



# THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

## **Autonomous Collision-Avoiding Car**

### **Group 3**

**Ryan Hay**

**Andy Hoang**

**Jackson Neese**

**Joey Presa**

**Ian Rainey**

Turned In: April 20, 2023

# Table of Contents

<b>1. Abstract.....</b>	<b>2</b>
<b>2. Device Production.....</b>	<b>4</b>
2.1. Chassis and Hardware Design.....	4
2.1.1. Equipment Used.....	4
2.1.2. Sensor Mount Design.....	11
Revision 1.....	12
Revision 2.....	13
Revision 3.....	14
Revision 4.....	15
2.1.3. Top Cover Design.....	15
Revision 1.....	16
Revision 2.....	17
Revision 3.....	18
Revision 4.....	19
2.1.4 Battery Case Design.....	20
Revision 1.....	20
Revision 2.....	21
2.2. Circuit Construction.....	21
2.3. Software.....	26
2.4. Data Processing.....	31
2.5. Room Scanning.....	34
<b>3. Theory of Operation.....</b>	<b>36</b>
<b>4. Tasks and Costs.....</b>	<b>38</b>
4.1. Project Roles.....	38
4.2. Hardware Costs.....	39
4.3. Schedule.....	42
<b>5. Calibration and Uncertainty.....</b>	<b>43</b>
5.1. Calibration.....	43
5.2. Uncertainty.....	45
5.3. Testing.....	45
<b>6. Lessons Learned.....</b>	<b>46</b>
<b>7. References.....</b>	<b>49</b>
<b>8. Appendix A: Full Code.....</b>	<b>50</b>
<b>9. Appendix B: Calibration Data.....</b>	<b>59</b>

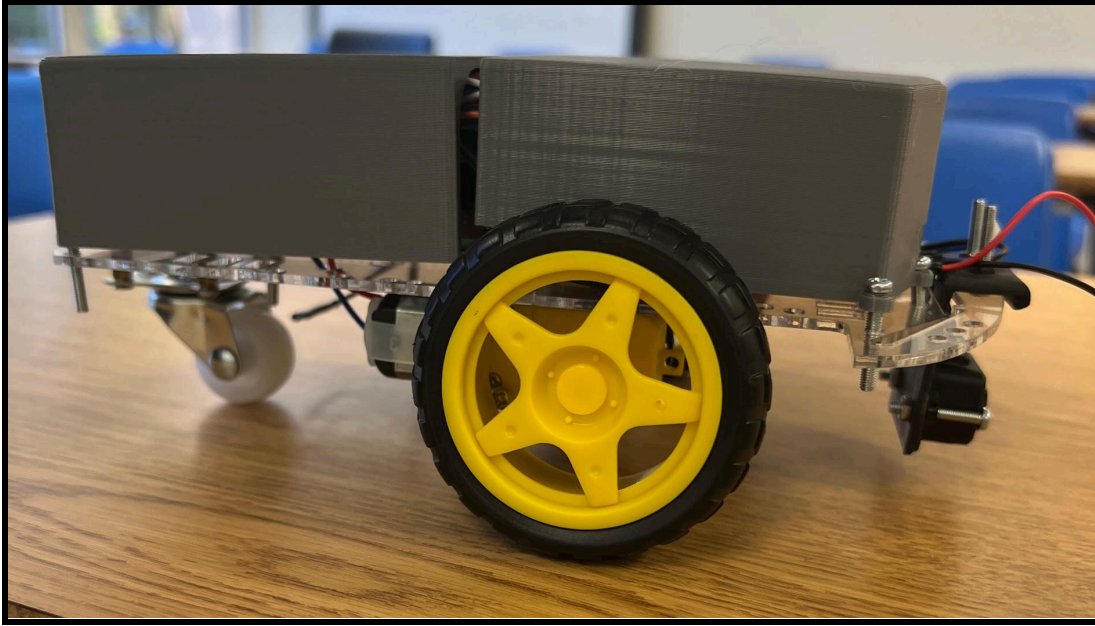
## 1. Abstract

The goal of this project is to measure the proximity distance of a collision-avoiding car with respect to its surroundings. The main electrical components used in this project include a TF-Luna Lidar Sensor, L298N Motor Driver Controller, MicroSD Card Module, and an Elegoo Arduino Uno. The TF-Luna Lidar Sensor is the main component for the measurement portion of our application. It is based on the time of flight principle in which the sensor emits modulation waves of near-infrared waves to nearby objects to which it is reflected and from this reflection the distance between the sensor and the object can be calculated/recorded. The TF-Luna Lidar sensor is a single-point range-finding sensor used for stable, accurate range detection at high speeds. As for the measurement itself, the Lidar produces very accurate results compared to other distance sensors like an ultrasonic sensor. These sensors work well with perpendicular surfaces but are not applicable to curved objects. Since most objects the collision-avoiding car will encounter will not always be perpendicular surfaces, the Lidar sensor was better suited for this application since it can accurately detect distance from the single-point laser.

Pairing this sensor with 2 DC motors and a motor driver controller, a regular RC car can become an autonomous collision-avoiding vehicle that collects distance measurements and avoids obstacles in its vicinity. Utilizing the MicroSD Card Module, the distance data recorded from the sensor can then be extracted into Excel in a .csv file for data analysis.

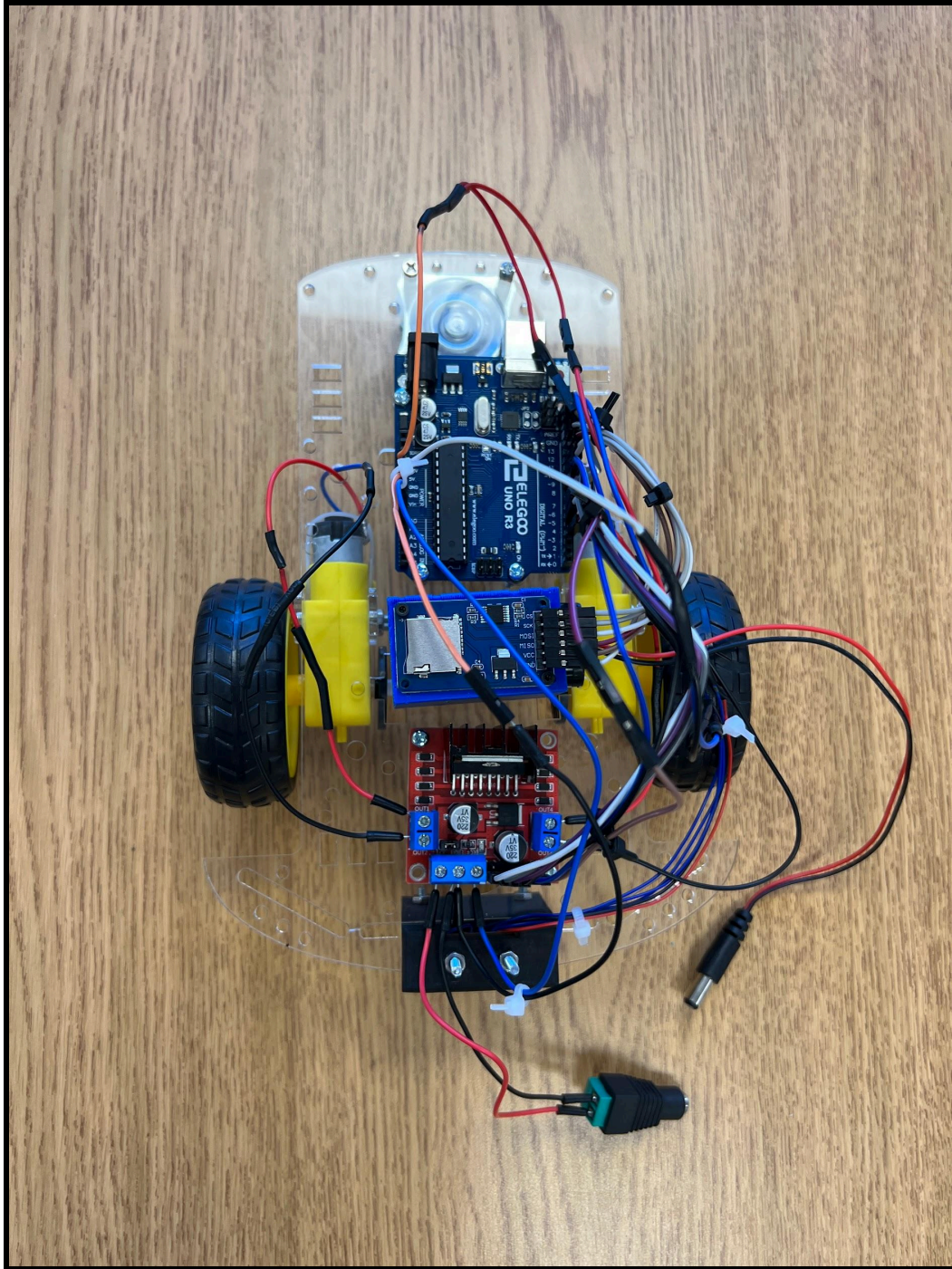
It was determined that with an application involving multiple electrical and mechanical components, skills such as wiring, soldering, crimping, heat shrinking, CAD modeling, and 3D printing play an important role in compiling everything in the project together. Calibration on the sensor was performed using a NIST-certified tape measurer in the lab, incorporating a test stand while collecting calibration data to allow for more accurate measurements. Ultimately, the collision-avoiding car performed as it was intended and was able to collect accurate distance data while also avoiding obstacles when the sensor detects the threshold distance. The total uncertainty of the sensor due to random and calibration errors was around 0.756% of the measurement which is roughly  $\pm 0.2722$  inches at a range of 3 ft and up to  $\pm 2.178$  inches at 24 ft.

Figures 1-3 show images of the final product with and without the final case design.

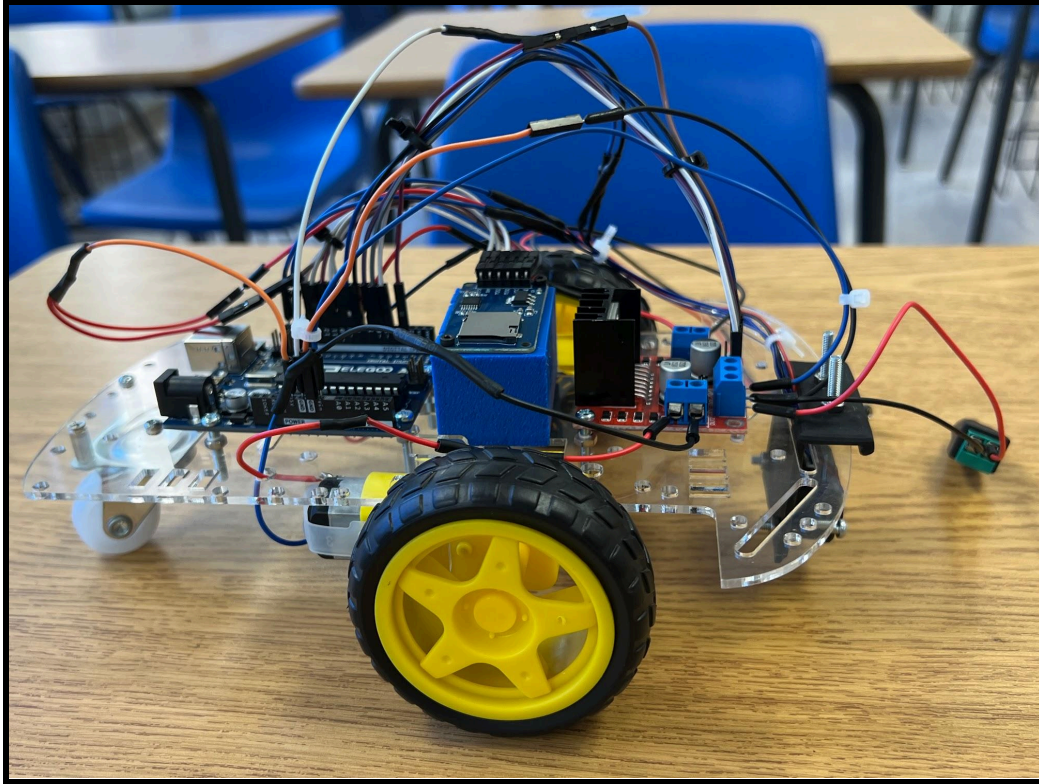


**Figure 1: Final Product With Case Design**





**Figure 2: Final Product Without Casing (Top-View)**



**Figure 3: Final Product Without Casing (Side-View)**

## **2. Device Production**

### **2.1. Chassis and Hardware Design**

#### **2.1.1. Equipment Used**

1. Elegoo Arduino Uno R3
2. TF-Luna Lidar Sensor
3. L298N Motor Drive Controller
4. Micro SD Card Module
5. Transcend 2 GB Micro SD card
6. 9V Battery
7. DIY Robot Smart Car Chassis Kit
8. (2) 12V DC Power Connector Power Jack Plugs
9. (9) M3 Steel Pan Head Machine Screws
10. (9) M3 Stainless Steel Hex Screw Nuts
11. (4) M2 Steel Hex Screws
12. (2) M2 Steel Hex Screw Nuts
13. (9) 4-inch Cable Ties
14. (18) Heat Shrink Tubing (3 mm diameter, 30 mm length)

- 15. Anycubic Photon Mono 4K Resin 3D Printer
  - a. Sensor Mount V1 -> Sensor Mount V4
- 16. Ender 3 FDM 3D Printer
  - a. Top Cover v2, Front Half, and Back Half

Figure 4 below shows the Arduino board that will be used to power and control all of the electronic components of the final product. The TF-Luna Lidar Sensor was chosen to be the primary distance recording sensor in this application instead of the HC-SR04 Ultrasonic Sensor. The main reason is that Lidar can measure more accurately on any surface since it uses a single-point laser to accurately record the distance in its line of sight, more detail on how Lidar works is in the theory of operation portion of the report. On the other hand, an ultrasonic sensor transmits ultrasonic waves to an object that reflects back to a transducer which records distance. This sensor is good for perpendicular surfaces, but it does not work well with curved objects or objects that can dampen the ultrasonic waves. Therefore, this sensor would not work well in obstacle detection for a robot car since it is limited to certain objects where it can record accurate data. Figure 5 below shows the TF-Luna Lidar Sensor used in this project as the primary distance-measuring device.



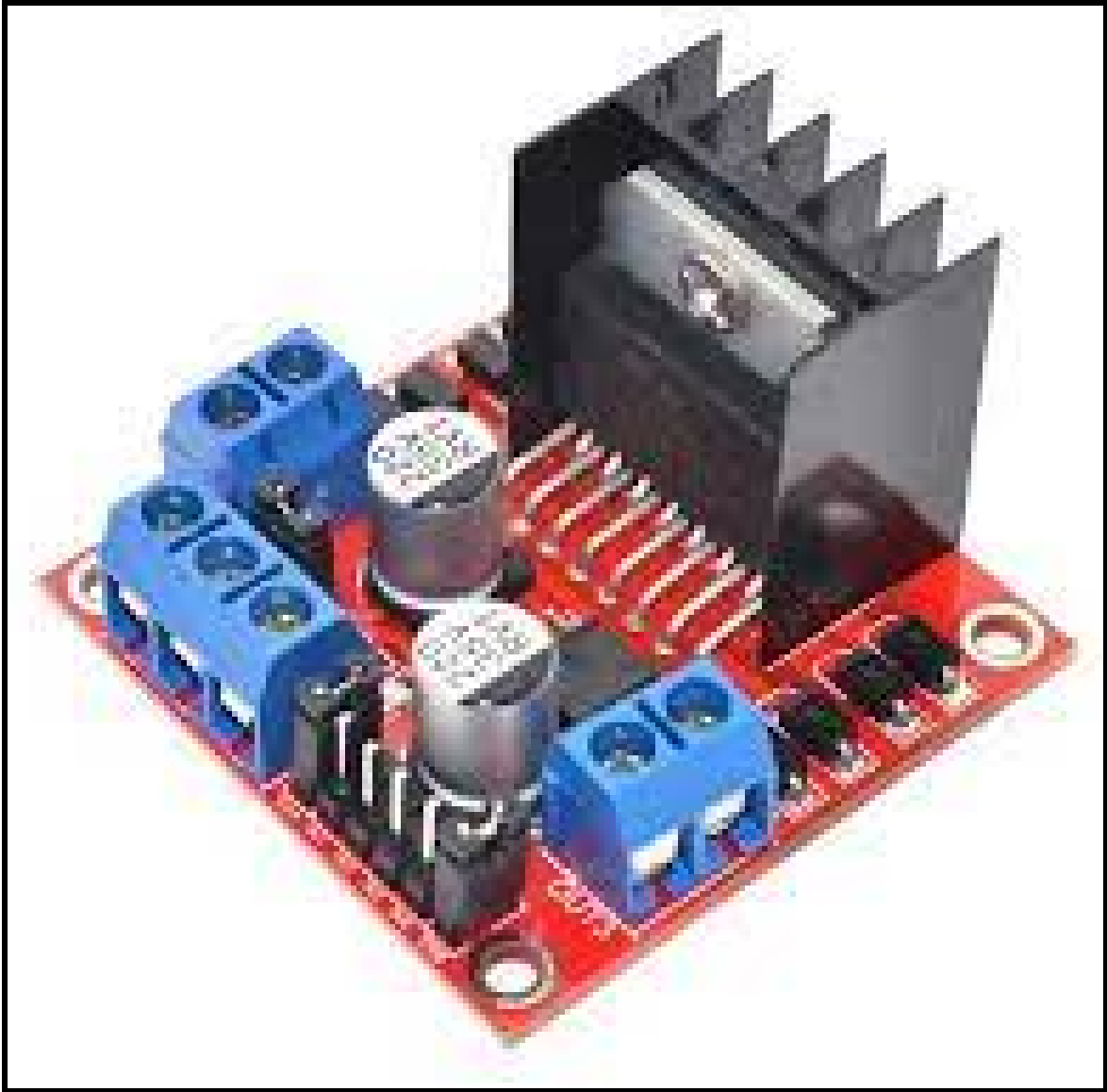
**Figure 4: Arduino Uno R3 Board**





**Figure 5: TF-Luna Lidar Sensor**

To power and move the obstacle-avoiding car, a motor driver module is necessary to control the speed of the motors as well as the direction. The L298N Motor Driver Controller from Figure 6 was used in this application since it can control and change the direction of 2 DC motors and it is fairly cheap costing around \$5 on Amazon. Next, the selected chassis for the car was the DIY Robot Smart Car Chassis Kit as seen in Figure 7. This kit had everything needed for our application which was 2 DC motors, 2 wheels, and a flat chassis with screw holes to house all of the other electrical components.

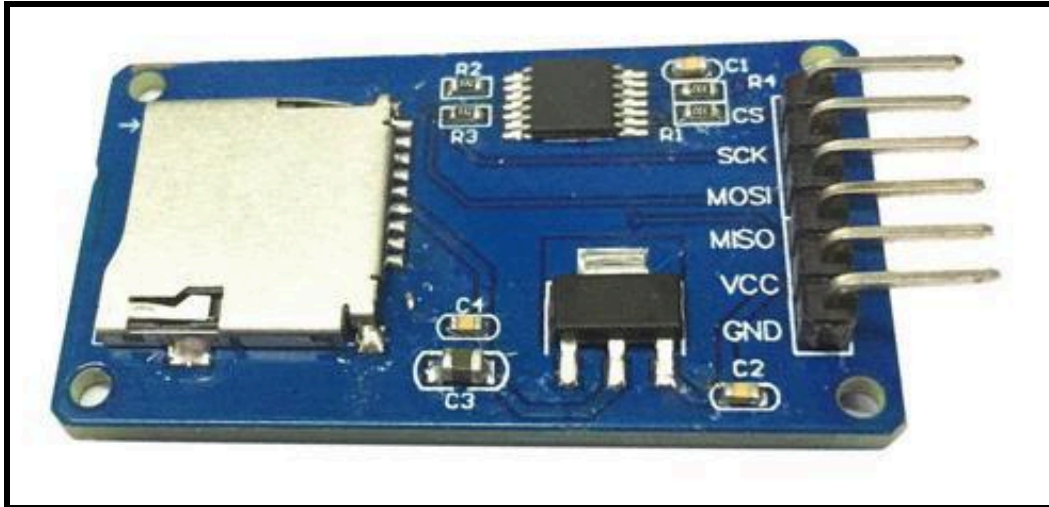


**Figure 6: L298N Motor Driver Controller**



**Figure 7: DIY Robot Smart Car Chassis Kit**

Finally, the last main component of the project was the Micro SD Card Module used to store the data from the Lidar sensor and open the data in a .csv Excel file which can then be used for data analysis, calibration, and uncertainty calculations. Figure 8 below shows the Micro SD Card Module used in the final product.



