Electric Vehicle Test-Bed

From: Christine Onita

Date: Wednesday 10th February 2021

Re: Electric Vehicle Testbed

Introduction

The aim of this project is to design an electric vehicle testbed using the knowledge we gained during this course. The following components were implemented in the system:

1. Collision avoidance
2. Interlocks
3. Security/Remote Control
4. Displays
5. Graphical User Interface (GUI)

The project was completed using Arduino and MATLAB (for the GUI). Details of the listed implemented parts above will be expatiated on below.

Method and Analysis/Discussion

Link to sketch (at the end of this document): Christine Onita Final Project Appendix

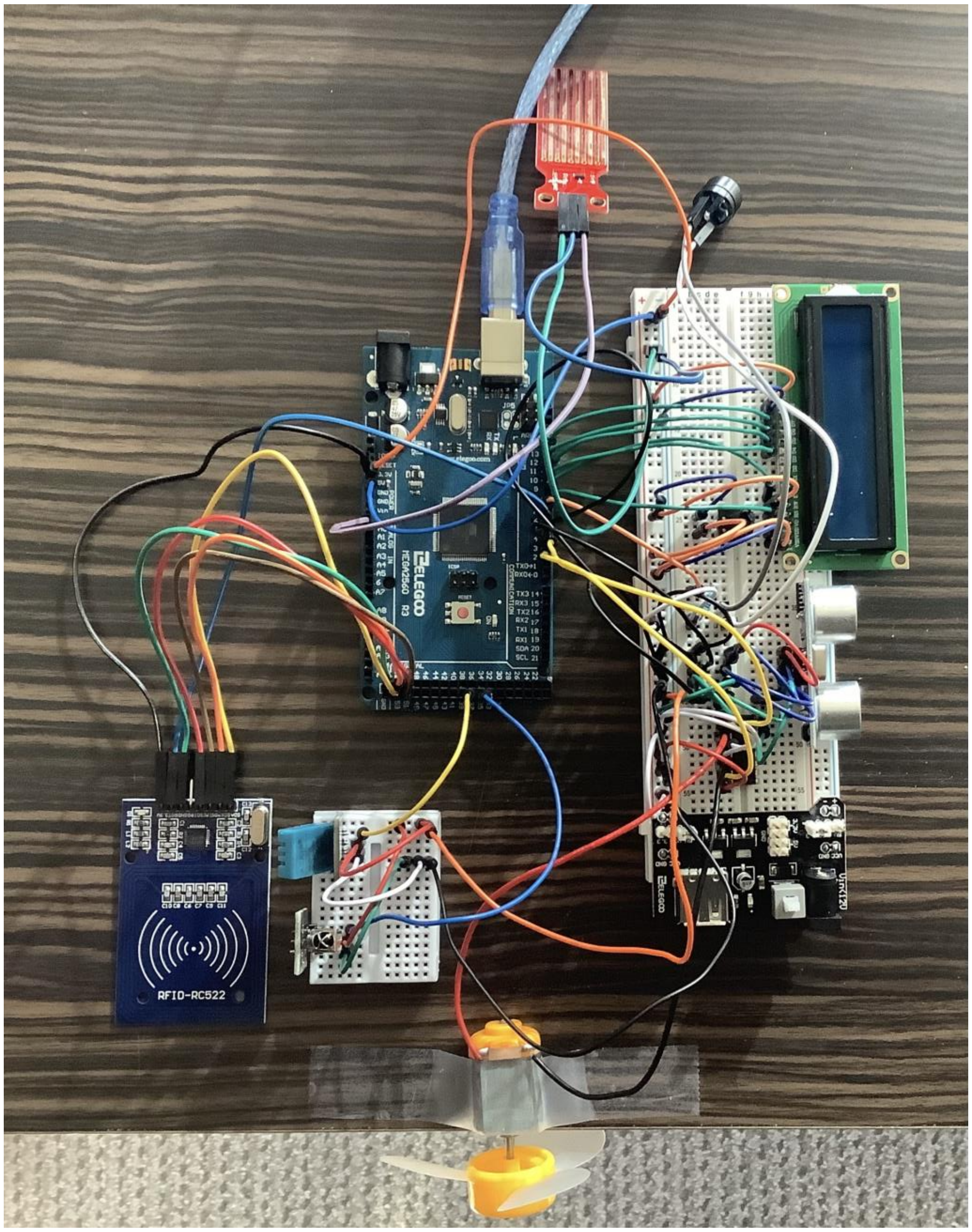
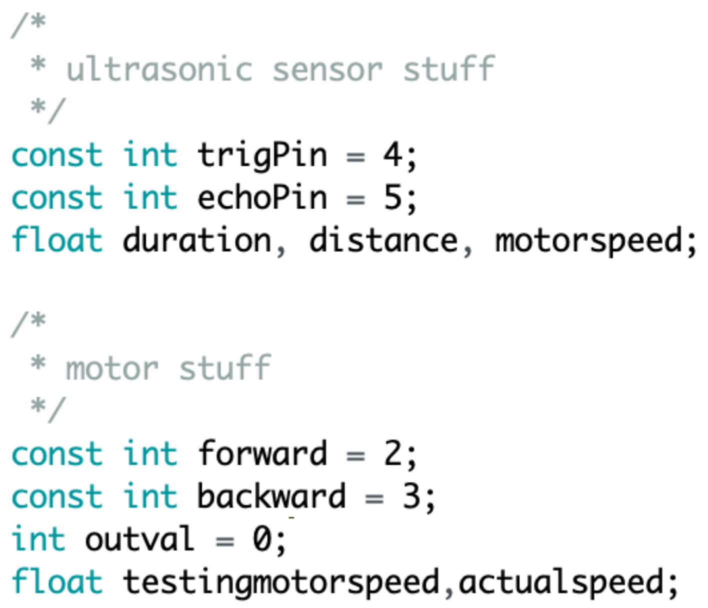


