ECE LABORATORY

DREXEL UNIVERSITY

To: **Dr. Peters**

From: **Ehi Simon**

Date: **03/17/23**

Re: **ECE 303 Final Project**

PURPOSE:

For our final project, the goal was to create an electric vehicle test bed. The project comprised of a collision avoidance system, dashboard development, security and remote control and sensor implementation. The following hardware was used: lcd screen, water level sensor, ultrasonic sensor, 2 LEDs as headlights controlled by an IR remote, a DC motor with fans, and an RFID security system which included a buzzer and a temperature sensor.

Discussion:

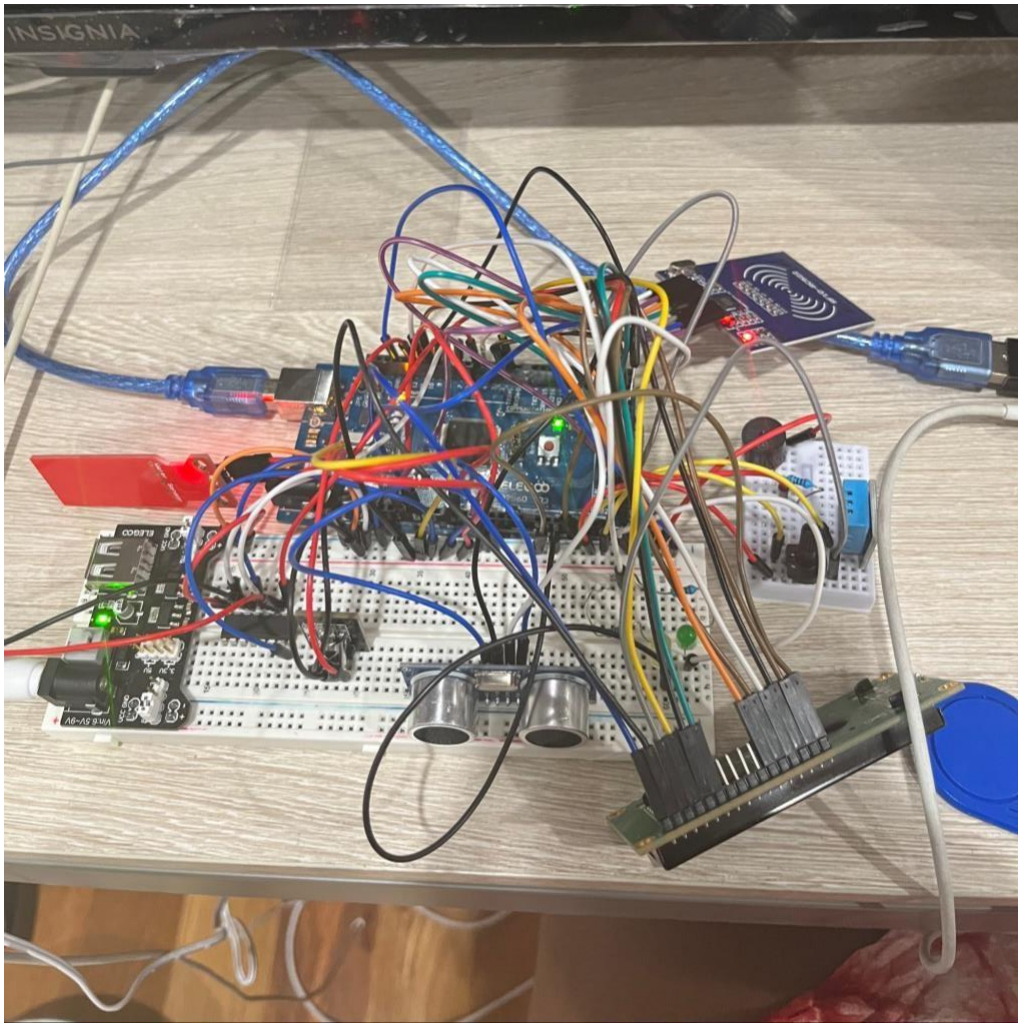

Fig. 1. Circuit Connection for Final Project 4

The circuit for the project was built like the one above. The project made use of many libraries and header files that can be seen below:

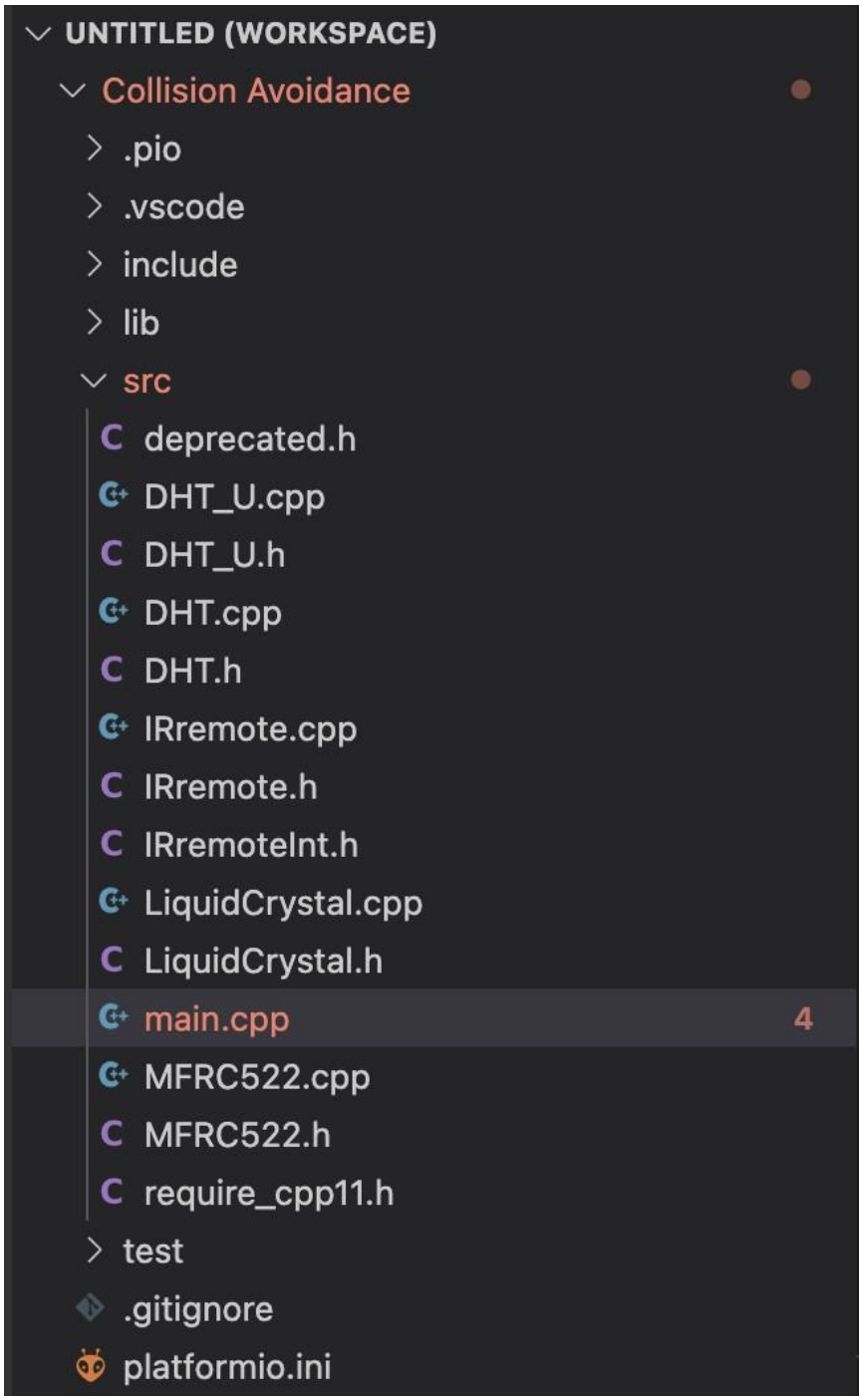

Fig. 2. Libraries and Source Files for Collision Avoidance System

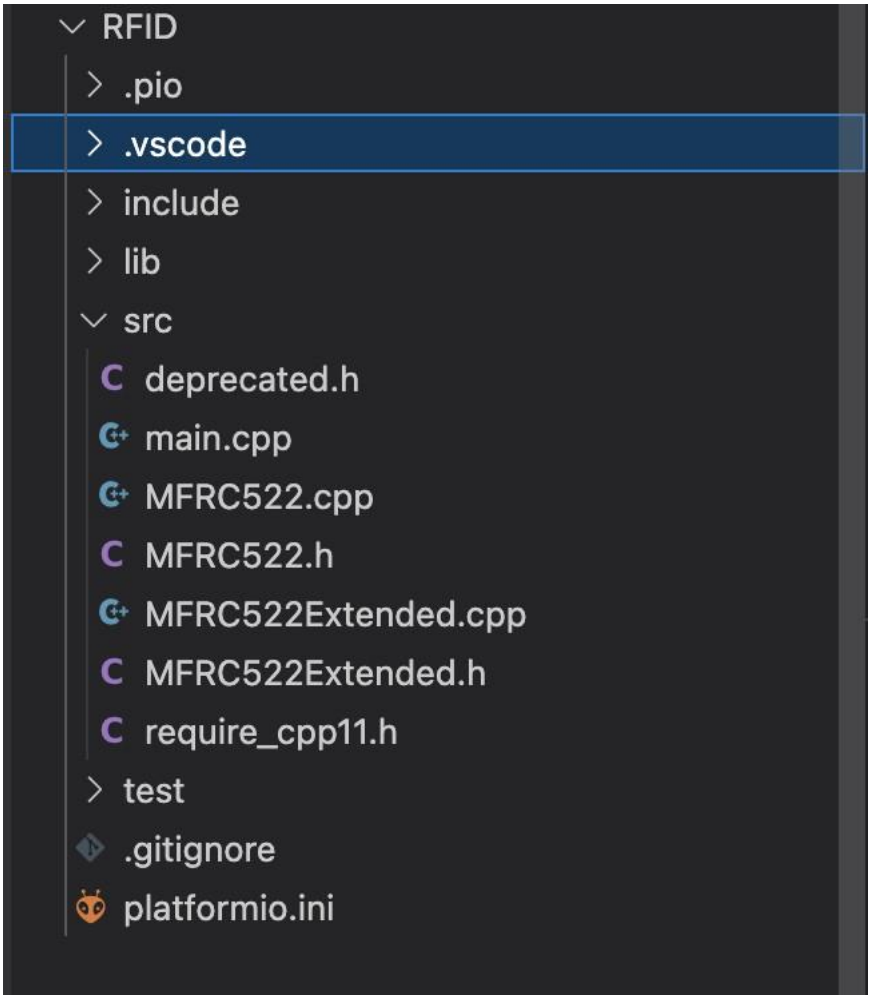

Fig. 3. Libraries and Source Files for RFID System

Main.cpp

The first thing I did in my main source file was to include header files and define pin numbers. This helped me import all the necessary helper functions and classes, as well as set variables that I used often in the rest of the code. This can be seen in figure 4 and 5 below.

```
#include <Arduino.h>
#include <LiquidCrystal.h>
#include "DHT.h"
#include "IRremote.h"
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN 6
#define SS_PIN 53
#define DHTPIN 10
#define DHTTYPE DHT11
#define ONE 16724175   // Define the key 1
#define TWO 16718055   // Define the key 2
#define THREE 16743045 // Define the key 3
#define FOUR 16716015  // Define the key 4
#define FIVE 16726215  // Define the key 5
#define SIX 16734885   // Define the key 6
#define SEVEN 16728765 // Define the key 7
#define EIGHT 16730805 // Define the key 8
#define NINE 16732845  // Define the key 9
#define ZERO 16738455  // Define the key 0
```

