MEMO

Drexel University

To: Dr. Christopher Peters, ECE 303 Fall 2020

From: Dinh Nguyen

Date: Nov 26th, 2020

Re: Term Project – Vehicle Test-bed

Purpose

The goal of this lab is to combine the studies of all the labs and lectures in the class to create a vehicle test-bed that will simulate a real vehicle system including collision avoidance, dashboard, security and remote control and sensor implementation. The lab also covers wide range of materials from serial communication, user interface, motors, RFID and IR remote control communication.

Discussion

Hardware: By combining a lot of individual circuit for RFID, water sensor, temperature sensor, motors, ultrasonic sensor, Figure 1 will show a photo of the final setup for the whole system.

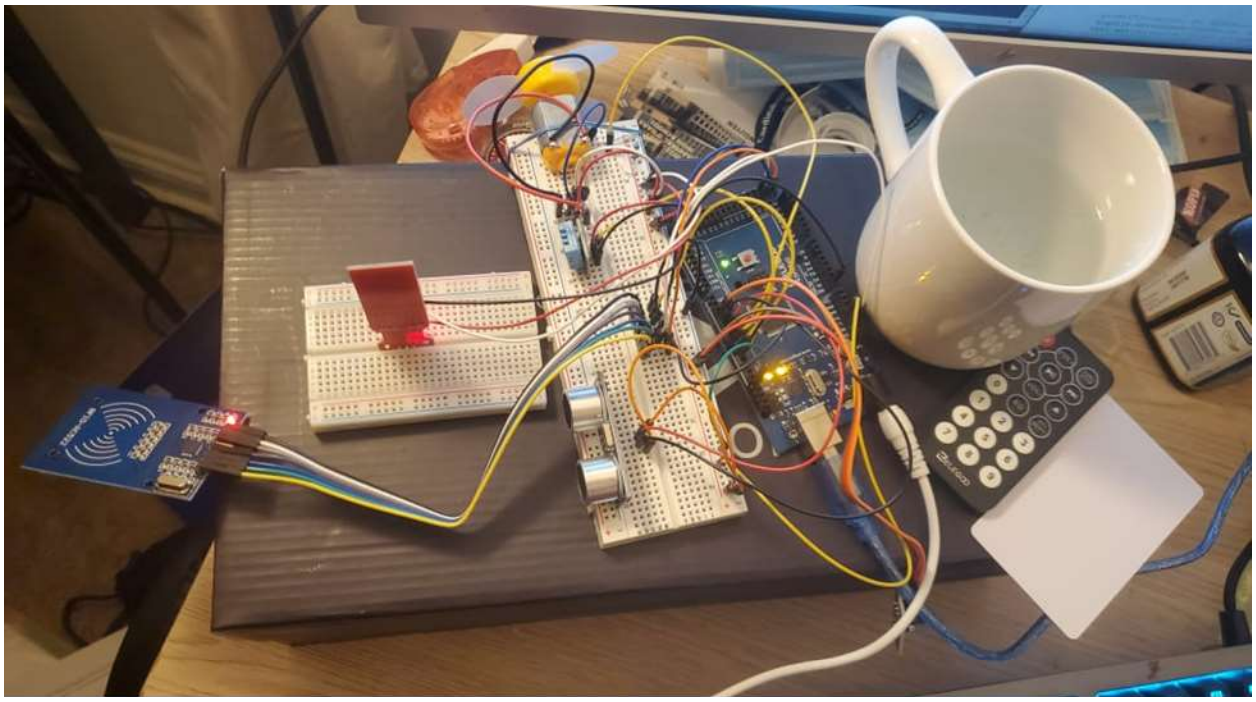
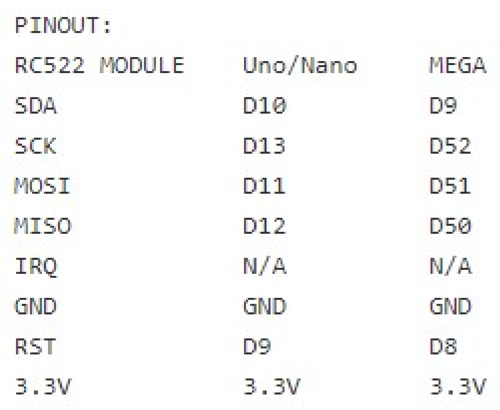