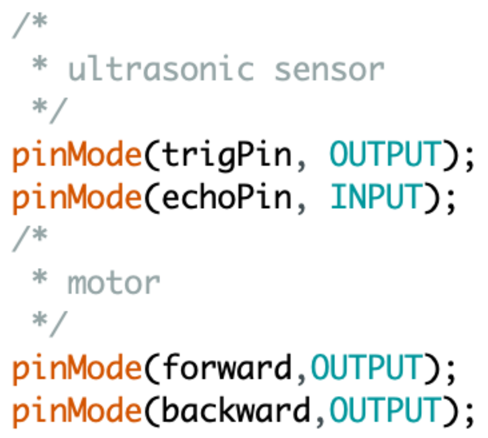
Fig 1: Arduino build (without 9V adapter connected)

Collision Avoidance



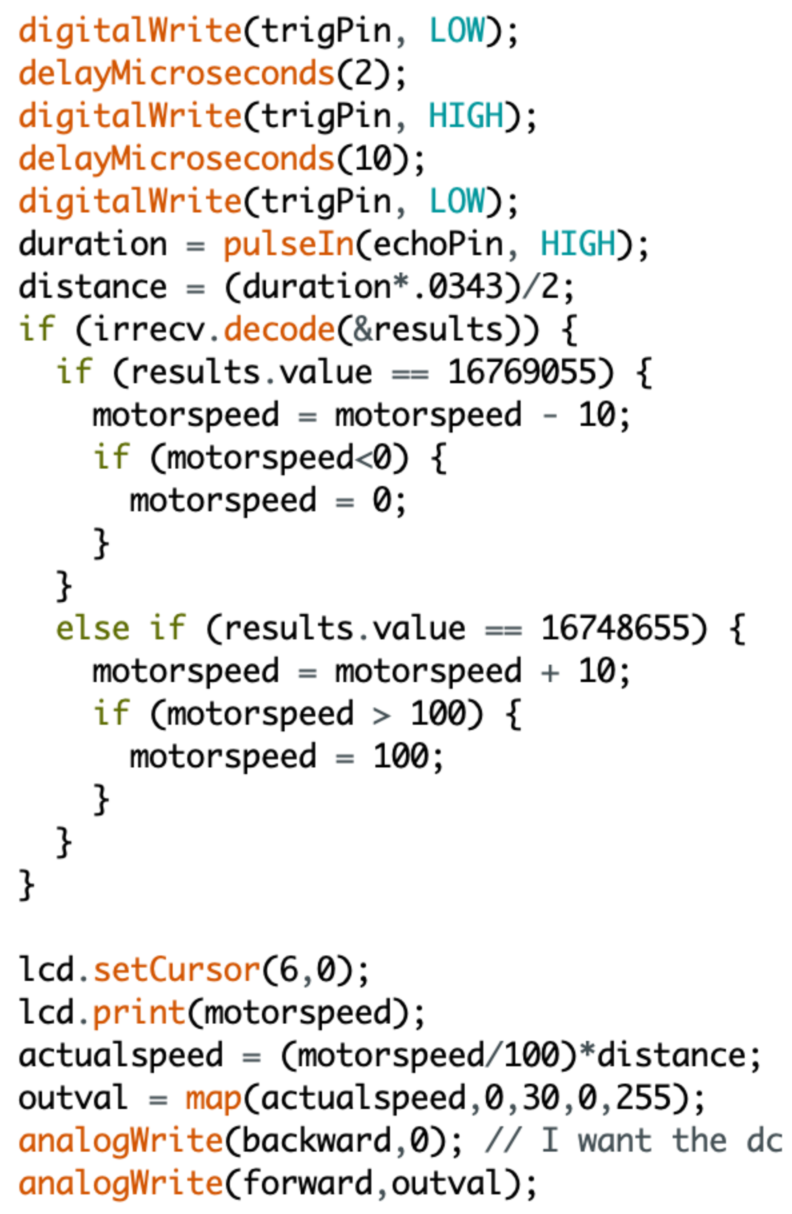
```
/*
 * ultrasonic sensor stuff
 */
const int trigPin = 4;
const int echoPin = 5;
float duration, distance, motorspeed;

/*
 * motor stuff
 */
const int forward = 2;
const int backward = 3;
int outval = 0;
float testingmotorspeed,actualspeed;
```

Defining values associated with the ultrasonic sensor and motor



```
/*
 * ultrasonic sensor
 */
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
/*
 * motor
 */
pinMode(forward,OUTPUT);
pinMode(backward,OUTPUT);
```

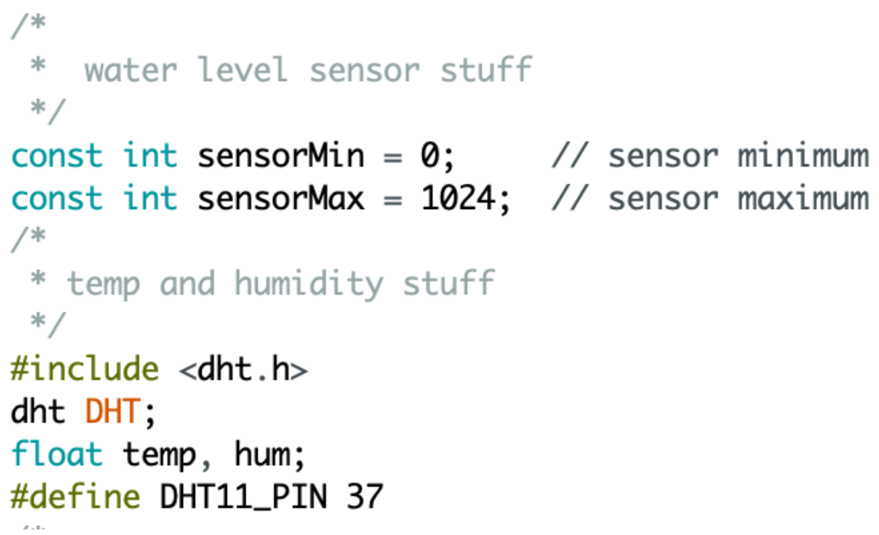
Assigning appropriate modes (in void setup())



```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration*.0343)/2;
if (irrecv.decode(&results)) {
  if (results.value == 16769055) {
    motorspeed = motorspeed - 10;
    if (motorspeed<0) {
      motorspeed = 0;
    }
  }
  else if (results.value == 16748655) {
    motorspeed = motorspeed + 10;
    if (motorspeed > 100) {
      motorspeed = 100;
    }
  }
}

lcd.setCursor(6,0);
lcd.print(motorspeed);
actualspeed = (motorspeed/100)*distance;
outval = map(actualspeed,0,30,0,255);
analogWrite(backward,0); // I want the dc
analogWrite(forward,outval);
```