Fig. 4. Including Header Files and Defining Pin Numbers and Key

```
// instances
IRrecv irrecv();
MFRC522 mfrc522(SS_PIN, RST_PIN);
DHT dht(DHTPIN, DHTTYPE);

// lcd
LiquidCrystal lcd(1, 30, 40, 38, 36, 9);

// headlights
int headlight1 = 11;
int headlight_intensity;

// variable definitions
const int trigPin = 2;
const int echoPin = 4;
int far = 100;
int safe = 10;
const int m1 = 7;
const int m2 = 5;
long duration;
int distance;
const int water = A0;
const int buzzer = 8;
int checkaccess = 0;
```

Fig. 5. Defining my Variables

The next thing I did was to set my pin modes in my setup function and begin serial communication. The trigPin and echoPin work with the ultrasonic sensor, while variables m1 and m2 work with the DC motor. The headlight1 variable is the pin that the LED is controlled by and will be turned on an off using an IR remote. This is what the irrecv.enableIRIn is for. In this function, the LCD display is also set up. This can be seen in figure 6 below.

```
void setup()
{

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(headlight1, OUTPUT);
  pinMode(buzzer, OUTPUT);

  irrecv.enableIRIn();

  SPI.begin();
  mfrc522.PCD_Init();
  dht.begin();

  pinMode(24, OUTPUT);
  analogWrite(24, 120);
  lcd.begin(16, 2);
  Serial.begin(9600);
}
```

Fig. 6. Setup Loop

After this, to prevent unauthorized access, I devised a for loop that handles access granting and denial. If the user possesses the appropriate fob for entry, a brief beep is produced by the buzzer, and the code starts to execute. Conversely, if the user doesn't have the correct fob, a prolonged unbroken beep sounds until the system is shut down. I had to find out the access code of the card I wanted to use to signify the authorized user. That code was then compared to the code of whichever card was read by the RFID. The code written can be found in figure 7 below and the output in the terminal after reading a card can be seen in figure 8 below.

```
void loop()
{
  if (checkaccess == 0)
  {
    if (!mfrc522.PICC_IsNewCardPresent())
    {
      return;
    }
    if (!mfrc522.PICC_ReadCardSerial())
    {
      return;
    }
    String access = "";
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
      access.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
      access.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
    access.toUpperCase();

    if (access.substring(1) == "99 9A 27 BC")
    {

      tone(buzzer, 1000);
      delay(1000);
      noTone(buzzer);
      delay(1000);
      checkaccess = 1;
      Serial.println("Access granted");
    }
    else
    {
      tone(buzzer, 1000);
```

