Principles of Measurement & Instrumentation I

Laboratory

PHYS417

Laboratory Report

Abdullah Burkan BEREKETOĞLU

Experiment 6 - User Interfaces

Submitted By: Abdullah Burkan BEREKETOĞLU

Submitted to: Deniz Orhun BOZTEMUR & Enver BULUR

Student Id: 2355170

Experiment Date: 7.1.2022

Submission Date: 19.1.2022

OBJECTIVES

To design an interface from python to Arduino that we can control the machinery that is dealt with by receiving and delivering commands to Arduino and from that to the machine. Pyserial, Tkinter, serial libraries for Arduino and Python were used.

INTRODUCTION

In the experiment, we tried to conduct an experiment by directly giving commands from the GUI that we wrote by clicking to a digital button on the screen and adjusting the PWM which will adjust the duty cycle and give us the adjustability of the speed, brightness by current and voltage. These buttons most of the time in backend programming are called Widgets and they are the ones that we used. We also optimized the GUI's height and width.

EQUIPMENT

• Arduino

• LED

• Resistor

• PC

• DC Motor

• L298N Motor driver shield

PROCEDURE

1-) In the first part, we designed a GUI only to set an LED or a motor on anything electrical that you can connect to Arduino and deliver energy to work just by on and off. We also used code from Arduino since it delivers the operation that code wants in python to the machine that is connected, hence we need to make what we write is understandable in a sense from Arduino. Also before starting with the GUI we first tried to make our

code work in the Arduino to see if we can deliver the voltage LOW, HIGH clearly to the device that is observed. We were able to do these because of the arduino serialcommand library that works well together with tkinter an pyserial.

```
//For providing logic to L298 IC to choose the direction of the DC motor

void setup()
{
pinMode(pwm,OUTPUT) ;   //we have to set PWM pin as output
pinMode(in_1,OUTPUT) ;  //Logic pins are also set as output
pinMode(in_2,OUTPUT) ;
}

void loop()
{
//For Clock wise motion , in_1 = High , in_2 = Low

digitalWrite(in_1,HIGH) ;
digitalWrite(in_2,LOW) ;
analogWrite(pwm,255) ;

/*setting pwm of the motor to 255
we can change the speed of rotaion
by chaning pwm input but we are only
using arduino so we are using higest
value to driver the motor  */

//Clockwise for 3 secs
delay(3000) ;

//For brake
digitalWrite(in_1,HIGH) ;
digitalWrite(in_2,HIGH) ;
delay(1000) ;

//For Anti Clock-wise motion - IN_1 = LOW , IN_2 = HIGH
digitalWrite(in_1,LOW) ;
digitalWrite(in_2,HIGH) ;
delay(3000) ;

//For brake
digitalWrite(in_1,HIGH) ;
digitalWrite(in_2,HIGH) ;
delay(1000) ;
 }
```