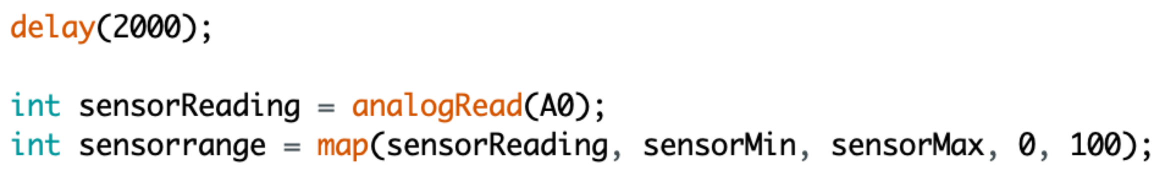
The trig pin as low first to make sure it is low before the first pulse. The duration is returned in microseconds. Speed of sound in cm/µs is 0.0343 and distance = speed *time. Here, the distance the wave travelled is halved because it travels to the object and back to the source. The resulting distance is displayed in centimeters. The motor speed is reduced by ten if the increase button is pressed on the remote (max motor speed = 100) and decreased by ten if the decrease button is pressed (min motor speed = 0).

Interlocks



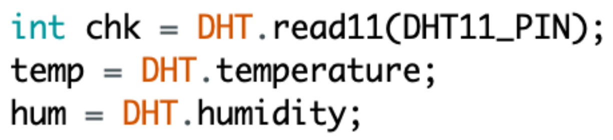
```
/*
 *  water level sensor stuff
 */
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum
/*
 * temp and humidity stuff
 */
#include <dht.h>
dht DHT;
float temp, hum;
#define DHT11_PIN 37
```

Defining values for temperature/humidity and water level sensors



```
delay(2000);

int sensorReading = analogRead(A0);
int sensorrange = map(sensorReading, sensorMin, sensorMax, 0, 100);
```

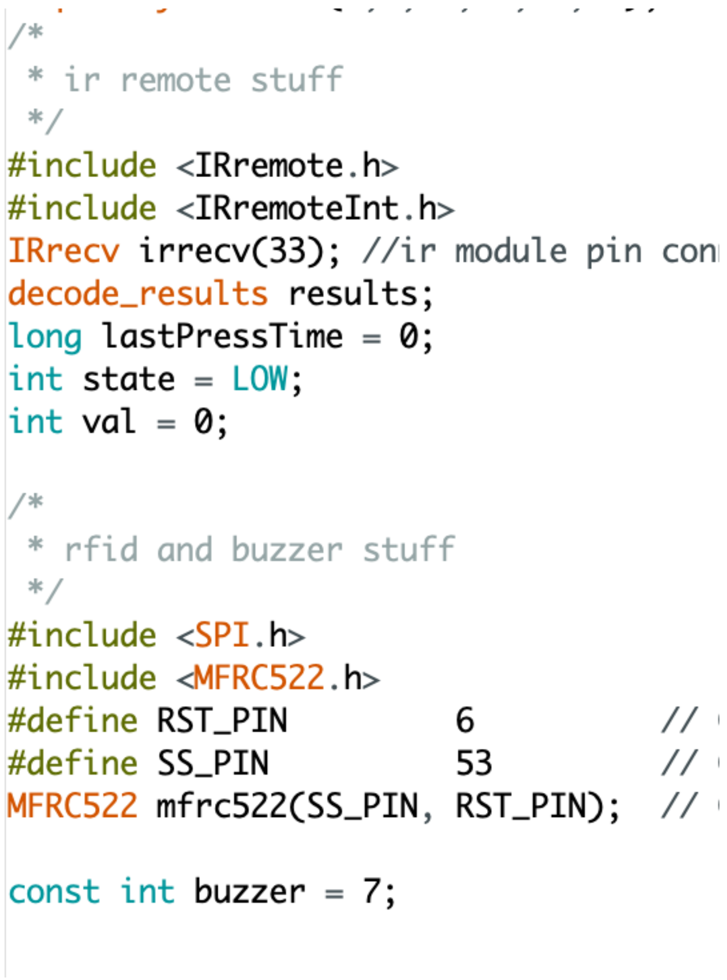
Void loop() code for water level sensor – the water level is printed on the lcd out of 100 (in %)



```
int chk = DHT.read11(DHT11_PIN);
temp = DHT.temperature;
hum = DHT.humidity;
```

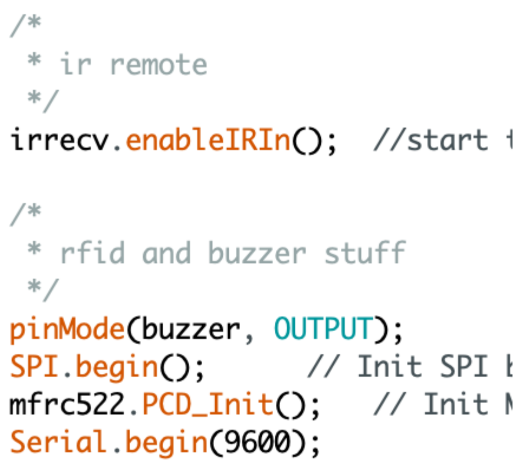
Void loop() code for temperature/humidity sensor – temperature is printed in ºC