Figure 1. Vehicle Test-bed system

RFID panel and card are used to verify the identity of the user. They will be connected to 3.3V voltage source instead of 5V source because the equipment only works with maximum 3.3V. Depending on the type of the signal pin, they will be connected to different pins on Arduino MEGA 2560 board:



PINOUT:

RC522 MODULE	Uno/Nano	MEGA
SDA	D10	D9
SCK	D13	D52
MOSI	D11	D51
MISO	D12	D50
IRQ	N/A	N/A
GND	GND	GND
RST	D9	D8
3.3V	3.3V	3.3V

The red rectangle panel is the water sensor which will be connected to A0 pin on the board to read the water level. A cup of water will be used to simulate the coolant tank of the vehicle system. The temperature is also connected to pin 7 to read the temperature of the system. If the temperature is too high or the coolant level is too low, the system will be locked for safety. The ultrasonic sensor will be connected to pin 4 and 5 to get the duration between the send and receive signals. From that, the distance with the object will be calculated. The motor is connected to pin 6 and the remote-control sensor/receiver is connected to pin 3. Except from the RFID, the rest will be powered by a 5.5V voltage source.

Software: The speed of the motor will simulate the velocity of our vehicle. It will wait for the user to scan his RFID card to authorize the user to control the vehicle and start the system at a certain speed according to the distance with the object in the front of it. If the distance is below 5 cm, it will turn on up to 4 LEDs distance sensors on the dashboard and will force the speed to 0 and block the user to increase the speed until the distance with the object is further. Tkinter Python Library with Matplotlib library will be used to build up the user interface for the dashboard. A speedometer on the dashboard will be built to reflect the velocity of the vehicle based on the distance and the remote control from user. The system is also setup with the remote-control codes to accept input from the users to increase or reduce the speed if the coolant, the temperature and the distance is fine. Separate libraries for the RFID, coolant, temperature and remote-control sensors will be downloaded to Arduino IDE. The distance data will then be sent to the python script using serial communication and the python script will handle the data and build up the interface from that.

Usage Instruction:

To start, load the Arduino sketch onto the board and then run the python script by python3 cardash.py. Click the START button. The system will then freeze and wait for user to scan the correct RFID card. If the RFID card is correct, the authorized button on dashboard will turn Green. After that, if there is no close object in front of the distance sensor, and if coolant and temperature are good, the fan will start spinning. Remote control up and down buttons are to increase and decrease the fan speed. Remote control button 1,2 and 3 is for turning headlight off, dim and on respectively. If the coolant is low or the temperature is high, their LED will turn RED and the system will reset to 0 and wait until lower temperature or higher coolant fluid. Figure 2 will illustrate an example of the running vehicle dashboard.