Figure 1 Arduino LOW, HIGH command code

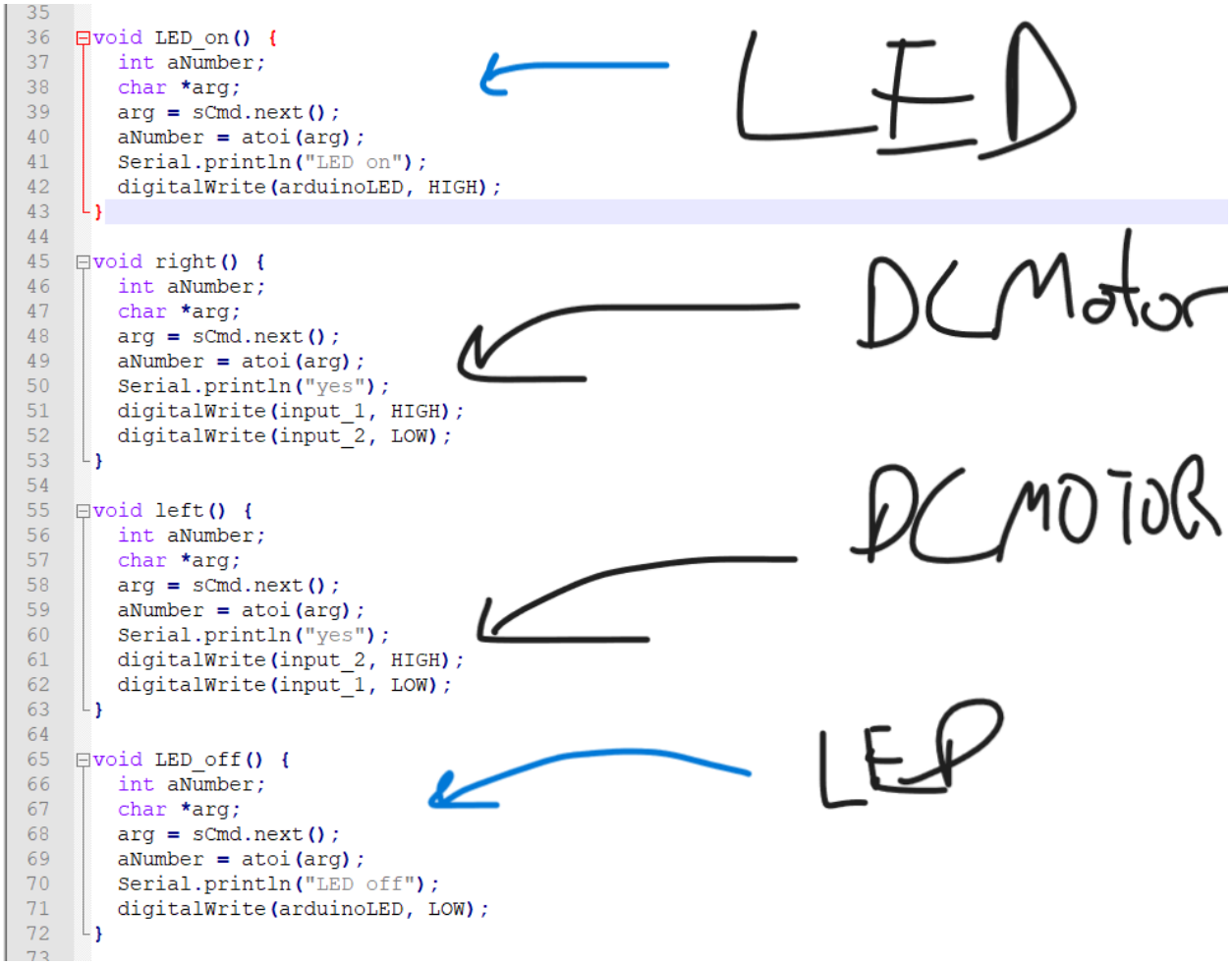
```
void setup() {
  pinMode(arduinoLED, OUTPUT);      // Configure the onboard LED for output
  digitalWrite(arduinoLED, LOW);    // default to LED off

  Serial.begin(9600);

  // Setup callbacks for SerialCommand commands
  sCmd.addCommand("ON",    LED_on);          // Turns LED on
  sCmd.addCommand("OFF",   LED_off);         // Turns LED off
  sCmd.addCommand("HELLO", sayHello);        // Echos the string argument back
  sCmd.addCommand("P",     processCommand);  // Converts two arguments to integers and echos them back
  sCmd.setDefaultHandler(unrecognized);      // Handler for command that isn't matched  (says "What?")
  Serial.println("Ready");
}

void loop() {
  sCmd.readSerial();     // We don't do much, just process serial commands
}


void LED_on() {
  int aNumber;
  char *arg;
  arg = sCmd.next();
  aNumber = atoi(arg);
  Serial.println("LED on");
  digitalWrite(arduinoLED, HIGH);
}

void LED_off() {
  int aNumber;
  char *arg;
  arg = sCmd.next();
  aNumber = atoi(arg);
  Serial.println("LED off");
  digitalWrite(arduinoLED, LOW);
}

void sayHello() {
  char *arg;
  arg = sCmd.next();    // Get the next argument from the SerialCommand object buffer
  if (arg != NULL) {    // As long as it existed, take it
```

Figure 2 Arduino to understand what comes from Python (for on and off)

Then in the second part of the first process, we added a PWM slider(scale) which gave us the ability to change the brightness of our light efficiently. We needed to make it from 0 to 255, we used 255 as the maximum because that is the maximum PWM register can get since it is 8 bits wide. So if we say analogWrite(255) which is what the scale gies at maximum, we will receive a 100% duty cycle. If 0 then 0% duty cycle. The Pin that we use for PWM is also important since Arduino's 5 and 6 pins will have 980 Hz which is the appropriate Hz for PWM but the rest of the pins are 490 Hz, which means the pin is important.

So to continue again with the second part, we can now say we saw different brightness levels of light and continued this in the second part of the experiment with the DC motor.

