ECE 303 LABORATORY

Drexel University

To: M. Shakir

From: Laura Douglas

Re: Final Project

Purpose: For our final project, the goal was to create a crash avoidance system. Implementing the following hardware: lcd screen, water level sensor, ultrasonic sensor, 2 led's as headlights controlled by an IR remote, a dc motor with fans, a rfid security system which included a buffer and a temperature sensor.

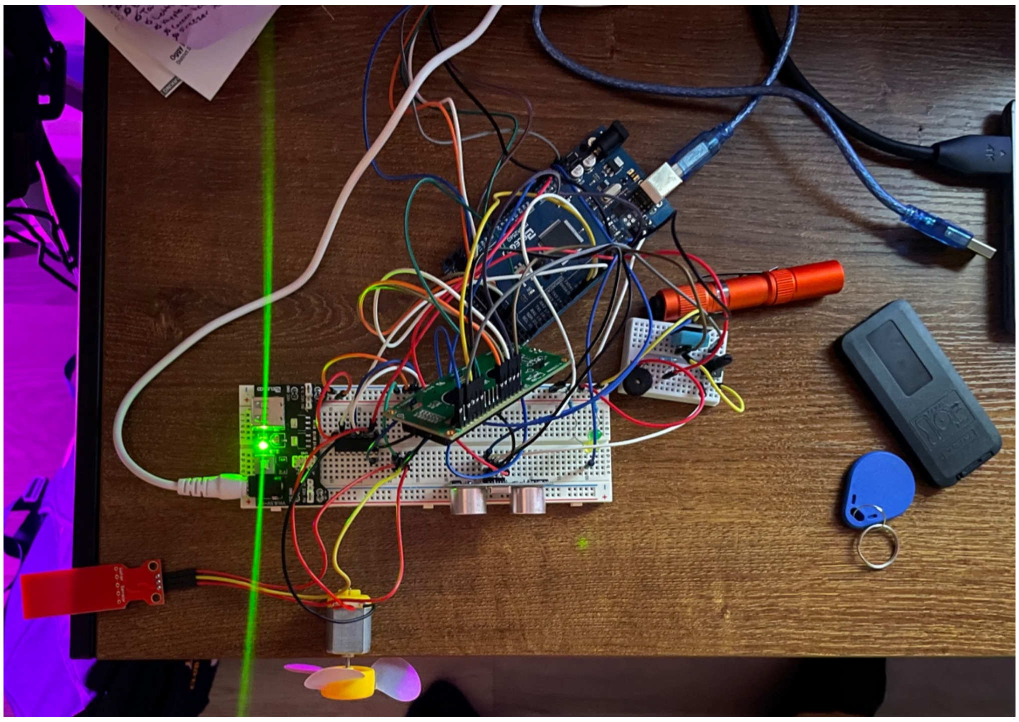
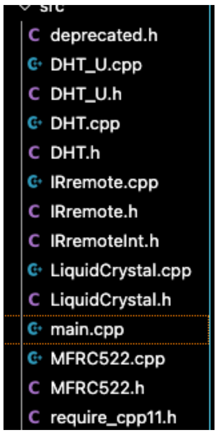


Figure 1: circuit connection

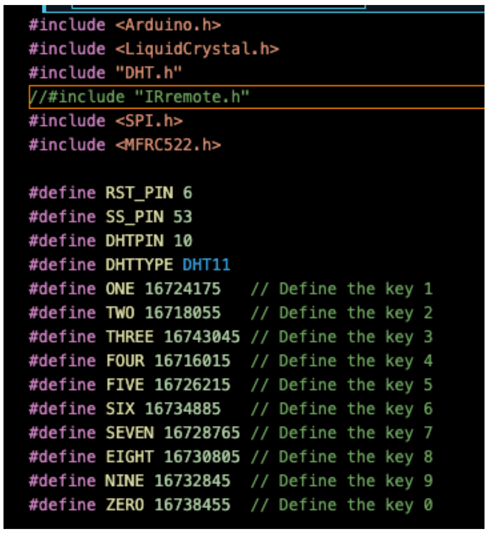
I began by including all necessary libraries to my project, as seen below:



```
src
  deprecated.h
  DHT_U.cpp
  DHT_U.h
  DHT.cpp
  DHT.h
  IRremote.cpp
  IRremote.h
  IRremoteInt.h
  LiquidCrystal.cpp
  LiquidCrystal.h
  main.cpp
  MFRC522.cpp
  MFRC522.h
  require_cpp11.h
```