Fig. 7. For Loop to check Access

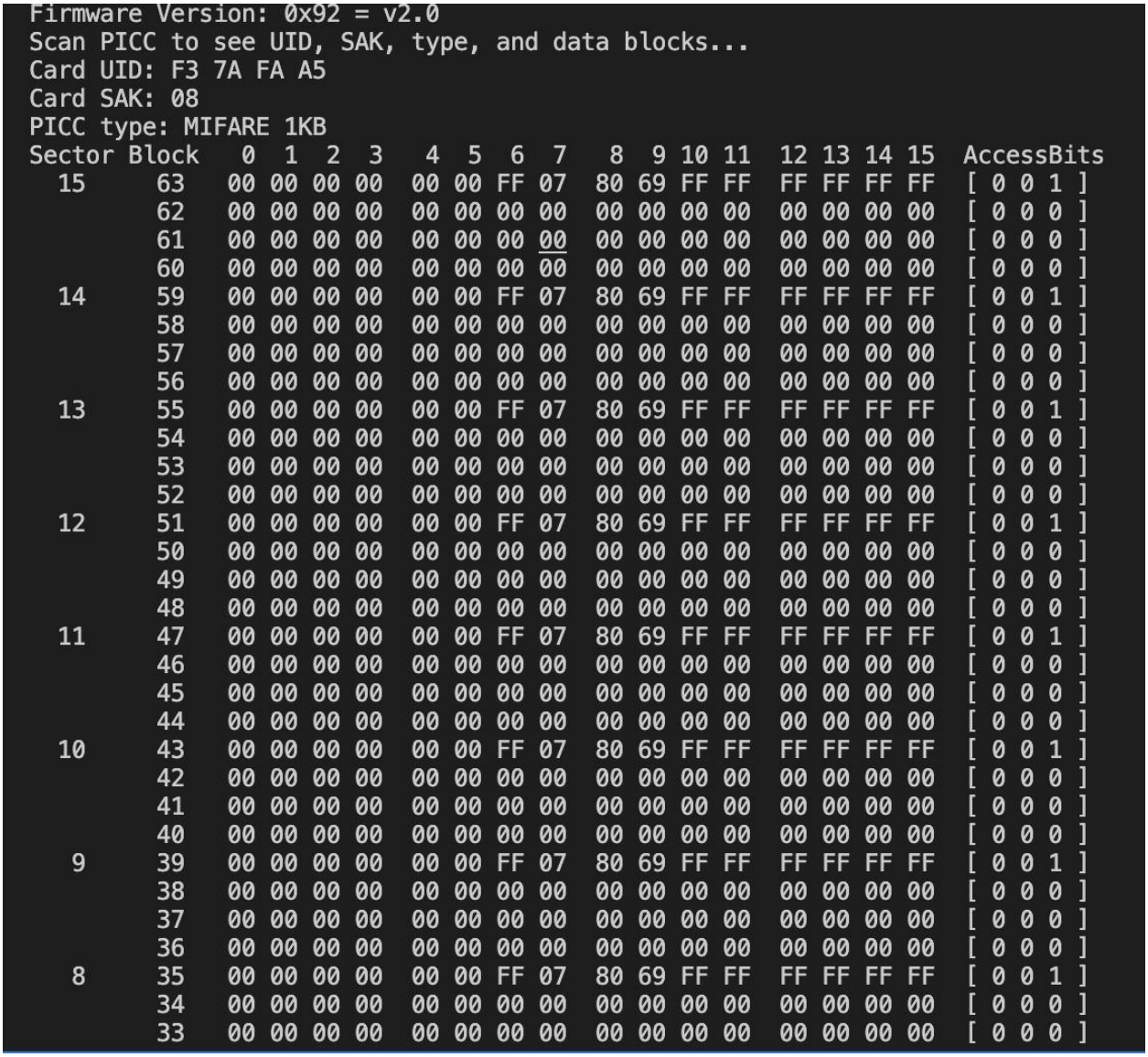

Fig. 8. If Statement for IR Remote to Control LEDs

Within the for loop, I wrote an if statement to use the IR remote to control the LED lights which acted as headlights for the vehicle. The code snippet can be found in the figure below.

```
//LED - IR remote
if (irrecv.decode(&results))
{
  // Serial.println(results.value);
  if (results.value == ONE)
  {                               // Check if the key 1 is pressed
    digitalWrite(headlight1, HIGH); // Turn on the headlight 1
    headlight_intensity = 100;

  }
  else if (results.value == TWO)
  {                                // Check if the key 2 is pressed
    analogWrite(headlight1, 150); // Turn on the headlight 1 (dim)
    headlight_intensity = 50;

  }
  else if (results.value == THREE)
  {                               // Check if the key 3 is pressed
    digitalWrite(headlight1, LOW); // Turn off the headlight 1
    headlight_intensity = 0;

  }
  irrecv.resume(); // Receive the next value
}
```

Fig. 9. If Statement for IR Remote to Control LEDs

Using the project from week 7, I was able to create a collision avoidance system with the DC motor. As an object grew closer to the ultrasonic sensor, the motor slowed down, and at a certain point stopped. I computed the motor speed as a percentage, measured and computed the temperature and water level with the sensors, and transformed these variables into strings. Then, I displayed all of this data on the LCD with intermittent pauses to allow the system to read and print the values accurately.

