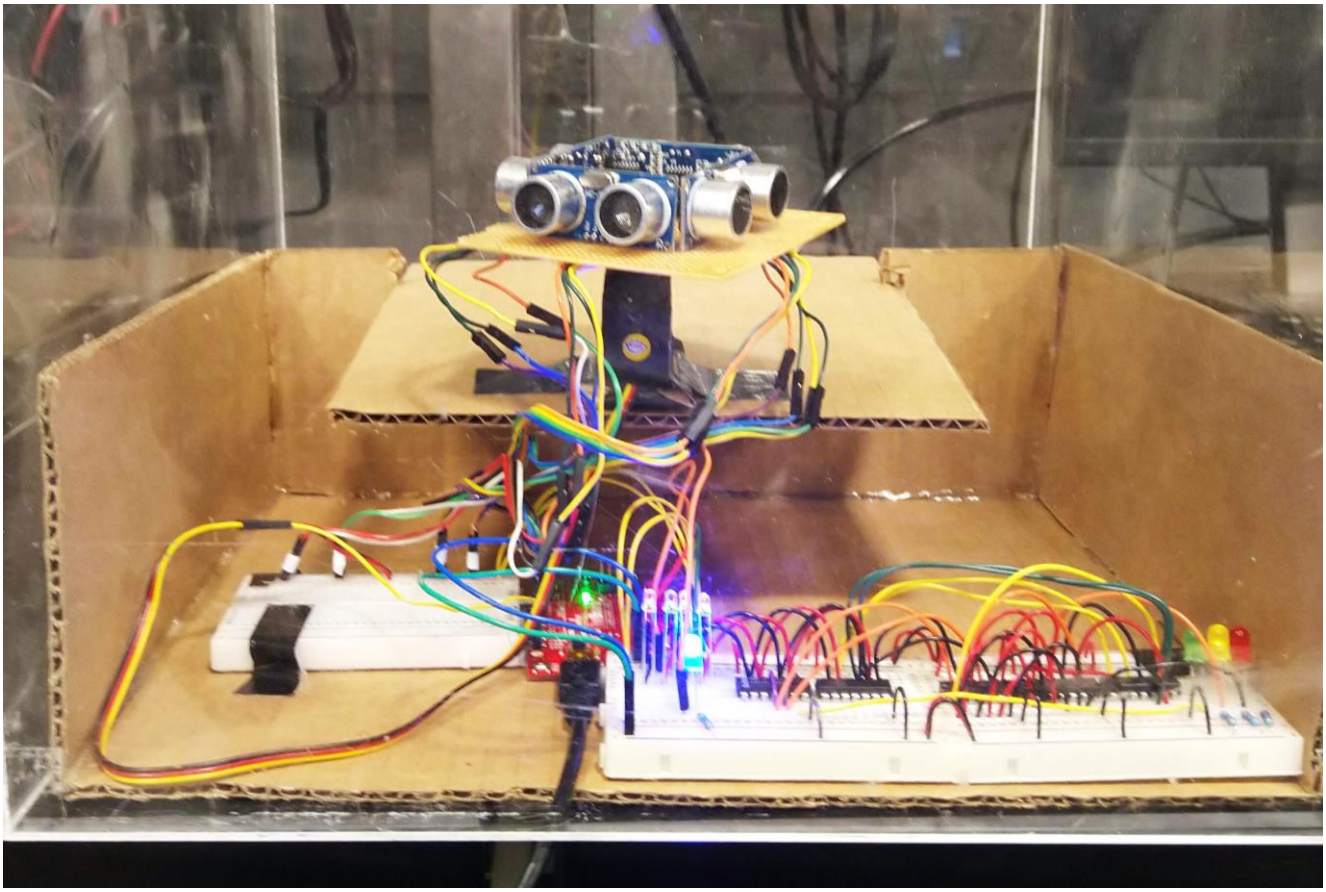
1. Four Ultrasonic sensors mounted on a Servo motor base
2. A breadboard responsible for the connections between the sensor
3. An Arduino that carries the code
4. And Finally another breadboard with LED indicators



There is a cardboard casing around the mapping robot. This case contains a cardboard platform on which the servo motor and the sensors are placed. The reason for placing the servo motor and the sensor in a higher position is that during the rotation of the circuit board, the connection wires require more area to move and rotate, resulting in entanglement to a certain degree. By placing the servo motor and the sensors at a higher level, this issue is easily resolved.

