ECE 303 Term Project Report

ELECTRIC VEHICLE TEST-BED

Illoh,Ekenedilichukwu

Drexel University

Table of Contents

Project Objective

The aim of this project was to design an electric vehicle test-bed using the skills and knowledge that we gained throughout the period of this course. The following parts were to be implemented in the system:

- Collision avoidance testing
- Dashboard development/ GUI
- Security and Remote control
- Sensor implementation

This project was completed through Arduino and Python codes as well as the circuit that the codes controlled.

Project 1 – Collision Avoidance

This is a concept familiar in modern automobiles. It uses sensors to determine if there is something in front of the car. There are signals that are sent to parts of the car to slow it down if need be. In this test-bed, the task was to simulate a collision avoidance system using the ultrasonic sensor and motor in our kits. The closer an object gets to the sensor, the slower the motor rotates imitating a vehicle coming to a stop if it is approaching an object in front of it.

Part 1: Ultrasonic Sensor Installation and Verification

The sensor pins were initialized in the Arduino code using the trigger and echo pins. It was given a maximum distance of 200 cm at which point it would not read any object in front of it. The ultrasonic sensor works using sound waves, it gives off sound waves and if there is an object in front of it, the waves hit the object and it bounces back to the sensor. The trigger pin triggers the transmitter to send out a sound wave and after the bounces back of the object, the echo pin reads the data of the receiver. The faster the sensor receives the waves bounced back, the closer the object is to it. In the code a formula was implemented to convert the travel time of the sound waves into distance in centimeters to directly show how far an object is from the sensor.

```
int getDistance() {

    digitalWrite(trigger, LOW);
    delayMicroseconds(2);
    digitalWrite(trigger, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigger, LOW);
    int distance;
    travTime = pulseIn(echo, HIGH);
    distance = (travTime*0.0343)/2;
    return distance;
}
```

Figure 1: Code initializing the sensor

Part 2: Map the angle of acceptance

This task was completed using the instructions from the project guide. I placed a flat object 30 cm directly in front of the sensor. Using a protractor, the object was positioned from 90 degrees in front of the sensor. A reading was then taken, the object was moved 5 degrees to

the left or the right of the sensor and took a reading again. This was done until readings were no longer gotten.

Part 3: DC Motor Installation and Verification

The DC Motor was connected on the same board as the ultrasonic sensor. In the Arduino code the pins were initialized for the motor using the power, dc_in1 and dc_in2 variables. The motor was set up in such a way that the speed could be controlled and determined by a number of different variables which will be discussed later. The pin could be set to 'LOW' to turn off the motor or the velocity could be adjusted to slow down or speed up. The motor moves in one direction due to the H bridge in the circuit.

Part 4: LED Installation and Verification

The LED's were set up and installed on the GUI using a python code. These were to be connected with the ultrasonic sensor and DC Motor. Three different LED's (green, yellow and red) were set up to serve as an indicator of when an object is too close to the sensor. If an object is between 15cm and 30cm, the green LED will come on in the dashboard. If an object is between 5cm and 15cm, both the yellow and green LED's will be displayed, and the motor speed will reduce. Finally, if the object is less than 5cm from the sensor, the red, yellow and green LED's will all be displayed, and the motor would come to a complete stop.

```
for i in line:
    if "Distance" in i:
        self.disText.delete(0.0, END)
        self.disText.insert(0.0, i.split(" ")[1])

        if float(i.split(" ")[1][:-1]) <= 0.05:
            self.colorTab3.create_oval(5, 5, 70, 70, width=0, fill='red')   # Red LED Canvas
            self.colorTab2.create_oval(5, 5, 70, 70, width=0, fill='yellow')    # Yellow LED Canvas
            self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='green')  # Green LED Canvas

        elif (float(i.split(" ")[1][:-1]) > 0.05) and (float(i.split(" ")[1][:-1]) <= 0.15):
            self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='green')  # Green LED Canvas
            self.colorTab2.create_oval(5, 5, 70, 70, width=0, fill='yellow')  # Yellow LED Canvas
            self.colorTab3.delete("all")

        elif (float(i.split(" ")[1][:-1]) > 0.15) and (float(i.split(" ")[1][:-1]) < 0.30):
            self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='green')  # Green LED Canvas
            self.colorTab2.delete("all")
            self.colorTab3.delete("all")

        else:
            self.colorTab.delete("all")
            self.colorTab2.delete("all")
            self.colorTab3.delete("all")
```