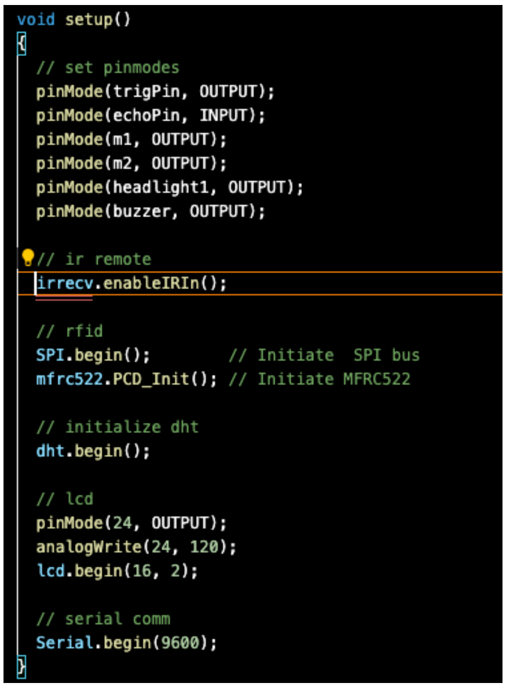
Figure 2: Libraries used

Next, I included all the .h files to access these libraries and defines my pin numbers for all my accessories:



```
#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
```

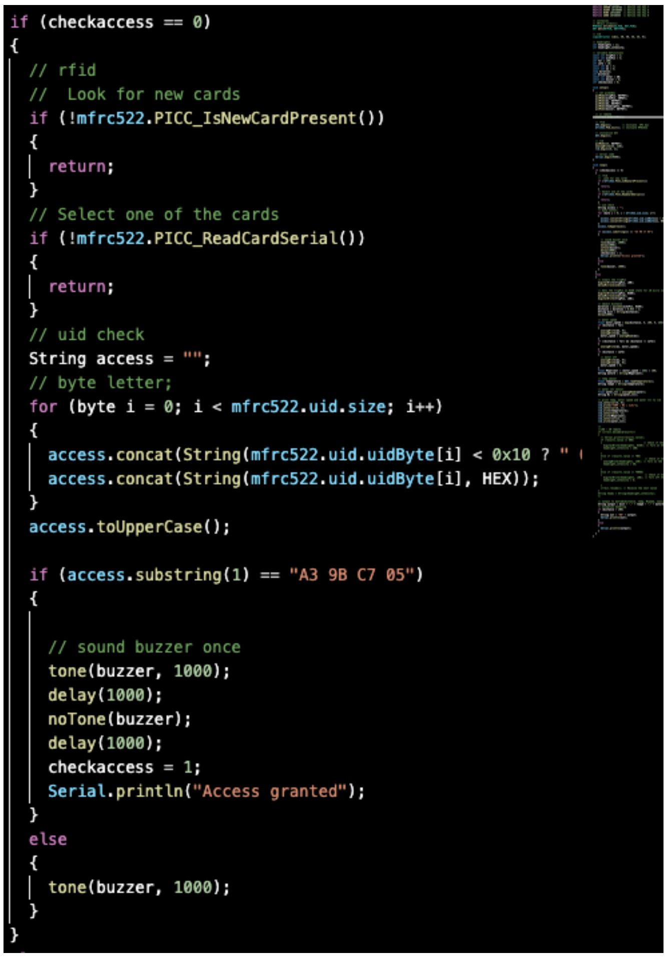
Figure 3: includes and some pin definitions



```
void setup()
{
  // set pinmodes
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(headlight1, OUTPUT);
  pinMode(buzzer, OUTPUT);

  // ir remote
  irrecv.enableIRIn();

  // rfid
  SPI.begin();          // Initiate  SPI bus
  mfrc522.PCD_Init(); // Initiate MFRC522

  // initialize dht
  dht.begin();

  // lcd
  pinMode(24, OUTPUT);
  analogWrite(24, 120);
  lcd.begin(16, 2);

  // serial comm
  Serial.begin(9600);
}
```

