Drexel University

To: Dr. Peters

From: Tyler Ostinato

Date: Dec 20, 2021

Re: ECE 303-Final Project

Purpose

Combine all the labs conducted over the past 8 weeks into one project. Be able to manage timer conflicts, current budget, and space management of components to create a system replicating the functions of a self-driving car.

Discussion

Ultrasonic Sensor | 1.1

Using the same code from my Lab 6 I was able to implement the ultrasonic sensor and the Arduino map function to detect the distance of an object from the system.

Map the Angle of Acceptance | 1.2

According to the datasheet of the Ultrasonic sensor provided in our kit the effect angle of acceptance is 30 degrees. This would be 15 degrees in each direction from the middle of the ultrasonic sensor (Figure 1).

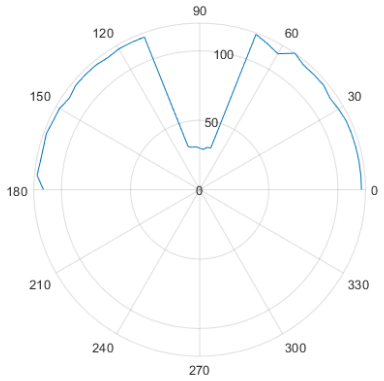

Figure 1: Two-way angle of acceptance for Ultrasonic Sensor.

Motor Installation | 1.3

Next, I set up the motor and connected it to the ultrasonic sensor like my week 6 lab. (Figure 2).

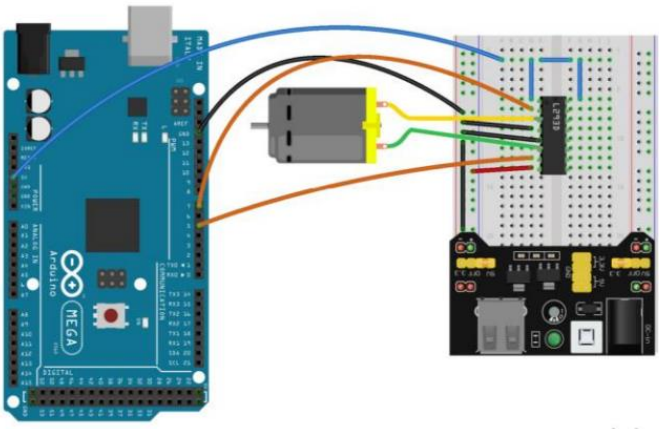

Figure 2: DC motor with H-bridge setup.

LED Installation | 1.4

The LED lights indicating the distance of an object will be implemented on the GUI section of this project, refer to section 2.1.

Graphical User Interface | 2.1

This user interface will need to convey a lot of information to the user. That means we need to communicate many values from the Serial terminal on the Arduino to MATLAB. To accomplish this, I sent all the necessary data delimited by a comma to MATALB and then processed the

data in MATLB. To read the data in MATALB I used regex to separate the data based on the comma. I was then able to index the data into its own variable and use them to trigger events in the MATLAB GUI (Figure 3).

```
while 1
    flush(arduino)
    write(arduino,2,'string');
    pause(0.5)
    a=read(arduino,16,'string');
    flush(arduino)
    c_str = regexp(num2str(a), ',', 'split');


    try
        distance = str2double(c_str(1));
        headlights_status = str2double(c_str(2));
        motor_speed = str2double(c_str(3));
        temp = str2double(c_str(4));
        coolant = str2double(c_str(5));
```

Figure 3: Reading multiple values from serial terminal into MATLAB.

In this final project I used a few new GUI components to display the water level and temperature of the sensor. Using the gauges from the MATALB GUI provided a more visibly satisfying and useful presentation of data (Figure 4).