2-) In the second part of the experiment, which is the second process now, we applied what we did for the LED to switch the light on and off with the Left and Right functions. We then by the PWM scaler powered the DC motor and changed the rotation direction to Left or Right by the buttons we made in Tkinter GUI.




```
35
36  void LED_on() {
37    int aNumber;
38    char *arg;
39    arg = sCmd.next();
40    aNumber = atoi(arg);
41    Serial.println("LED on");
42    digitalWrite(arduinoLED, HIGH);
43  }
44
45  void right() {
46    int aNumber;
47    char *arg;
48    arg = sCmd.next();
49    aNumber = atoi(arg);
50    Serial.println("yes");
51    digitalWrite(input_1, HIGH);
52    digitalWrite(input_2, LOW);
53  }
54
55  void left() {
56    int aNumber;
57    char *arg;
58    arg = sCmd.next();
59    aNumber = atoi(arg);
60    Serial.println("yes");
61    digitalWrite(input_2, HIGH);
62    digitalWrite(input_1, LOW);
63  }
64
65  void LED_off() {
66    int aNumber;
67    char *arg;
68    arg = sCmd.next();
69    aNumber = atoi(arg);
70    Serial.println("LED off");
71    digitalWrite(arduinoLED, LOW);
72  }
```

Figure 3 Arduino to understand what comes from Python for all (DC Motor or LED)

Now since we discussed both of the parts of the experiment let's talk about the codes now.

In figure 3 we see 4 functions that take aNumber and *arg which come from the terminal directly from python output to Arduino input and converted to a value with atoi(). We can also see how to turn the way of DC Motor turns with the left and right functions which are also directly getting their input from python to Arduino to the motor. With digitalWrite we make it turn the other way.

```
  if (arg != NULL) {    // As long as it existed, take it
    Serial.print("Hello ");
    Serial.println(arg);
  }
  else {
    Serial.println("Hello, whoever you are");
  }
}


void processCommand() {
  int aNumber;
  char *arg;

  Serial.println("We're in processCommand");
  arg = sCmd.next();
  if (arg != NULL) {
    aNumber = atoi(arg);    // Converts a char string to an integer
    Serial.print("First argument was: ");
    Serial.println(aNumber);
    analogWrite(arduinoLED,aNumber);
  }
  else {
    Serial.println("No arguments");
  }

  arg = sCmd.next();
  if (arg != NULL) {
    aNumber = atol(arg);
    Serial.print("Second argument was: ");
    Serial.println(aNumber);
    analogWrite(arduinoLED,aNumber);
  }
  else {
    Serial.println("No second argument");
  }
}

// This gets set as the default handler, and gets called when no other command matches.
void unrecognized(const char *command) {
  Serial.println("What?");
}
```

Figure 4 Arduino code process command

Here we see the conditions to e able to pass our commands from serial monitor as the first task that is needed to be done.

Let’s show Python code parts then show what does it pop-up as a GUI.

```
import tkinter as tk
from tkinter import ttk
import serial
import time

# root window
root = tk.Tk()
root.geometry('350x275')
root.resizable(False, False)
root.title('LED Brightness Control')

# arduino
arduino = serial.Serial('COM4', 9600)

#columnconfig
root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=3)


# slider current value
current_value = tk.DoubleVar()



def get_current_value():
    return '{: .2f}'.format(current_value.get())


def slider_changed(event):
    value_label.configure(text=get_current_value())

def led_on():
    message = "ON"
    message_2 = "\r\n"
    arduino.write(message.encode()+message_2.encode())
    print(message.encode()+message_2.encode())

def led_off():
    message = "OFF"
```

Figure 5 First part of the python code

In this part one can see that we have the value get_current_value() function and led_on function to open the led, and so. Led_on sends bytes to arduino to make the command successfully processed.

```
def get_current_value():
    return '{: .2f}'.format(current_value.get())


def slider_changed(event):
    value_label.configure(text=get_current_value())

def led_on():
    message = "ON"
    message_2 = "\r\n"
    arduino.write(message.encode()+message_2.encode())
    print(message.encode()+message_2.encode())

def led_off():
    message = "OFF"
    message_2 = "\r\n"
    arduino.write(message.encode()+message_2.encode())
    print(message.encode()+message_2.encode())

def send_pwm():
    messg = "P "
    message = str(current_value.get())
    message_2 = "\r\n"
    arduino.write(messg.encode()+message.encode() + message_2.encode())
    print(messg.encode()+message.encode() + message_2.encode())

def right():
    message = "R"
    message_2 = "\r\n"
    arduino.write(message.encode()+message_2.encode())
    print(message.encode() + message_2.encode())

def left():
    message = "L"
    message_2 = "\r\n"
    arduino.write(message.encode()+message_2.encode())
```

Figure 5 second part of the python code

Here this part is more detailed with pwm function, right and left function in which right and left as led_on, led_off sends bytes to make the specific process to be run for the

body connected. So if motor is started to run by send_pwm, then when you make use of left motor will rotate to left for right the opposite.

```
    print(message.encode() + message_2.encode())

def left():
    message = "L"
    message_2 = "\r\n"
    arduino.write(message.encode()+message_2.encode())
    print(message.encode() + message_2.encode())


# label for the slider
slider_label = ttk.Label(
    root,
    text='PWM:'
)

slider_label.grid(
    column=0,
    row=0,
    sticky='w'
)

#  slider
slider = tk.Scale(
    root,
    from_=0,
    to=255,
    orient='horizontal',  # vertical
    command=slider_changed,
    variable=current_value
)

slider.grid(
    column=1,
    row=0,
    sticky='we'
)

# Button 1
button_1 = ttk.Button(
```