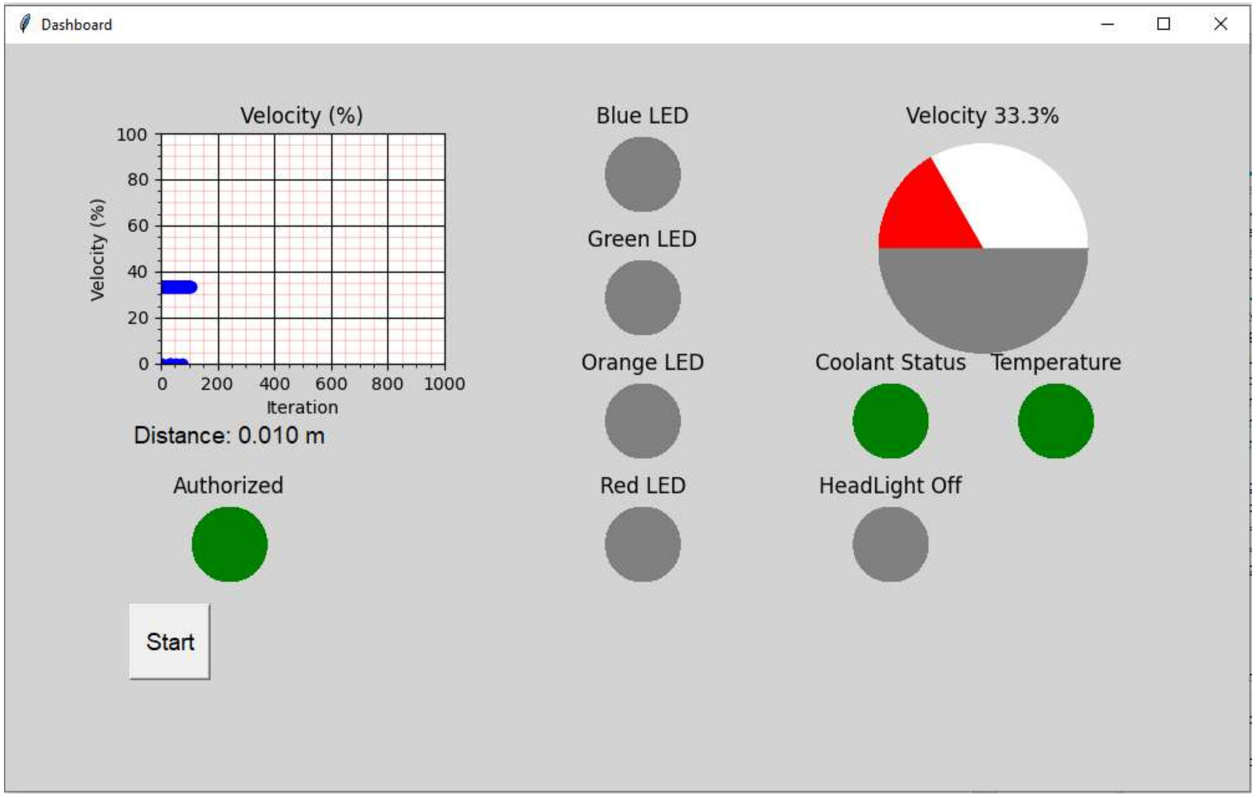
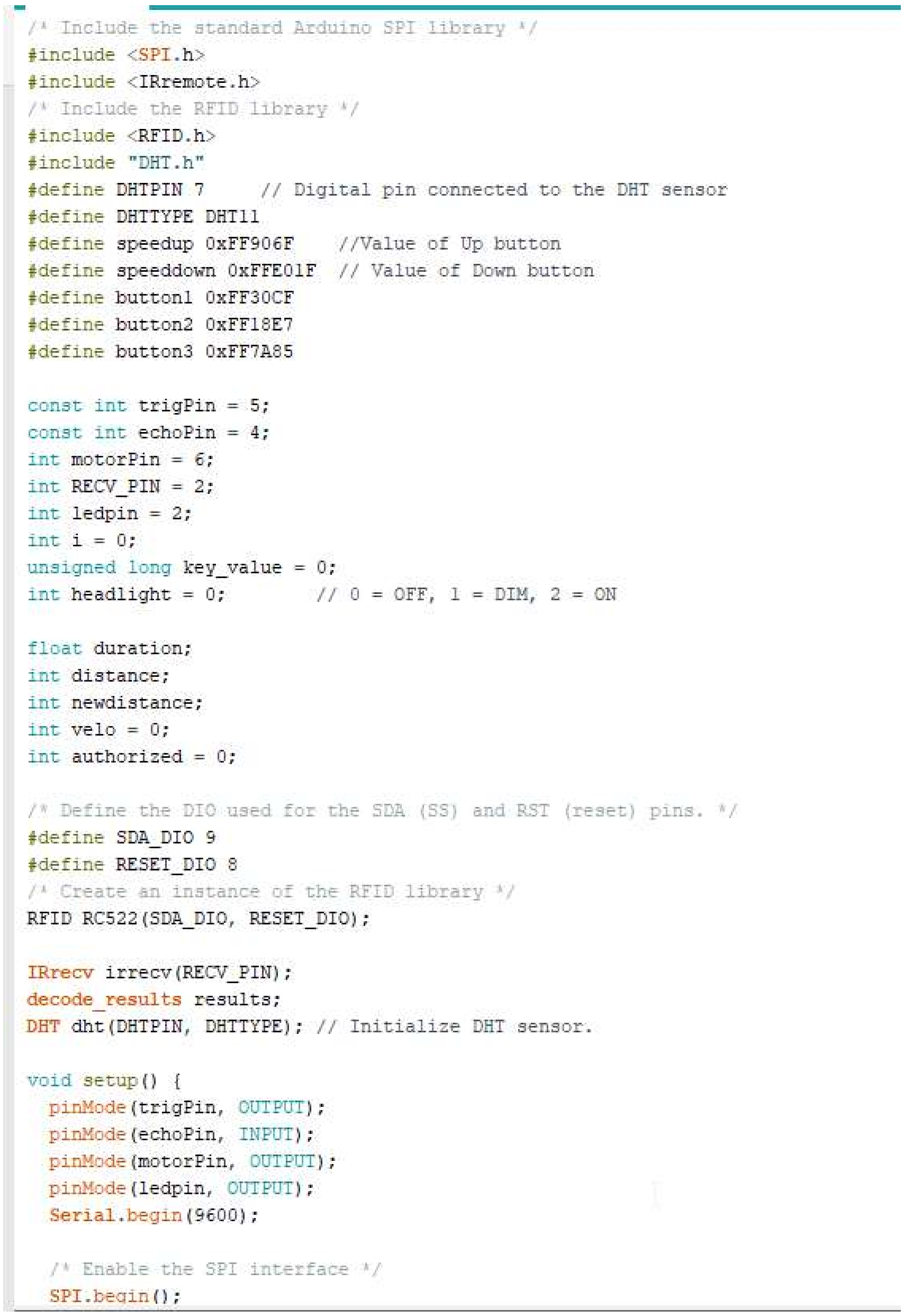


