ECE 303 Final Project

Drexel University

To: Dr. Christopher Peters

From: Zheren Gu

cc: Amirhosein Chahe

Date: 6/8/2023

Re: Final Project

Purpose

The purpose of this final project is to take all the labs that we have been working on throughout the term and incorporate the techniques we used in them all to complete this task of creating an electric vehicle test bed.

Discussion

This final project was quite challenging as there were many moving pieces that all needed to work with each other. There were 4 main objectives to this project. The first was Collision Avoidance, we are trying to simulate a car, so we need to make sure it doesn't crash. This part was the same as Lab 7 about motors. Then we needed to incorporate a GUI from Lab 6. Then in Part 3 we added factors that force the system to shut down, in our case if the water level goes under 100 or the temperature goes over 90 degrees. Finally in part 4 we were to add security measures and the ability to use a remote. This was accomplished using an RFID sensor to detect if we used a particular key card that came with our kit. Lastly, we added a display to show the parameters that are being measured by the device.

Recommendation

Unfortunately, I was unable to figure out fully how to incorporate the remote into the interface. This was the only part of the assignment that I was unable to complete as I couldn't figure out how to get the remote button timings correct in the void loop part of the code, as it loops through it was near impossible to time it perfectly to get a reading. Had I done this again this is the main thing that I would try to fix.

```
#include <Arduino.h>

#include "DigitalPin.h"


DigitalPin::DigitalPin(int pin){

    _pin = pin;

    this->set_pin();

    this->set_output_mode();


}

void DigitalPin::set_off(){

    digitalWrite(_pin, LOW);

}

void DigitalPin::set_on(){

    digitalWrite(_pin, HIGH);

}


void DigitalPin::set_TCCRA(int val){

    switch (_pin){
```

```
        case 44:
            TCCR5A = val;
            break;
        case 11:
            TCCR1A = val;
            break;
        case 6:
            TCCR4A = val;
            break;
        case 5:
            TCCR3A = val;
            break;

    }
}

void DigitalPin::set_TCCRB(int val){ // WGM and CS values would change based on timer
    switch (_pin){
        case 44:
            TCCR5B = val;
            TCCR5B |= (1 << WGM52);
            TCCR5B |= (1 << CS52);
            break;
        case 11:
```

```
            TCCR1B = val;
            TCCR1B |= (1 << WGM12);
            TCCR1B |= (1 << CS12);
            break;
        case 6:
            TCCR4B = val;
            TCCR4B |= (1<<WGM42);
            TCCR4B |= (1<<CS42);
            break;
        case 5:
            TCCR3B = val;
            TCCR3B |= (1 << WGM32); //ctc mode
            TCCR3B |= (1<< CS32) | (0<< CS31) | (0<< CS30); // 256 prescalar
            break;

    }
}

void DigitalPin::set_TCNT(int val){
    switch (_pin){
        case 44:
            TCNT5 = val;
            break;
        case 11:
```

```
            TCNT1 = val;
            break;
        case 6:
            TCNT4 = val;
            break;
        case 5:
            TCNT3 = val;
            break;

    }
}
void DigitalPin::set_OCR(int val){
    switch (_pin){
        case 44:
            OCR5A = val;
            break;
        case 11:
            OCR1A = val;
            break;
        case 6:
            OCR4A = val;
            break;
        case 5:
            OCR3A = val;
```

```
            break;
    }
}

void DigitalPin::factor_OCR(int factor){
    switch (_pin){
        case 44:
            OCR5A /= factor;
            break;
        case 11:
            OCR1A /= factor;
            break;
        case 6:
            OCR4A /= factor;
            break;
        case 5:
            OCR3A /= factor;
            break;
    }
}

void DigitalPin::set_TIMSK(int val){
    if (val == 1){
        switch (_pin){
```

```
        case 44:
            TIMSK5 = 0;
            TIMSK5 |= (1<<OCIE0A);
            break;
        case 11:
            TIMSK1 = 0;
            TIMSK1 |= (1<<OCIE1A);
            break;
        case 6:
            TIMSK4 = 0;
            TIMSK4 |= (1<<OCIE4A);
            break;
        case 5:
            TIMSK3 = 0;
            TIMSK3 |= (1<<OCIE3A);
            break;

    }
}
if (val == 0){
    switch (_pin){
        case 44:
            TIMSK5 = 0;
            break;
```

```
            case 11:
                TIMSK1 = 0;
                break;
            case 6:
                TIMSK4 = 0;
                break;
            case 5:
                TIMSK3 = 0;
                break;

        }
    }
}

// Homework 3 methods and On

void DigitalPin::set_pin(){
    if (_pin == 5){
        _DDR = &DDRE;
        _PORT = &PORTE;
        _PIN = &PINE;
        _bits = (1 << (3));
    }
    else if (_pin == 6){
```

```
        _DDR = &DDRH;
        _PORT = &PORTH;
        _PIN = &PINH;
        _bits = (1 << (3));
    }
    else if (_pin == 11){
        _DDR = &DDRB;
        _PORT = &PORTB;
        _PIN = &PINB;
        _bits = (1 << (5));
    }
    else if (_pin == 44){
        _DDR = &DDRL;
        _PORT = &PORTL;
        _PIN = &PINL;
        _bits = (1 << (5));
    }
}

void DigitalPin::set_output_mode(){
    *(_DDR) |= _bits;
}
void DigitalPin::on(){
    *(_PORT) |= _bits;
```

```
}
void DigitalPin::off(){
    *(_PORT) &=  B00000000;
}
void DigitalPin::invert_pin(){
    *(_PIN) |= (_bits);
}
void DigitalPin::set_input_mode(){
    *(_DDR) &= ~(_bits);
}

// Homework 4 methods

void DigitalPin::set_ICR(){
    switch (_pin){
        case 6:
            TCCR4A = (1<<WGM41) | (1<<COM4A1);
            TCCR4B = (1<<WGM43) | (1<<CS40);
            ICR4 = 255;
            OCR4A = 0;
            TCNT4 = 0;
            break;
    }
}
```

```
void DigitalPin::set_duty_cycle(int val){
    switch (_pin){
        case 6:
            OCR4A = (val * 255) / 100;
            break;
    }
}
#ifndef DigitalPin_
#define DigitalPin_
#include <Arduino.h>

class DigitalPin
{
    public:

        DigitalPin(int pin);
        void set_TCCRA(int val);
        void set_TCCRB(int val);
        void set_TCNT(int val);
        void set_OCR(int val);
        void factor_OCR(int factor);
        void set_TIMSK(int val);
        void set_off();
        void set_on();
```

```
        void set_pin();
        void set_output_mode();
        void set_input_mode();
        void on();
        void off();
        void invert_pin();

        void set_ICR();
        void set_duty_cycle(int val);

    private:
        int _pin;
        volatile uint8_t* _PORT;
        uint8_t _bits;
        volatile uint8_t* _DDR;
        volatile uint8_t* _PIN;
};

#endif
#include <Arduino.h>
#include <Servo.h>
#include <RFID.h>
#include <IRremote.h>
```

```
#include <IRremoteInt.h>
#include <LiquidCrystal.h>
#define REMOTEPIN 2
#define UP 4127850240
#define DOWN 4161273600
#define ONE 4077715200
#define TWO 3877175040
#define THREE 2707357440
#define SS_PIN 53
#define RST_PIN 45

LiquidCrystal lcd(46,35,37,33,47,49);
MFRC522 mfrc522(SS_PIN , RST_PIN);

const int forward = 5;
const int backward = 7;
int time_delay=3000;
int val = 0;
int speed = 0;

int degree = 0;
int servo_pin = 9;
Servo servo;
```

```
int trigPin = 10;
int echoPin = 11;
long duration, cm, inches;

const int max = 255;
const int min = 100;
const int thresholdDistance = 20;
const int stopDistance = 10;
int current = max;

int power_pin = 4;
int signal_pin = A0;
int waterval = 0;

int ThermistorPin = A1;
int Vo;
float R1 = 10000;
float logR2, R2, T;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
bool access = false;
const int buzzer = 8;

void go_forward(int speed){
  analogWrite(forward, speed);
```

```
}
void go_backward(){
  analogWrite(backward, 0);
}

void brake(){
  digitalWrite(forward, 0);
  digitalWrite(backward, 0);
}

IRrecv irrecv(REMOTEPIN);
void RemoteChange(){
    if (irrecv.decode()){
        if (irrecv.decodedIRData.decodedRawData == UP){
            // increase motor speed by 10%
            Serial.println("UP!");
            current += current * 0.10;
            if (current > max) { current = max;}
            go_forward(current);
            }

        if (irrecv.decodedIRData.decodedRawData == DOWN ){
            // decrease DC motor by 10
            Serial.println("Down");
```

```
            current -= current * 0.10;
            if (current < min) { current = min;}
            go_forward(current);

        }
      irrecv.resume();
    }

}

void setup() {

  lcd.begin(16,2);

  irrecv.enableIRIn();
  mfrc522.PCD_Init();
  servo.attach(servo_pin);
  servo.write(0);
  pinMode(buzzer, OUTPUT);

  pinMode(forward, OUTPUT);
  pinMode(backward, OUTPUT);
  digitalWrite(backward, 0);
  digitalWrite(forward, 1);
```

```
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(power_pin, OUTPUT);
  digitalWrite(power_pin, 0);

  SPI.begin();
  Serial.begin (9600);
  while (!access){
    access = rfid();
    delay(500);
   }
}

void loop() {

  RemoteChange();
  digitalWrite(power_pin, 1);
  delay(10);
  waterval = analogRead(signal_pin);
  digitalWrite(power_pin, 0);
  lcd.setCursor(0,0);
  lcd.print("Water:");
  lcd.print(waterval);
```