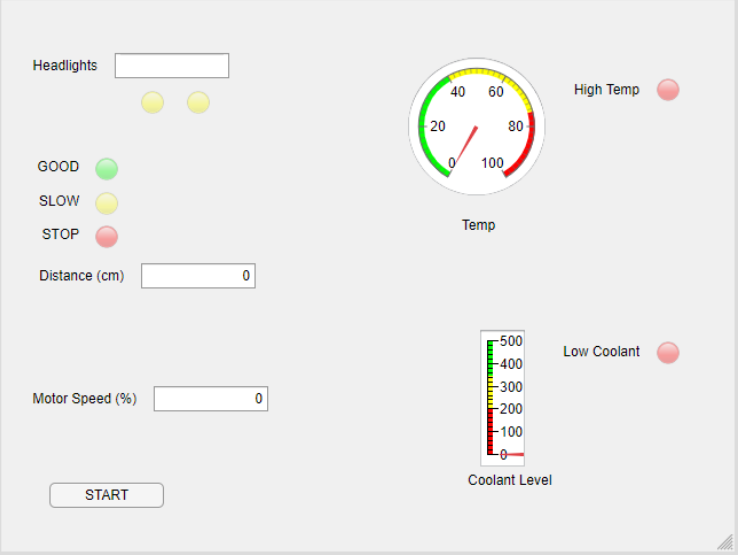

Figure 4: MATLAB GUI.

Interlocks | 3.1

To capture the temperature of the system I chose to use the DHT11 temperature sensor. The is very simple to use and can display the current temperature using DHT.temperature(). For the water level sensor, it was also very easy to implement. Apply power, ground, and an analog input we can get a reading to determine the amount of water. After getting this information I just made a simple conditional that would turn off all components when the temp became too high, or the water level was too low (Figure 5).

```
// Check temperature and water level
int chk = DHT.read11(DHT11_PIN);
water_level = analogRead(A0);
temp = DHT.temperature;
if(temp > 75 || water_level < 200){
  distance=0;
  analogWrite(led1, 0);
  analogWrite(led2, 0);
  lcd.clear();
}
```

Figure 5: High temperature or low water system shutoff logic.

Security and Remote | 4.1

This part of the project was the most difficult for me. Implementing the RFID and IR remote separately don't pose a challenge, but when putting them together a timer conflict will arise. It took a long time to figure out how to fix this. After lots of research I was able to determine how to change the timer of the IR remote. In the Arduino libraries we can go to IRremote/src/private/IRTimer.hpp. In this file there is a section for the Arduino Mega 2560 that contains 5 timer options. I commented out timer 2 and selected timer 3 instead (Figure 6). This fixed the timer conflict between my RFID card reader and Infrared remote.

```
// Arduino Mega
#elif defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#  if !defined(IR_USE_AVR_TIMER1) && !defined(IR_USE_AVR_TIMER2) && !defined(IR_USE_AVR_TIMER3)
//#define IR_USE_AVR_TIMER1   // send pin = pin 11
// #define IR_USE_AVR_TIMER2     // send pin = pin 9
#define IR_USE_AVR_TIMER3   // send pin = pin 5
//#define IR_USE_AVR_TIMER4   // send pin = pin 6
//#define IR_USE_AVR_TIMER5   // send pin = pin 46
#  endif
```

Figure 6: Changing IRremote library default timer.

LCD Display | 5.1

Implementation of the LCD was straightforward following the fritzing diagram provided (figure 7). I was then simply able to write the necessary information to the LCD display using lcd.print().