Figure 4: pins initialised in setup

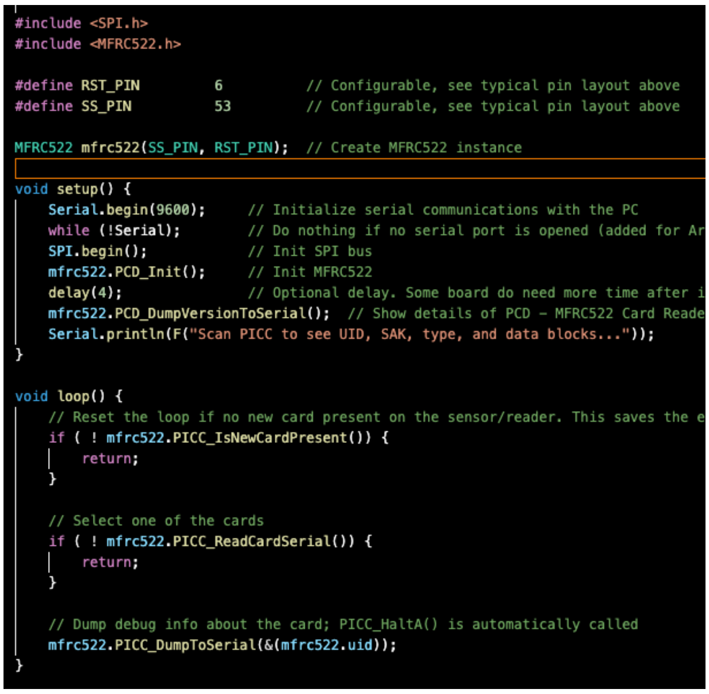
After initializing all my pins, I created a for loop for the access granting/denial. If the user has the right fob for access, a short beep is emitted from the buzzer and the code begins to run; otherwise, a long uninterrupted beep plays until you close the system. This was implemented to prevent unauthorized access.



```
if (checkaccess == 0)
{
  // rfid
  //  Look for new cards
  if (!mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  // uid check
  String access = "";
  // byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    access.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 
    access.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  access.toUpperCase();

  if (access.substring(1) == "A3 9B C7 05")
  {

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

Figure 5: Access granting/denial

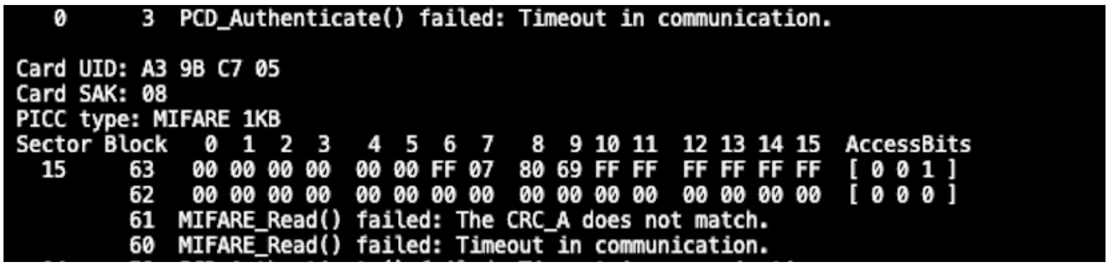
However, in order to get my UID value for comparison in my loop, I needed to run the following code:



```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         6          // Configurable, see typical pin layout above
#define SS_PIN          53         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
    Serial.begin(9600);     // Initialize serial communications with the PC
    while (!Serial);        // Do nothing if no serial port is opened (added for Ar
    SPI.begin();            // Init SPI bus
    mfrc522.PCD_Init();     // Init MFRC522
    delay(4);               // Optional delay. Some board do need more time after i
    mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reade
    Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void loop() {
    // Reset the loop if no new card present on the sensor/reader. This saves the e
    if ( ! mfrc522.PICC_IsNewCardPresent()) {
        return;
    }

    // Select one of the cards
    if ( ! mfrc522.PICC_ReadCardSerial()) {
        return;
    }

    // Dump debug info about the card; PICC_HaltA() is automatically called
    mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Figure 6: code to get UID number of key fob/card

Which output the following to terminal:

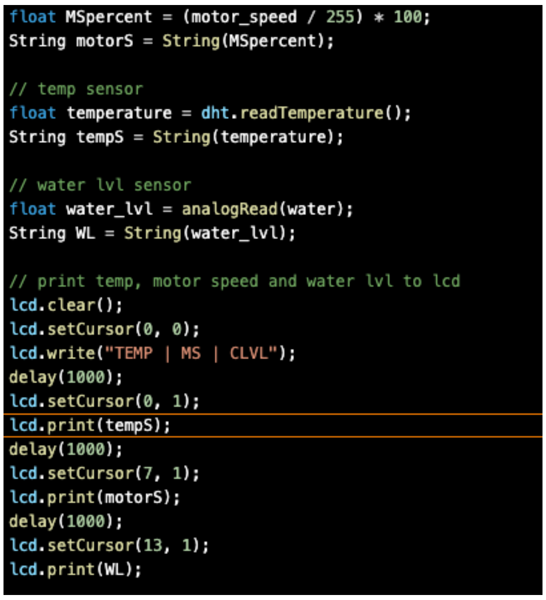


```
   0      3  PCD_Authenticate() failed: Timeout in communication.

Card UID: A3 9B C7 05
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block   0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15  AccessBits
  15     63   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         62   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         61  MIFARE_Read() failed: The CRC_A does not match.
         60  MIFARE_Read() failed: Timeout in communication.
```

Figure 7: UID number

Next following the previous lab, I implemented the motor decreasing as the object gets closer to the ultrasonic sensor. [refer to lab 7].

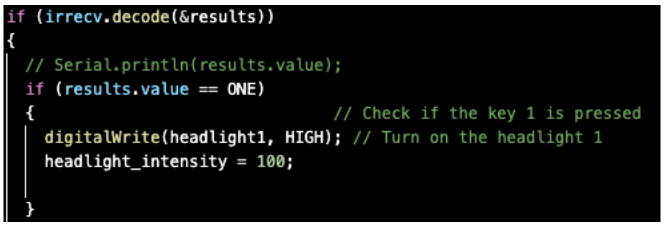
I calculated motor speed in percentage, read and calculated temperature, and red water level, converted these variables to strings and output all of this to the LCD with delays in between to give the system time to read and print the values



```
float MSpercent = (motor_speed / 255) * 100;
String motorS = String(MSpercent);

// temp sensor
float temperature = dht.readTemperature();
String tempS = String(temperature);

// water lvl sensor
float water_lvl = analogRead(water);
String WL = String(water_lvl);

// print temp, motor speed and water lvl to lcd
lcd.clear();
lcd.setCursor(0, 0);
lcd.write("TEMP | MS | CLVL");
delay(1000);
lcd.setCursor(0, 1);
lcd.print(tempS);
delay(1000);
lcd.setCursor(7, 1);
lcd.print(motorS);
delay(1000);
lcd.setCursor(13, 1);
lcd.print(WL);
```

Figure 8: string conversion and LCD display

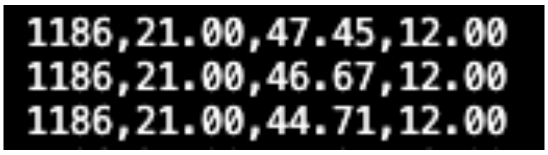
Using the pre-defined values for ONE, TWO and THREE on the IR remote, I made of several if/else statements, to change the brightness of my LED from off, on(bright), and on(dim).



```
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;

  }
```

Figure 9: if/else snippet for IR remote and LED relationship.

Finally, I concatenated all my integer outputs, that have been converted to strings and returned them in a single string, which I split up in MATLAB for processing with the gui.



```
1186,21.00,47.45,12.00
1186,21.00,46.67,12.00
1186,21.00,44.71,12.00
```