**Figure 8: Micro SD Card Module**

To fabricate components designed in SolidEdge, the two following 3D printers were used. The sensor mount was created with the AnyCubic Photon Mono 4k resin printer, and the Top Cover and Battery Case were created using the Ender 3 FDM printer. Both are shown in Figures 9 and 10.



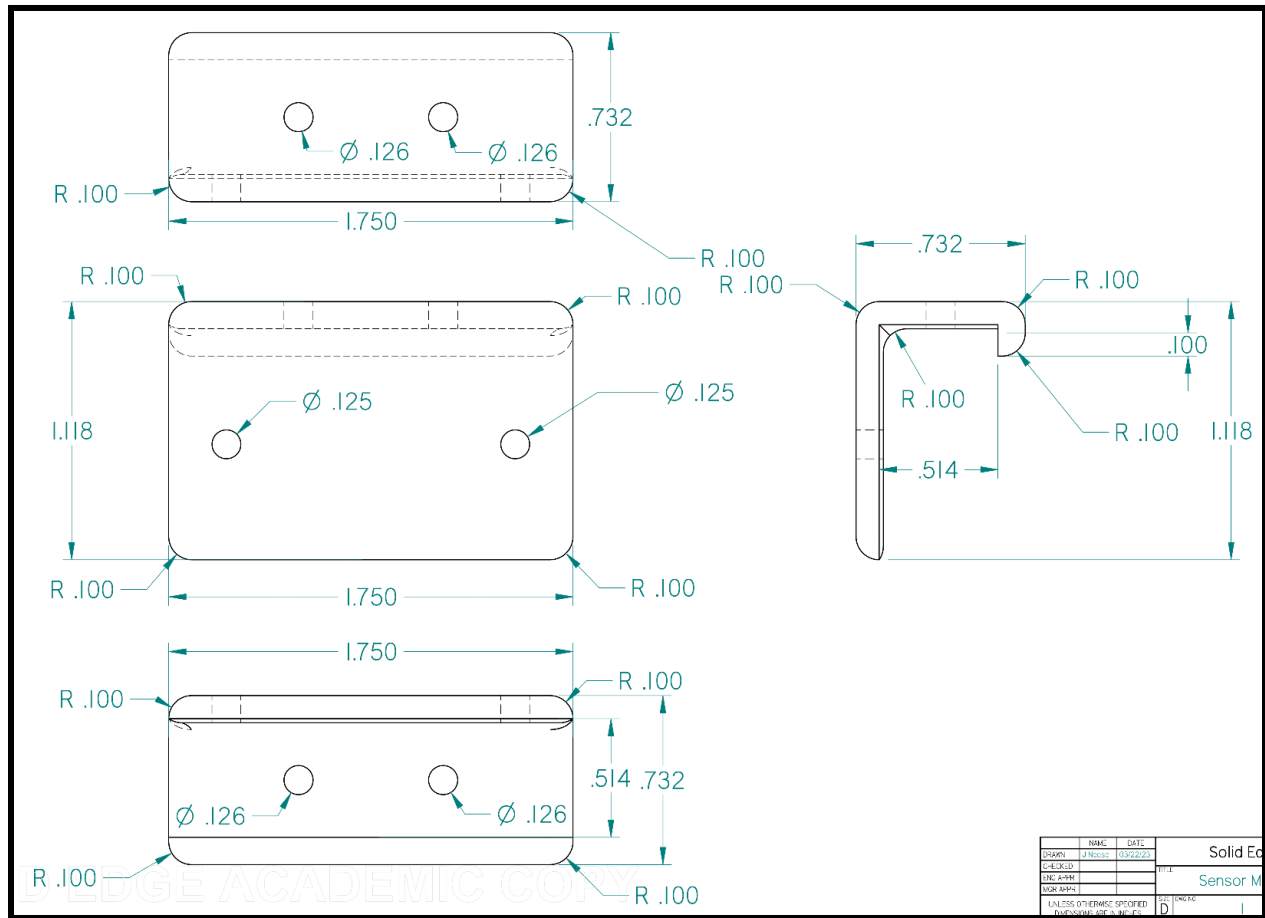
**Figure 9: AnyCubic Proton Mono 4k Resin 3D Printer and Wash Station**





**Figure 10: Ender 3 FDM 3D Printer**

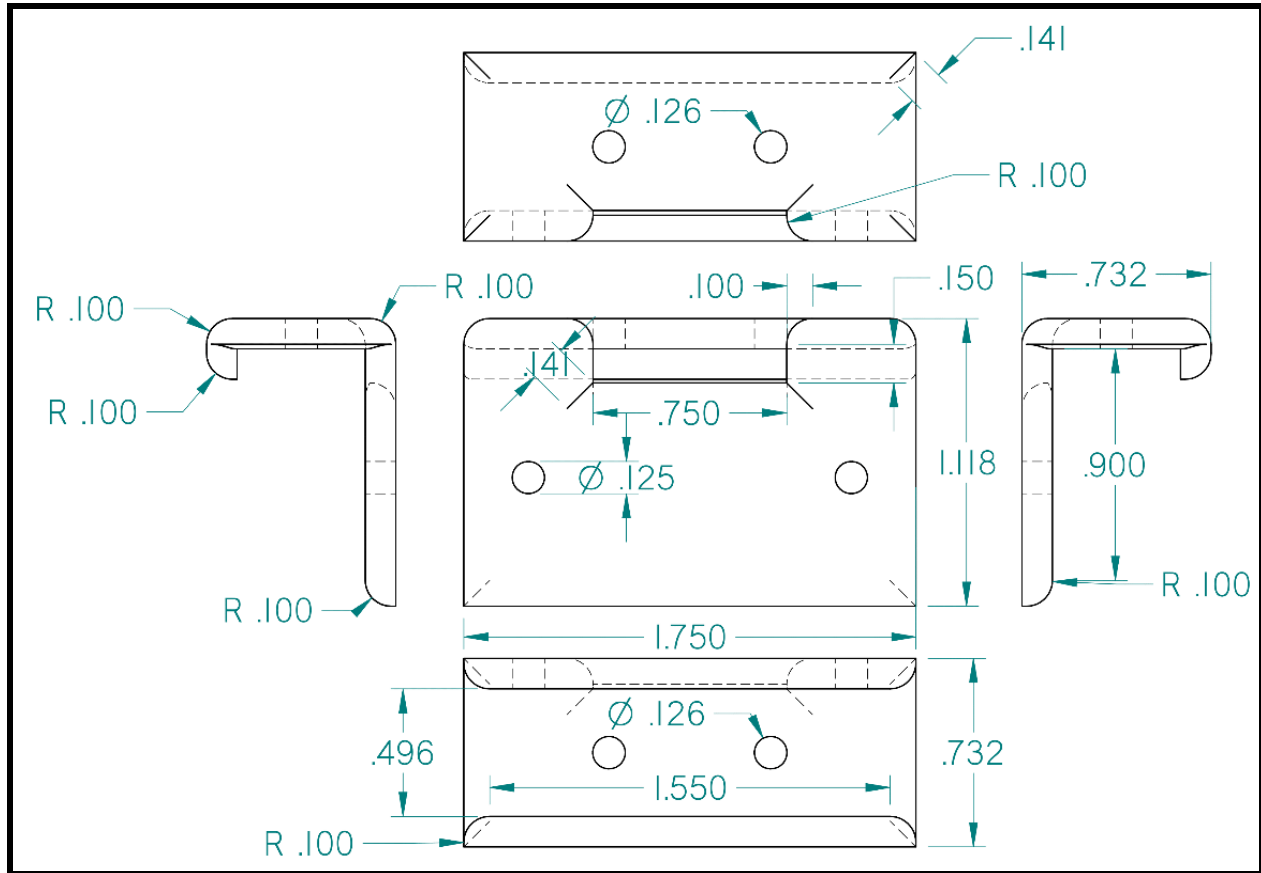
### 2.1.2. Sensor Mount Design



**Figure 11: Sensor Mount Revision 1**

#### Revision 1

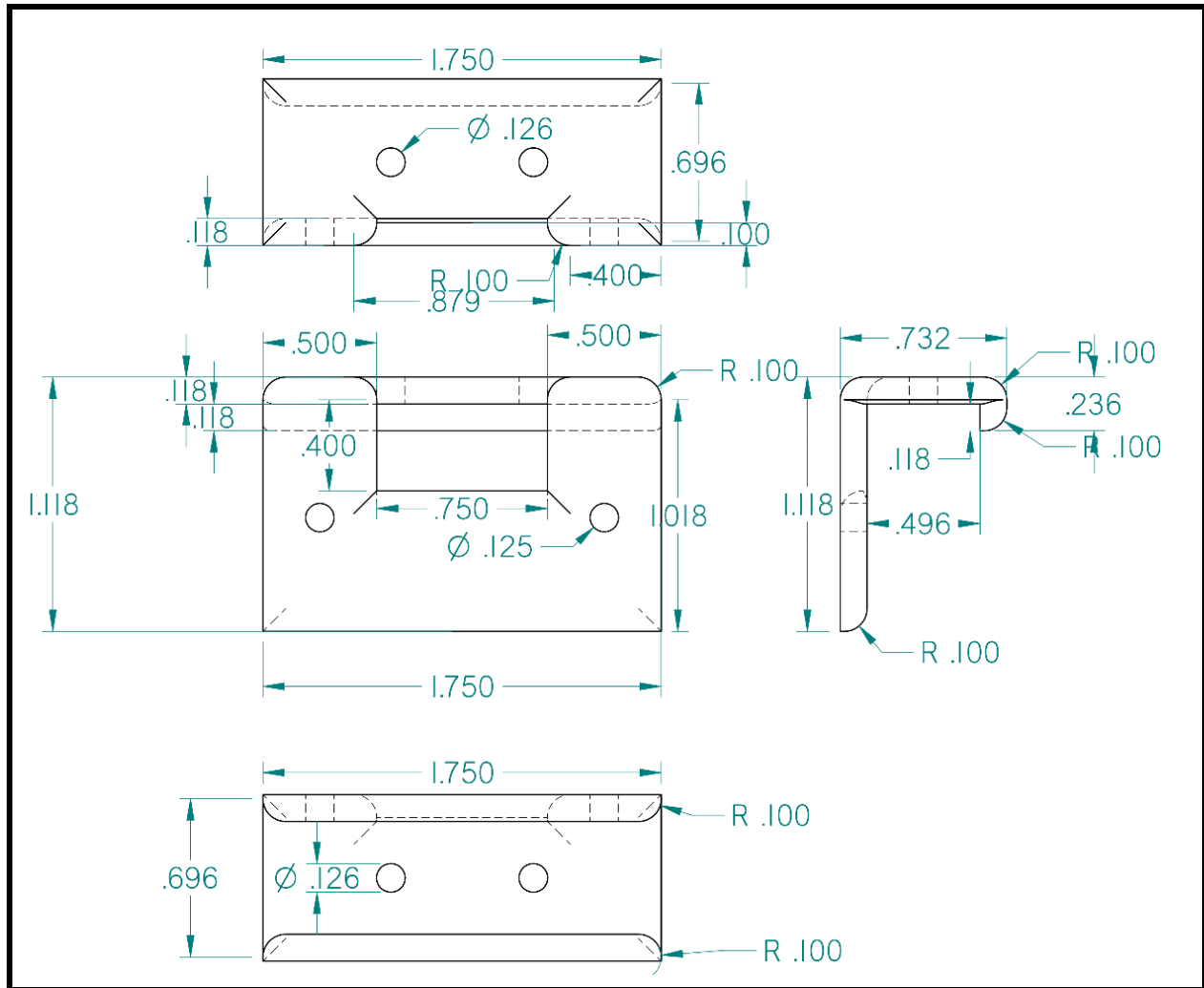
Using a mockup of the base plate, a simple design was made for a piece to hang off the front with screw holes to mount the LIDAR sensor. It utilizes a slot and holes pre-drilled into the base plate to hold it into place and keep it from jostling while in motion, with a simple panel hanging down to act as the mounting plane. With this design, the sensor should have a clear view of the car's direction of travel and be situated at a necessary height to provide relevant data to be used in the code.



**Figure 12: Sensor Mount Revision 2**

### Revision 2

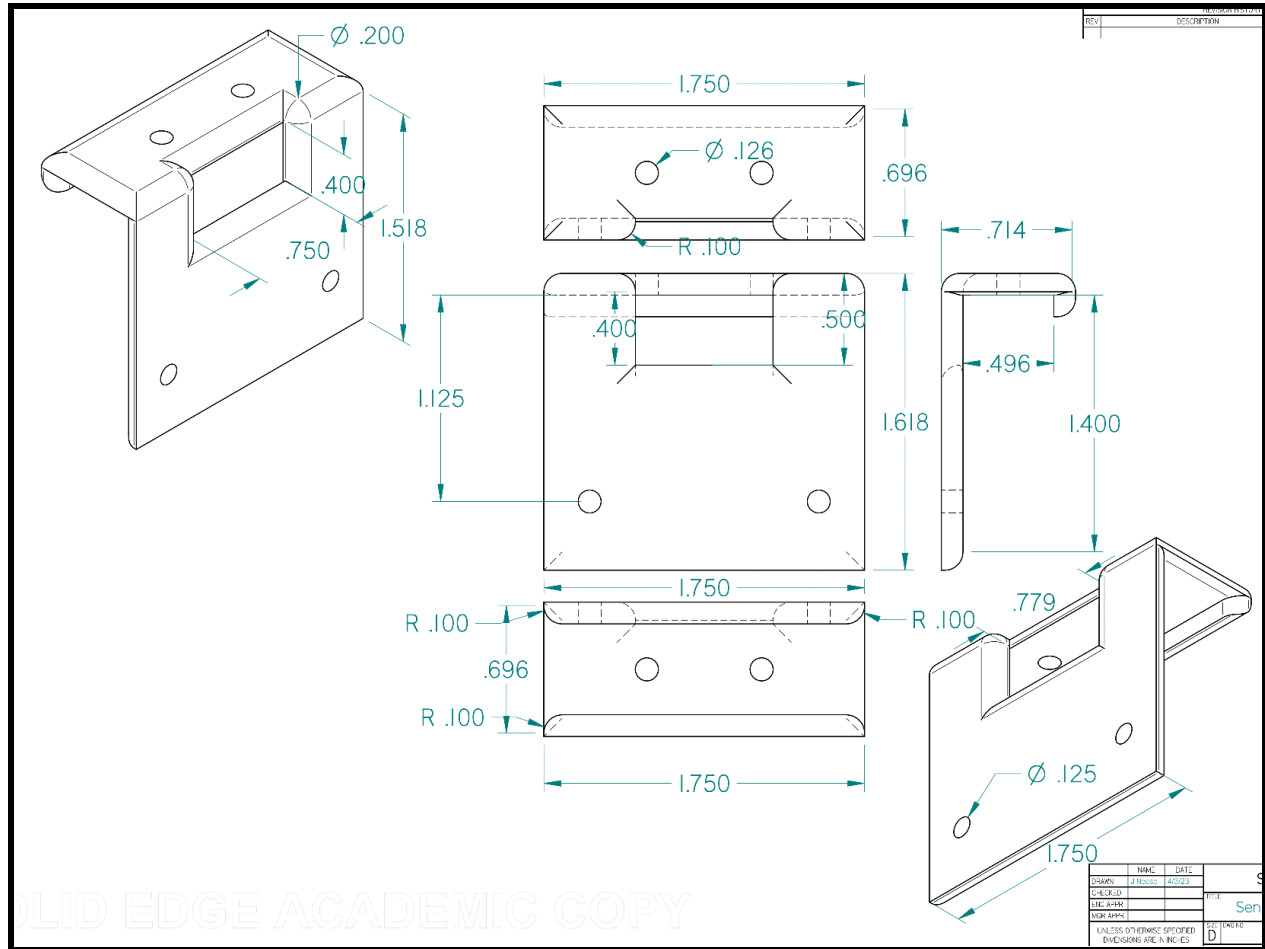
With a printed version of Revision 1, it was determined turning it backwards, where the mounting bracket section goes through the slot on the baseplate would be a better way to put it on. Essentially, it being backward makes it safer for the sensor since it wouldn't be the first thing to impact anything in case of navigation failure. Additionally, when testing the sensor with the wires, it needs a place to pass the wires through since it otherwise would bend around the mount or put unnecessary strain on the wires. I added a small hole in the top section to allow us to thread the wires through to the top section. Also, since the mount would be used in a backward manner, I changed the curved edges to the other side, complimenting the sensor's new position. Also, a rounded edge in the inside corner of the first version was eliminated due to it shortening the contact edge between the mount and the base plate.



**Figure 13: Sensor Mount Revision 3**

### Revision 3

With the new hole on Revision 2, the angle in which it would need to pass the wires through would still add undue stress on the bend point, possibly causing wear and breakage over time. Also, with the small hole near the top, it only had a small clearance point due to its close proximity to the wall of the base plate. I expanded the hole down somewhat to allow more space for the wires to flow through. However, the problem continued. The sensor was too high up in the bracket's corner, making the wires hit the edges regardless of the hole's size.