Figure 2: GUI code for LED's

Project 2 – Graphical User Interface

This aspect of the project was developed to display the real-time data of the system as it is running as a way to see the system performance. To construct the GUI, the 'tkinter' package was used in python to create the interface.
The GUI displayed the following data in real-time:

- Distance LED's based on distance
- The Headlight status
- The Motor speed in percent
- The coolant alarm status
- The temperature reading in degrees Celsius
- The temperature alarm status

The serial command in python enabled the GUI to be connected to the serial port of the circuit in order to receive the data being displayed. The inputs that were received by the code helped update the dashboard to display the real-time data of the system.

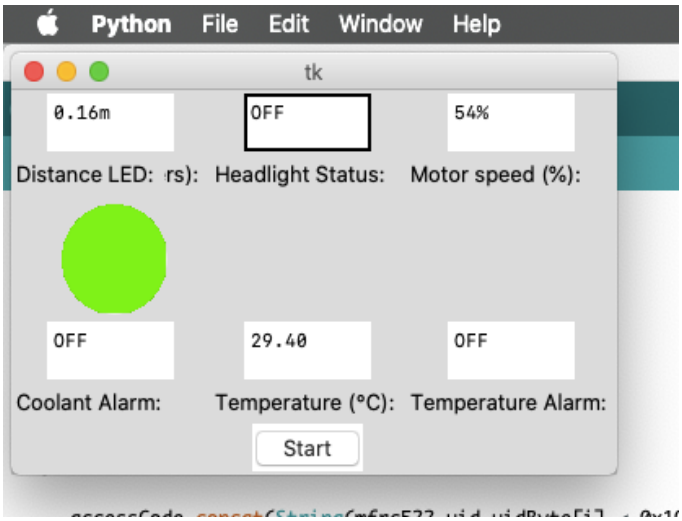

Figure 3: Screenshot of Dashboard

Project 3 – Interlocks

This part of the project was based on the temperature and water level of the system. The system was to shut down if the water level is low or if the temperature is too high. These occurrences would also be reflected in the dashboard by turning the coolant alarm or temperature alarm on respectively. Once these parameters are back to normal i.e. the

temperature is below the max or the water level is no longer low, then the system should come back on.

The water level and temperature are set to initial values at the beginning of the Arduino code. Once the code is uploaded into the circuit, those values begin to update to the real-time values.

```
bool interlocks() {
  if ((pinTemp > 35) || (Dist1 < 5) || (watlvl < 200)){
    TurnOff();
    return true;
  }
  else {
    return false;
  }

}
```

Figure 4: Code to set Interlocks

As seen in the function above, the system is set to turn off if any one of those parameters occur, i.e. if the temperature goes above 35℃, the water level drops below 200 cm or the distance of an object to the sensor is less than 5cm as discussed earlier in the report.

Project 4 – Security and Remote Control

For this part of the project, a security measure was to be implemented to prevent unauthorized personnel from starting the system. A remote control feature was also added to control the speed of the motor and the headlights as well.