Serial.println(waterval);

Vo = analogRead(ThermistorPin);

R2 = R1 * (1023.0 / (float)Vo - 1.0);

logR2 = log(R2);

T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));

T = T - 273.15;

T = (T * 9.0)/ 5.0 + 32.0;

lcd.setCursor(0,1);

lcd.print("Temp(F):");

lcd.print(T);

Serial.println(T);

```
if (waterval>=100 && T < 90){
  digitalWrite(trigPin, 0);
  delayMicroseconds(5);
  digitalWrite(trigPin, 1);
  delayMicroseconds(10);
  digitalWrite(trigPin, 0);


  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, 1);

  cm = (duration/2) / 29.1;
  Serial.println(cm);
  if (cm > 25) {
    go_forward(255);
    go_backward();
    servo.write(180);
  } else if (cm > 20) {
```

```
      go_forward(150);
      go_backward();
      servo.write(135);
    } else if (cm > 15) {
      go_forward(100);
      go_backward();
      servo.write(90);
    } else {
      brake();
      servo.write(45);
    }
    delay(1000);
    }
  else{
    brake();
    servo.write(0);
    cm = 0;
    Serial.println(cm);
    delay(1000);
  }
}
#include <RFID.h>


bool rfid(){

  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return false;
  }

  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return false;
  }
  String content= "";
  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) == "5C 38 18 E0")
  {
    tone(buzzer, 2500, 1000);
    return true;

  }

 else{
    Serial.println("0");
    Serial.println("0");
    Serial.println("0");
    tone(buzzer, 500, 1000);
    return false;

  }
}

#ifndef RFID_
#define RFID_

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

extern MFRC522 mfrc522;
extern const int buzzer;
bool rfid();

#endif
```