The servo motor is glued onto the circuit board, carrying the four ultrasonic sensors, using a glue gun. The casing is also made via a similar procedure. The Ultrasonic sensors are soldered onto the circuit board for better stability and connectivity.

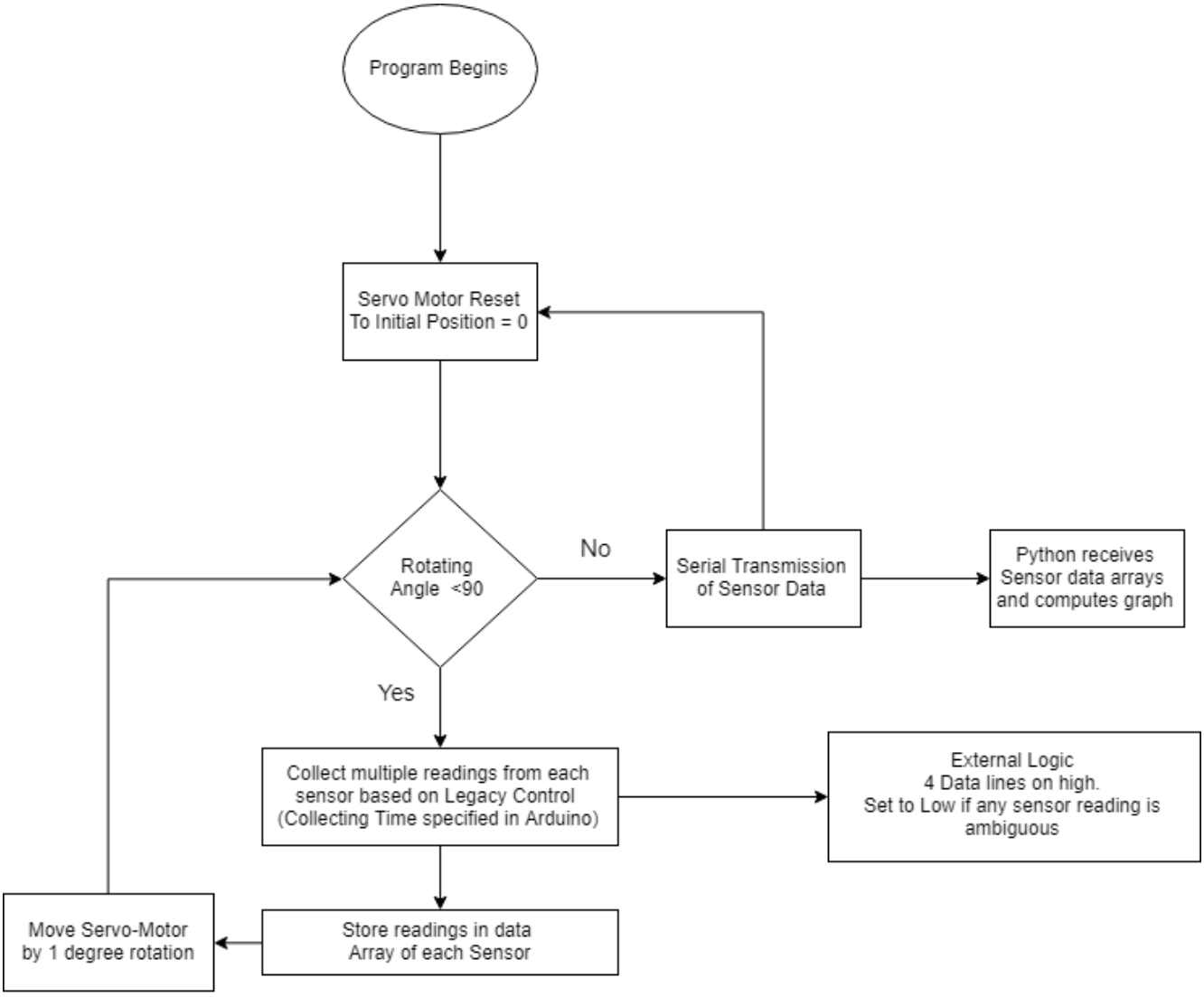
The connecting wires are also soldered on to the sensors for the same purpose. The breadboard and Arduino-Redboard are taped together using a duct tape to prevent any connection wires from popping out of their slots. All the components are fixed to the case by using an industrial tape. The only potential defect that we could not solve in the time frame was a support for high weight of the servo motor. As a result, the supporting platform is slanted downward at a small angle. Although that is not a significant issue, it can be improved and put more thought into.