Security/Remote Control



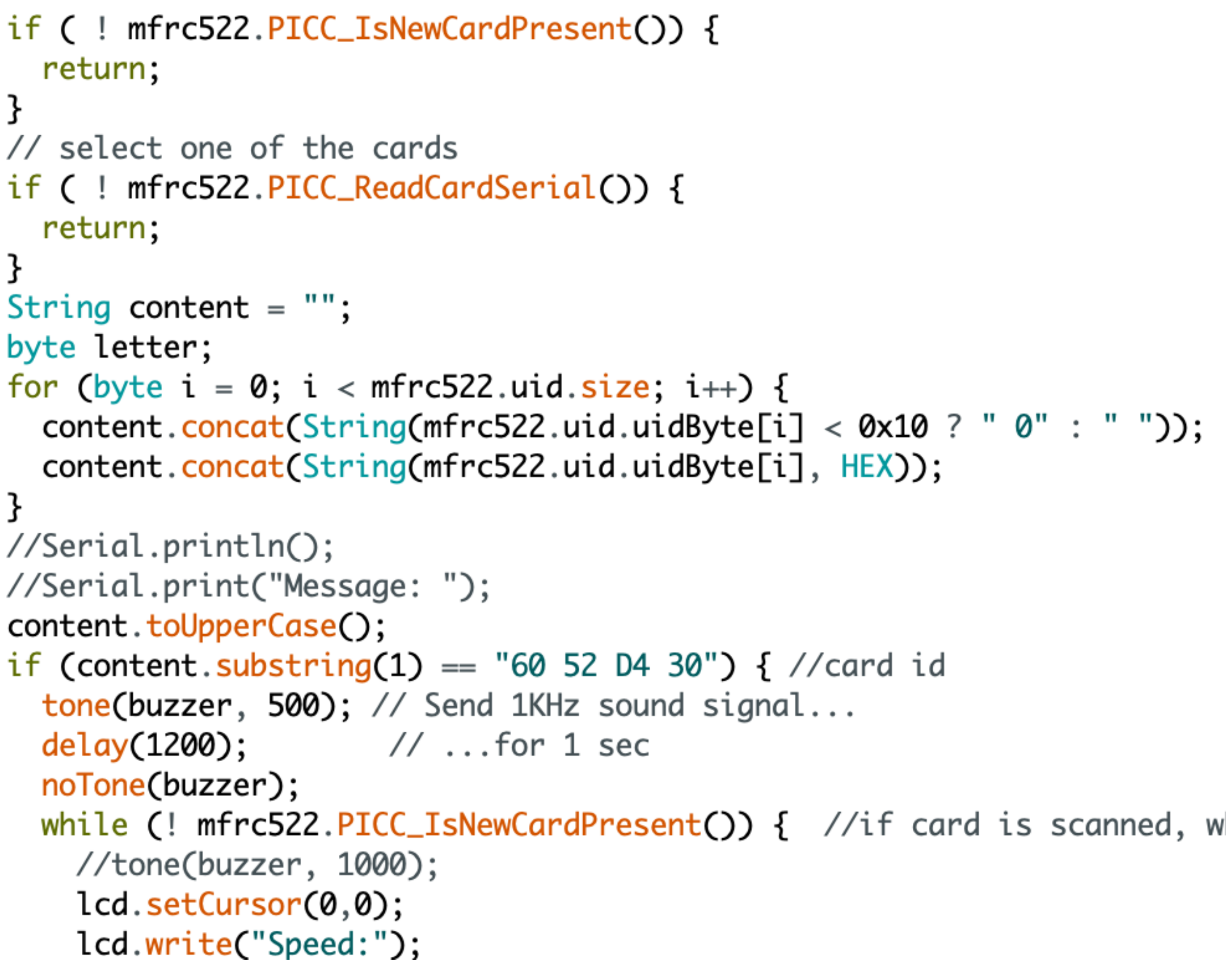
```
/*
 * ir remote stuff
 */
#include <IRremote.h>
#include <IRremoteInt.h>
IRrecv irrecv(33); //ir module pin con
decode_results results;
long lastPressTime = 0;
int state = LOW;
int val = 0;

/*
 * rfid and buzzer stuff
 */
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN          6           //
#define SS_PIN           53          //
MFRC522 mfrc522(SS_PIN, RST_PIN);  //

const int buzzer = 7;
```

Defining values and including libraries for the remote, RFID scanner and buzzer



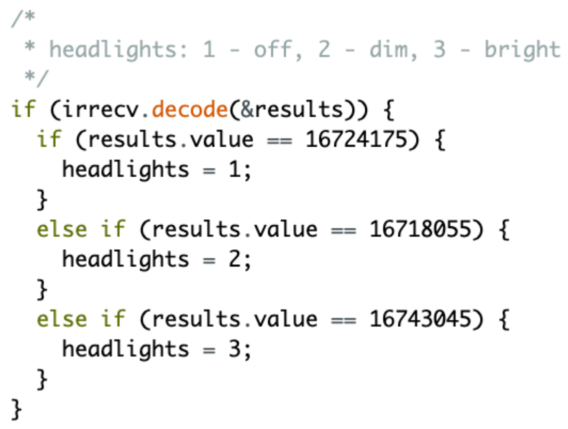
```
/*
 * ir remote
 */
irrecv.enableIRIn();  //start

/*
 * rfid and buzzer stuff
 */
pinMode(buzzer, OUTPUT);
SPI.begin();       // Init SPI
mfrc522.PCD_Init();   // Init
Serial.begin(9600);
```

In void setup() the Infrared receiver is started, and the RFID scanner is ready to detect card/fob.



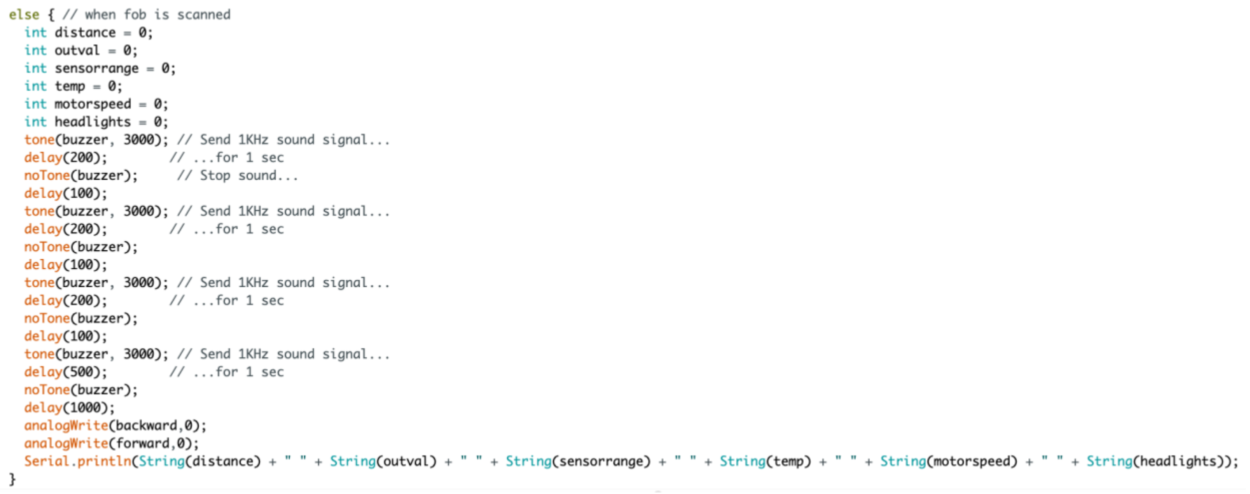
```
if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;
}
// select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;
}
String content = "";
byte letter;
for (byte i = 0; i < mfrc522.uid.size; i++) {
  content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
  content.concat(String(mfrc522.uid.uidByte[i], HEX));
}
//Serial.println();
//Serial.print("Message: ");
content.toUpperCase();
if (content.substring(1) == "60 52 D4 30") { //card id
  tone(buzzer, 500); // Send 1KHz sound signal...
  delay(1200);        // ...for 1 sec
  noTone(buzzer);
  while (! mfrc522.PICC_IsNewCardPresent()) {  //if card is scanned, w
    //tone(buzzer, 1000);
    lcd.setCursor(0,0);
    lcd.write("Speed:");
```