Figure 10: string output example

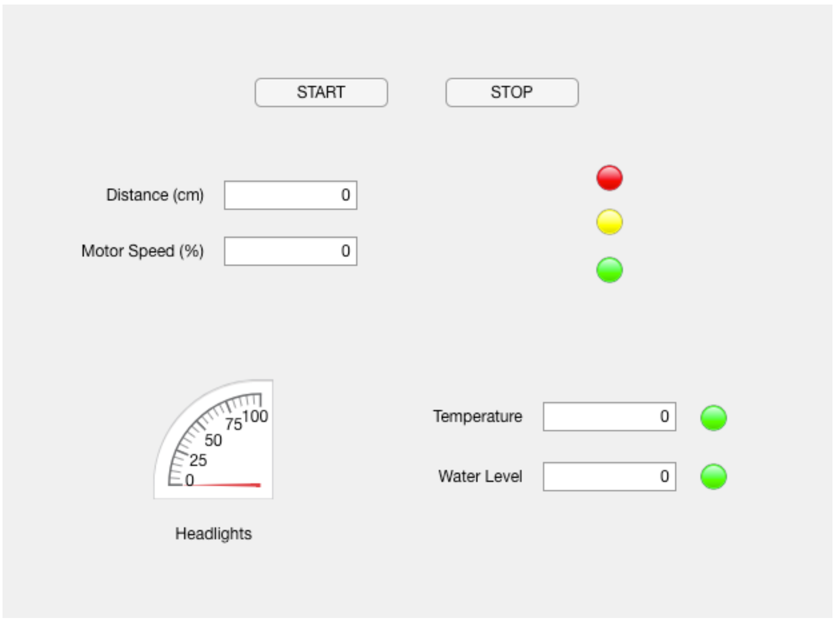
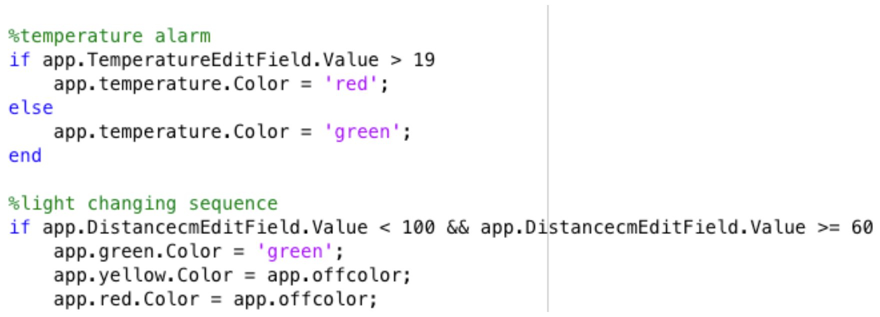


Figure 11: design view of gui

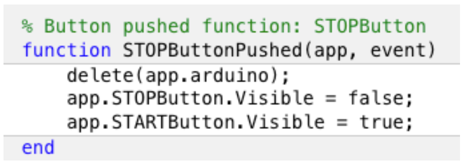
When the start button is clicked, the Arduino starts, authorized user must scan their fob for the system to begin running. After this scan, the string is split and the strings are converted to data type double, str2double in MATLAB. These values are then assigned to their respective data edit fields. Using this data, and several if/else statements, I am able to manipulate the led's and turn the temperature and water level sensors red, whenever the threshold is exceeded.



```
%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;
```

Figure 12: if/else snippet for sensors and led/distance change

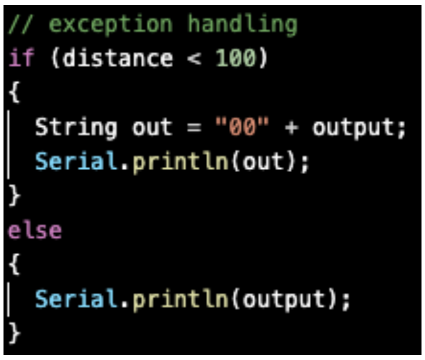
Next, I added a stop button to allow the user to stop the system on their own accord.



```
% Button pushed function: STOPButton
function STOPButtonPushed(app, event)
    delete(app.arduino);
    app.STOPButton.Visible = false;
    app.STARTButton.Visible = true;
end
```

Figure 13: stop button function

One problem faced when reading from Arduino to MATLAB was with the distance fields. If the distance is less than a certain number of characters, it messed up the 'read' Arduino function in MATLAB, causing the entire gui to fail. In order to handle this exception, I padded my value in vscode, so that is the distance was less than a threshold number, ‘00’ was added in front, so all distance string had a minimum of 3 characters.



```
// exception handling
if (distance < 100)
{
  String out = "00" + output;
  Serial.println(out);
}
else
{
  Serial.println(output);
}
```

Figure 14: exception handling