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.

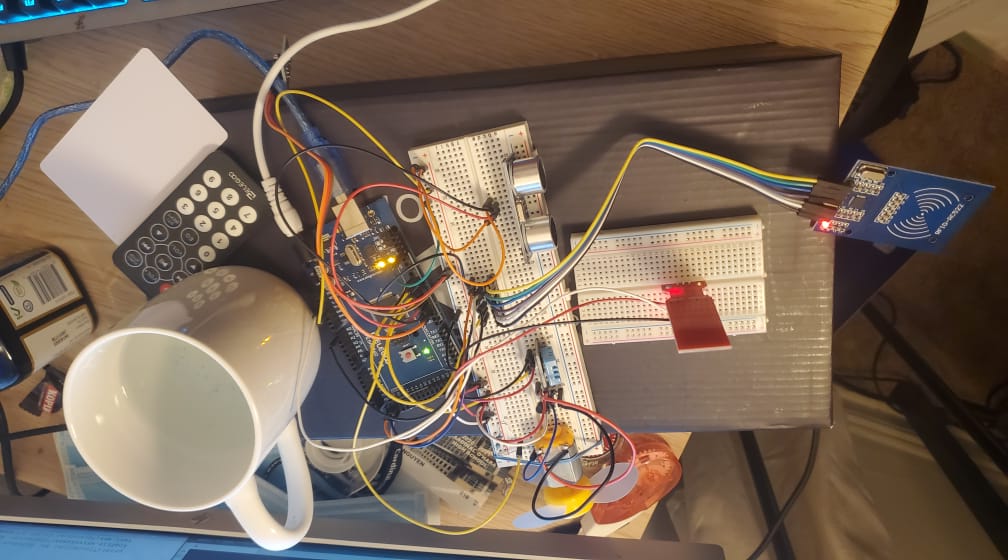
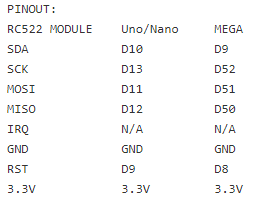


Figure 1. Vehicle Test-bed system

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:



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.

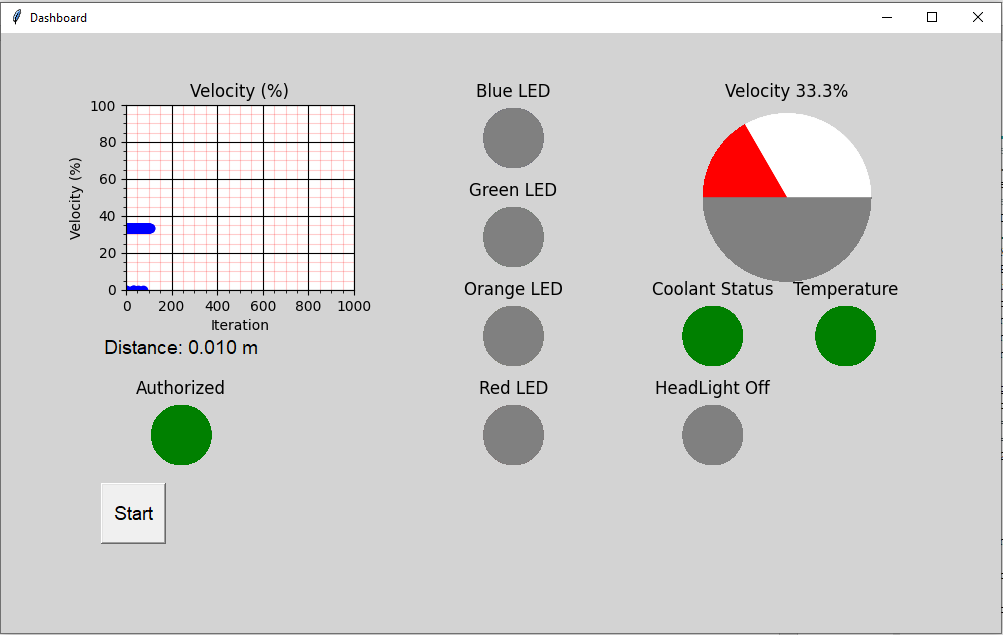


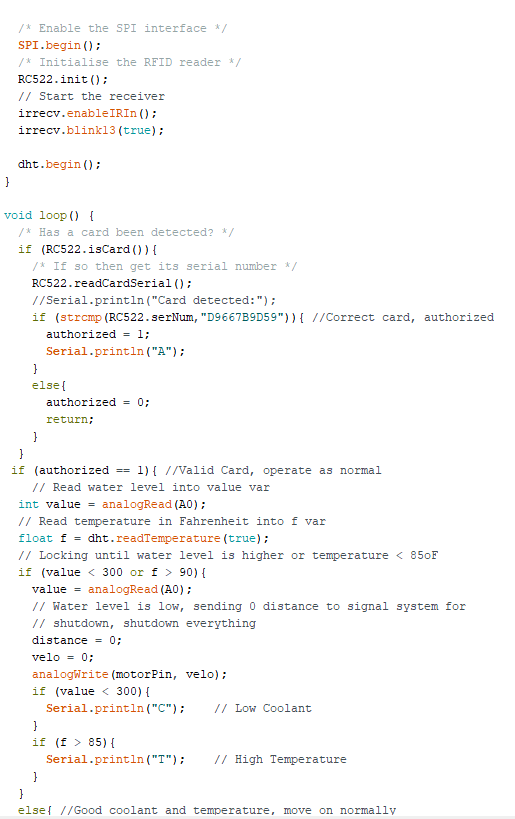
Figure 2. Vehicle Dashboard

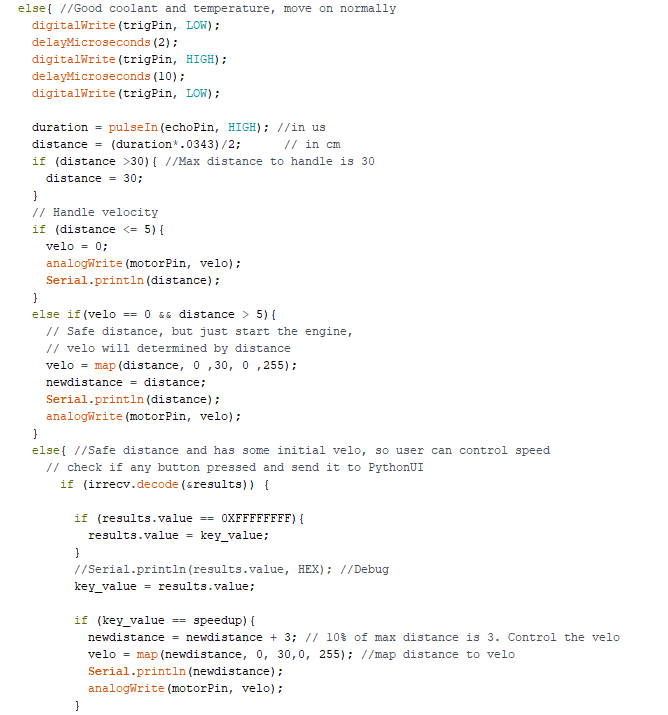
**Recommendation**

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**









**Python Script**

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 start

            TemperatureLED = [100,0,0] # order = off, normal, critical - off on start

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

            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, command = startCallBack)

start\_button.place(x=100, y=450)

root.mainloop()