If the card is detected, while there is nothing for the RFID to detect, the system proceeds as planned (measuring distance, temperature, e.t.c.).



```
/*
 * headlights: 1 - off, 2 - dim, 3 - bright
 */
if (irrecv.decode(&results)) {
  if (results.value == 16724175) {
    headlights = 1;
  }
  else if (results.value == 16718055) {
    headlights = 2;
  }
  else if (results.value == 16743045) {
    headlights = 3;
  }
}
```

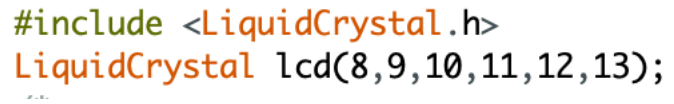
When 1, 2 or 3 is pressed on the remote, the value is printed in the array which will be used in the MATLAB code here.



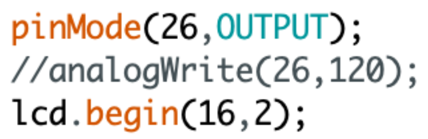
```
else { // when fob is scanned
  int distance = 0;
  int outval = 0;
  int sensorrange = 0;
  int temp = 0;
  int motorspeed = 0;
  int headlights = 0;
  tone(buzzer, 3000); // Send 1KHz sound signal...
  delay(200);        // ...for 1 sec
  noTone(buzzer);     // Stop sound...
  delay(100);
  tone(buzzer, 3000); // Send 1KHz sound signal...
  delay(200);        // ...for 1 sec
  noTone(buzzer);
  delay(100);
  tone(buzzer, 3000); // Send 1KHz sound signal...
  delay(200);        // ...for 1 sec
  noTone(buzzer);
  delay(100);
  tone(buzzer, 3000); // Send 1KHz sound signal...
  delay(500);        // ...for 1 sec
  noTone(buzzer);
  delay(1000);
  analogWrite(backward,0);
  analogWrite(forward,0);
  Serial.println(String(distance) + " " + String(outval) + " " + String(sensorrange) + " " + String(temp) + " " + String(motorspeed) + " " + String(headlights));
}
```

If the fob is detected, all values are set to zero and an array of zeros are printed (which will be used in the GUI).

Displays

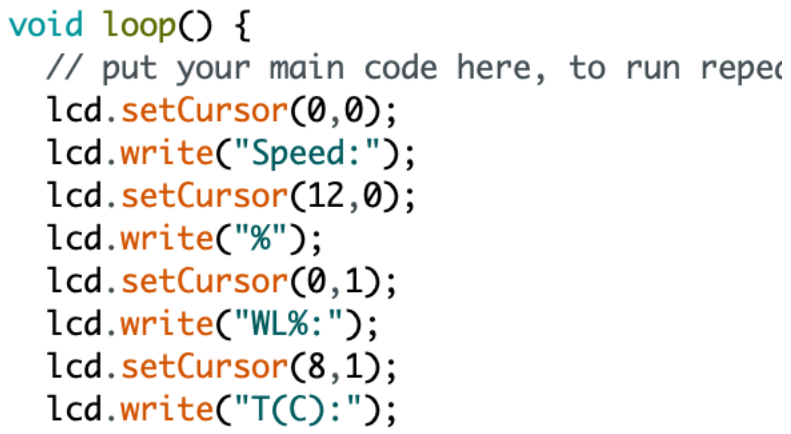


```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,10,11,12,13);
```



```
pinMode(26,OUTPUT);
//analogWrite(26,120);
lcd.begin(16,2);
```