Figure 2. Vehicle Dashboard

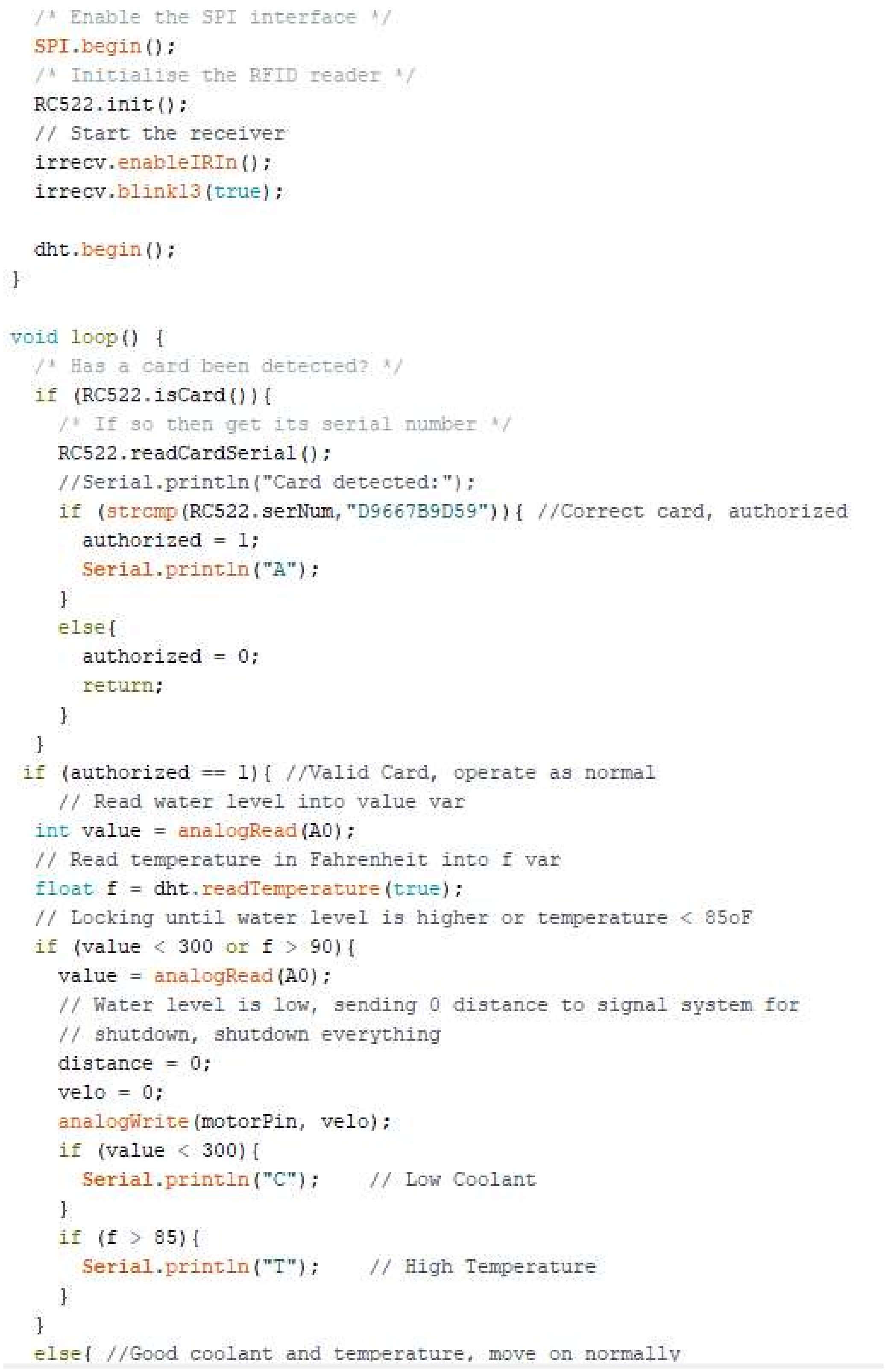
Recommendation

From testing, we can observe an inconsistency in using the IR remote control. The number specified for each of the button is sometimes not recognized by the IR receiver which causes some trouble reading user inputs to the Arduino and the Python UI. Thus, better equipment for the remote control is recommended. Examples of test-bed will be recorded in a separate video.