Security Measure

A function that detects when there is a card close to the RFID was implemented and once it detected a card, it called a read serial function that reads the serial number of the card. This was used to fetch the serial number of the card that was to have access to start the system. Any other card would have a serial number and so a function was written in Arduino to check if the serial number matches and if there is no match, the system does not start. This grants only proper personnel access to the system.

```
void authorize() {
 if (mfrc522.PICC_ReadCardSerial()) {

  String accessCode= "";
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {

     accessCode.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     accessCode.concat(String(mfrc522.uid.uidByte[i], HEX));
  }

  if (accessCode.substring(1) == "39 0a 09 b4")
  {

    tone(activeBuzzer, 200, 3000);
    while (! mfrc522.PICC_IsNewCardPresent()){
      TurnOn();
    }
  }

  else {
    TurnOff();
    for (int i=0; i<2; i++){
      tone(activeBuzzer, 500, 1000);
      delay(500);
      noTone(activeBuzzer);
      delay(1000);
  }

  }

  }
}
```

Figure 5: Code to implement Security Measures

Furthermore, a buzzer was implemented in the code above to give off different sounds informing the user whether or not they are authorized.

Remote Control

A function was written in the Arduino code using an IR module. The circuit receives instructions from the remote in the form of infrared waves. In this project, the motor speed and headlights could also be controlled using the remote. The motor speed could be increased or decreased in 10% increments while the headlights could be set to off/dim/high depending on what buttons are pressed on the remote. The active buttons of the remote were coded using the case function as can be seen in the code below.

```
void RemoteControl(){
  if (irrecv.decode(&results)){

       switch(results.value){
         case 0xFFE01F:
         reduceSpeed();
         break;

         case 0xFF906F:
         increaseSpeed();
         break;

         case 0xFF30CF:
         analogWrite(led1, 0);
         analogWrite(led2, 0);
         lights = "OFF";
         break;

         case 0xFF18E7:
         analogWrite(led1, 100);
         analogWrite(led2, 100);

         lights = "DIM";
         break;

         case 0xFF7A85:
         analogWrite(led1, 255);
         analogWrite(led2, 255);

         lights = "HIGH";
         break;

       }

       irrecv.resume();
    }
}
```