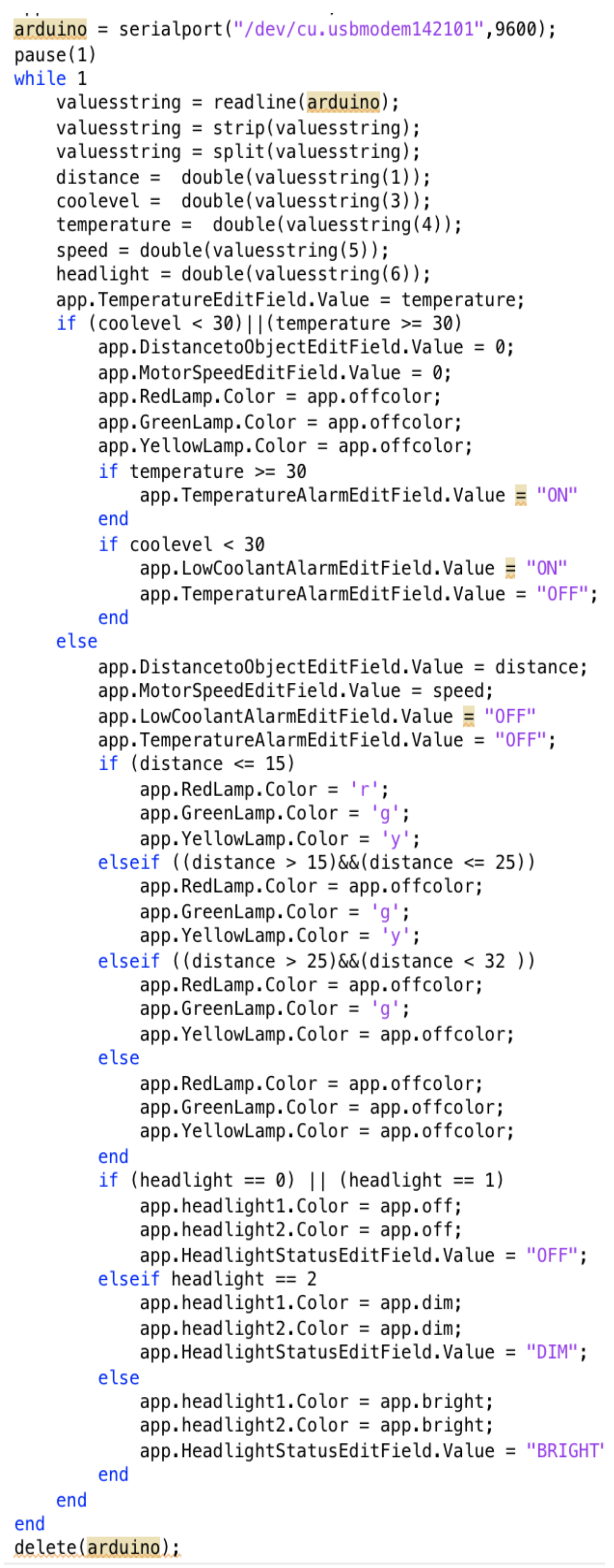
Defining corresponding pins and mode



```
void loop() {
  // put your main code here, to run repe
  lcd.setCursor(0,0);
  lcd.write("Speed:");
  lcd.setCursor(12,0);
  lcd.write("%");
  lcd.setCursor(0,1);
  lcd.write("WL%:");
  lcd.setCursor(8,1);
  lcd.write("T(C):");
```

Printing the string part at the beginning of loop code since they never change

Graphical User Interface



```
arduino = serialport("/dev/cu.usbmodem142101",9600);
pause(1)
while 1
    valuesstring = readline(arduino);
    valuesstring = strip(valuesstring);
    valuesstring = split(valuesstring);
    distance =  double(valuesstring(1));
    coolevel =  double(valuesstring(3));
    temperature =  double(valuesstring(4));
    speed = double(valuesstring(5));
    headlight = double(valuesstring(6));
    app.TemperatureEditField.Value = temperature;
    if (coolevel < 30)||(temperature >= 30)
        app.DistancetoObjectEditField.Value = 0;
        app.MotorSpeedEditField.Value = 0;
        app.RedLamp.Color = app.offcolor;
        app.GreenLamp.Color = app.offcolor;
        app.YellowLamp.Color = app.offcolor;
        if temperature >= 30
            app.TemperatureAlarmEditField.Value = "ON"
        end
        if coolevel < 30
            app.LowCoolantAlarmEditField.Value = "ON"
            app.TemperatureAlarmEditField.Value = "OFF";
        end
    else
        app.DistancetoObjectEditField.Value = distance;
        app.MotorSpeedEditField.Value = speed;
        app.LowCoolantAlarmEditField.Value = "OFF"
        app.TemperatureAlarmEditField.Value = "OFF";
        if (distance <= 15)
            app.RedLamp.Color = 'r';
            app.GreenLamp.Color = 'g';
            app.YellowLamp.Color = 'y';
        elseif ((distance > 15)&&(distance <= 25))
            app.RedLamp.Color = app.offcolor;
            app.GreenLamp.Color = 'g';
            app.YellowLamp.Color = 'y';
        elseif ((distance > 25)&&(distance < 32 ))
            app.RedLamp.Color = app.offcolor;
            app.GreenLamp.Color = 'g';
            app.YellowLamp.Color = app.offcolor;
        else
            app.RedLamp.Color = app.offcolor;
            app.GreenLamp.Color = app.offcolor;
            app.YellowLamp.Color = app.offcolor;
        end
        if (headlight == 0) || (headlight == 1)
            app.headlight1.Color = app.off;
            app.headlight2.Color = app.off;
            app.HeadlightStatusEditField.Value = "OFF";
        elseif headlight == 2
            app.headlight1.Color = app.dim;
            app.headlight2.Color = app.dim;
            app.HeadlightStatusEditField.Value = "DIM";
        else
            app.headlight1.Color = app.bright;
            app.headlight2.Color = app.bright;
            app.HeadlightStatusEditField.Value = "BRIGHT'
        end
    end
end
delete(arduino);
```

The array of (six) values printed in the serial monitor is initially stored in 'valuesstring'. 'Strip' is used to remove any white space and 'split' is used to separate the different values. The 'double' command converts the string to a double so the values can be treated as integers. As noticed, the 1st value in the array is distance, 3rd is coolant level, 4th is temperature (in ºC), 5th is motor speed (in %), and the 6th is the headlight status.

If the temperature is above 30ºC, the temperature alarm is set as 'on' otherwise, it's set as 'off'.

If the coolant level falls below 30, the coolant alarm is set to 'on', otherwise it is set as 'off'.

As distance between object and sensor decreases more LEDs (lamps in the GUI) are turned on.

If the value for headlights in the array is:

1. The headlights are off.
2. The headlights are dim.
3. The headlights are at their brightest.

Problems I encountered (and solutions)

1. After implementing the GUI, the motor stopped after each reading and continued to get the next.
    - Solution: Initially, I used the 'serial' command instead of 'serialport' to establish a connection to the Arduino board; and the 'fclose(arduino)' was in the while loop meaning that after each reading, the connection would be broken (hence the periodic rotation of the motor). Using 'serialport' and getting rid of 'fclose' made the GUI and motor behave as originally planned.
2. The motor speed wouldn't go past 30% and LCD screen displayed jumbled text.
    - Solution: In 'void setup()' I had a line that read 'irrecv.blink13(true)' to enable the blinking of the LED while input is received from the remote. It is not a vital part of the code so after multiple trials and errors, I took out that line which made the characters on the LCD appear as expected.

Conclusion

Interfacing a collision avoidance system with other components (GUI, temperature sensor, coolant level, e.t.c.) was not a straightforward as the previous project (lab 6 – collision avoidance with motor and LEDs) but tackling each element independently before combining it all together made reaching the final destination quite a breeze - except for some minor issues discussed above.