Arduino Sketch

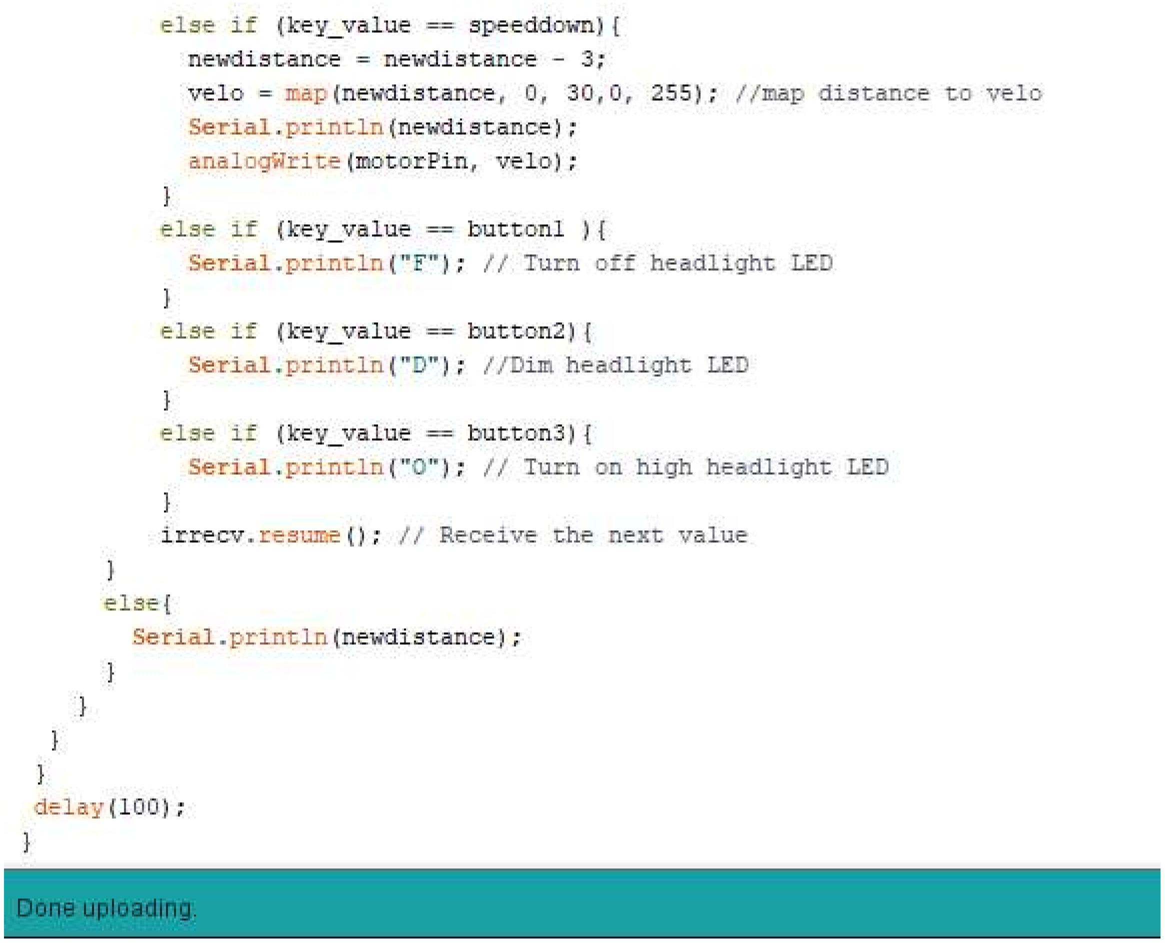


```
/* Include the standard Arduino SPI library */
#include <SPI.h>
#include <IRremote.h>
/* Include the RFID library */
#include <RFID.h>
#include "DHT.h"
#define DHTPIN 7     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11
#define speedup 0xFF906F    //Value of Up button
#define speeddown 0xFFE01F  // Value of Down button
#define button1 0xFF30CF
#define button2 0xFF18E7
#define button3 0xFF7A85

const int trigPin = 5;
const int echoPin = 4;
int motorPin = 6;
int RECV_PIN = 2;
int ledpin = 2;
int i = 0;
unsigned long key_value = 0;
int headlight = 0;        // 0 = OFF, 1 = DIM, 2 = ON

float duration;
int distance;
int newdistance;
int velo = 0;
int authorized = 0;

/* Define the DIO used for the SDA (SS) and RST (reset) pins. */
#define SDA_DIO 9
#define RESET_DIO 8
/* Create an instance of the RFID library */
RFID RC522(SDA_DIO, RESET_DIO);

IRrecv irrecv(RECV_PIN);
decode_results results;
DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor.

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(motorPin, OUTPUT);
  pinMode(ledpin, OUTPUT);
  Serial.begin(9600);

  /* Enable the SPI interface */
  SPI.begin();
```



```
  /* Enable the SPI interface */
  SPI.begin();
  /* Initialise the RFID reader */
  RC522.init();
  // Start the receiver
  irrecv.enableIRIn();
  irrecv.blink13(true);

  dht.begin();
}

void loop() {
  /* Has a card been detected? */
  if (RC522.isCard()){
    /* If so then get its serial number */
    RC522.readCardSerial();
    //Serial.println("Card detected:");
    if (strcmp(RC522.serNum,"D9667B9D59")){ //Correct card, authorized
      authorized = 1;
      Serial.println("A");
    }
    else{
      authorized = 0;
      return;
    }
  }
 if (authorized == 1){ //Valid Card, operate as normal
    // Read water level into value var
  int value = analogRead(A0);
  // Read temperature in Fahrenheit into f var
  float f = dht.readTemperature(true);
  // Locking until water level is higher or temperature < 85oF
  if (value < 300 or f > 90){
    value = analogRead(A0);
    // Water level is low, sending 0 distance to signal system for
    // shutdown, shutdown everything
    distance = 0;
    velo = 0;
    analogWrite(motorPin, velo);