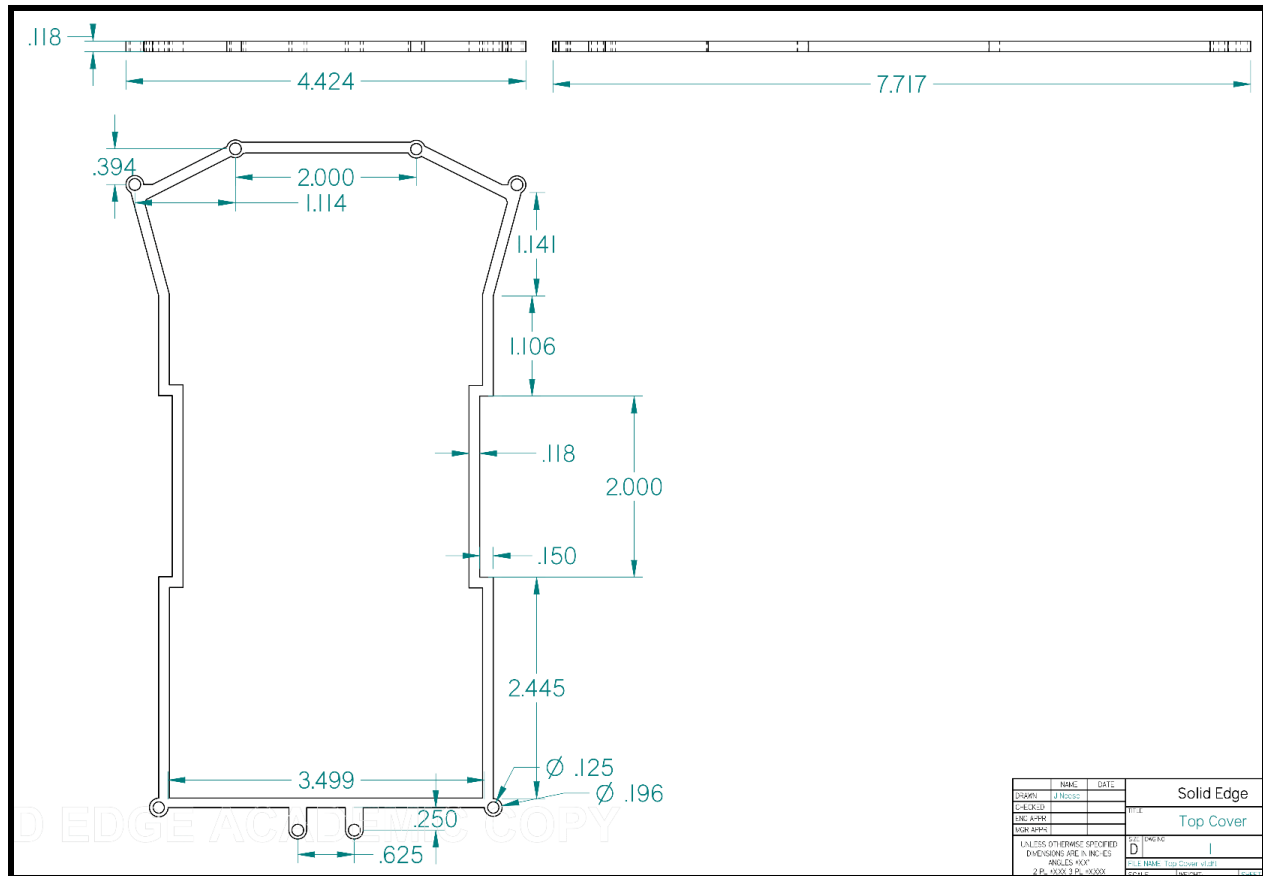


**Figure 14: Sensor Mount Revision 4**

### Revision 4

Confirming the source of the issue, the large panel holding the sensor was extended downward and the mounting holes were lowered to allow greater clearance of the wires. In mounting this system, the wires flowed through sufficiently and the motor fit snugly in the mounting holes. However, the holes at the top for mounting the piece to the base plate were slightly out of line. To mitigate this small issue, they were manually ground out with hand tools to align correctly. This led to a fully successful use of this part.

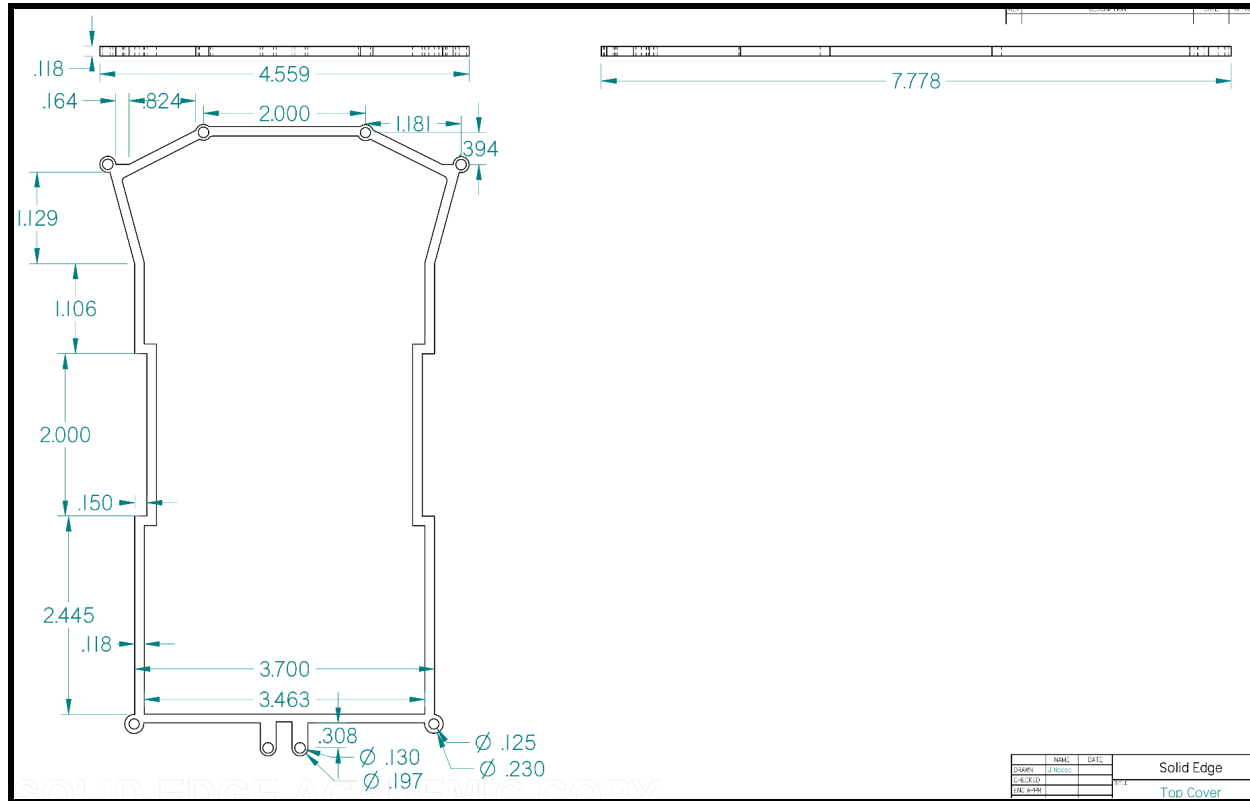
### 2.1.3. Top Cover Design



**Figure 15: Top Cover Revision 1**

## Revision 1

To cover the sensitive electronic parts and protect them, a piece was designed to go over the whole assembly and screw into some pre-existing holes. To do this, measurements were made over selected screw holes in the base plate. It only needs to be 3mm thick as to not waste 3D filament since this part shouldn't see much wear in regular use. A section was made to move around the motor mount and to avoid pre-existing hardware as well. To test the placement of the holes, only the outer rim has been constructed. The part drawing is measured in inches, but some components are converted from centimeters as the base plate seemed to use both in its design considerations.



**Figure 16: Top Cover Revision 2**

## Revision 2

After a basic print to test for the hole placement, it was determined that there were some inaccuracies in the base plate model. In the next version, the plate was directly measured to ensure accuracy. Some aesthetic changes were also added to mitigate failure points near the holes. The part drawing is measured in inches, but some components are converted from centimeters as the base plate seemed to use both in its design considerations.

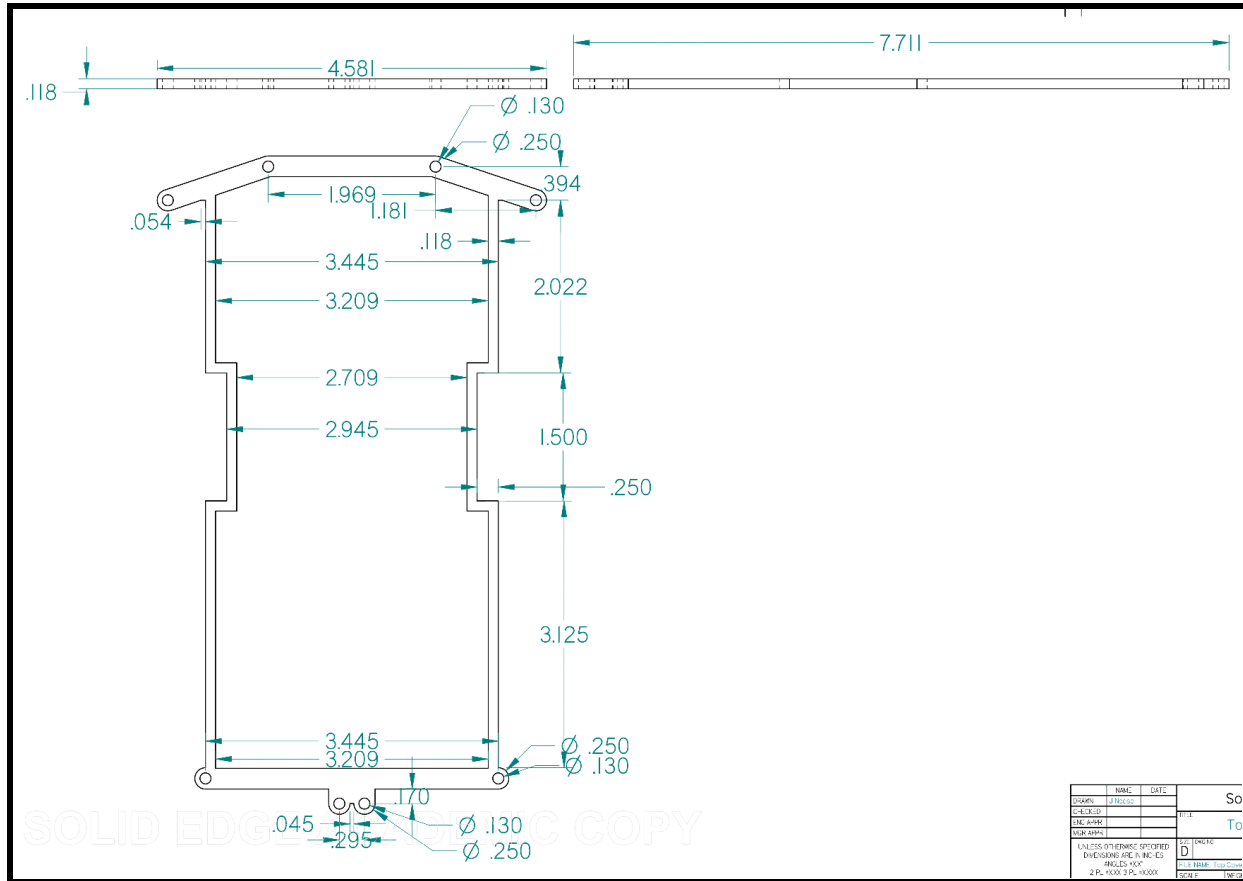


Figure 17: Top Cover Revision 3

### Revision 3

There was once again an issue with the hole placement, and another test piece was constructed. The front and back sides were thickened to reinforce them as there are many screw holes there. The part drawing is measured in inches, but some components are converted from centimeters as the base plate seemed to use both in its design considerations.



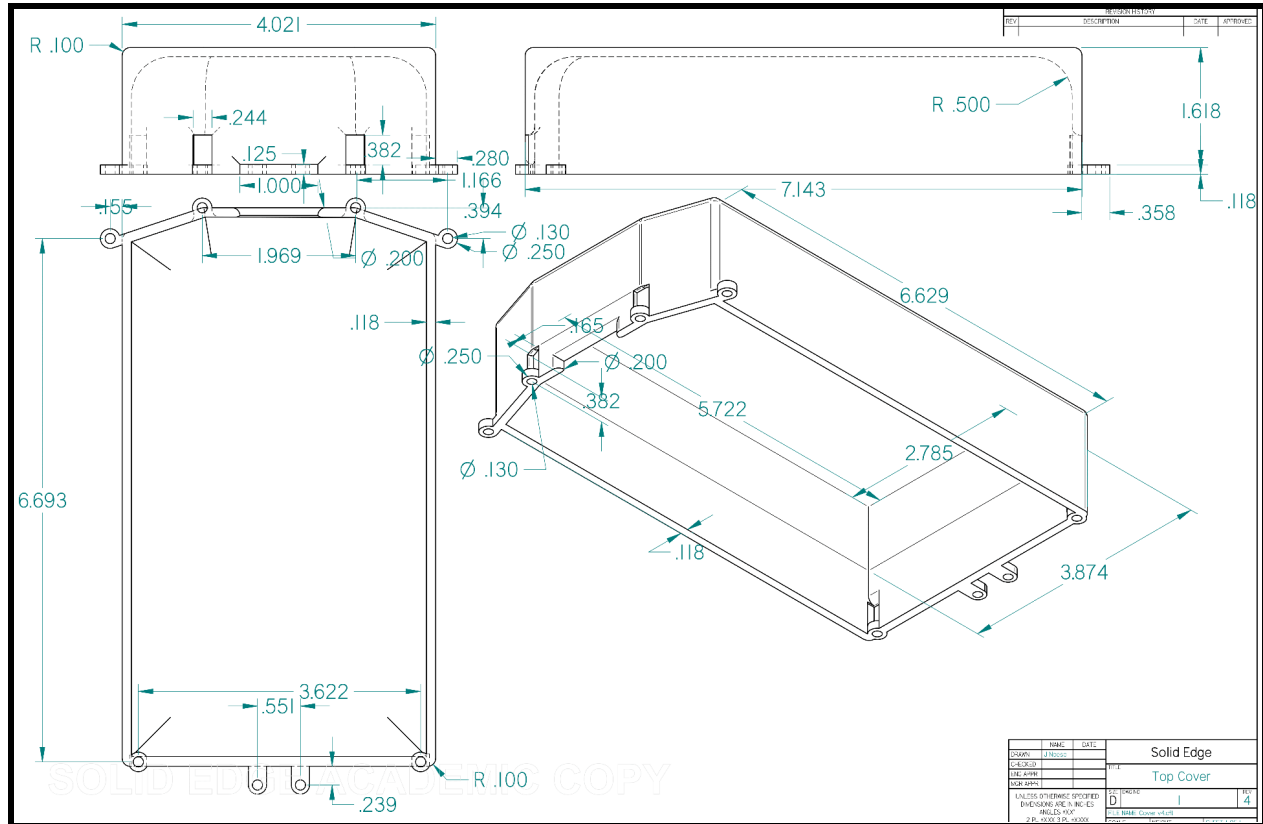
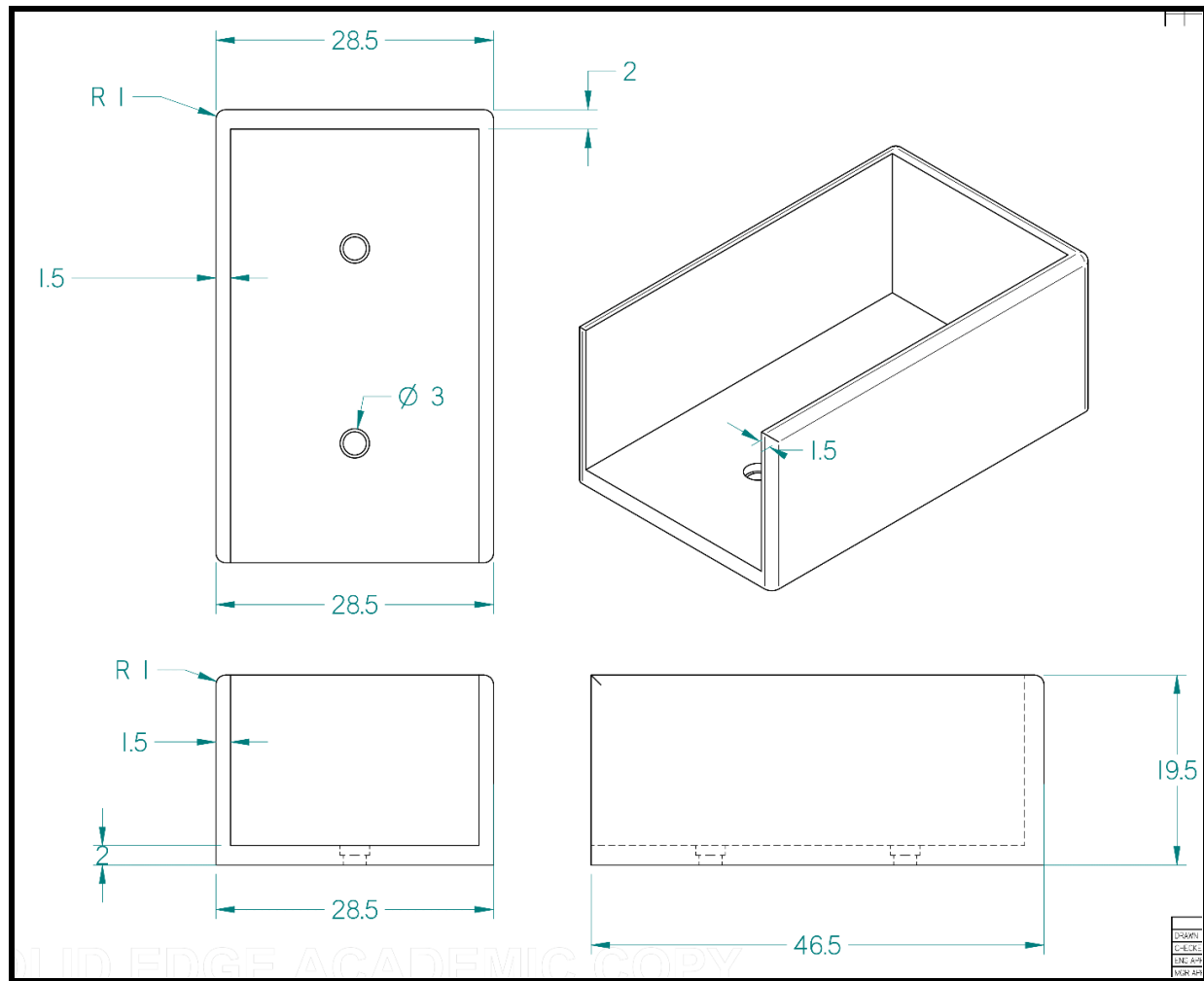


Figure 18: Top Cover Revision 4

### Revision 4

Following more issues with hole placement and the CAD software changing certain measurements as a result of others being modified, the measurements were refined, double-checked, and compared to a photo of the base plate. Then, a small section was extended up and clearances were cut for wires and screws. Following these cuts, the walls were further extended, a ceiling formed, and edges rounded for aesthetic consideration and ease of handling. The part drawing is measured in inches, but some components are converted from centimeters as the base plate seemed to use both in its design considerations.

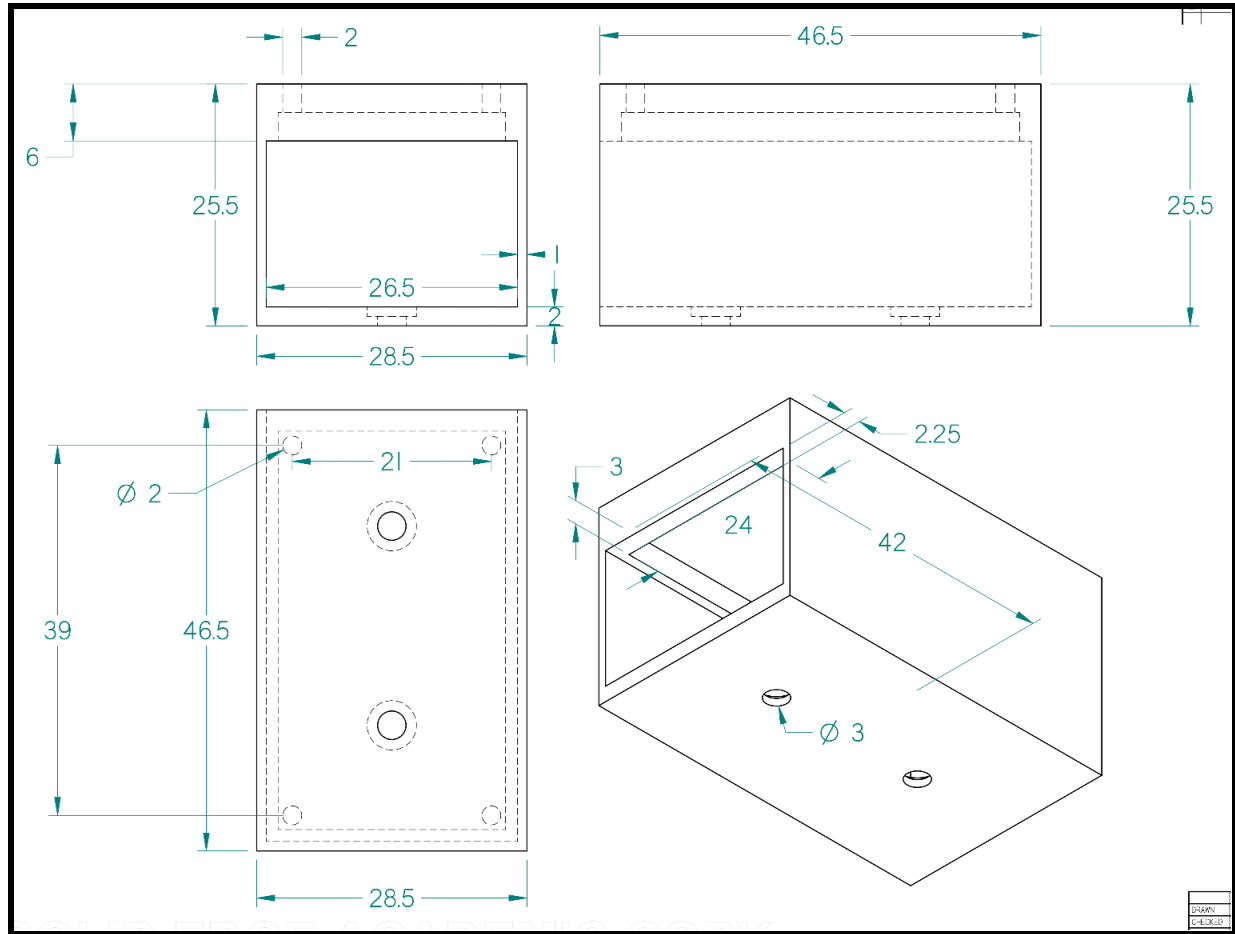
### 2.1.4 Battery Case Design



**Figure 19: Battery Case Revision 1**

#### Revision 1

To keep the battery from moving when the car is in motion, a casing was designed to contain the 9-Volt battery and bolt into the base plate. To do this, a 9-Volt battery was measured to build a housing and a set of holes was selected on the base plate to size and arrange the holes on the base to mesh well with where it would be going. The part drawing is measured in millimeters.