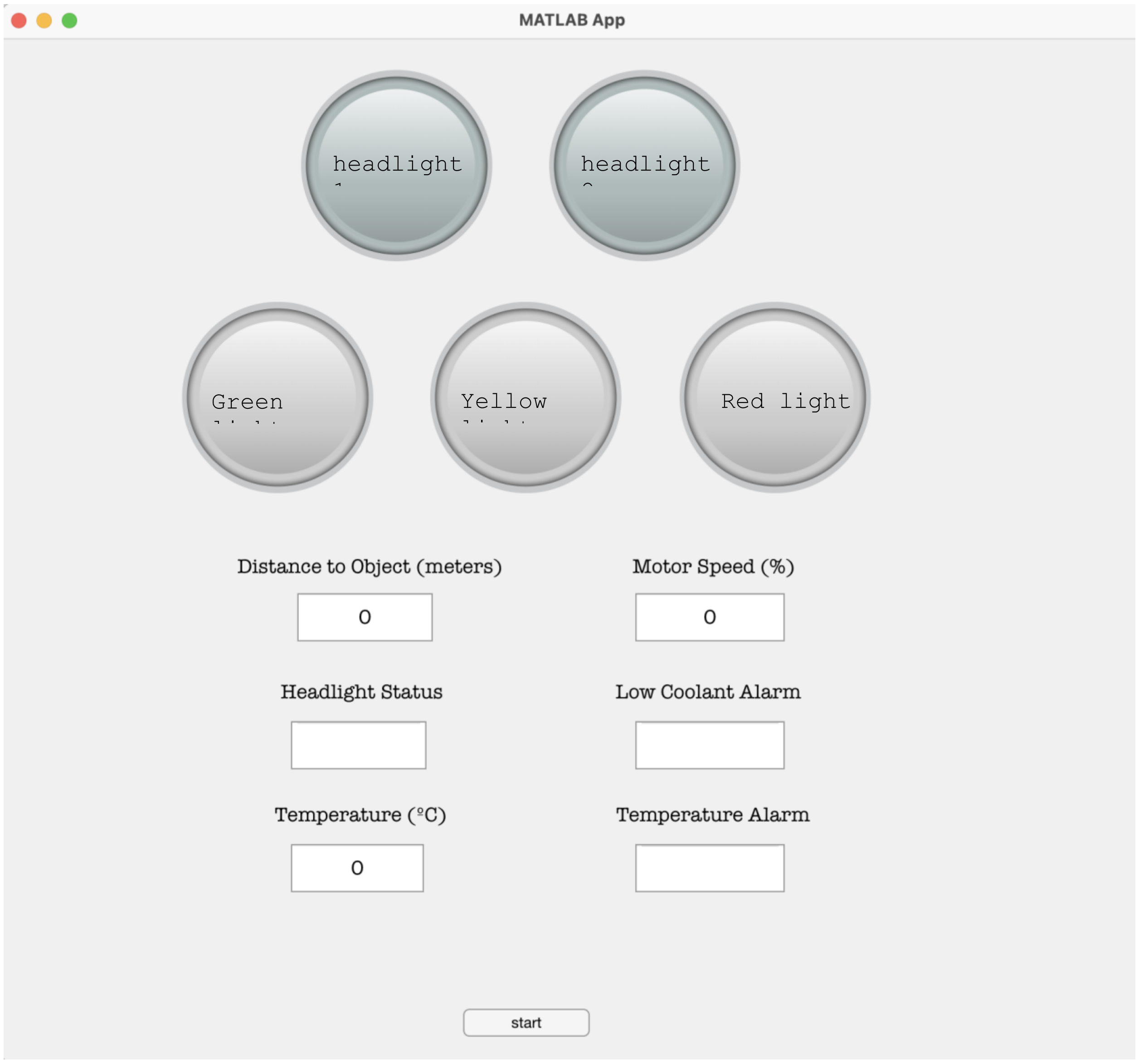
Appendix

Final Project Sketch

```
*
 * ultrasonic sensor stuff
 */
const int trigPin = 4;
const int echoPin = 5;
float duration, distance, motorspeed;

/*
 * motor stuff
 */
const int forward = 2;
const int backward = 3;
int headlights;
int outval = 0;
float testingmotorspeed,actualspeed;
/*
 *  water level sensor stuff
 */
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum
/*
 * temp and humidity stuff
 */
#include <dht.h>
dht DHT;
float temp, hum;
#define DHT11_PIN 37
/*
 * lcd stuff
 */
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,10,11,12,13);
/*
 * ir remote stuff
 */
#include <IRremote.h>
#include <IRremoteInt.h>
IRrecv irrecv(33); //ir module pin connection
decode_results results;
long lastPressTime = 0;
int state = LOW;
int val = 0;

/*
 * rfid and buzzer stuff
 */
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN         6
#define SS_PIN          53
MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance
```

```
const int buzzer = 7;


void setup() {
  /*
   * ultrasonic sensor
   */
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  /*
   * motor
   */
  pinMode(forward,OUTPUT);
  pinMode(backward,OUTPUT);
  /*
   * lcd
   */
  pinMode(26,OUTPUT);
  //analogWrite(26,120);
  lcd.begin(16,2);
  /*
   * ir remote
   */
  irrecv.enableIRIn();  //start the reciever

  /*
   * rfid and buzzer stuff
   */
  pinMode(buzzer, OUTPUT);
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init();   // Init MFRC522
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  //Serial.println();
  //Serial.print("Message: ");
  content.toUpperCase();
  if (content.substring(1) == "60 52 D4 30") { //card id
    tone(buzzer, 500); // Send 1KHz sound signal...
    delay(1200);        // ...for 1 sec
    noTone(buzzer);
    while (! mfrc522.PICC_IsNewCardPresent()) {  //if card is scanned, while
no card is there, sound at 1000Hz
      lcd.setCursor(0,0);
      lcd.write("Speed:");
      lcd.setCursor(12,0);
      lcd.write("%");
      lcd.setCursor(0,1);
      lcd.write("WL%:");
      lcd.setCursor(8,1);
      lcd.write("T(C):");
      delay(2000);
      int sensorReading = analogRead(A0);
      int sensorrange = map(sensorReading, sensorMin, sensorMax, 0, 100);
      if (sensorrange < 30) {
        analogWrite(backward,0); // I want the dc motor to move in only one
direction (forward)
      analogWrite(forward,0);
      }
      lcd.setCursor(4,1);
      lcd.print(sensorrange);


      digitalWrite(trigPin, LOW);
      delayMicroseconds(2);
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);
      duration = pulseIn(echoPin, HIGH);
      distance = (duration*.0343)/2;
      if (irrecv.decode(&results)) {
        if (results.value == 16769055) {
          motorspeed = motorspeed - 10;
          if (motorspeed<0) {
            motorspeed = 0;
          }
        }
        else if (results.value == 16748655) {
          motorspeed = motorspeed + 10;
          if (motorspeed > 100) {
            motorspeed = 100;
          }
        }
      }
      /*
       * headlights: 1 - off, 2 - dim, 3 - bright
       */
      if (irrecv.decode(&results)) {
        if (results.value == 16724175) {
          headlights = 1;
        }
        else if (results.value == 16718055) {
          headlights = 2;
        }
        else if (results.value == 16743045) {
          headlights = 3;
        }
      }
      lcd.setCursor(6,0);
      lcd.print(motorspeed);
      actualspeed = (motorspeed/100)*distance;
      outval = map(actualspeed,0,30,0,255);
      analogWrite(backward,0); // I want the dc motor to move in only one
direction (forward)
      analogWrite(forward,outval);
      if (sensorrange < 30) {
        analogWrite(backward,0); // I want the dc motor to move in only one
direction (forward)
      analogWrite(forward,0);
      }

      // temp and humidty loop stuff
      int chk = DHT.read11(DHT11_PIN);
      temp = DHT.temperature;
      hum = DHT.humidity;
      if (temp > 30) {
        analogWrite(backward,0); // I want the dc motor to move in only one