Figure 6: Code for Remote Control

Python Code for the GUI

```
import tkinter as tk
from tkinter import *
import serial


class mainApp:

    def __init__(self):
        self.arduino = serial.Serial('/dev/cu.usbmodem14201', 9600, timeout=5)  # Communicates with the serial port
        self.window = Tk()
        self.window.configure(bg="light gray")

        self.disLabel = Label(self.window, text="Distance (meters): ", bg="light gray")
        self.disLabel.grid(row=1, column=0, sticky=W)

        self.disText = Text(self.window, height=2, width=10)
        self.disText.grid(row=0, column=0, pady=2)

        self.disLEDLabel = Label(self.window, text="Distance LED: ", bg="light gray")
        self.disLEDLabel.grid(row=1, column=0, sticky=W)

        #   Canvas for drawing circles
        self.colorTab = Canvas(self.window, width=70, height=70, bg='light gray', highlightthickness=0)
        self.colorTab.grid(row=2, column=0, pady=2)
        self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='white')

        self.colorTab2 = Canvas(self.window, width=70, height=70, bg='light gray', highlightthickness=0)
        self.colorTab2.grid(row=2, column=1, pady=2)
        self.colorTab2.create_oval(5, 5, 69, 69, width=0, fill='white')

        self.colorTab3 = Canvas(self.window, width=70, height=70, bg='light gray', highlightthickness=0)
        self.colorTab3.grid(row=2, column=2, pady=2, padx=2)
        self.colorTab3.create_oval(5, 5, 69, 69, width=0, fill='white')
```

```
        self.headLightLabel = Label(self.window, text="Headlight Status: ", bg="light gray")
        self.headLightLabel.grid(row=1, column=1, sticky=W)

        self.headLightText = Text(self.window, height=2, width=10)
        self.headLightText.grid(row=0, column=1, pady=2)

        self.motorVelocityTab = Label(self.window, text="Motor speed (%): ", bg="light gray")
        self.motorVelocityTab.grid(row=1, column=2, sticky=W)

        self.motorVelocityText = Text(self.window, height=2, width=10)
        self.motorVelocityText.grid(row=0, column=2, pady=2)


        self.coolantAlarmLabel = Label(self.window, text="Coolant Alarm: ", bg="light gray")
        self.coolantAlarmLabel.grid(row=4, column=0, sticky=W)

        self.coolantAlarmText = Text(self.window, height=2, width=10)
        self.coolantAlarmText.grid(row=3, column=0, pady=2)


        self.TempLabel = Label(self.window, text="Temperature (°C): ", bg="light gray")
        self.TempLabel.grid(row=4, column=1, sticky=W)

        self.TempText = Text(self.window, height=2, width=10)
        self.TempText.grid(row=3, column=1, pady=2)

        self.TempAlarmLabel = Label(self.window, text="Temperature Alarm: ", bg="light gray")
        self.TempAlarmLabel.grid(row=4, column=2, sticky=W)

        self.TempAlarmText = Text(self.window, height=2, width=10)
        self.TempAlarmText.grid(row=3, column=2, pady=2)

        self.startButton = Button(self.window, text="Start", bg="green", width=5, command=self.main)
        self.startButton.grid(row=5, column=1, pady=2)
```

```
        self.window.mainloop()

    def main(self):
        while True:
            line = self.arduino.readline().decode("utf-8")
            line = line.split(", ")
            print(line)
            for i in line:
                if "Distance" in i:
                    self.disText.delete(0.0, END)
                    self.disText.insert(0.0, i.split(" ")[1])

                    if float(i.split(" ")[1][:-1]) <= 0.05:
                        self.colorTab3.create_oval(5, 5, 70, 70, width=0, fill='red')   # Red LED Canvas
                        self.colorTab2.create_oval(5, 5, 70, 70, width=0, fill='yellow')    # Yellow LED Canvas
                        self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='green')  # Green LED Canvas

                    elif (float(i.split(" ")[1][:-1]) > 0.05) and (float(i.split(" ")[1][:-1]) <= 0.15):
                        self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='green')  # Green LED Canvas
                        self.colorTab2.create_oval(5, 5, 70, 70, width=0, fill='yellow')  # Yellow LED Canvas
                        self.colorTab3.delete("all")

                    elif (float(i.split(" ")[1][:-1]) > 0.15) and (float(i.split(" ")[1][:-1]) < 0.30):
                        self.colorTab.create_oval(5, 5, 69, 69, width=0, fill='green')  # Green LED Canvas
                        self.colorTab2.delete("all")
                        self.colorTab3.delete("all")

                    else:
                        self.colorTab.delete("all")
                        self.colorTab2.delete("all")
                        self.colorTab3.delete("all")
```

```
            elif "Motor" in i:
                self.motorVelocityText.delete(0.0, END)
                self.motorVelocityText.insert(0.0, i.split(" ")[2])

            elif "Temperature" in i:
                self.TempText.delete(0.0, END)
                self.TempText.insert(0.0, i.split(" ")[1])

                if float(i.split(" ")[1]) > 30:
                    self.TempAlarmText.delete(0.0, END)
                    self.TempAlarmText.insert(0.0, "ON")
                    self.motorVelocityText.delete(0.0, END)
                    self.motorVelocityText.insert(0.0, 0)
                else:
                    self.TempAlarmText.delete(0.0, END)
                    self.TempAlarmText.insert(0.0, "OFF")

            elif "Headlights" in i:
                self.headLightText.delete(0.0, END)
                self.headLightText.insert(0.0, (i.split(" ")[1]).upper())

            elif "Water" in i:
                if int(i.split(" ")[2]) <= 100:
                    self.coolantAlarmText.delete(0.0, END)
                    self.coolantAlarmText.insert(0.0, "ON")
                    self.motorVelocityText.delete(0.0, END)
                    self.motorVelocityText.insert(0.0, 0)
                else:
                    self.coolantAlarmText.delete(0.0, END)
                    self.coolantAlarmText.insert(0.0, "OFF")

        self.window.update()
```

```
                    self.motorVelocityText.insert(0.0, 0)
                else:
                    self.TempAlarmText.delete(0.0, END)
                    self.TempAlarmText.insert(0.0, "OFF")

            elif "Headlights" in i:
                self.headLightText.delete(0.0, END)
                self.headLightText.insert(0.0, (i.split(" ")[1]).upper())

            elif "Water" in i:
                if int(i.split(" ")[2]) <= 100:
                    self.coolantAlarmText.delete(0.0, END)
                    self.coolantAlarmText.insert(0.0, "ON")
                    self.motorVelocityText.delete(0.0, END)
                    self.motorVelocityText.insert(0.0, 0)
                else:
                    self.coolantAlarmText.delete(0.0, END)
                    self.coolantAlarmText.insert(0.0, "OFF")

            self.window.update()


if __name__ == "__main__":
    app = mainApp()
```