**Figure 20: Battery Case Revision 2**

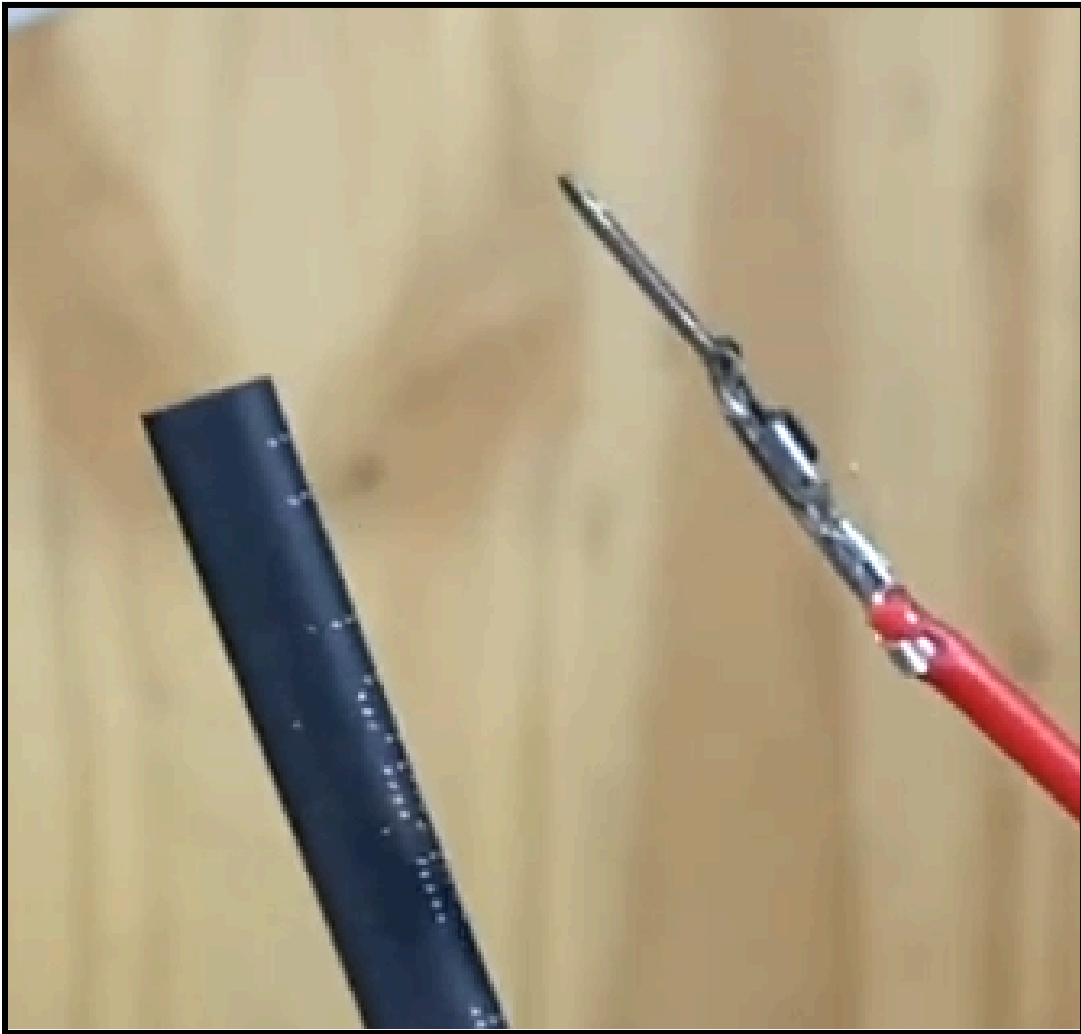
### Revision 2

While Revision 1 was successful, it was revised to include a place to mount the SD card reader chip in an effort to both secure it to the car and save space. The part drawing is measured in millimeters.

## 2.2. Circuit Construction

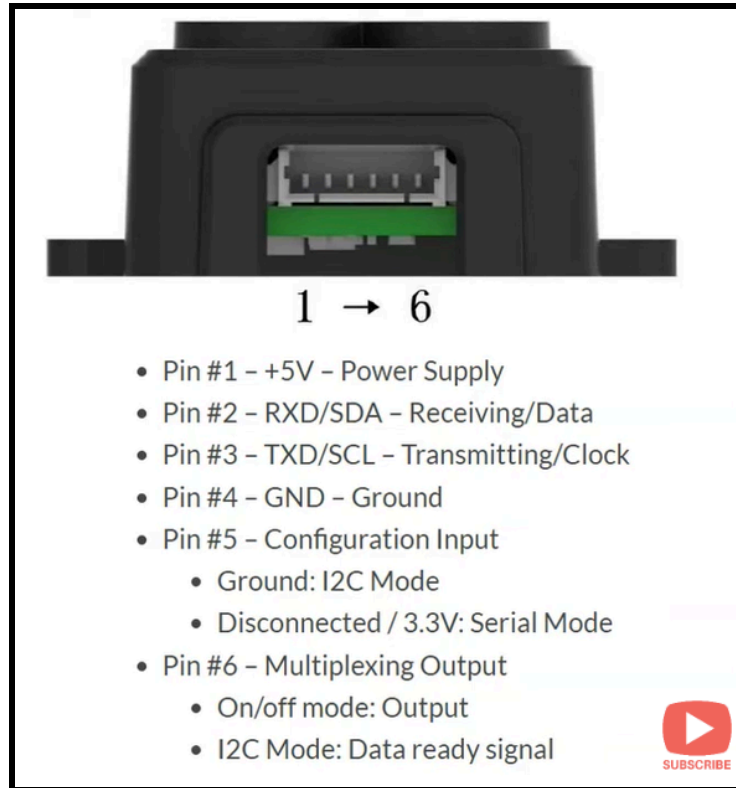
This project utilizes many different pin connections to the Arduino, and some of them could not fit because the wires were too thin or there was not enough space on the Arduino to connect them. Because of this, we resorted to wire crimping methods and soldering to ensure that there was enough pin space on the Arduino as well as consistent wire connections. The wires on the TF-Luna Lidar Sensor were too thin to make any strong connection to the Arduino. Due to this issue, we used male pin connectors paired with heat-shrinking techniques to make the wiring stronger and neater. Figure 21 below shows the overview look at how the thin wires were crimped to a male pin connector after which the exposed part of the connector was wrapped with

heat shrink and applied heat to. This process was repeated for all of the wires on the TF-Luna Lidar Sensor.



**Figure 21: Crimping and Heat-Shrinking Wire [2]**

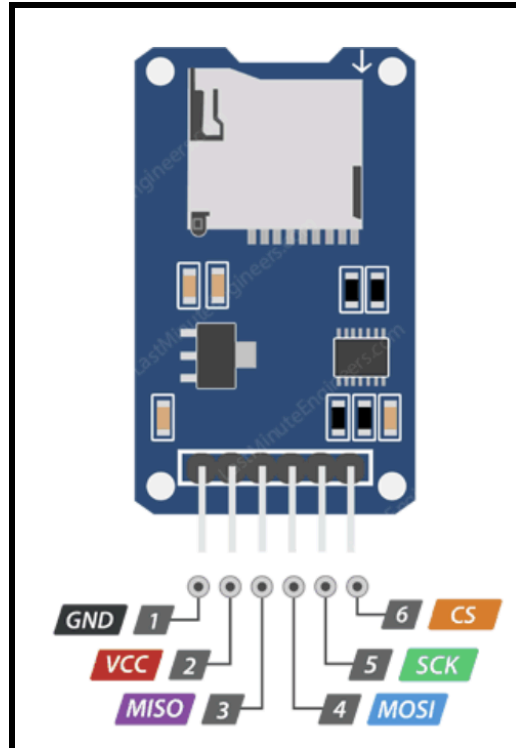
The TF-Luna Lidar Sensor has 6 pins: pin #1 is the 5V power supply, pin #2 is for receiving data, pin #3 is the transmitting clock, and pins #4 and #5 are ground pins where pin #5 is grounded for the I2C mode, which will be the main interface used to record and read the data from the serial monitor [1]. Pin #6 is not used in the circuit design and is therefore cut to make room for cable management. Since the I2C interface will be used, pins #2 and #3 are wired to the I2C pins on the Arduino (SCL and SDA pins). Figure 22 shows the pin diagram for the TF-Luna Lidar sensor.



**Figure 22: TF-Luna Lidar Sensor Pin Diagram [1]**

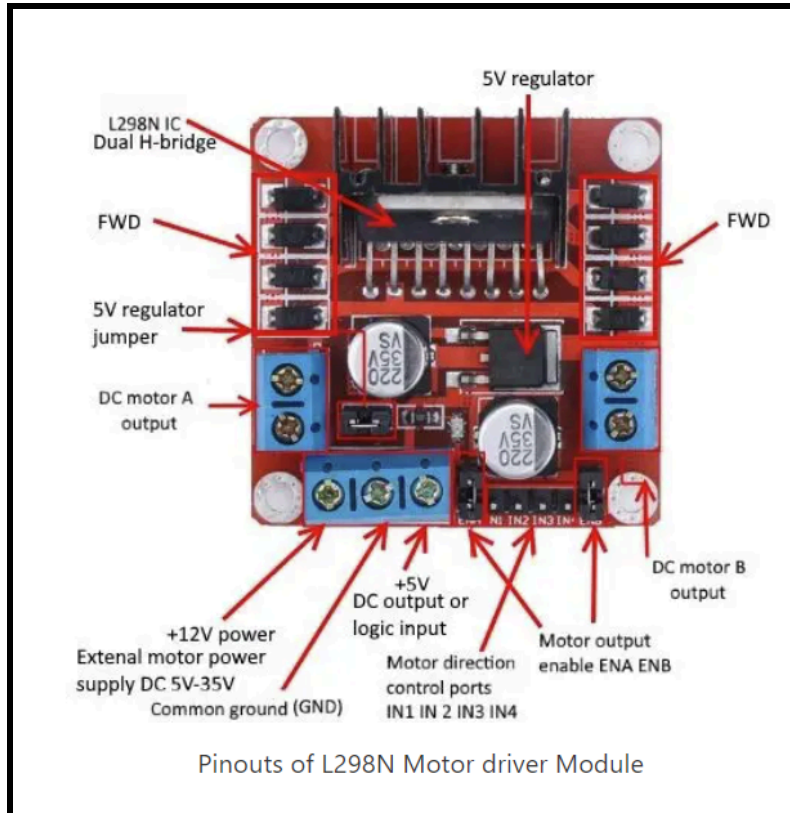
Because the TF-Luna Lidar Sensor has 2 ground wires, soldering had to be done to ensure that there were enough ground pins for the other components of the product. We used the soldering kit provided in the lab to perform all of the solderings. We soldered the two ground wires of the sensor, two female connectors to one male connector for the 5V output, and 2 wires to each DC motor. Once all of the soldering and crimping were finished, all of the wires can be connected to the Arduino.

The MicroSD card module also has 6 different pins that are all used in the circuit of the project. The VCC pin provides power to the MicroSD card module and is connected to the Arduino's 5V pin. The GND pin is the ground pin and goes to the ground. The MISO (Master In Slave Out) is the SPI output from the MicroSD card module and goes to pin 12. The MOSI (Master Out Slave In) is the SPI input to the MicroSD card module and goes to pin 11. The SCK (Serial Clock) pin synchronizes the data transmission and is connected to pin 13. Finally, the CS (chip select) pin is used to send and receive data to the MicroSD card module and is connected to pin 10. Figure 23 below shows the pin diagram for the MicroSD card module used in the project.



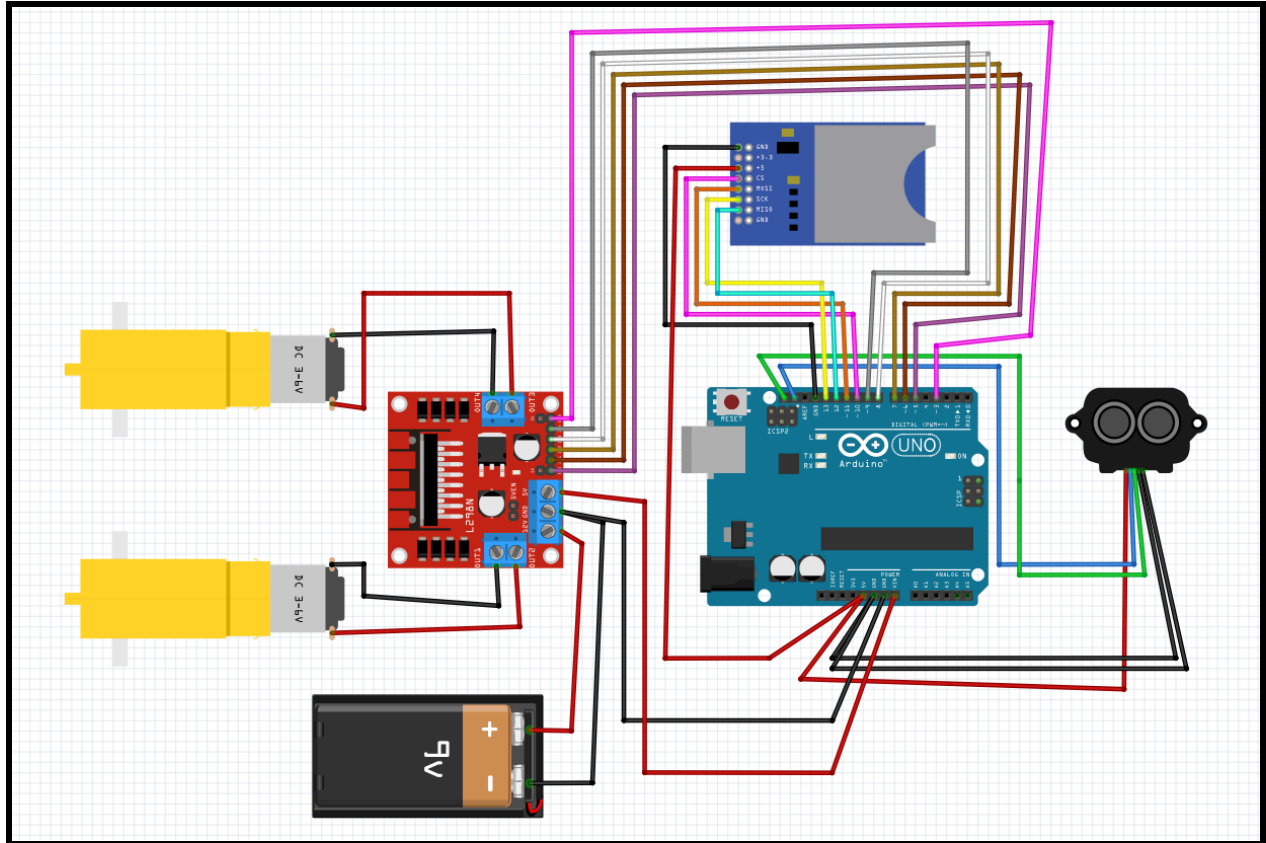
**Figure 23: MicroSD Card Module Pin Diagram [6]**

The L298N Motor Driver Controller has 2 DC motor output pins, a 12V external motor power supply, motor direction pins (IN1, IN2, IN3, and IN4), motor enabling pins (ENA, ENB), and a heat sink. The VCC pin on the Arduino supplies the power to the motor. The GND pin is used to ground the motor controller and the 5V supplies power to the switching logic circuitry inside the IC. The ENA pin controls the speed of motor A, setting this pin to high will cause the motor to rotate while setting it to low will cause the motor to stop. The IN1 and IN2 pins are used to control the direction of motor A. IN1 and IN2 must be opposite in voltage output, meaning an IN1 at HIGH and IN2 at low will cause the motor to rotate one way and vice versa to make the motor rotate the opposite direction. Pins ENB, IN3, and IN4 are the same but for motor B. Finally, pins OUT1 and OUT2 are connected to the wires of motor A and OUT3 and OUT4 are connected to the wires of motor B. Figure 24 shows the pin diagram for the L298N Motor Driver Controller.



**Figure 24: L298N Motor Driver Controller Pin Diagram [7]**

Figure 25 below shows the full wiring schematic created on fritzing software, including all of the components in the final product.



**Figure 25: Final Product Wiring Schematic**

### 2.3. Software

Since this project involves the use of controls and measurements simultaneously, many pins, variables, and libraries such as the “SD.h”, “SPI.h”, “Arduino.h”, “Wire.h”, “TFLI2C.h” libraries were initiated. The “Arduino.h”, “Wire.h”, and “TFLI2C.h” libraries were used for the TF-Luna Lidar sensor. The wire library is instantiated and the TF-Luna Library is pre-downloaded and called to record the distance from the lidar sensor and send the data to the serial monitor. The “SD.h” and “SPI.h” libraries are used to capture the recorded distance data from the sensor/serial monitor and store it on the microSD card as a comma-delimited value file. This file can then be used in post-processing data analysis.