The Project

Design Diagrams

Arduino Functioning - Block Diagram



As shown above, the flowchart of the entire design is not complicated. The servo motor drives four ultrasonic sensors placed on it to different angle positions. The sensors measure distances multiple times (reason for that will be explained later) and store each measured data in an array. After the entire data collecting process, Arduino sends those data to Python (a detailed explanation of python code will be performed later). Also, during the process, four output signals from Arduino will go through an external logic that tests if each sensor is working proper or not.

Sensor – Data Stream Error-Detection

The Arduino code measures sensor data, multiple times at each degree up to 90. At each point it gets an array of readings. These readings are compared and averaged. Four pins on the Arduino, from A0 to A3 are initially set to high. They correspond to each of the 4 sensors. If the deviation between readings at each degree vary greater than numerical value ten, then the corresponding pin to the faulty sensor is set to low.

The Logic gate circuit receives 4 Inputs in the form of A, B, C and D, each corresponding to one sensor. Based on the input being received, different health state lights, Red, Yellow and Green, are switched on. The K-Map logic for them is as follows.

GREEN	C'.D'	C'.D	C.D	C.D'
A'.B'	0	0	0	0
A'.B	0	0	0	0
A.B	0	0	1	0
A.B'	0	0	0	0

YELLOW	C'.D'	C'.D	C.D	C.D'
A'.B'	0	0	0	0
A'.B	0	0	1	0
A.B	0	1	0	1
A.B'	0	0	1	0

RED	C'.D'	C'.D	C.D	C.D'
A'.B'	1	1	1	1
A'.B	1	1	0	1
A.B	1	0	0	0
A.B'	1	1	0	1

Boolean Expressions for the various stream Health where

Green = All Sensors working as expected GREEN = A.B.C.D

Yellow = One of the Sensor's is relaying false data YELLOW = A.B.(C+D) + C.D.(A+B)

Red = Two or more Sensor's relaying false data RED = A'. (B'+C'+D') + B'(C'+D') + C'. D'

This board also has a Data Light – Blue, which is switched on when data is being transmitted from the Arduino via Serial Transmission to Python for graphing and mapping.

Python Code Analysis

I make use of three Python libraries: serial, matplotlib, and numpy. The "serial" library establishes connection with Arduino board. The "numpy" library performs the data preprocessing. The "matplotlib" library draws the graph. As I found that there are very limited resources on the Internet about getting Python to "cowork" with Arduino, I will provide more details in this section about some tricks of making the Python code run smoothly.

The first thing to notice is that the datatype transmitted through the Serial port is "bytes" rather than integers. Thus, it's crucial to do a data conversion before making use of these data.

```
def data_conv(cur_data):
    cur_int=""
    for i in range(len(cur_data)):
        if cur_data[i].isnumeric():
            cur_int=cur_int+str(cur_data[i])
    return int(cur_int)
```

The above is a simple data conversion function which should be very straightforward and self-explanatory.