Arduino Code

```
/*
Project One - Collision Avoidance
1. Ultrasonic Sensor
2. DC Motor
 */
int power = 11;
int dc_in1 = 8;
int dc_in2 = 7;

int trigger = 9;
int echo = 12;
int distThreshold = 200;
int Dist1;
int Dist2 = 0;
int motVelocity = 0;
long travTime;


/*
Project Two - GUI
1. LEDS
2. Headlights
 */
int led1 = 3;
int led2 = 5;

String lights = "off";


/*
Project Three - Interlocks
1. Temperature Sensor
2. Water level sensor
 */
#include "DHT.h"
#define DHTTYPE DHT11
#define DHTPIN 6


#define sen_power 45
#define senPIN A0

DHT dht(DHTPIN, DHTTYPE);

int watlvl=0;
float pinTemp;


/*
 Project 4 - Security & Remote Control & Displays
 */
#include <IRremote.h>
int REMOTE_PIN = 2;
IRrecv irrecv(REMOTE_PIN);
decode_results results;
unsigned long value = 0;


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

#define SS_PIN 53
#define RST_PIN 49
MFRC522 mfrc522(SS_PIN, RST_PIN);

#include <LiquidCrystal.h>
const int rs = 4, en = 13, d4 = 47, d5 = 48, d6 = 45, d7 = 43;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int activeBuzzer = 46;
```

```
void setup() {
  // put your setup code here, to run once:

  /*
   Set all the motor control pins to outputs
   */
  pinMode(power, OUTPUT);
  pinMode(dc_in1, OUTPUT);
  pinMode(dc_in2, OUTPUT);


  Serial.begin(9600);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);

  /*
   Setting motors to initial state
   */
  digitalWrite(dc_in1, LOW);
  digitalWrite(dc_in2, LOW);

  /*
   Setting LED's to initial state
   */
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);

  /*
   Sets the power through the sensor to LOW
   */
  digitalWrite(sen_power, HIGH);



  /*
   IR Remote Setup
   */
  attachInterrupt(digitalPinToInterrupt(REMOTE_PIN), RemoteControl, CHANGE);
  irrecv.enableIRIn();

  /*
   Setting the temperature sensor
   */
  dht.begin();

  /*
   Setting the OUTPUT to D7
   */
  pinMode(sen_power, OUTPUT);


  /*
   Choosing the number of rows and columns for the LCD
   */
  lcd.begin(16, 2);

  /*
   Setup for the RFID
   */
  SPI.begin();
  mfrc522.PCD_Init();

  /*
   Initializing the Buzzer
   */
  pinMode(activeBuzzer, OUTPUT);

}
```

```
/*
 * Functionality Functions
 */

void authorize() {
 if (mfrc522.PICC_ReadCardSerial()) {

  String accessCode= "";
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {

     accessCode.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     accessCode.concat(String(mfrc522.uid.uidByte[i], HEX));
  }

  if (accessCode.substring(1) == "39 0a 09 b4")
  {

    tone(activeBuzzer, 200, 3000);
    while (! mfrc522.PICC_IsNewCardPresent()){
      TurnOn();
    }
  }

  else {
    TurnOff();
    for (int i=0; i<2; i++){
      tone(activeBuzzer, 500, 1000);
      delay(500);
      noTone(activeBuzzer);
      delay(1000);
  }

  }

  }
}

void RemoteControl(){
  if (irrecv.decode(&results)){

        switch(results.value){
          case 0xFFE01F:
          reduceSpeed();
          break;

          case 0xFF906F:
          increaseSpeed();
          break;

          case 0xFF30CF:
          analogWrite(led1, 0);
          analogWrite(led2, 0);
          lights = "OFF";
          break;

          case 0xFF18E7:
          analogWrite(led1, 100);
          analogWrite(led2, 100);