After that, the variables and pins are initialized. A variable named `tfDist` is set up and will be used to call the TF-Luna sensor library and to represent the distance the sensor records in centimeters. Another variable is initialized to use the I2C interface since the default interface for the sensor is UART. Then, the SD card data variables and pins are initialized. The chip-select pin on the MicroSD Card Module is set to pin 10, a file variable is set up for the SD card to write to, and two time variables are used to plot the time the over the distance is measured as well as time offset variable for the time plot if an SD card is not present in the module. Finally, the motor pins can be initiated for the movement and control of the car. Some of these pins will be connected to



the PWM (Pulse Width Modulation) to set the motor at desired speeds. A `currentMillis` and `previousMillis` are used as delays so that the car can record distance measurements very quickly while detecting the threshold distance to turn at. Figure 26 displays all of the library, pins, and variables initialized for the entire code of the project.

```
#include <SD.h> // Load the SD library
#include <SPI.h> // Load the SPI communication library

#include <Arduino.h>
#include <Wire.h> //Instantiate the Wire library
#include <TFLI2C.h> // TFLuna-I2C Library

TFLI2C tfI2C;

int16_t tfDist; // distance in centimeters
int16_t tfAddr = TFL_DEF_ADDR; // Use this default I2C address

const int chipSelect = 10; // Set chipSelect = 10
File mySensorData; // Variable for working with our file object
unsigned int timePlot; // Variable for plotting time
unsigned int timeOffset; // Variable for time offset

// Initialize Motor Pins
int ena = 5;
int in1 = 6;
int in2 = 7;
int in3 = 8;
int in4 = 9;
int enb = 3;

// use millis as delays to make measurements are quicker
long previousMillis = 0;
unsigned long currentMillis = 0;
```

**Figure 26: Library, Pin, and Variable Initialization Code**

In the void setup of the code, the serial monitor is initialized and read at a 115200 baud rate to get fast and accurate data collection. The wire library, chip select pin, and time variables are also initialized. Once the sketch has been uploaded to the Arduino, the code opens any existing files on the SD card and deletes them using the “O\_TRUNC” command in the SD

library. This is to make the data collection run more smoothly and not add any more data to existing files since it can affect the time plot values recorded. Finally, the motor pins are all set to OUTPUT since they will be supplying power to the L298N Motor Driver Controller. Figure 27 below shows the void setup code.

```
void setup() {  
  // SD Card Setup  
  Serial.begin(115200); // Initialize serial port  
  Wire.begin(); // Initialize Wire library  
  SD.begin(chipSelect); // Initialize the SD card with chipSelect connected to pin 10  
  timeOffset = millis(); // For when an SD card is not inserted, the time offset is considered  
  mySensorData = SD.open("DisData.csv", FILE_WRITE | O_TRUNC); // deletes any old data when sketch is uploaded  
  
  // Motor Setup  
  pinMode(ena, OUTPUT);  
  pinMode(in1, OUTPUT);  
  pinMode(in2, OUTPUT);  
  pinMode(enb, OUTPUT);  
  pinMode(in3, OUTPUT);  
  pinMode(in4, OUTPUT);  
  previousMillis = millis();  
}
```

**Figure 28: Void Setup Code**

In the main void loop, the file variable “mySensorData” is opened and written to. After that, an if-statement is called only if the data file has opened successfully. Then, the data recording and collection can begin. Calling the tfDist variable from the I2C interface, the distance measured by the sensor is recorded and printed in the serial monitor in centimeters and inches as well as the time the distance was recorded. Once recorded in the serial monitor, the sensor data file is written to, recording the time and distance in centimeters and inches in a .csv file, after which the file is closed for the next iteration of the loop. Then, a delay of 50 milliseconds is used to pause in between readings from the sensor so that it can record more accurately, while still being able to record quickly. Figure 29 below shows data recording and logging code in the void loop section.

```

void loop() {

mySensorData = SD.open("DisData.csv", FILE_WRITE); // Open DisData.txt (DistanceData) on the SD card as a file to write to

if (mySensorData) { // Only record data if data file opened successfully

  if (tfII2C.getData(tfDist, tfAddr)){
    Serial.println(String(tfDist)+" cm / " + String(tfDist/2.54) + " inches");

    timePlot = millis() - timeOffset; // record time the distance is measured
    Serial.println(timePlot); // print time

    mySensorData.print(timePlot); // Write time data to the SD card
    mySensorData.print(","); // Write comma to the line for comma delimited file
    mySensorData.print(tfDist); // Write distance data in cm and go to next line
    mySensorData.print(","); // Write comma to the line for comma delimited file
    mySensorData.println(tfDist/2.54); // Write distance data in inches
    mySensorData.close(); // Close the file

    delay(50); // pause between readings
  }
}
}

```

**Figure 29: Data Recording and Logging Code**

The next portion of the void loop is programmed solely for the 2 motors. If the sensor is collecting data, then the motors will always move forward. The in1 and in3 digital pins are set high while the in2 and in4 are set low to ensure the motors rotate in a clockwise direction. Then, the analogWrite command is used to set the speed of both motors. Figure 30 shows the code used to constantly make the motors move forward unless a certain condition is met.

```

// Constantly move forward unless close to object

//RIGHT MOTOR CLOCKWISE MAX SPEED
digitalWrite(in1,HIGH);
digitalWrite(in2,LOW);
analogWrite(ena, 155);

//LEFT MOTOR CLOCKWISE MAX SPEED
digitalWrite(in3,HIGH);
digitalWrite(in4,LOW);
analogWrite(enb, 155);

```

**Figure 30: Motor Control Code**

Since this is an autonomous collision-avoiding car, there must be a threshold distance for which the sensor reads, and if that threshold is met then it will turn to avoid the obstacle. For this application, the threshold distance is set to 20 cm and if the sensor measures any distance below 20 cm then it will turn left at a certain speed and continue moving forward if the distance recorded by the sensor is greater than 20 cm. If the threshold has been reached, the built-in millisecond counter is used as a delay for the turning so that the Lidar can continuously record distances at high speeds. If the difference between the current and the previous millisecond counter is less than 100 milliseconds then the car will turn left. The right motor is set to move in the same clockwise direction while the left motor moves in a counterclockwise direction by switching the in3 pin to LOW and the in4 pin to HIGH. A 200 millisecond delay is used so that the car has time to do a full left turn before continuously reading more distance data from the sensor. Once the current millisecond counter is equal to the previous millisecond counter, the previous millisecond counter will reset to the value of the current to allow the conditional statement to be repeated if the threshold is reached again. Finally, the last else, statement is nested outside the sensor data code. This conditional statement is used if the Arduino does not detect an SD card to write to or does not open the file correctly. In this conditional statement, the monitor will constantly print that the file did not open successfully. The time offset is also taken into account and is subtracted from the time plot variable when an SD card is detected and a file can be written in it. Figure 31 shows the conditional code if the lidar sensor measures a distance below the threshold.

```
// if RC car within a certain distance of object then turn left
if(tfDist < 20){
    currentMillis = millis();
    // Use millis to make measurements quicker with less delay
    if (currentMillis - previousMillis <= 100){
        // Turn Left
        //RIGHT MOTOR CLOCKWISE SPEED
        digitalWrite(in1,HIGH);
        digitalWrite(in2,LOW);
        analogWrite(ena, 100);

        //LEFT MOTOR COUNTERCLOCKWISE SPEED
        digitalWrite(in3,LOW);
        digitalWrite(in4,HIGH);
        analogWrite(enb, 100);
        delay(200);
    }
    // reset previousMillis if currentMillis goes above 100 ms
    else {
        previousMillis = millis();
    }
}
}
}
else {
    Serial.println("File did not open successfully");
    timeOffset = millis();
}
```

Figure 31: Conditional Threshold Statement Code

## 2.4. Data Processing

The distance data is written to a .csv file where the first column is the time that the sensor records the distance in milliseconds, the second column is the distance recorded in centimeters, and the third column is the distance in inches using the conversion factor 1 in = 2.54 cm. The collision-avoiding car was placed in a 37 x 30 x 5-inch box as seen in Figure 32 and allowed to move freely in the box for about 30 seconds while avoiding collisions with the outer edges.





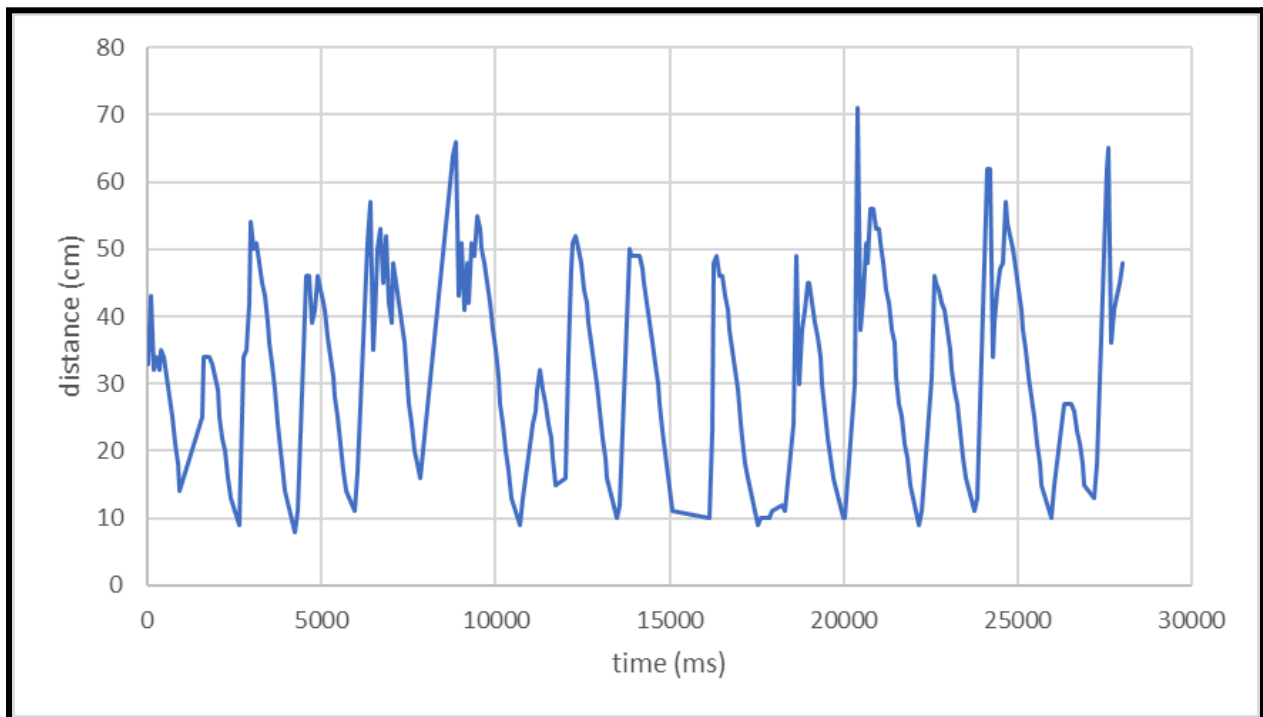
**Figure 32: Data Collection Testing Apparatus**

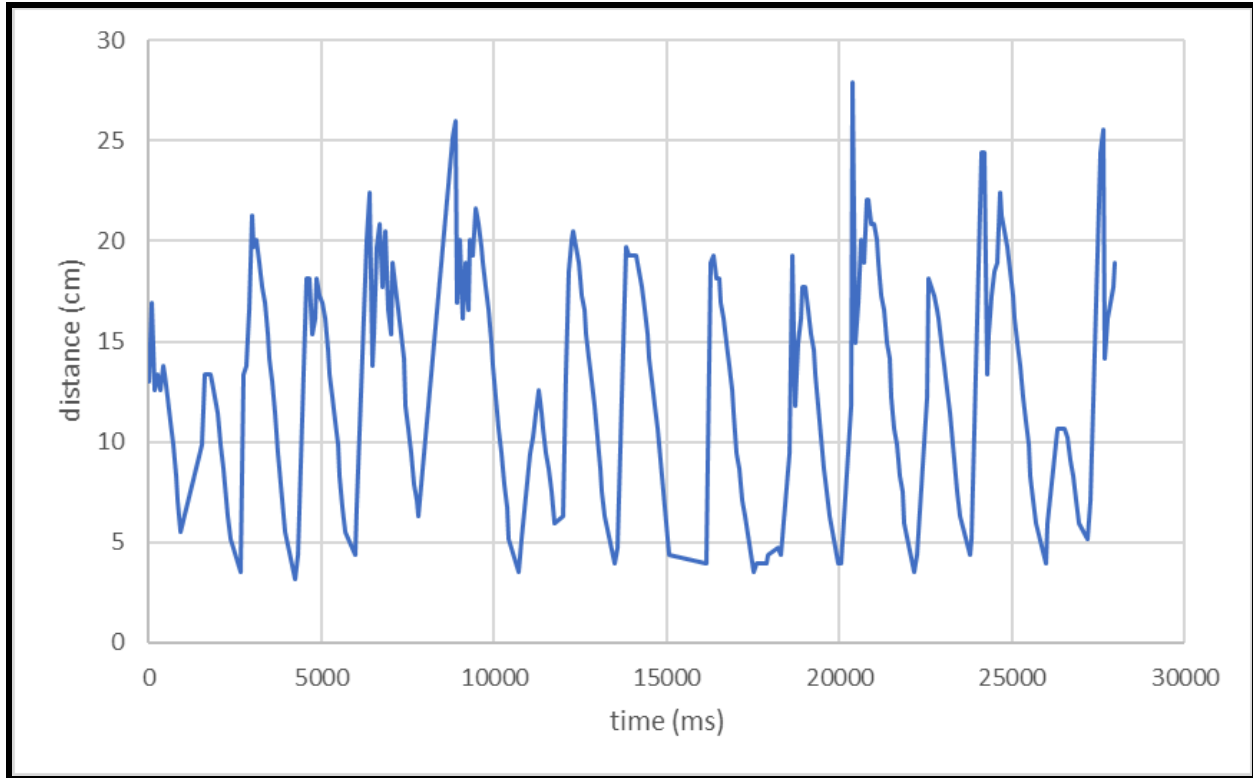
Once the data collection is complete, the SD card is extracted from the SD card module and inserted into a computer. From there, a .csv file named “DISDATA” has been created and can be opened directly in Excel for data analysis. Table 1 shows a sample dataset of the .csv file created from the data collection. Figures 33 and 34 are the distance versus time plots graphed from the dataset in both centimeters and inches.

It is important to note that, while the threshold distance is set to 20 cm, the lidar sensor can still record distances below 20 cm. The 20 cm threshold is a conditional statement, and once met, will program the collision-avoiding car to turn left until the distance between an object and the sensor is greater than 20 cm, and then will proceed to move forward. However, if the car continues to meet the threshold, for example, if it is in the corner of the test apparatus, then it will measure very small distances until the threshold is no longer reached.

**Table 1: Sample Dataset for Time and Distance**

Time (ms)	Distance (cm)	Distance (in)
15	33	12.99
105	43	16.93
179	32	12.6
253	34	13.39
328	32	12.6
402	35	13.78
477	34	13.39
558	31	12.2

**Figure 33: Distance (cm) vs Time**



**Figure 34: Distance (in) vs Time**

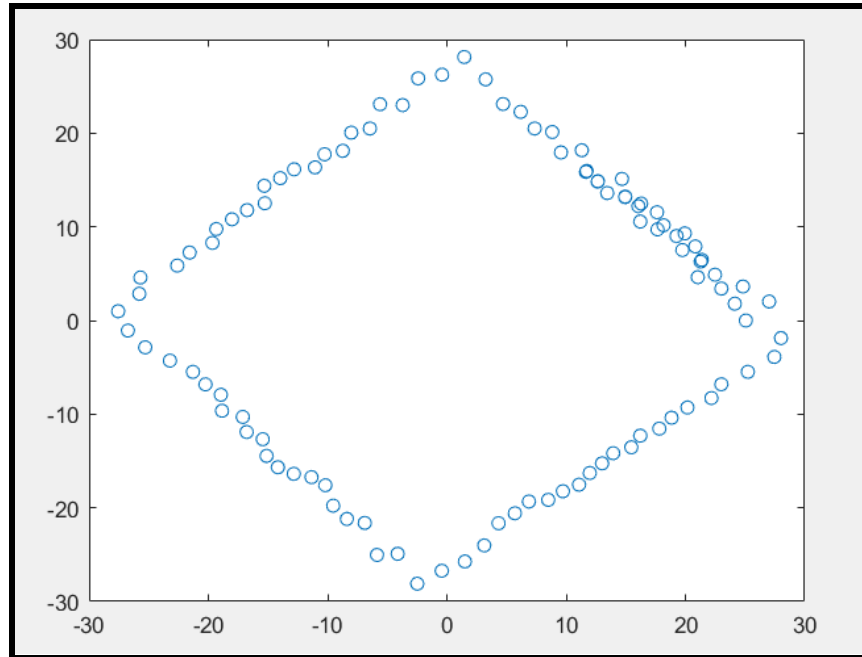
## 2.5. Room Scanning

The benefit of the sensor having its own locomotion is that it can move itself to take different measurements, as the robot does when moving and avoiding obstacles. By the same token, the sensor can instead record measurements while moving in a circle in place, and with enough granularity, gain a 2D picture of the space it is situated in.

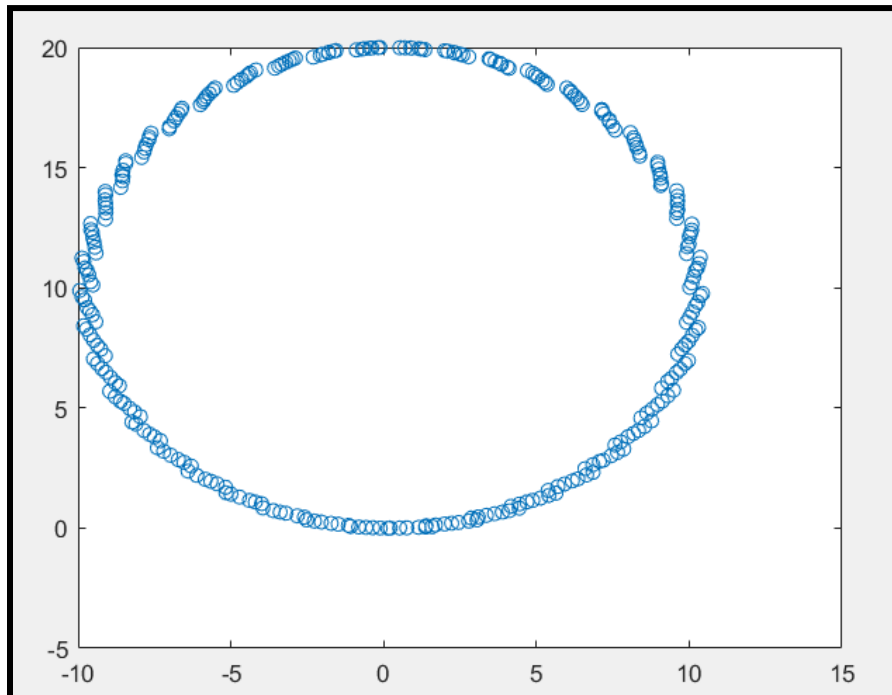
While higher-end LIDAR sensors send out pulses in all directions to create what is known as a “point cloud”, the TF-Luna LIDAR sensor only detects the distance immediately in front of it. By making the robot spin in place and constantly take distance measurements, however, a scan of the room can be achieved.

One of the main hurdles in getting a proper scan is not knowing how fast the robot is moving, which means that the time taken to spin 360 degrees cannot be assumed. We must, however, assume that the robot spins at a largely constant speed throughout the scan for the sake of simplicity. Finding the time taken to spin 360 degrees is done by taking a set of points at the beginning of the dataset, and comparing it to sets of points later in the dataset. When the difference between these two datasets is below a certain threshold, the time taken to spin is considered to be found. This is checked for the entire list, so we know the first proper time the values align. The time value at this state is our phase, which allows us to properly plot the distance data collected by the robot.





**Figure 35: Example Mapping of Square Room with Added Noise**



**Figure 36: Example Mapping of Circular Room with Added Noise**

### 3. Theory of Operation

LIDAR, short for Light Detection And Ranging [8], is a digital sensor that uses light to sense distance. The basic principle relies on the fact that light travels at a more-or-less known and constant speed; using this principle, a LIDAR sensor emits a pulse of light of a known wavelength and detects the same wavelength reflected back. By measuring the time elapsed between emitting the pulse and receiving its reflection, the distance to whatever reflected the light can be easily calculated. This principle is useful for near-instantaneous ranging and mapping. It is commonly used in mapping, surveying and navigation, such as in archaeology, construction, and space exploration [9]. It is also an important component of many self-driving cars currently in development [8]. Figure 37 is a diagram displaying the basic operation of a LIDAR sensor.