There was also the GUI section of the project. The GUI was designed as seen in figure 10 below.

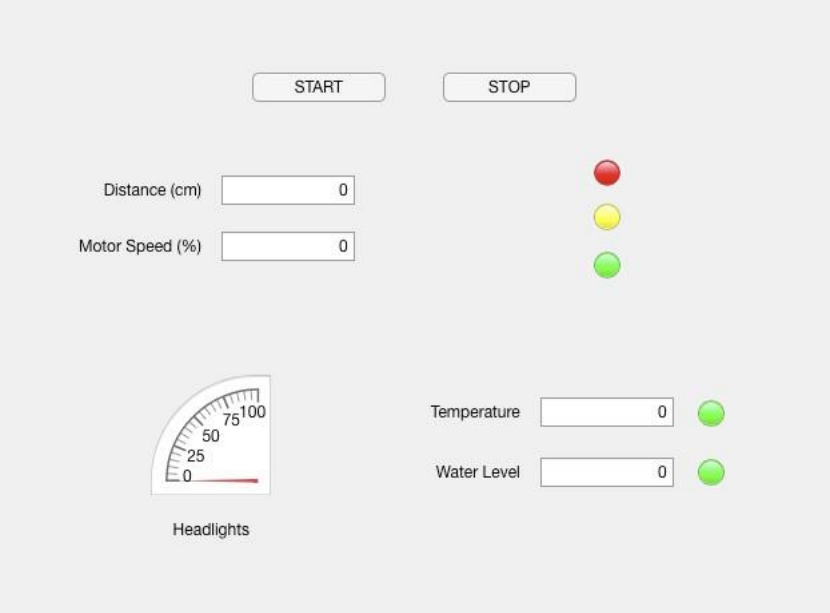

Fig. 10. Design of MATLAB GUI

First, the user must click the start button, then the Arduino starts to run and the correct key card must be used before access can be granted and the system can start running. Then values from the Arduino are read into MATLAB and the strings are converted into values for the MATLAB code. Depending on the values read from the Arduino, the colors in the GUI can change to represent if the water level is too high or if temperature is too high. This GUI ties the project together. Snippets of the MATLAB code to control the GUI can be found below.

```
%temperature alarm
if app.TemperatureEditField.Value > 19
    app.temperature.Color = 'red';
else
    app.temperature.Color = 'green';
end

%light changing sequence
if app.DistancecmEditField.Value < 100 && app.DistancecmEditField.Value >= 60
    app.green.Color = 'green';
    app.yellow.Color = app.offcolor;
    app.red.Color = app.offcolor;
```

Fig. 11. Code to Control MATLAB GUI

Conclusion

This project made use of everything we learnt in the class so far. It was a refresher on all our previous projects as well. The current budget for the project can be found in the table below:

Device	Arduino - Current (mA)	Pin Used	Power Supply - Current (mA)
DC Motor	NA	Power Supply Module	250
Yellow LED	20	11	
Green LED	20	11	
H-Bridge	NA	5V	
LCD	1	Power Supply Module	150
Temperature/Humidity Sensor	2	5V	
IR Module	20	Power Supply Module	
Active Buzzer	20	8	
Water Detection Sensor	15	5V	
Ultrasonic Sensor	15	5V	
RFID Module	25	3.3V	
Power Supply Module	25		
Potentiometer	NA		

Table 1. Current Budget Showing Too Much Current was Not Used