    if (value < 300){
      Serial.println("C");    // Low Coolant
    }
    if (f > 85){
      Serial.println("T");    // High Temperature
    }
  }
  else{ //Good coolant and temperature, move on normally
```

```
else{ //Good coolant and temperature, move on normally
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH); //in us
  distance = (duration*.0343)/2;      // in cm
  if (distance >30){ //Max distance to handle is 30
    distance = 30;
  }
  // Handle velocity
  if (distance <= 5){
    velo = 0;
    analogWrite(motorPin, velo);
    Serial.println(distance);
  }
  else if(velo == 0 && distance > 5){
    // Safe distance, but just start the engine,
    // velo will determined by distance
    velo = map(distance, 0 ,30, 0 ,255);
    newdistance = distance;
    Serial.println(distance);
    analogWrite(motorPin, velo);
  }
  else{ //Safe distance and has some initial velo, so user can control speed
    // check if any button pressed and send it to PythonUI
      if (irrecv.decode(&results)) {

        if (results.value == 0XFFFFFFFF){
          results.value = key_value;
        }
        //Serial.println(results.value, HEX); //Debug
        key_value = results.value;

        if (key_value == speedup){
          newdistance = newdistance + 3; // 10% of max distance is 3. Control the velo
          velo = map(newdistance, 0, 30,0, 255); //map distance to velo
          Serial.println(newdistance);
          analogWrite(motorPin, velo);
        }
```



```
          else if (key_value == speeddown){
            newdistance = newdistance - 3;
            velo = map(newdistance, 0, 30,0, 255); //map distance to velo
            Serial.println(newdistance);
            analogWrite(motorPin, velo);
          }
          else if (key_value == button1 ){
            Serial.println("F"); // Turn off headlight LED
          }
          else if (key_value == button2){
            Serial.println("D"); //Dim headlight LED
          }
          else if (key_value == button3){
            Serial.println("O"); // Turn on high headlight LED
          }
          irrecv.resume(); // Receive the next value
      }
      else{
        Serial.println(newdistance);
      }
    }
  }
 }
 delay(100);
}
```

Done uploading.