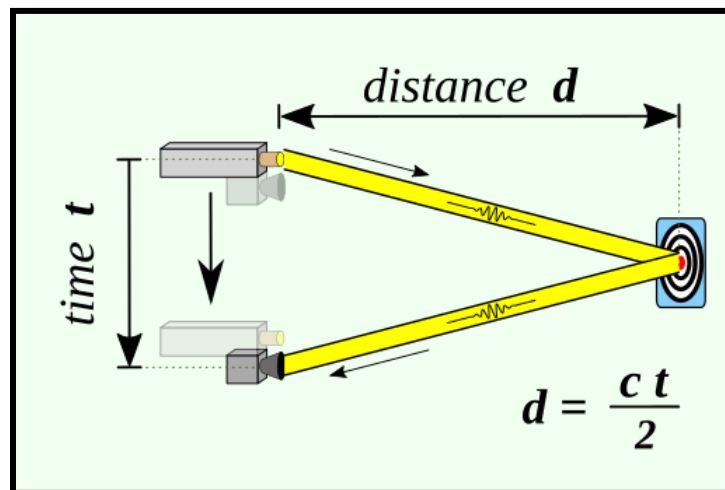
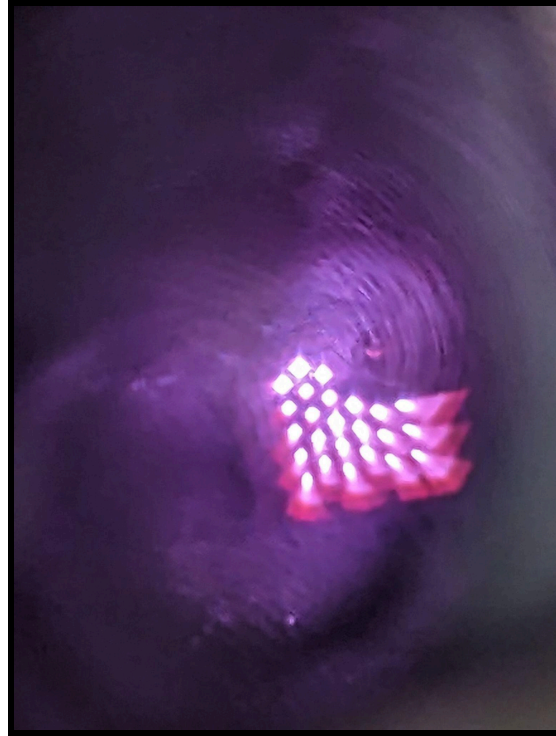


Figure 37: Diagram of LIDAR [10]

Most LIDAR sensors, including the sensor chosen for this project, use near-infrared lasers, as they are not visible to the human eye. However, many smartphone cameras are sensitive to near-infrared. Figure 38 shows a photo of the laser grid in the sensor, taken using a group member's smartphone.



**Figure 38: Laser Grid in LIDAR**

This photo shows both the grid of lasers as well as, faintly in the top right corner of the grid, the sensor that receives the light.

Originally, it was planned to use an ultrasonic sensor for this project. However, LIDAR was chosen instead. Ultrasonic sensors work on the same time-of-flight principle as LIDAR, but instead of infrared light, it uses ultrasonic sound waves, or sound of such high frequency that humans cannot hear it. While ultrasonic sensors are cheap and useful in many situations, there were multiple considerations that lead to the decision to use LIDAR instead. Ultrasonic sensors can struggle with detecting curved surfaces, or surfaces that could absorb soundwaves such as fabric. While ultrasonic sensors are useful for parking sensors in real cars, for a small car like the one in this project, it would encounter obstacles that would be better detected with LIDAR.

The specific LIDAR model used in this project, as stated earlier, is the TF-Luna LIDAR. This LIDAR is lightweight and short-ranged; it has a range of 0.2 m-8 m, with an accuracy of 6 cm or 2%, whichever is greater. It pulses at a frequency of 100 Hz, with a field of view of 2°. It emits 850 nm light, very-near-infrared, from Vertical-Cavity Surface-Emitting Lasers (VCSEL), in two semi-grids of 23 emitters, one behind each lens. Each set is paired with a receiver. [11] The distance data, recorded as stated at a rate of 100 Hz, is sent to the 2GB MicroSD card.

## 4. Tasks and Costs

### 4.1. Project Roles

#### Andy Hoang

I was responsible for researching and developing the code for the TF-Luna Lidar sensor, L298N Motor Controller Module, and Micro SD Card Module. I compiled all of the individual sensor/module codes and combined them together to make the car behave like an autonomous obstacle-avoiding vehicle while simultaneously recording distance measurements to be extracted into an Excel file. From this, I learned how to wire the electronics directly to the Arduino by using wire crimping/heat shrinking techniques and soldering methods with the help of Ryan. I was also responsible for supplying some of the equipment that was needed such as the Arduino and purchasing all of the products on Amazon. I had help from Jackson and Joey to print my design of the battery/SD card module case in Solid Edge. I also helped Rainey record and collect the calibration data from the sensor.

#### Joey Presa

I was responsible for the fabrication of the top cover and assisted in writing and bug fixing the code for the TF-Luna Lidar sensor, L298N Motor Controller Module, and Micro SD Card Module. I worked with Andy on both troubleshooting issues with the Micro SD Card Module as well as prototypes of the wiring used in the final version of the robot. I also worked with Jackson on the top cover of the robot, as my experience with FDM painting on my Ender 3 allowed us to adjust the design requirements of the cover itself to not only fit to the measurements of the base plate but also to be feasibly printable. After some failed prototypes, I was able to find accurate dimensions from the product's specifications and print the top cover with the correct measurements on the first attempt. Along with Jackson and Andy, I assisted with the final assembly of the robot. Lastly, I exclusively worked on the code for the Room Mapping functionality of the robot.

#### Jackson Neese

I was responsible for developing, fabricating, and testing the sensor mount, battery case, and top cover. With my previous experience in CAD, I was able to break down the necessary components into geometry that was easily sketched in the software. Using measurements from the base plate and other given objects, I assembled each component in SolidEdge. Printing each component using the AnyCubic Photon Mono 4k or the Ender 3, I tested each component's fit with the parts it would interact with and adjusted the SolidEdge specifications accordingly. I also created the drawings that list measurements for the components' part geometry and different views for the parts. I assisted with assembling the robot's prefabricated components and helped Andy and Joey with troubleshooting a few times when necessary. As for the circuit, I extended the premade sensor wires by crimping, heat shrinking, and binding new wires.

**Ian Rainey**

I was responsible for processing the collected data, analyzing said data for calibration, and producing the uncertainty estimates. My primary contribution was in the assessment of testing methods, improving and directing calibration data collection trials, and producing figures to represent data precision and accuracy. In addition, I took on an assistant role in the manufacturing process, primarily through occasional soldering when Ryan was indisposed, wrapping and preparing wiring, and generally assisting in construction wherever needed.

**Ryan Hay**

I was responsible for research and soldering connections on the car. As one of few members with experience soldering as well as the only member who enjoys soldering, I did the majority of all of the required soldering. This included both soldering multiple wires together, such as to lengthen wires, change the type of connection on one end of a wire, and to create a wire with multiple connections, and soldering connections between wires and components, in particular the motors. I also researched the theory and principles of operation of LIDAR.

**4.2. Hardware Costs**

The original budget of the project from the project proposal was estimated to be about \$59.44 to \$76.44 which did not include products and equipment that we already had. However, there were many deviations from the equipment needed to be used and the testing costs that dramatically impacted the total costs of the project. One of the main deviations in the original project proposal budget was changing the distance recording sensor from the HC-SR04 Ultrasonic Sensor to the TF-Luna Lidar Sensor. The cost of the ultrasonic sensor would have been around \$3.50 on Amazon, but it was not suitable for our application for an obstacle-avoiding car since it would have only measured accurately for perpendicular surfaces. Therefore, the team decided to switch to a Lidar sensor for more accurate single-point distance measurements that can also account for curved surfaces. Because Lidar is more accurate and sensitive, the cost of the sensor was about \$25.99 which is roughly 9 times more expensive than the ultrasonic sensor.

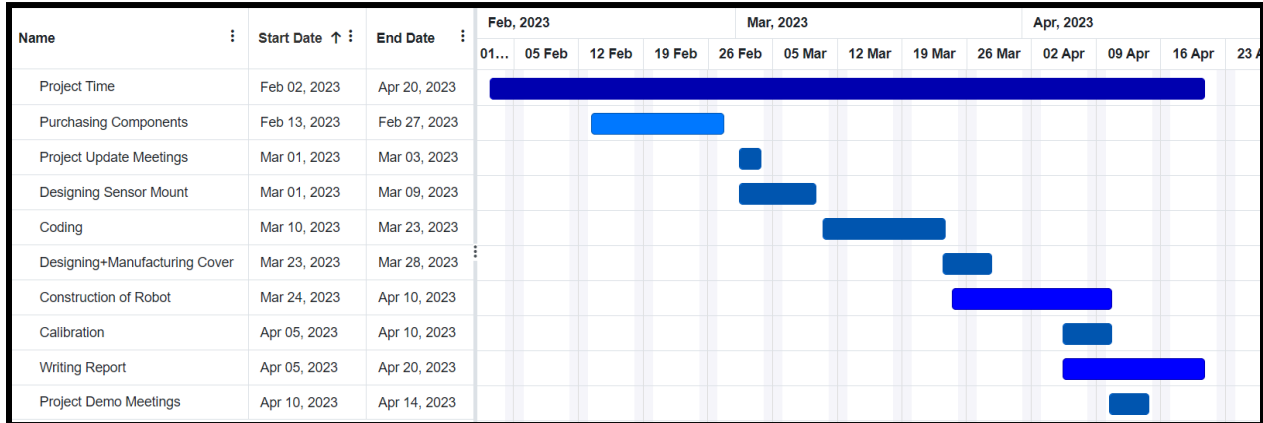
Another reason why the overall budget of the project increased was due to product malfunction and false purchases. There were a total of 3 products that were bought and not used in the final product, totaling to \$24.35 of the overall costs. One example of product malfunction was the Adafruit MicroSD Card Breakout Board+ which had an issue with pins that resulted in consistent wiring and connection issues. We read forums that said that soldering was the only solution to the problem that we were having, however, some said that soldering still could not fix the same problem [5]. The team concluded that purchasing a new data logging product was the best solution to not only have fewer wiring problems but also to decrease the amount of space taken up on the chassis of the car. We purchased the MicroSD Card Module since it had the same

pins used as the Adafruit Breakout Board, was smaller in size, and had male connectors out of the pins, ensuring proper and consistent wiring. The other two products that were purchased and not used in the final product were simply purchased by mistake. One, it was either thought of and when it arrived, the product was no longer needed in the end. And the other reason was the description of the product did not match the specific application we needed it for. For example, the 3 ft silicone wire was purchased for when any soldering or crimping of extra wires was needed, however, the silicone wire was too thick to be crimped or soldered to any of the other wire connections that were currently being used. Table 2 shows the specific costs of each product purchased as well as the location from where it was purchased, the person who purchased it, and if it was used in the final product. The item costs in red represent a return on the item and were, therefore, not included in the overall cost of the project.

**Table 2: Total Project Costs List**

<b>Product</b>	<b>Price (before tax)</b>	<b>Purchased By</b>	<b>Purchased From</b>	<b>Used in Final Product</b>
TF-Luna Lidar Sensor	\$25.99	Andy	Amazon	Y
Adafruit MicroSD Card Breakout Board	\$10.48	Andy	Amazon	N
Robot Smart Car Chassis Kit	\$17.05	Andy	Amazon	Y
12 V DC Power Connector Power Jack Plug	\$4.99	Andy	Amazon	Y
Heat Shrink Tubing	\$8.49	Andy	Amazon	Y
Male and Female Pin Connectors	\$7.88	Andy	Amazon	Y
L298N Motor Driver Controller Board Module	\$5.99	Andy	Amazon	Y
3 ft silicone wire	\$6.88	Andy	Amazon	N
Male to Female Jumper Wires	\$5.99	Andy	Amazon	Y
M3 Steel Pan Head Machine Screws	\$4.33	Andy	Amazon	Y
M3 Stainless Steel Hex Screw Nuts	\$6.69	Andy	Amazon	Y
Transcend 2 GB Micro SD card	(\$9.97)	Andy	Amazon	N
Micro SD card Module	\$12.99	Andy	Amazon	Y
M2 Steel Hex Screws and Nuts	\$12.99	Andy	Amazon	Y
2-3 pin Terminal Block Connector	(\$6.99)	Andy	Amazon	N
9 V batteries	\$8.99	Andy	Publix	Y
4 inch Cable Ties	\$1.97	Andy	Walmart	Y
Resin 3D Prints	\$2.22	Jackson	Amazon	Y
FDM 3D Prints	\$2.20	Joey	Amazon	Y
<b>Total Cost (before tax)</b>	\$129.16			

### 4.3. Schedule



**Figure 38: Project Schedule**

There were many factors that contributed to delaying the initially projected schedule in the project proposal as seen in Figure 38. We spent far more time constructing the robot itself than anticipated due to various difficulties with wiring and soldering all components properly. Designing the sensor mount also took more iterations than expected, due to tolerance issues and problems with the 3D printer used to manufacture it. We also needed to purchase new components due to difficulties with the initial Adafruit MicroSD Breakout Board.

Due to these situations, much of the testing had to be performed for each item separately and then combined into one final product. Since we had the Lidar sensor and the code working very early on, we would couple the sensor with one of the other electrical components that were available. While we were waiting for the Adafruit MicroSD card to be replaced by the MicroSD Card Module, we tested the L298N Motor Driver Controller paired with the Lidar sensor and developed code for the collision-avoiding car to perform turns based on a threshold distance measured from the Lidar sensor. Once we had complications with the DC motor wires not staying on the terminals, we would test the Lidar sensor paired with the MicroSD card module. We tested code for the sensor to capture the distance data recorded and open it in a .csv file on the MicroSD card after which it can be extracted into Excel for data analysis. After all of our components were available and ready to be tested, we were able to assemble a final product and with our previous testing of the individual components together, we were able to combine the code and have it perform all of the tasks that we set it to do. Despite the challenges we faced throughout the project, we were able to build a final product we were satisfied with and were also able to meet some of the projected scheduled deadlines such as calibration, project demonstration, and report writing.



## 5. Calibration and Uncertainty

### 5.1. Calibration

Calibration of the TF Luna LiDAR was confirmed through a series of static tests in which distance measurements were made under near-optimal conditions. According to the product manual for the TF Luna [12], known range and performance data come from tests indoors with a light surface of 90% reflectivity. For this reason, all calibration tests were performed in a well-lit hallway, with the sensor aimed with respect to the white walls being used as the sensing distance. Eight trials were performed at intervals of three feet from the referenced wall, starting at 3ft and ending at 24ft. Roughly 100 points of data were collected at each position, with 50 being used in the uncertainty analysis. A NIST-certified Stanley tape measure (LTM 696, Appendix J) was used to confirm the distance from the test stand to the wall. The test stand, pictured in Figure 39, consisted of a box on which the LiDAR sensor was taped. This allowed for a stable stand with consistent orientation when slotted against the base of the wall and floor, ensuring that data was collected from the same wall location. No test collected a point more than 2 inches from the expected value.



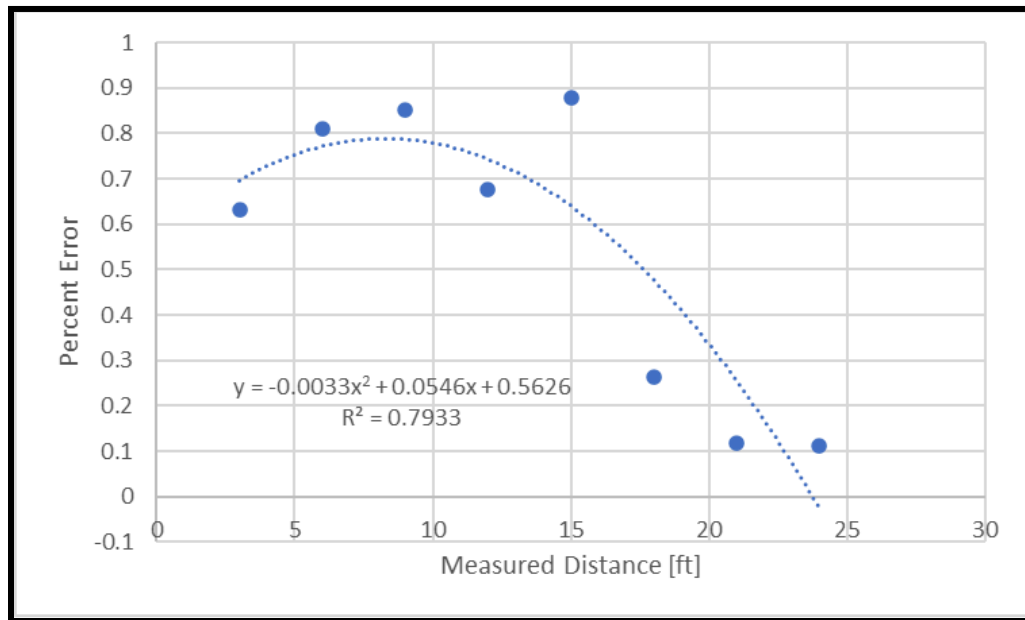
**Figure 39. Calibration Stand**

Percent error for the average distance measured at each location was calculated using Equation 1.

$$\delta = 100 * \frac{|d_{expected} - d_{measured}|}{d_{expected}} = 100 * \frac{|3 - 3.018983|}{3} = 0.632778\% \quad (1)$$

The percent error is plotted against the measured distance in Figure 40. Analysis of this data shows a drop in percent error with distance as the absolute error remains relatively unchanged, though the standard deviation and direction in which error changes does increase, as shown in the excerpt from Table 3. The polynomial curve fit indicates that slight correction could

be made to data as range decreases, though doing so was deemed unnecessary for this application.

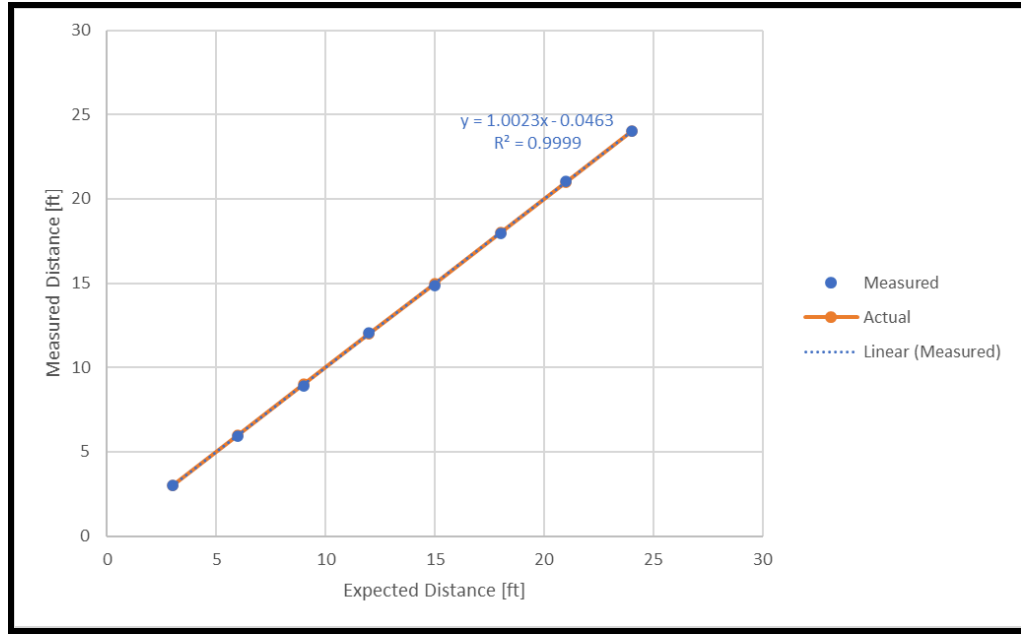


**Figure 40. Percent Error versus Measured Distance**

**Table 3: Standard Deviation and Difference with Distance**

	3 ft	6 ft	9 ft	12 ft	15 ft	18 ft	21 ft	24 ft
Standard Deviation [in]	0.055154	0.261814	0.097806	0.220455	0.926032	0.420795	0.332566	1.073813
Measured-Expected [in]	0.018983	-0.04865	-0.07652	0.081317	0.1319	0.04735	0.024967	0.026917

Figure 41 shows the average of 50 points collected at each interval plotted against the expected value, producing an  $R^2$  value of 0.9999, according to Excel. It was produced using the unprocessed data, presented in Appendix K and L.