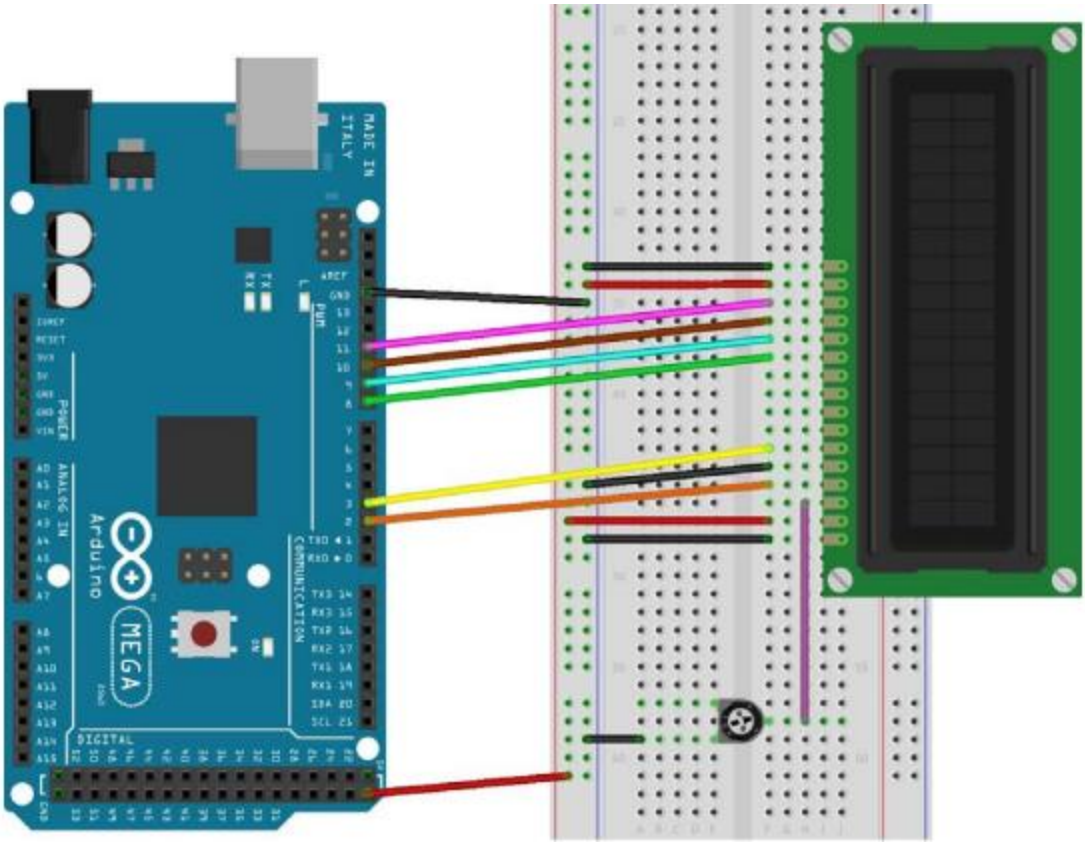

Figure 7: Schematic of connecting LCD display to Arduino.

Current Budget | 6.1

The current budget for my project is as follows. After a few Google searches I was able to determine a few key notes. The power supply can support a max of 700mA of current at any given time. The 3.3v pin on the Arduino can support 150mA of current. Any individual IO pin on the Arduino can also supply 40mA individually and 200mA between them all. After looking at the data sheets for all my components I believe my current budget should be fine. I would like to highlight a few of the key components to better understand why. The DC motor in my kit takes a maximum of 100mA with no load. Assuming the fan on the motor does not provide much resistance, we can assume its drawing around 100mA of current. The servo motor will draw around 120mA with no load. The RFID and water level sensor will use power off the Arduino directly and will not take current from the power supply. Now accounting for LEDS, LCD, Buzzer, RFID, Ultrasonic sensor, IR sensor, and temp sensor they will draw around 380mA. Our DC power supply from the Arduino should have more than enough current to supply to the components.

Component	Current Consumption (mA)
DC Motor	100
Servo Motor	120
RFID Module	9
IR Module	20
Ultrasonic Sensor	15
Temperature Sensor	2.5
Water Sensor	20
LCD Display	18.5
Passive Buzzer	30
White LEDs (x2)	50
Total	380 mA

Conclusion

When I signed up for this class, I was not sure what to expect, but I am glad to say I have not been disappointed. I enjoyed each and everyone of these labs. I got a great deal of experience using the Arduino and learning how to program the hardware and using the breadboard. I felt this final project did an excellent job at wrapping up everything I've learned in the last 8 weeks. I am very proud of my final project and plan on keeping it together because I think it looks awesome.