Python Script

```python
import serial
import matplotlib.pyplot as plt
import time
import sys
import numpy
import tkinter as tk
import tkinter.font as tkFont
from tkinter import *
from tkinter import messagebox
import math as m
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

plt.close('all')

root = tk.Tk()
root.geometry("1000x600")
root.title("Dashboard")
root.configure(bg='grey')

figure1 = plt.Figure(figsize=(12,6), dpi=100)

ax1 = figure1.add_subplot(231)

fig1 = FigureCanvasTkAgg(figure1,root)
fig1.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH)
figure1.set_facecolor('lightgrey')

line1, = ax1.plot([],[],'bo')

ax1.set_xlabel('Iteration', fontsize=10)
ax1.set_ylabel('Velocity (%)', fontsize=10)
ax1.set_title('Velocity (%)')
ax1.grid(b=True, which='major', color='k', linestyle = '-')
ax1.grid(b=True, which='minor', color='r', linestyle = '-', alpha = 0.2)
ax1.minorticks_on()

figure1.subplots_adjust(hspace=0.5)
fontStyle= tkFont.Font(family="Lucida Grande", size=14)

var = StringVar()
var1 = StringVar()

#label= Label( root,textvariable=var,font=fontStyle,bg='lightgrey')
```

```
label1 = Label(root,textvariable=var1,font=fontStyle,bg='lightgrey')
#label2 = Label(root,textvariable='LEDs',font=fontStyle,bg='lightgrey')
#frame1 = Frame(width=20, height=20, bg="white",relief=SUNKEN)
#label.place(x=100,y=400)
label1.place(x=100,y=300)
#label2.place(x=600,y=300)
#frame1.place(x=600,y=400)

OnAuthorized = False

#system one-time status var -----------------------------------------------------
----------

Temp_Authorized = [100,0]      # order = denied, allowed - denied on start
sys_Authorized = figure1.add_subplot(5,6,19)
sys_Authorized.pie(Temp_Authorized, colors=['red','green'])
sys_Authorized.set_title('Unauthorized')
sys_Authorized.axis('equal')
#-------------------------------------------------------------------------------
--

def startCallBack():
    global OnAuthorized
    arduino=serial.Serial('COM4',9600,timeout=10)
    time.sleep(1)
    DC=[]
    V_pc=[]
    velo=[50,0,50]
    print("**********Lab5**************")

    #system long-term status var ------------------------------------------------
---------------
    Authorized = [100,0]
    HL_status = [100,0,0]      # order = off,dim,full - off on start
    #---------------------------------------------------------------------------
------



    with open('lab5logs.txt', 'w') as logFile:
        logFile.writelines("**********Lab5**************\n")
        logFile.writelines("Measurements:\n")

        for i in range(1000):
            DC.append(i)
            distance=0
            arduino.reset_input_buffer()
            arduino.reset_output_buffer()
            time.sleep(0.1)
            arduino.write(b'1')
            data = arduino.readline().decode("utf-8").strip()
            # Handle data collected from Arduino:
            # null, distance, "low coolant" or "high temperature"

            #system short-term status var --------------------------------------
------------------------
            CoolantLED = [100,0,0]     # order = off, normal, critical - off on s
tart
            TemperatureLED = [100,0,0] # order = off, normal, critical - off on s
tart
            #--------------------------------------------------------------------
--------------


            if data:
                print(data)
                if data == "C":
                    a = 0
                    CoolantLED = [0,0,100] #coolant low
                    # Set Coolant LED
                else:
                    CoolantLED = [0,100,0]
                    if data == "T":
                        a = 0
                        TemperatureLED = [0,0,100] #over heat
                        # Set Temperature LED
                    else:
                        TemperatureLED = [0,100,0]
                        if data == "A":
                            OnAuthorized = True
                            a = 0
                            Authorized = [0,100] #authorized
                        # Set Authorized LED
                        elif data == "F":
                            HL_status = [100,0,0]
                            # set Headlight LED off
                        elif data == "D":
                            HL_status = [0,100,0]
                            # Set headlight lED dim
                        elif data =="O":
                            HL_status = [0,0,100]