Figure 6 Graphical scaling component build code

```
# Button 1
button_1 = ttk.Button(
    root,
    text='SEND PWM TO ARDUINO',
    command=send_pwm)

button_1.grid(
    row=1,
    columnspan=2,
    sticky='n',
    ipadx=5,
    ipady=5
)

# Button 2
button_2 = ttk.Button(
    root,
    text='LED ON',
    command = led_on)

button_2.grid(
    row=2,
    columnspan=2,
    sticky='n',
    ipadx=5,
    ipady=5
)

# Button 3
button_3 = ttk.Button(
    root,
    text='LED OFF',
    command = led_off)

button_3.grid(
    row=3,
    columnspan=2,
    sticky='n',
```

Figure 7 Python GUI buttons with specific command and texts.

```
    ipadx=5,
    ipady=5
)


# Button 4
button_4 = ttk.Button(
    root,
    text='RIGHT',
    command=right)

button_4.grid(
    row=4,
    columnspan=2,
    sticky='n',
    ipadx=5,
    ipady=5
)

# Button 5
button_5 = ttk.Button(
    root,
    text='LEFT',
    command=left)

button_5.grid(
    row=5,
    columnspan=2,
    sticky='n',
    ipadx=5,
    ipady=5
)

# current value label
current_value_label = ttk.Label(
    root,
    text='PWM Value:'
)
```

Figure 8 Again another button setting with python tkinter GUI library

```
    columnspan=2,
    sticky='n',
    ipadx=5,
    ipady=5
)

# current value label
current_value_label = ttk.Label(
    root,
    text='PWM Value:'
)

current_value_label.grid(
    row=6,
    columnspan=2,
    sticky='n',
    ipadx=10,
    ipady=10
)

# value label
value_label = ttk.Label(
    root,
    text=get_current_value()
)
value_label.grid(
    row=7,
    columnspan=2,
    sticky='n'
)


root.mainloop()
```

Figure 9 Python label with ttk version (tk and ttk differ with graphics)

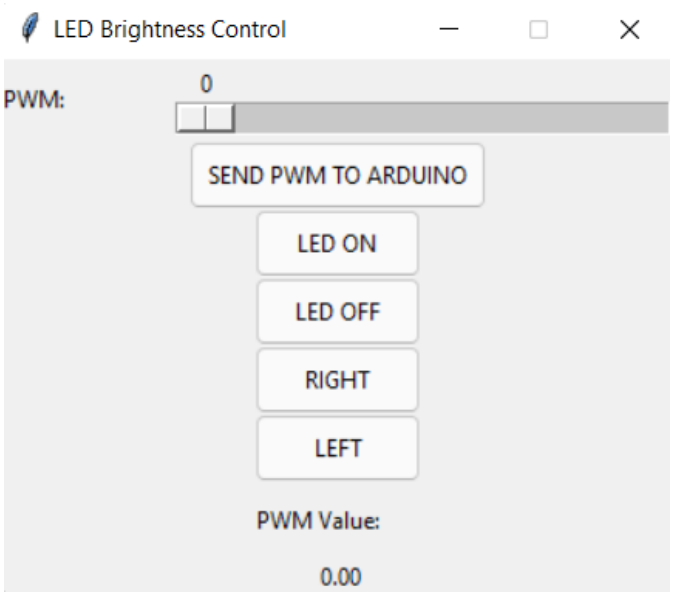


Figure 10 The end result to Control LED Brightness, Motor Speed, ON-OFF, Rotation direction with PWM Value visible at the label.

CALCULATIONS

Since if the circuit and the program are not read and used before the lab, the equipment such as Arduino or LED, DC Motor can burn and also if there is a much more expensive circuitry in the hands of the user that may get damaged if there is no preventive system that protects it from changing currents, voltage, heat and so. Calibrating those parts especially for the User Interface experiment is really essential so the one who is doing the experiment (not only this) needs to be prepared with pre-calculated values to not make any hazardous or to not damage valuable property.

DISCUSSION & COMMENTS

In this experiment, we needed to learn Tkinter and Python GUI building basics beforehand, also how to connect them to Arduino, thanks to the bits of help of TA Deniz Boztemur and our friend Fatih DUMAN we were able to finish the logical part smoothly, the latter part was to write the code which I believe I did nicely after learning the logic behind. With the systems that we use now, it is clear to me that in the future I may be able to design a small system with Communication Protocols: I^2C & SPI Compatible Sensors to visualize the information we get from the sensors we designed.