          lights = "DIM";
          break;

          case 0xFF7A85:
          analogWrite(led1, 255);
          analogWrite(led2, 255);

          lights = "HIGH";
          break;

        }

        irrecv.resume();
    }
}
```

```
void increaseSpeed() {
  if (motVelocity <= 100){
    motVelocity += 10;
    analogWrite(power, ((float)motVelocity/100)*255);
  }
  else if (motVelocity > 90 && motVelocity < 100){
    motVelocity = 100;
    analogWrite(power, ((float)motVelocity/100)*255);
  }

}

void reduceSpeed() {
  if (motVelocity > 0){
    motVelocity -= 10;
    analogWrite(power, ((float)motVelocity/100)*255);
  }
}


void updateSpeed() {
       digitalWrite(dc_in1, LOW);
       digitalWrite(dc_in2, HIGH);
       analogWrite(power, 255);
       float diff = Dist1-Dist2;

       if (Dist2 > Dist1){
         int velDiff = (((-1.0*diff)/30.0)*100.0);
         motVelocity = motVelocity - velDiff;

       }
       else{
         int velDiff = round(((float)(diff)/30)*100);
         motVelocity = motVelocity + velDiff;

       }

     if (motVelocity > 100){
       motVelocity = 100;
       analogWrite(power, ((float)motVelocity/100)*255);
     }
     else if (motVelocity < 0 ){
       motVelocity = 0;
       analogWrite(power, ((float)motVelocity/100)*255);
     }
     else {
       analogWrite(power, ((float)motVelocity/100)*255);

     Dist2 = Dist1;
}


bool interlocks() {
  if ((pinTemp > 35) || (Dist1 < 10) || (watlvl < 200)){
    TurnOff();
    return true;
  }
  else {
    return false;
  }

}

void TurnOn(){
    attachInterrupt(digitalPinToInterrupt(REMOTE_PIN), RemoteControl, CHANGE);

    /*
     Getting Sensor Readings
     */
    Dist1 = getDistance();
    pinTemp = Temp();
    watlvl = getWaterLevel();
```

```
    if (!interlocks()){
      updateSpeed();
      delay(1000);
    }


    Serial.print("Distance: ");
    Serial.print((float) Dist1/100);
    Serial.print("m, ");

    Serial.print("Motor speed: ");
    Serial.print(motVelocity);
    Serial.print("%");

    Serial.print(", Temperature: ");
    Serial.print(pinTemp);
    Serial.print(" *C, ");

    Serial.print("Water level: ");
    Serial.print(watlvl);

    Serial.print(", Headlights: ");
    Serial.println(lights);


    funcPrint();

    delay(500);

}


void TurnOff(){
  motVelocity = 0;
  digitalWrite(dc_in1, LOW);
  digitalWrite(dc_in2, LOW);
```

```
  digitalWrite(trigger, LOW);
  digitalWrite(power, LOW);


  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);


  detachInterrupt(digitalPinToInterrupt(REMOTE_PIN));

}


void funcPrint(){

    lcd.clear();

    lcd.setCursor(0, 0);
    lcd.print("S ");
    lcd.print(String(motVelocity));

    lcd.print("WL ");
    lcd.print(watlvl);


    lcd.setCursor(0, 1);
    lcd.print("T ");
    lcd.print(pinTemp);

}
```

Conclusion

This project enabled me to practice using timers and interrupts (with the remote control) and pulse width modulation (with the LED's and the DC motor). I was also given the opportunity to further understand how certain automobile systems works such as the collision avoidance and the security and remote-control systems. Being able to implement these systems using Arduino helped solidify the concepts that have been covered throughout this course.