The second thing that's worth explaining is the outliers-rejecting algorithm. The reason why we make multiple data measurements earlier is that the HC-SR04 sensors are not of high precision and usually behave not properly. Also, it's undeniable that sensors of good quality also don't have 100% accuracy. Therefore, I use a simple outliers-removing algorithm to suppress these spikes.

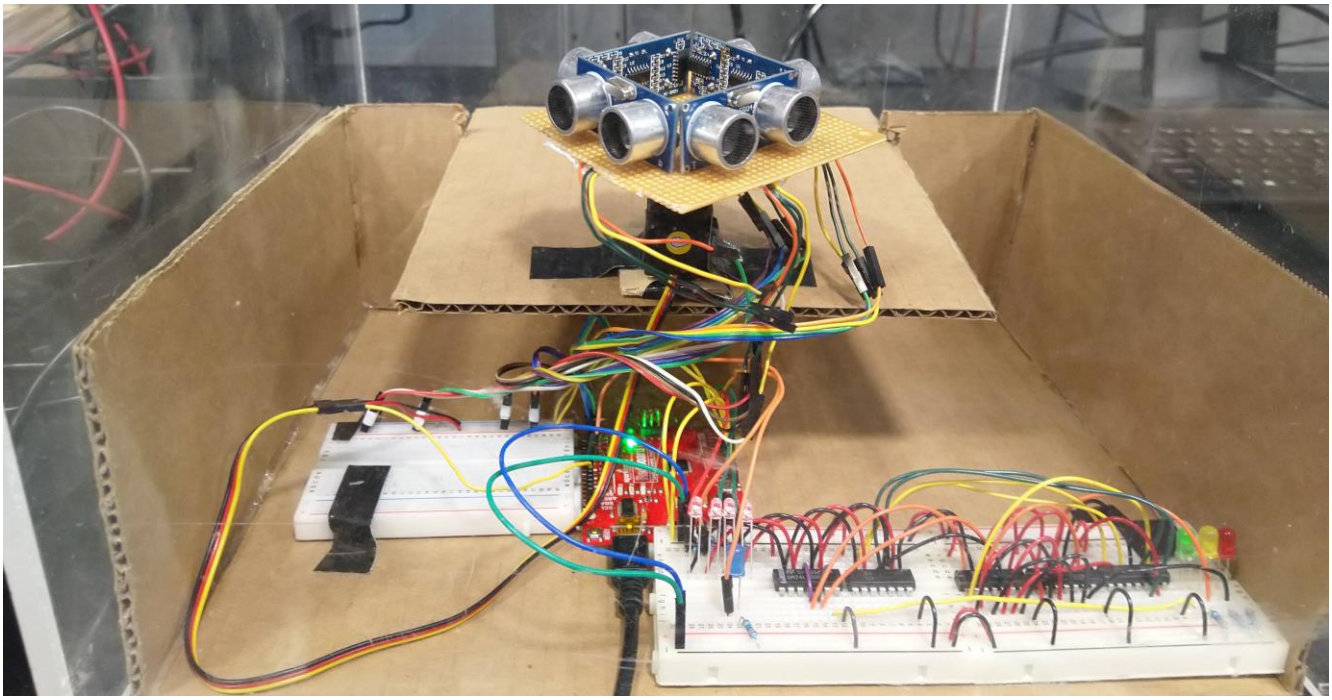
```
def reject_outliers(data, m = 2.):
    d = np.abs(data - np.median(data))
    mdev = np.median(d)
    s = d/mdev if mdev else 0.
    return data[s<m]

"""def reject_outliers(data, m=2):
    return data[abs(data - np.mean(data)) < m * np.std(data)]"""
```

These two outliers removing algorithms are both good to use. Since we're currently only using three to five data points at each degree position, it's not worthy of employing some complicated machine learning and statistical methods to remove outliers. However, for purposes that need high precision data, it's possible to measure distances many times and use regression or IQRs to detect outliers.