Appendix

Arduino Sketch

```
// Init DC Motor
int dc_motor=4;
int motor_speed=0;
int motor_prcnt=0;

// Init Ultrasonic Sensor
long duration=0; // variable for the duration of sound wave travel
int distance=0; // variable for the distance measurement
#define echoPin 23 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 22 //attach pin D3 Arduino to pin Trig of HC-SR04

// Init Servo Motor
#include <Servo.h>
Servo servo;
int deg=0;
int servo_pin=6;

#include <dht.h>
dht DHT;
#define DHT11_PIN 30
int temp=0;

// Init Water Level Sensor
int water_level=0;

// Init RFID
#include <SPI.h>
```

```
#include <MFRC522.h>

#define SS_PIN 53
#define RST_PIN 7
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
int access_flag=0;

// Init IR Remote
// Changed timer in IRemote/src/private/IRTimer.hpp -> #define IR_USE_AVR_TIMER3   // send pin = pin 5
#include <IRremote.h>
#include <IRremoteInt.h>
IRrecv irrecv(42);
decode_results results;
long lastPressTime = 0;
int state = LOW;
int max_speed=255;
#define SEND_PWM_BY_TIMER=0

// Init Buzzer
const int buzzer = 5; //buzzer to arduino pin 9
int piezoPin = 5;

// Init LCD Display
#include <LiquidCrystal.h>
#include <stdlib.h>
LiquidCrystal lcd(2,3,8,9,10,12);

// Init Headlights
int led1 = 13;
int led2 = 11;
int headlight_status=0;
```

```
void accepted_sound(){
  tone(piezoPin, 5000, 250);
}

void alarm_sound(){
  tone(piezoPin, 1000, 100);
//  delay(250);
  tone(piezoPin, 1000, 100);
//  delay(250);
  tone(piezoPin, 1000, 100);
//  delay(250);
  tone(piezoPin, 1000, 100);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  // DC Motor setup
  pinMode(dc_motor, OUTPUT);

  // Ultrasonic Sensor setup
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

  // Servo Motor setup
  servo.attach(servo_pin);
  servo.write(0);
```

```
  // RFID setup
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522

  // IR Remote setup
  irrecv.enableIRIn(); // Start the receiver

  // Buzzer setup
  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

  // Setup LCD Display
  pinMode(26,OUTPUT);
  analogWrite(26,120);
  lcd.begin(16,2);

  // Headlights setup
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
}



void loop() {
  if(access_flag==0){ // change back to zero
    analogWrite(dc_motor,0);
    // Look for new cards
    if ( ! mfrc522.PICC_IsNewCardPresent())
    {
      return;
    }
    // Select one of the cards
    if ( ! mfrc522.PICC_ReadCardSerial())
```

```
  {
    return;
  }
  //Show UID on serial monitor
  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));
  }
  content.toUpperCase();
  if(content.substring(1) == "1C C6 0C 4A"){
    access_flag=1;
    accepted_sound();
  }
  else{
    access_flag=0;
    alarm_sound();
  }
 }


 if (access_flag==1){
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
//   delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
//   delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
```

```
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance = duration * 0.034 / 2;



// Check temperature and water level
int chk = DHT.read11(DHT11_PIN);
water_level = analogRead(A0);
temp = DHT.temperature;
if(temp > 75 || water_level < 200){
  distance=0;
  analogWrite(led1, 0);
  analogWrite(led2, 0);
  lcd.clear();
}

if(distance > 30){
  distance = 30;
}

// Check for motor speed change from remote
if(irrecv.decode(&results)){
  if(results.value==16748655){
    max_speed+=25;
  }
  else if(results.value==16769055){
    max_speed-=25;
  }
  irrecv.resume();
```

```
    // Headlights logic
    if(results.value==16753245){
      analogWrite(led1, 255);
      analogWrite(led2, 255);
      headlight_status=2;
    }
    if(results.value==16769565){
      analogWrite(led1, 0);
      analogWrite(led2, 0);
      headlight_status=0;
    }
    if(results.value==16736925){
      analogWrite(led1, 50);
      analogWrite(led2, 50);
      headlight_status=1;
    }
  }
  if(max_speed>255){max_speed=255;motor_prcnt=100;}
  if(max_speed<0){max_speed=0;motor_prcnt=0;}

  // Change motor speed based on 0-30cm distance
  motor_speed=map(distance,0,30,0,max_speed);
  motor_prcnt=map(motor_speed,0,255,0,100); // Still need to have LCD speed change in respect to the remote speed change-
  analogWrite(dc_motor,motor_speed);

  // Call the breaks
  servo.attach(servo_pin);
  if(distance < 11){
   servo.write(180);
  }
```

```
else{
  servo.write(0);
}

// Serial Terminal to MATLAB
Serial.print(distance);
Serial.print(",");
Serial.print(headlight_status);
Serial.print(",");
Serial.print(motor_prcnt);
Serial.print(",");
Serial.print(temp);
Serial.print(",");
Serial.print(water_level);
Serial.println(",");

// LCD Display Logic
lcd.clear();
lcd.setCursor(0,0);
lcd.print("S: %");
lcd.setCursor(4,0);
lcd.print(motor_prcnt);
if(water_level < 200){
  lcd.setCursor(0,1);
  lcd.print("W: ");
  lcd.setCursor(3,1);
  lcd.print("LOW");
}
if(water_level > 200 && water_level < 350){
  lcd.setCursor(0,1);
  lcd.print("W: ");
```

```
    lcd.setCursor(3,1);
    lcd.print("MED");
   }
   if(water_level > 350){
    lcd.setCursor(0,1);
    lcd.print("W: ");
    lcd.setCursor(3,1);
    lcd.print("HIGH");
   }
   lcd.setCursor(9,1);
   lcd.print("T: ");
   lcd.setCursor(12,1);-
   lcd.print((int)DHT.temperature);
   delay(250);
 }
}
```