**Figure 41. Measured Distance versus Expected Distance**

## 5.2. Uncertainty

Calibration and random error are combined to produce a total uncertainty for the TF Luna sensor. The calibration uncertainty was calculated in a manner similar to the error from part 5.1. This time, the percent error is calculated for each of the 400 points via Equation 2, then averaged. This produces the estimate for calibration error, which comes to approximately 0.573%. The corresponding excerpt can be found in Appendix O.

The random error was found by first calculating the total standard deviation of each trial through the stdev function in Excel. Then, the random uncertainty for each trial can be found via Equation 2, where the calculation for the 3ft trial is demonstrated.

$$u_{random} = 100 * \frac{2*\sigma}{d_{expected}} = 100 * \frac{2*0.055154}{36 \text{ in}} = 0.3064 \quad (2)$$

In this instance,  $\sigma$  is the standard deviation of a particular 50 point trial and  $d_{expected}$  is the anticipated value in inches at that range. An average of the random uncertainty for the 8 trials produces a total random uncertainty of 0.4939%

The total uncertainty is then found via Equation 3.

$$u_{total} = \sqrt{u_{random}^2 + u_{calibration}^2} = \sqrt{0.573^2 + 0.439^2} = 0.756\% \quad (3)$$

This produces an equivalent uncertainty of  $\pm 0.2722$  inches at a range of 3 ft, up to  $\pm 2.178$  inches at 24 ft, consistent with the collected data.

An important point to note is the presence of resolution error. Though the TF Luna does not suffer from the shortcomings of sensors that relay data through a voltage range, data is still collected in 1cm intervals, as indicated in the manual [12]. This is notable, as it partially explains the large calibration error that decreases with increasing distance, as seen in Figure 40. At 24 ft,  $\pm 1$ cm uncertainty is only 0.137%, but at 3 ft it is responsible for 1.09% uncertainty. Thus, appropriate processing of LiDAR data ideally accounts for this when assessing measurements at close range and in close grouping.

### 5.3. Testing

In general, we tested the collision-avoiding car in 4 different ways to ensure that the code, wiring, and assembly were consistent throughout the project. We first started with the Lidar sensor itself since it will be the main component of our project. The first test involved uploading code to the Arduino with the Lidar sensor properly wired and tested to see if the sensor was producing relatively accurate distance measurements. After the first test was completed, we transitioned to combining the existing code with the code for the motor driver controller. The purpose of this test was to be able to have the car perform a turn if a distance was measured below a certain value. The third test was similar to the first test in which we only needed the Lidar sensor to measure distances and record them in the serial monitor. However, instead of only recording it to the serial monitor, we added code for it to open a .csv file on a MicroSD card and record the data onto the file. The fourth and final test incorporated all of the previous successful tests into one running code that will then be uploaded to the final assembly of the car. This final test would allow the collision-avoiding car to roam in a testing apparatus, avoid obstacles if the sensor measures a distance below a threshold, and send the distance data to the MicroSD card. Finally, once the test is finished the MicroSD card is extracted from the assembly and the file can be opened in Excel for data analysis. Table 4 shows a summarized description of the 4 different tests that were performed.

**Table 4: Tests and Descriptions**

<b>Test</b>	<b>Description</b>
Lidar	Ensured the code was working and that the Lidar sensor was recording accurate distances by moving objects towards and away from the sensor
Lidar with Motor Driver Controller	Combined code for the lidar sensor and the motor driver controller for the collision-avoiding car to perform a turn if a threshold distance was met
Lidar with MicroSD Card Module	Combined code for the lidar sensor and the MicroSD Card Module to have the sensor record data, send it to a .csv file on the MicroSD card, and have it extracted and analyzed on Excel
Lidar, Motor Driver Controller, and MicroSD Card Module	Combined the code for the lidar sensor, motor driver controller and MicroSD card module to have the collision-avoiding car avoid obstacles, record distance data, and save it in the MicroSD card

## 6. Lessons Learned

### Cad Modeling

In SolidEdge's relationship-based sketch environment, modifying a pre-established measurement can often have unintended consequences on other measurements in the sketch. As such, many measurements, particularly on the unusual geometry of the top cover, can be slightly off, unbeknownst to the user. Because of this, the top cover had various issues in its design and required more iterations than should have been necessary. Additionally, the base plate itself seemed to have been constructed using a combination of metric and US imperial measurements. This caused several issues as the measurements were sometimes not a traditional number, such as a fraction or whole number. Because of this, gathering measurements on the plate was challenging due to a lack of specifications from the manufacturer and limited access to precision measuring devices. These issues reinforced the "Measure Twice, Cut Once" philosophy when it comes to implementation. The importance of verifying one's measurements was clearly demonstrated in the struggles in designing the project's components in CAD.

## Data Logging

When attempting to record the data measured to an SD card, we first tried to use an Adafruit SD Breakout Board+ and a 32 GB SD card. Both of these components, however, had various issues that showed us a lot about how these components function. The main difficulty was that the SD card would not initialize, which precluded us from taking any data at all. Surprisingly, software was not the failure point of this system, as both the SD Breakout Board and the SD Card itself were faulty. The Breakout Board had loosely fitting holes intended to be soldered together, but we were not prepared to solder at the time, so the connections were unreliable, and caused the system to fail. The SD card's failure points were twofold: it was far too large in memory for the Breakout Board to properly initialize, and it was formatted incorrectly. Both of these issues were solved with a 2GB SD card in the FAT16 format as well as a new MicroSD card module with easier-to-connect pins. The main takeaway from this for us was to always read and understand the documentation of a board and what it is built to handle, no matter how simple the board is to use. However, we noticed that when using the new MicroSD card module, with better wire connections, we were able to log data onto the 32 GB MicroSD card. This meant that the main cause of our SD card problem was solely on the wiring of the Adafruit MicroSD card Breakout Board+. Because of this, we no longer needed the 2 GB MicroSD card and returned the card to make back some extra money.

## Wiring

After deviating from the ultrasonic sensor to the lidar sensor as our distance measuring medium, we found that the level of complexity for the wiring increased with the new change. The wires on the TF-Luna Lidar sensor were too thin to connect directly to the Arduino. Because of this, it would cause the connection to the Arduino to be unstable and possibly record inaccurate data. To combat this problem, we researched how to properly crimp wires. Crimping wires is essentially connecting wires to a metal terminal that can then be used to connect to another terminal, for example, the Arduino. We bought male and female pin connectors, cut/stripped the wires to a desired length, and then wrapped the connectors around the wire making sure that the copper part of the wire was in contact with the connector. Once the wire was secured to the connector, the wires could be plugged into the Arduino without any loose connections. To ensure that the wires can't come loose from the connectors, we added heat shrink to the exposed part of the wire and applied enough heat to it to get a tight fit around the wire and connector. We would repeat this process for wires that needed to be longer or shorter in length. We learned that crimping wires is very beneficial for not only ensuring a strong, reliable connection from the electronic components to the Arduino but also to vary the length of wires to make cable management of the final product more efficient.

## Soldering

Our project used many electrical components that required connection to the 5V pin and GND pin on the Arduino. The main problem we had after we were able to get proper connections

from the wires was the amount of space we could save on the Arduino. With the L298N Motor Driver Module, MicroSD Card Module, TF-Luna Lidar sensor, and 2 DC motors there was simply not enough space for each wire to have its own pin connection on the Arduino. Because of this, we resulted to soldering some of the wires together so that we could connect 2 ground wires to 1 ground pin on the Arduino board. Since the TF-Luna Lidar sensor had 2 ground wires, we soldered a wire to connect both ground wires together and then we were able to connect both wires to 1 ground pin. We also soldered a wire that could connect to the 5V pin and had 2 female connectors so that the Lidar and SD card module could receive power from the same pin. Finally, once we solved the issue with pin space, we had a problem with the wires on both DC motors ripping off constantly. Since we were testing the car numerous times and transporting it back and forth as well, the wires on the motors underwent a lot of stress and caused many connection issues. We soldered the wires constantly and yet, we were still having the same issue. We figured out that the main problem was not the soldering, but it was the strength of the copper in the wire not holding onto the terminals of the motor. After repeated soldering, the terminals of the motors were wearing out, and we were cutting it close to how far we could solder to the terminals without damaging any internal components. We were able to solder the wires to the motors one last time and the problem has not come up since. From this problem, we learned to always take into consideration the integrity of the wire strength when soldering as well as considering the amount of stress the wires go through when testing and transporting it.

### **Budgeting**

During the entirety of this project, we ran into problems with budgeting to keep our total costs at a minimum. We knew the cost of the total project would be relatively more expensive than our proposal since we did not factor in the costs of the motor driver controller, car chassis, 2 DC Motors, or the transition from ultrasonic sensor to lidar sensor. However, many small miscellaneous expenses involved in testing such as batteries, screws, heat shrinking tubes, male and female pin connectors, and a micro SD card were not considered and greatly impacted the costs. We also came across issues purchasing items that were either not what we needed or malfunctioned during testing and had to be replaced. This issue also added more to the total costs of the project. Overall, there were many expenses that we did not foresee, were not used, or malfunctioned and had to be replaced. We learned that when budgeting for a project that involves purchasing many electrical and mechanical components we should perform more detailed research on the specific parts that are needed and decide if they are 100% applicable to the project before purchasing the items. This cost-estimating research should be done far in advance so that the items can arrive in time for assembling, calibrating, and testing.

### **Uncertainty and Processing**

While the data collected in the calibration phase was sufficient for identifying characteristics of this sensor setup, it became clear that additional testing of a different variety could have been beneficial, particularly if we were planning to use the setup for closer testing



rather than long range scans. Since the LiDAR sensor we chose to work with has a flat resolution uncertainty of 1cm, close measurements tend to have a far higher uncertainty due to the higher proportionality. Starting again, testing could have proceeded from the very minimum detection range up to the maximum, with a shorter interval within the first few meters. This would have made analyzing the exact impact of resolution more clear, as identifying its effects in the intervals we collected is still relatively unsure. The same result could be achieved by simply taking more tests on smaller intervals, even if fewer overall data points are collected.

## 7. References

1. "Lidar with Arduino - TF-Luna Sensor." *YouTube*, YouTube, 2 June 2022, [https://www.youtube.com/watch?v=QMW1H0owzdY&ab\\_channel=DIYEngineers](https://www.youtube.com/watch?v=QMW1H0owzdY&ab_channel=DIYEngineers).
2. "Soldering & Crimping Wires for DIY Arduino Projects." *YouTube*, YouTube, 19 Apr. 2021, [https://www.youtube.com/watch?v=6ipeC9eb1lM&ab\\_channel=SoundSimulator](https://www.youtube.com/watch?v=6ipeC9eb1lM&ab_channel=SoundSimulator).
3. "Lesson 21: Data Logging from Arduino to SD Card." *YouTube*, YouTube, 29 July 2014, [https://www.youtube.com/watch?v=-nXSUXJz4QQ&ab\\_channel=PaulMcWhorter](https://www.youtube.com/watch?v=-nXSUXJz4QQ&ab_channel=PaulMcWhorter).
4. "How to Control a DC Motor with L298N Driver and Arduino Uno." *YouTube*, YouTube, 3 Nov. 2020, [https://www.youtube.com/watch?v=Ey4xoG970Go&ab\\_channel=educ8s.tv](https://www.youtube.com/watch?v=Ey4xoG970Go&ab_channel=educ8s.tv).
5. "Forums Index." *Micro SD Breakout Board Not Initializing*, <https://forums.adafruit.com/viewtopic.php?f=22&t=138806>.
6. Last Minute Engineers. "In-Depth Tutorial to Interface Micro SD Card Module with Arduino." *Last Minute Engineers*, Last Minute Engineers, 10 Oct. 2022, <https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/>.
7. Kakarlapudi, Mani Kumar. "L298n Motor Driver Pin Diagram, Working, Datasheet & Arduino Connection." *ETechnophiles*, 1 Apr. 2023, <https://www.etechnophiles.com/l298n-motor-driver-pin-diagram/>.
8. Velodyne Lidar. "What Is Lidar? Learn How Lidar Works | Velodyne Lidar." *Velodyne Lidar*, 3 June 2022, [velodynelidar.com/what-is-lidar](https://velodynelidar.com/what-is-lidar).
9. Ackerman, Evan. "How NASA Designed a Helicopter That Could Fly Autonomously on Mars." *IEEE Spectrum*, 18 Aug. 2022, [spectrum.ieee.org/nasa-designed-perseverance-helicopter-rover-fly-autonomously-mars](https://spectrum.ieee.org/nasa-designed-perseverance-helicopter-rover-fly-autonomously-mars).

10. *File:20200501 Time of flight.svg - Wikimedia Commons*. 1 May 2020, [commons.wikimedia.org/wiki/File:20200501\\_Time\\_of\\_flight.svg](https://commons.wikimedia.org/wiki/File:20200501_Time_of_flight.svg).

11. “TF-Luna LiDAR Module - Short-Range Distance Sensor.” *Seeed Studio*, [www.seeedstudio.com/TF-Luna-LiDAR-Module-Short-Range-Distance-Sensor-p-4561.html](https://www.seeedstudio.com/TF-Luna-LiDAR-Module-Short-Range-Distance-Sensor-p-4561.html).

12. “Product Manual of TF-Luna” *Benewake*, acc 4 April 2023, [SJ-PM-TF-Luna+A01+Product+Manual.pdf](#)

## 8. Appendix A: Full Code

### Appendix A - Lidar Sensor Code

```
1  #include <Arduino.h>
2  #include <Wire.h>      //Instantiate the Wire library
3  #include <TFLI2C.h>    // TFLuna-I2C Library
4
5  TFLI2C tfLI2C;
6
7  int16_t tfDist;        // distance in centimeters
8  int16_t tfAddr = TFL_DEF_ADDR; // Use this default I2C address
9
10 void setup() {
11     Serial.begin(115200); // Initialize serial port
12     Wire.begin();         // Initialize Wire library
13 }
14
15 void loop() {
16     if (tfLI2C.getData(tfDist, tfAddr)){
17         Serial.println(String(tfDist)+" cm / " + String(tfDist/2.54) + " inches");
18     }
19     delay(50);
20 }
21
```

**Appendix B - Motor Driver Code**

```
1 //MOTOR1 PINS
2 int ena = 5;
3 int in1 = 6;
4 int in2 = 7;
5 int in3 = 8;
6 int in4 = 9;
7 int enb = 3;
8
9 void setup() {
10
11   Serial.begin(9600);
12   pinMode(ena, OUTPUT);
13   pinMode(in1, OUTPUT);
14   pinMode(in2, OUTPUT);
15   pinMode(enb, OUTPUT);
16   pinMode(in3, OUTPUT);
17   pinMode(in4, OUTPUT);
18
19 }
20
```

**Appendix C - Motor Driver Code (cont.)**

```
21 void loop() {  
22  
23 // Forwards  
24  
25 //RIGHT MOTOR CLOCKWISE MAX SPEED  
26 digitalWrite(in1,HIGH);  
27 digitalWrite(in2,LOW);  
28 analogWrite(ena, 255);  
29  
30 //LEFT MOTOR CLOCKWISE MAX SPEED  
31 digitalWrite(in3,HIGH);  
32 digitalWrite(in4,LOW);  
33 analogWrite(enb, 255);  
34 delay(2000);  
35  
36 //STOP  
37 digitalWrite(in1,LOW);  
38 digitalWrite(in2,LOW);  
39 digitalWrite(in3,LOW);  
40 digitalWrite(in4,LOW);  
41 delay(2000);  
42  
43 //backwards  
44  
45 //RIGHT MOTOR COUNTERCLOCKWISE MAX SPEED  
46 digitalWrite(in1,LOW);  
47 digitalWrite(in2,HIGH);  
48 analogWrite(ena, 255);  
49  
50 //LEFT MOTOR COUNTERCLOCKWISE MAX SPEED  
51 digitalWrite(in3,LOW);  
52 digitalWrite(in4,HIGH);  
53 analogWrite(enb, 255);  
54  
55 delay(2000);  
56  
57 //STOP  
58 digitalWrite(in1,LOW);  
59 digitalWrite(in2,LOW);  
60 digitalWrite(in3,LOW);  
61 digitalWrite(in4,LOW);  
62 delay(2000);  
}
```

**Appendix D - Motor Driver Code (cont.)**

```
64 // Turn Left
65
66 //RIGHT MOTOR CLOCKWISE SPEED
67 digitalWrite(in1,HIGH);
68 digitalWrite(in2,LOW);
69 analogWrite(ena, 95);
70
71 //LEFT MOTOR COUNTERCLOCKWISE SPEED
72 digitalWrite(in3,LOW);
73 digitalWrite(in4,HIGH);
74 analogWrite(enb, 95);
75 delay(1000);
76
77 //STOP
78 digitalWrite(in1,LOW);
79 digitalWrite(in2,LOW);
80 digitalWrite(in3,LOW);
81 digitalWrite(in4,LOW);
82 delay(2000);
83
84 // Turn right
85
86 //RIGHT MOTOR CLOCKWISE SPEED
87 digitalWrite(in1,LOW);
88 digitalWrite(in2,HIGH);
89 analogWrite(ena, 95);
90
91 //LEFT MOTOR COUNTERCLOCKWISE SPEED
92 digitalWrite(in3,HIGH);
93 digitalWrite(in4,LOW);
94 analogWrite(enb, 95);
95 delay(1000);
96
97 //STOP
98 digitalWrite(in1,LOW);
99 digitalWrite(in2,LOW);
100 digitalWrite(in3,LOW);
101 digitalWrite(in4,LOW);
102 delay(2000);
103 }
```

**Appendix E - Motor Driver and Lidar Sensor Code**

```
1  #include <Arduino.h>
2  #include <Wire.h>      //Instantiate the Wire library
3  #include <TFLI2C.h>    // TFLuna-I2C Library
4
5  TFLI2C tfI2C;
6
7  int16_t tfDist;        // distance in centimeters
8  int16_t tfAddr = TFL_DEF_ADDR; // Use this default I2C address
9
10 // Initialize Motor Pins
11 int ena = 5;
12 int in1 = 6;
13 int in2 = 7;
14 int in3 = 8;
15 int in4 = 9;
16 int enb = 3;
17
18 // use millis as delays to make measurements quicker
19 long previousMillis = 0;
20 unsigned long currentMillis = 0;
21
22 void setup() {
23     Serial.begin(115200); // Initialize serial port
24     Wire.begin();         // Initialize Wire library
25     pinMode(ena, OUTPUT);
26     pinMode(in1, OUTPUT);
27     pinMode(in2, OUTPUT);
28     pinMode(enb, OUTPUT);
29     pinMode(in3, OUTPUT);
30     pinMode(in4, OUTPUT);
31     previousMillis = millis();
32 }
33
```