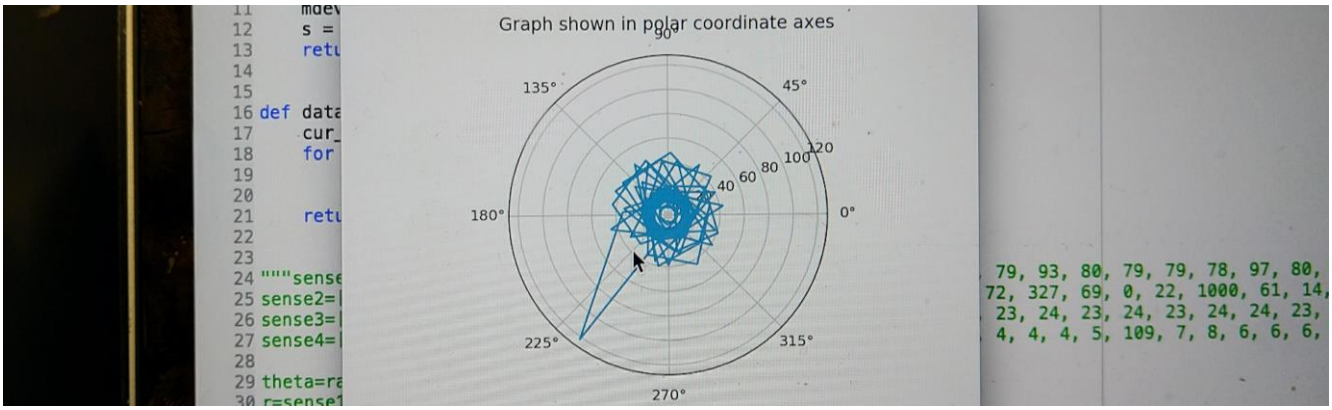
To wrap up, the data transmitted from python is stored in a two dimensional numpy array. We run the outliers_rejecting algorithm in the second dimensional sub arrays and take the average value of the remaining clean data in each subarrays. In this case, the two dimensional arrays will be transformed into a one dimensional array.

RESULTS & PROJECT DEMO

Working Demo of Our Project



Graph Generated from Sensor data – (Note the abnormal high value indicates that a sensor is not working properly. The container was square and can be depicted by the remaining 3 sensors



Scan the Barcode to view the project in motion

CONCLUSION AND SELF-ASSESSMENT

Future Improvements and Applications

There are numerous potential improvements of this project. For example, we can add another motor and elevate the platform of ultrasonic sensors to generate a 3D graph in python. Also, this detector could be placed inside some pipes of important use and operate continuously. Once a leak or crack appears on its side, a spike should be displayed in the graph and indicator light should be turned on. This would also allow us to reduce moving the whole apparatus manually. Automatic movement would make the readings more precise.

Lessons Learnt

One of the biggest challenges facing during the construction of this robot were to troubleshoot why the Arduino won't drive the servo motor and the ultrasonic sensors when both components are at a 5V external input. At first, we thought it might be some issue in the code. But after going through it several time, the code appeared to work just fine with the individual components. In the end, we found out that the sensors rely on Ground and VCC=+5V. By having an external voltage for the components, the ground value for them became arbitrary and way off the Arduino ground resulting in odd readings.

Self-Assessment

While beginning the project we wanted to build a smart-chair, one that would automatically align itself to the table when commanded to via a wireless module. Seemingly simple-task, right? Wrong. As we delved into it, we realized that to have an autonomous chair, it needs to know its location and map the surroundings. From there we focused on individual component of mapping and worked on perfecting that and ended up changing the theme entirely. But now, seeing everything work to our satisfaction, the three of us are eager to complete our initial project next semester.