MATLAB GUI Script

```
% Set up communications
arduino=serialport("COM8",9600,"Timeout",15);
pause(1)

while 1
    flush(arduino)
    write(arduino,2,'string');
    pause(0.5)
    a=read(arduino,16,'string');
    flush(arduino)
    c_str = regexp(num2str(a), ',', 'split');


    try
        distance = str2double(c_str(1));
        headlights_status = str2double(c_str(2));
        motor_speed = str2double(c_str(3));
        temp = str2double(c_str(4));
        coolant = str2double(c_str(5));
```

```
                % DISTANCE
                app.DistancecmEditField.Value=distance; % String wrong when <
10 cm

                % MOTOR SPEED
                app.MotorSpeedEditField.Value=motor_speed;

                % TEMP
                app.TempGauge.Value=temp;
                if temp >= 75
                    app.HighTempLamp.Enable = 'on';
                else
                    app.HighTempLamp.Enable = 'off';
                end


                % COOLANT
                app.CoolantGauge.Value=coolant;
                if coolant <= 200
                    app.LowCoolantLamp.Enable = 'on';
                else
                    app.LowCoolantLamp.Enable = 'off';
                end


                % HEADLIGHTS
                if headlights_status == 0
                    app.HeadlightsEditField.Value="OFF";
                    app.HL1.Enable = 'off';
                    app.HL2.Enable = 'off';
                end

                if headlights_status == 1
                    app.HeadlightsEditField.Value="DIM";
                    app.HL1.Enable = 'on';
                    app.HL2.Enable = 'on';
                end

                if headlights_status == 2
                    app.HeadlightsEditField.Value="HIGH";
                    app.HL1.Enable = 'on';
                    app.HL2.Enable = 'on';
                end


                %LEDS
                if distance < 12
                    app.GOODLamp.Enable = 'on';
                    app.SLOWLamp.Enable = 'on';
```

```
            app.STOPLamp.Enable = 'on';
        end

        if distance > 10 && distance < 16
            app.GOODLamp.Enable = 'on';
            app.SLOWLamp.Enable = 'on';
            app.STOPLamp.Enable = 'off';
        end

        if distance > 16 && distance < 24
            app.GOODLamp.Enable = 'on';
            app.SLOWLamp.Enable = 'off';
            app.STOPLamp.Enable = 'off';
        end

        if distance > 24
            app.GOODLamp.Enable = 'off';
            app.SLOWLamp.Enable = 'off';
            app.STOPLamp.Enable = 'off';
        end

    catch
        continue
    end

end
delete(arduino);
clear arduino;
```