## Appendix F - Motor Driver and Lidar Sensor Code (cont.)

```

34 void loop() {
35
36   if (tflr2c.getData(tfDist, tfAddr)){
37     Serial.println(String(tfDist)+" cm / " + String(tfDist/2.54) + " inches");
38     delay(50);
39
40     // Constantly move forward unless close to object
41
42     //RIGHT MOTOR CLOCKWISE MAX SPEED
43     digitalWrite(in1,HIGH);
44     digitalWrite(in2,LOW);
45     analogWrite(ena, 155);
46
47     //LEFT MOTOR CLOCKWISE MAX SPEED
48     digitalWrite(in3,HIGH);
49     digitalWrite(in4,LOW);
50     analogWrite(enb, 155);
51
52     // if RC car within a certain distance of object then go backwards and then move left
53     if(tfDist < 20){
54       currentMillis = millis();
55
56       if(currentMillis - previousMillis <= 100){
57         Serial.println("BACK");
58         //backwards
59         //RIGHT MOTOR COUNTERCLOCKWISE MAX SPEED
60         digitalWrite(in1,LOW);
61         digitalWrite(in2,HIGH);
62         analogWrite(ena, 155);
63
64         //LEFT MOTOR COUNTERCLOCKWISE MAX SPEED
65         digitalWrite(in3,LOW);
66         digitalWrite(in4,HIGH);
67         analogWrite(enb, 155);
68         delay(200);
69       }
70
71       else if (currentMillis - previousMillis <= 1000){
72         Serial.println("LEFT");
73         // Turn Left
74         //RIGHT MOTOR CLOCKWISE SPEED
75         digitalWrite(in1,HIGH);
76         digitalWrite(in2,LOW);
77         analogWrite(ena, 85);
78
79         //LEFT MOTOR COUNTERCLOCKWISE SPEED
80         digitalWrite(in3,LOW);
81         digitalWrite(in4,HIGH);
82         analogWrite(enb, 85);
83         delay(200);
84       }
85
86       // reset previousMillis if currentMillis goes above 1000
87       else {
88         previousMillis = millis();
89       }
90     }
91   }
92 }
93
94 }
95

```



## Appendix G - Lidar Sensor and MicroSD Card Module Code

```

1  #include <SD.h> // Load the SD library
2  #include <SPI.h> // Load the SPI communication library
3
4  #include <Arduino.h>
5  #include <Wire.h> //Instantiate the Wire library
6  #include <TFI2C.h> // TFLuna-I2C Library
7
8  TFI2C tflI2C;
9
10 int16_t tfDist; // distance in centimeters
11 int16_t tfAddr = TFL_DEF_ADR; // Use this default I2C address
12
13 const int chipSelect = 10; // Set chipSelect = 10
14 File mySensorData; // Variable for working with our file object
15 unsigned int timePlot;
16 unsigned int timeOffset;
17
18 void setup() {
19     Serial.begin(115200); // Initialize serial port
20     Wire.begin(); // Initialize Wire library
21     SD.begin(chipSelect); // Initialize the SD card with chipSelect connected to pin 10
22     timeOffset = millis(); // accounts for time offset
23     mySensorData = SD.open("DisData.csv", FILE_WRITE | O_TRUNC); // deletes any old data when sketch is uploaded
24 }
25
26 void loop() {
27
28     mySensorData = SD.open("DisData.csv", FILE_WRITE); // Open DisData.txt (DistanceData) on the SD card as a file to write to
29
30     if (mySensorData) { // Only record data if data file opened successfully
31
32         if (tflI2C.getData(tfDist, tfAddr)) {
33             Serial.println(String(tfDist)+" cm / " + String(tfDist/2.54) + " inches"); // print distance
34
35             timePlot = millis() - timeOffset; // record time the distance is measured
36             Serial.println(timePlot); // print time
37
38             mySensorData.print(timePlot); // Write time data to the SD card
39             mySensorData.print(","); // Write comma to the line for comma delimited file
40             mySensorData.print(tfDist); // Write distance data in cm and go to next line
41             mySensorData.print(",");
42             mySensorData.println(tfDist/2.54);
43             mySensorData.close(); // Close the file
44
45             delay(50); // pause between readings
46         }
47     }
48 }
49
50 else{
51     Serial.println("File did not open successfully");
52     timeOffset = millis();
53 }
54
55 }

```

## Appendix H - Lidar Sensor, MicroSD Card Module, and Motor Driver Code

```

1  #include <SD.h> // Load the SD library
2  #include <SPI.h> // Load the SPI communication library
3
4  #include <Arduino.h>
5  #include <Wire.h> //Instantiate the Wire library
6  #include <TFLI2C.h> // TFLuna-I2C Library
7
8  TFLI2C tflI2C;
9
10 int16_t tflDist; // distance in centimeters
11 int16_t tflAddr = TFL_DEF_ADR; // Use this default I2C address
12
13 const int chipSelect = 10; // Set chipSelect = 10
14 File mySensorData; // Variable for working with our file object
15 unsigned int timePlot; // Variable for plotting time
16 unsigned int timeOffset; // Variable for time offset
17
18 // Initialize Motor Pins
19 int ena = 5;
20 int in1 = 6;
21 int in2 = 7;
22 int in3 = 8;
23 int in4 = 9;
24 int enb = 3;
25
26 // use millis as delays to make measurements are quicker
27 long previousMillis = 0;
28 unsigned long currentMillis = 0;
29
30 void setup() {
31     // SD Card Setup
32     Serial.begin(115200); // Initialize serial port
33     Wire.begin(); // Initialize Wire library
34     SD.begin(chipSelect); // Initialize the SD card with chipSelect connected to pin 10
35     timeOffset = millis(); // For when an SD card is not inserted, the time offset is considered
36     mySensorData = SD.open("DisData.csv", FILE_WRITE | O_TRUNC); // deletes any old data when sketch is uploaded
37
38     // Motor Setup
39     pinMode(ena, OUTPUT);
40     pinMode(in1, OUTPUT);
41     pinMode(in2, OUTPUT);
42     pinMode(enb, OUTPUT);
43     pinMode(in3, OUTPUT);
44     pinMode(in4, OUTPUT);
45     previousMillis = millis();
46 }

```

## Appendix I - Lidar Sensor, MicroSD Card Module, and Motor Driver Code (cont.)

```

48 void loop() {
49
50 mySensorData = SD.open("DisData.csv", FILE_WRITE); // Open DisData.txt (DistanceData) on the SD card as a file to write to
51
52 if (mySensorData) { // Only record data if data file opened successfully
53
54     if (tfll2c.getData(tfDist, tfAddr)){
55         Serial.println(String(tfDist)+" cm / " + String(tfDist/2.54) + " inches");
56
57         timePlot = millis() - timeOffset; // record time the distance is measured
58         Serial.println(timePlot); // print time
59
60         mySensorData.print(timePlot); // Write time data to the SD card
61         mySensorData.print(","); // Write comma to the line for comma delimited file
62         mySensorData.print(tfDist); // Write distance data in cm and go to next line
63         mySensorData.print(","); // Write comma to the line for comma delimited file
64         mySensorData.println(tfDist/2.54); // Write distance data in inches
65         mySensorData.close(); // Close the file
66
67         delay(50); // pause between readings
68
69         // Constantly move forward unless close to object
70
71         //RIGHT MOTOR CLOCKWISE MAX SPEED
72         digitalWrite(in1,HIGH);
73         digitalWrite(in2,LOW);
74         analogWrite(ena, 155);
75
76         //LEFT MOTOR CLOCKWISE MAX SPEED
77         digitalWrite(in3,HIGH);
78         digitalWrite(in4,LOW);
79         analogWrite(enb, 155);
80
81         // if RC car within a certain distance of object then turn left
82         if(tfDist < 20){
83             currentMillis = millis();
84             // Use millis to make measurements quicker with less delay
85             if (currentMillis - previousMillis <= 100){
86                 // Turn Left
87                 //RIGHT MOTOR CLOCKWISE SPEED
88                 digitalWrite(in1,HIGH);
89                 digitalWrite(in2,LOW);
90                 analogWrite(ena, 100);
91
92                 //LEFT MOTOR COUNTERCLOCKWISE SPEED
93                 digitalWrite(in3,LOW);
94                 digitalWrite(in4,HIGH);
95                 analogWrite(enb, 100);
96                 delay(200);
97             }
98             // reset previousMillis if currentMillis goes above 100 ms
99             else {
100                 previousMillis = millis();
101             }
102         }
103     }
104 }
105 else {
106     Serial.println("File did not open successfully");
107     timeOffset = millis();
108 }
109
110 }
111

```

## 9. Appendix B: Calibration Data

## Appendix J - LTM 696 Tape Measure NIST Certification

**Lixer Tools LLC**  
363 W. 300 N. Lindon, UT 84042 801-796-7066

**Certificate of Calibration for Tape Measures**

Tape Measure Model: Stanley 25' Power Lock 33-425 Serial Number: LTM 696

Lixer Tools certifies this tape measure has been inspected in accordance with our inspection procedure. All measurements and calibrations are traceable to the National Institute of Standards and Technology (NIST).

Calibration Standard 6 inch: Mitutoyo: Certificate # TO9D01994	Calibration Standard for tape measure blades: Starrett SLC #13419598
Calibration Standard for tape measure end hooks: Lixer Tools: Certificate # LM 033	LTM 327 Simco Certificate #6719332 Traceable Humidity/Temperature Meter Cert. No: 4800-6636881/ 4800-6592208

Calibration Conditions:  
Temperature: 71 ° F Humidity: 31 %

Detectable End Hook Errors measured at 6" push -.002"  
Detectable End Hook Errors measured at 6" pull -.001"  
Uncertainty .001

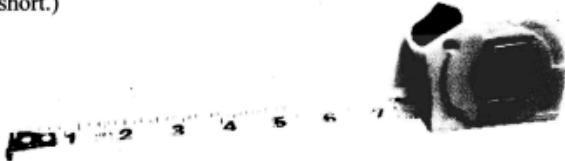
Tape Measure Blade pull measurements compared against 40' NIST Traceable scale.


Blade errors identified at:


2' +.001" 5' +.003" 8' +.004" 11' +.001" 15' +.002" 20' +.002" 24' +.015"

Uncertainty .005"

(Note: Measurements are to the approximate center of the mark on the tape measure blade. A(+) measurement indicates the tape measure reads too long, a (-) measurement indicates the tape measure reads too short.)



Inspected by:  Date: September 10, 2015  
D. Christiansen



**Appendix K - Calibration Data [1]**

3ft	6ft	9ft	12ft	15ft	18ft	21ft	24ft
36.22	72.05	107.09	145.67	178.35	214.57	252.76	288.98
36.22	70.87	107.09	144.88	178.74	214.57	251.97	288.58
36.22	71.65	107.09	144.88	179.53	215.35	251.97	288.19
36.22	70.87	107.09	144.88	178.74	214.57	252.36	287.4
36.22	71.65	107.09	145.67	178.74	216.14	252.36	286.61
36.22	71.26	107.09	145.28	178.74	215.75	252.76	288.58
36.22	71.26	107.09	145.28	177.95	215.35	252.36	285.83
36.22	71.65	106.69	145.28	178.74	215.35	251.97	287.8
36.22	70.87	107.09	145.28	178.74	215.35	252.36	288.19
36.22	71.26	107.09	145.28	178.74	215.75	252.76	288.58
36.22	71.65	107.09	145.28	177.56	214.96	252.36	285.83
36.22	71.26	107.09	144.88	177.56	215.75	251.97	287.01
36.22	71.65	107.09	144.88	177.17	215.35	252.76	288.19
36.22	71.26	107.09	144.88	177.56	215.75	252.36	287.01
36.22	71.26	107.09	145.28	177.56	214.96	252.36	287.8
36.22	71.65	107.48	144.88	177.95	214.96	252.76	287.4
36.22	71.65	107.09	145.28	177.95	215.35	252.36	287.4
36.22	71.65	107.09	144.88	177.17	215.35	251.97	287.4
36.22	71.65	107.09	144.88	177.56	214.57	252.36	287.8
36.22	71.65	107.09	144.88	178.74	215.35	252.36	289.76
36.22	71.65	107.09	144.88	178.35	215.35	251.97	288.19
36.22	71.65	107.09	144.88	178.74	214.96	252.76	288.98
36.22	71.26	107.09	144.88	177.95	215.35	252.36	287.4
36.22	71.26	107.09	144.88	178.35	214.57	252.76	287.01
36.22	71.65	107.09	144.88	179.13	215.35	252.76	288.98
36.22	71.65	107.09	144.88	178.74	215.35	252.36	287.8
36.22	71.26	107.09	144.88	178.74	215.35	251.97	289.37
36.22	71.65	107.09	144.88	178.74	215.35	252.36	289.37
36.22	71.26	107.09	144.88	178.74	215.75	251.97	288.58
36.22	71.26	107.09	144.88	179.13	215.75	252.36	287.8
36.22	70.87	107.09	144.88	179.53	215.35	251.97	290.94

**Appendix L - Calibration Data Cont. [2]**

36.22	71.26	107.09	144.88	179.53	215.35	251.57	289.37
36.22	71.26	107.09	144.88	179.13	215.75	251.97	289.76
36.22	71.26	107.09	145.28	179.13	216.14	251.97	287.4
36.22	71.26	107.09	144.88	179.13	215.35	252.36	288.58
36.22	71.65	107.09	144.88	178.35	215.35	252.36	288.19
36.22	71.65	107.09	144.88	178.35	215.35	252.76	288.19
36.61	71.65	107.09	144.88	177.17	215.35	251.97	289.76
36.22	71.65	107.09	144.88	179.53	215.75	251.97	289.37
36.22	71.65	107.09	144.88	178.74	216.14	252.36	287.8
36.22	71.65	107.09	144.88	178.74	216.14	252.76	290.16
36.22	71.26	106.69	144.88	179.13	215.75	251.97	288.58
36.22	71.26	107.09	144.88	179.13	215.75	251.97	287.4
36.22	71.26	107.09	144.88	179.13	215.75	251.97	287.8
36.22	71.26	107.09	144.88	179.13	216.14	252.76	288.58
36.22	71.65	107.09	144.88	179.13	215.75	252.36	289.76
36.22	71.26	107.09	144.49	178.74	215.75	252.36	289.37
36.22	71.26	107.09	144.88	178.35	215.35	252.36	289.37
36.22	71.26	107.09	144.88	175.59	215.75	252.76	288.58
36.22	71.26	107.09	144.88	174.8	215.35	251.57	289.37

**Appendix M - Measured Averages**

Anticipated Distance [ft]	3ft	6ft	9ft	12ft	15ft	18ft	21ft	24ft
Measured Average [ft]	3.01898	5.95135	8.92348	12.0813	14.8681	17.9527	21.025	24.0269
Measured Average [in]	36.2278	71.4162	107.0818	144.9758	178.4172	215.4318	252.2996	288.323

**Appendix N - Deviation and Difference of Calibration Data**

	3 ft	6 ft	9 ft	12 ft	15 ft	18 ft	21 ft	24 ft
Standard Deviation [in]	0.055154	0.261814	0.097806	0.220455	0.926032	0.420795	0.332566	1.073813
Measured-Expected [in]	0.018983	-0.04865	-0.07652	0.081317	0.1319	0.04735	0.024967	0.026917



**Appendix O - Individual Calibration Error [1]**

3ft	6ft	9ft	12ft	15ft	18ft	21ft	24ft
Calibration Error (decimal)							
0.006111	0.000694	0.008426	0.011597	0.009167	0.00662	0.003016	0.003403
0.006111	0.015694	0.008426	0.006111	0.007	0.00662	0.000119	0.002014
0.006111	0.004861	0.008426	0.006111	0.002611	0.003009	0.000119	0.00066
0.006111	0.015694	0.008426	0.006111	0.007	0.00662	0.001429	0.002083
0.006111	0.004861	0.008426	0.011597	0.007	0.000648	0.001429	0.004826
0.006111	0.010278	0.008426	0.008889	0.007	0.001157	0.003016	0.002014
0.006111	0.010278	0.008426	0.008889	0.011389	0.003009	0.001429	0.007535
0.006111	0.004861	0.01213	0.008889	0.007	0.003009	0.000119	0.000694
0.006111	0.015694	0.008426	0.008889	0.007	0.003009	0.001429	0.00066
0.006111	0.010278	0.008426	0.008889	0.007	0.001157	0.003016	0.002014
0.006111	0.004861	0.008426	0.008889	0.013556	0.004815	0.001429	0.007535
0.006111	0.010278	0.008426	0.006111	0.013556	0.001157	0.000119	0.003438
0.006111	0.004861	0.008426	0.006111	0.015722	0.003009	0.003016	0.00066
0.006111	0.010278	0.008426	0.006111	0.013556	0.001157	0.001429	0.003438
0.006111	0.010278	0.008426	0.008889	0.013556	0.004815	0.001429	0.000694
0.006111	0.004861	0.004815	0.006111	0.011389	0.004815	0.003016	0.002083
0.006111	0.004861	0.008426	0.008889	0.011389	0.003009	0.001429	0.002083
0.006111	0.004861	0.008426	0.006111	0.015722	0.003009	0.000119	0.002083
0.006111	0.004861	0.008426	0.006111	0.013556	0.00662	0.001429	0.000694
0.006111	0.004861	0.008426	0.006111	0.007	0.003009	0.001429	0.006111
0.006111	0.004861	0.008426	0.006111	0.009167	0.003009	0.000119	0.00066
0.006111	0.004861	0.008426	0.006111	0.007	0.004815	0.003016	0.003403
0.006111	0.010278	0.008426	0.006111	0.011389	0.003009	0.001429	0.002083
0.006111	0.010278	0.008426	0.006111	0.009167	0.00662	0.003016	0.003438
0.006111	0.004861	0.008426	0.006111	0.004833	0.003009	0.003016	0.003403
0.006111	0.004861	0.008426	0.006111	0.007	0.003009	0.001429	0.000694
0.006111	0.010278	0.008426	0.006111	0.007	0.003009	0.000119	0.004757
0.006111	0.004861	0.008426	0.006111	0.007	0.003009	0.001429	0.004757
0.006111	0.010278	0.008426	0.006111	0.007	0.001157	0.000119	0.002014
0.006111	0.010278	0.008426	0.006111	0.004833	0.001157	0.001429	0.000694
0.006111	0.015694	0.008426	0.006111	0.002611	0.003009	0.000119	0.010208
0.006111	0.010278	0.008426	0.006111	0.002611	0.003009	0.001706	0.004757



**Appendix P - Individual Calibration Error cont. [2]**

0.006111	0.010278	0.008426	0.006111	0.004833	0.001157	0.000119	0.006111
0.006111	0.010278	0.008426	0.008889	0.004833	0.000648	0.000119	0.002083
0.006111	0.010278	0.008426	0.006111	0.004833	0.003009	0.001429	0.002014
0.006111	0.004861	0.008426	0.006111	0.009167	0.003009	0.001429	0.00066
0.006111	0.004861	0.008426	0.006111	0.009167	0.003009	0.003016	0.00066
0.016944	0.004861	0.008426	0.006111	0.015722	0.003009	0.000119	0.006111
0.006111	0.004861	0.008426	0.006111	0.002611	0.001157	0.000119	0.004757
0.006111	0.004861	0.008426	0.006111	0.007	0.000648	0.001429	0.000694
0.006111	0.004861	0.008426	0.006111	0.007	0.000648	0.003016	0.0075
0.006111	0.010278	0.01213	0.006111	0.004833	0.001157	0.000119	0.002014
0.006111	0.010278	0.008426	0.006111	0.004833	0.001157	0.000119	0.002083
0.006111	0.010278	0.008426	0.006111	0.004833	0.001157	0.000119	0.000694
0.006111	0.010278	0.008426	0.006111	0.004833	0.000648	0.003016	0.002014
0.006111	0.004861	0.008426	0.006111	0.004833	0.001157	0.001429	0.006111
0.006111	0.010278	0.008426	0.003403	0.007	0.001157	0.001429	0.004757
0.006111	0.010278	0.008426	0.006111	0.009167	0.003009	0.001429	0.004757
0.006111	0.010278	0.008426	0.006111	0.0245	0.001157	0.003016	0.002014
0.006111	0.010278	0.008426	0.006111	0.028889	0.003009	0.001706	0.004757

**Appendix Q - Error Solutions**

Cal Avg Err %	Rand Avg Err %
0.57256093	0.49386
Total Error %	Absolute Err [in]
0.75612331	0.197