direction (forward)
      analogWrite(forward,0);
      }

      lcd.setCursor(13,1);
      lcd.print(String(int(temp)));

      irrecv.resume();
      Serial.println(String(distance) + " " + String(outval) + " " +
String(sensorrange) + " " + String(temp) + " " + String(motorspeed) + " " +
String(headlights));
    }
  }


  else { // when fob is scanned
    int distance = 0;
    int outval = 0;
    int sensorrange = 0;
    int temp = 0;
    int motorspeed = 0;
    int headlights = 0;
    tone(buzzer, 3000); // Send 1KHz sound signal...
    delay(200);        // ...for 1 sec
    noTone(buzzer);     // Stop sound...
    delay(100);
    tone(buzzer, 3000); // Send 1KHz sound signal...
    delay(200);        // ...for 1 sec
    noTone(buzzer);
    delay(100);
    tone(buzzer, 3000); // Send 1KHz sound signal...
    delay(200);        // ...for 1 sec
    noTone(buzzer);
    delay(100);
    tone(buzzer, 3000); // Send 1KHz sound signal...
    delay(500);        // ...for 1 sec
    noTone(buzzer);
    delay(1000);
    analogWrite(backward,0);
    analogWrite(forward,0);
    Serial.println(String(distance) + " " + String(outval) + " " +
String(sensorrange) + " " + String(temp) + " " + String(motorspeed) + " " +
String(headlights));
  }
}
```

Final MATLAB GUI Design



Distance to Object (meters): 0

Motor Speed (%): 0

Headlight Status:

Low Coolant Alarm:

Temperature (°C): 0

Temperature Alarm:

Final MATLAB GUI Code (properties)

```
      properties (Access = private)
```

```
arduino;
offcolor = [0.80,0.80,0.80];
off = [0.67,0.73,0.73];
 dim = [0.2,0.6,0.6];
bright = [0,1,1];
end
```

Final MATLAB GUI Code (for start button)

```
arduino = serialport("/dev/cu.usbmodem142101",9600);
pause(1)
while 1
    valuesstring = readline(arduino);
    valuesstring = strip(valuesstring);
    valuesstring = split(valuesstring);
    distance =  double(valuesstring(1));
    coolevel =  double(valuesstring(3));
    temperature =  double(valuesstring(4));
    speed = double(valuesstring(5));
    headlight = double(valuesstring(6));
    app.TemperatureEditField.Value = temperature;
    if (coolevel < 30)||(temperature >= 30)
        app.DistancetoObjectEditField.Value = 0;
        app.MotorSpeedEditField.Value = 0;
        app.RedLamp.Color = app.offcolor;
        app.GreenLamp.Color = app.offcolor;
        app.YellowLamp.Color = app.offcolor;
        if temperature >= 30
            app.TemperatureAlarmEditField.Value = "ON"
        end
        if coolevel < 30
```

```
            app.LowCoolantAlarmEditField.Value = "ON"
            app.TemperatureAlarmEditField.Value = "OFF";
        end
    else
        app.DistancetoObjectEditField.Value = distance;
        app.MotorSpeedEditField.Value = speed;
        app.LowCoolantAlarmEditField.Value = "OFF"
        app.TemperatureAlarmEditField.Value = "OFF";
        if (distance <= 15)
            app.RedLamp.Color = 'r';
            app.GreenLamp.Color = 'g';
            app.YellowLamp.Color = 'y';
        elseif ((distance > 15)&&(distance <= 25))
            app.RedLamp.Color = app.offcolor;
            app.GreenLamp.Color = 'g';
            app.YellowLamp.Color = 'y';
        elseif ((distance > 25)&&(distance < 32 ))
            app.RedLamp.Color = app.offcolor;
            app.GreenLamp.Color = 'g';
            app.YellowLamp.Color = app.offcolor;
        else
            app.RedLamp.Color = app.offcolor;
            app.GreenLamp.Color = app.offcolor;
            app.YellowLamp.Color = app.offcolor;
        end
        if (headlight == 0) || (headlight == 1)
            app.headlight1.Color = app.off;
            app.headlight2.Color = app.off;
            app.HeadlightStatusEditField.Value = "OFF";
        elseif headlight == 2
            app.headlight1.Color = app.dim;
            app.headlight2.Color = app.dim;
            app.HeadlightStatusEditField.Value = "DIM";
        else
            app.headlight1.Color = app.bright;
            app.headlight2.Color = app.bright;
            app.HeadlightStatusEditField.Value = "BRIGHT";
        end
    end
end
delete(arduino);
```