                            # setheadlight LED ON
                        else:
                            a= int(data)
                            OnAuthorized = True
                            Authorized = [0,100] #authorized
            else:
                a = 0
            distance = a/1000.0  # in meters, a in cm


            velo[1] = distance*100/0.03
            V_pc.append(velo[1])


            print("Iteration: %d, Distance: %f m, Velo: %f " % (i,distance,velo[1
]))
            logFile.writelines("Iteration: %d, Distance: %f m, Velo: %f \n" % (i,
distance,velo[1]))
            var1.set("Distance: {:.3f} m".format(distance))
            #var.set("Velocity: {:.3f} %".format(velo[1]))
            # Figure
            line1.set_data(DC,V_pc)
            axa = fig1.figure.axes[0]
            axa.set_xlim(0,1000)
            axa.set_ylim(0, 100)

            # Speedometer
            realvelo = velo[1]
            velo[1] = velo[1]/2
            velo[0] = 50-
velo[1] # Only draw half of the circle for the speedometer
            Rounded_velo = round(realvelo*10)/10  #round num
            percentage_speed = str(Rounded_velo) #convert to string
            ax2 = figure1.add_subplot(233)
            ax2.pie(velo,colors=['white','red','gray'])
            ax2.set_title('Velocity ' + percentage_speed +'%')
            ax2.axis('equal')

            # system status LEDs ------------------------------------------------
-----------

            sys_coolant = figure1.add_subplot(5,6,17)

            sys_coolant.pie(CoolantLED, colors=['gray','green','red'])
            sys_coolant.set_title('Coolant Status')
            sys_coolant.axis('equal')
            sys_temperature = figure1.add_subplot(5,6,18)
            sys_temperature.pie(TemperatureLED, colors=['gray','green','red'])
            sys_temperature.set_title('Temperature')
            sys_temperature.axis('equal')

            sys_Authorized = figure1.add_subplot(5,6,19)
            sys_Authorized.pie(Authorized, colors=['red','green'])
            if OnAuthorized:
                sys_Authorized.set_title('Authorized')
            else:
                sys_Authorized.set_title('Unauthorized')
            sys_Authorized.axis('equal')

            sys_HeadLight = figure1.add_subplot(5,6,23)
            sys_HeadLight.pie(HL_status, colors=['gray','pink','red'])
            if HL_status[0] == 100:
                sys_HeadLight.set_title('HeadLight Off')
            if HL_status[1] == 100:
                sys_HeadLight.set_title('HeadLight Dim')
            if HL_status[2] == 100:
                sys_HeadLight.set_title('HeadLight Full')
            sys_HeadLight.axis('equal')

            # ------------------------------------------------------------------
-------

            # LEDs --------------------------------------------------------------
------

            if distance <= 5/1000:
                LED1 = [100,0]
            else:
                LED1 = [0,100]
            if distance <= 4/1000:
                LED2 = [100,0]
            else:
                LED2 = [0,100]
            if distance <= 3/1000:
                LED3 = [100,0]
            else:
                LED3 = [0,100]
            if distance <= 2/1000:
                LED4 = [100,0]
            else:
                LED4 = [0,100]
            ax3 = figure1.add_subplot(5,1,1)
            ax3.pie(LED1, colors = ['blue','gray'])
            ax3.set_title('Blue LED')
            ax3.axis('equal')

            ax4 = figure1.add_subplot(5,1,2)
            ax4.pie(LED2, colors=['green','gray'])
            ax4.set_title('Green LED')
            ax4.axis('equal')

            ax5 = figure1.add_subplot(5,1,3)
            ax5.pie(LED3, colors=['orange','gray'])
            ax5.set_title('Orange LED')
            ax5.axis('equal')

            ax6 = figure1.add_subplot(5,1,4)
            ax6.pie(LED4, colors=['red','gray'])
            ax6.set_title('Red LED')
            ax6.axis('equal')
            # -------------------------------------------------------------------
-------
            # Draw tools -  Do not touch
            fig1.draw()
            plt.pause(0.05)
            plt.show()

        logFile.close()

    arduino.close()
    sys.exit(0)
    return

# Create Start button
start_button = Button(root, text ="Start", font=fontStyle, height=2, width=5, com
mand = startCallBack)
start_button.place(x=100, y=450)
root.mainloop()
```