ROBOTIC ARM

Prototype of Industrial Robotic Arm Systems

Project: Robotic Arm
Project Members: Deval Malhotra & Saif Chhipa
Submission Date: November 30, 2024

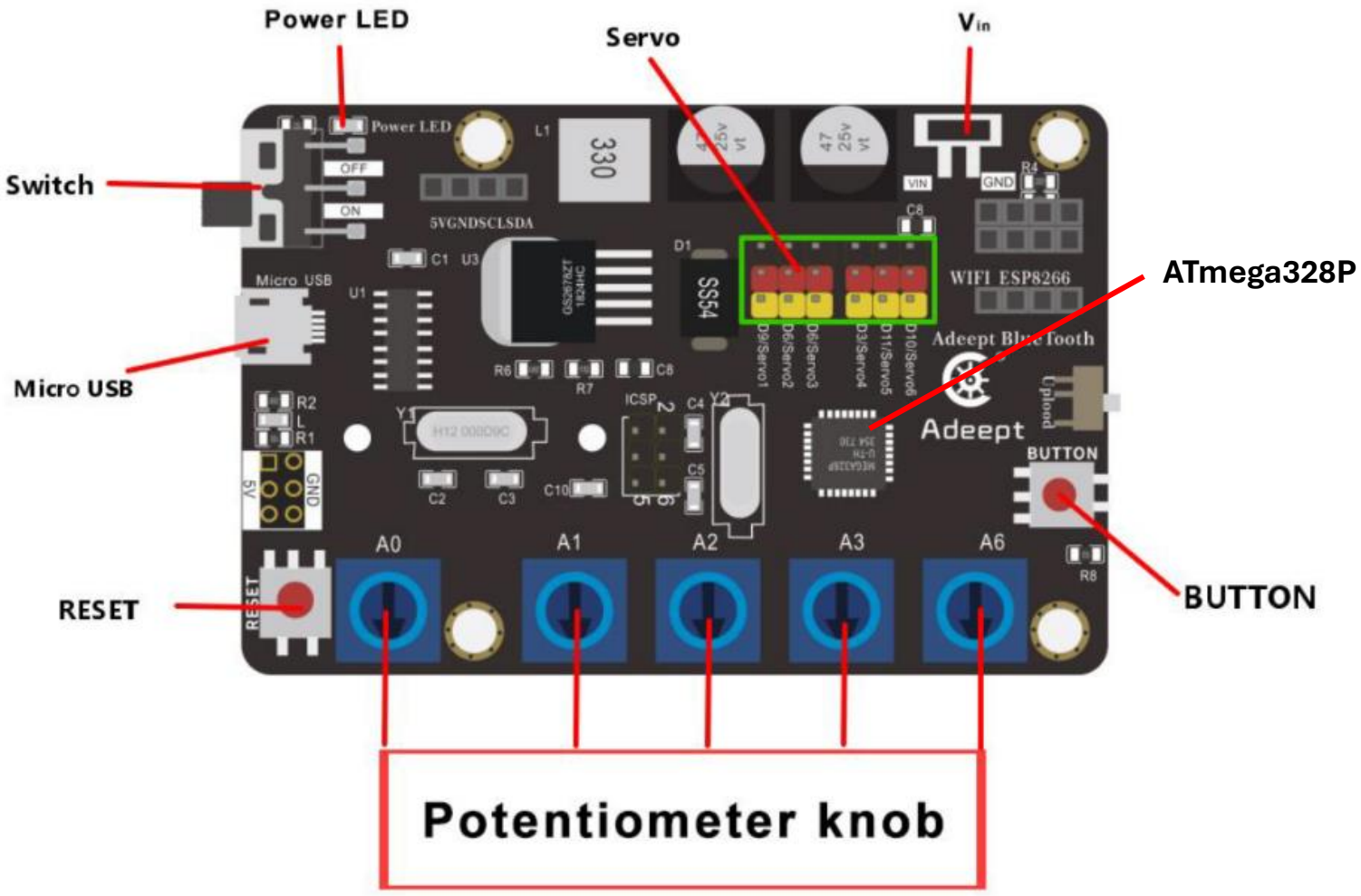
British Columbia Institute of Technology

Objective: Construction of a robotic arm, capable of lifting lightweight objects.

Features: Lifts small, lightweight objects. It can be controlled through Potentiometers, a Web Interface or a Graphical Dashboard. Five servo motors provide extensive flexibility and control over the robotic arm, capable of grabbing, lifting, transporting and dropping small objects. Designed to prototype heavy-duty robotic arms used in industrial control and manufacturing. An OLED screen is used to display messages to the user. The Robotic Arm can also learn a control action and perform the action in a "loop".

Project Goals & Rubric Alignment:

- **Project Design & Creativity:** The project is designed to prototype the much larger, and robust Industrial Robotic Arm Systems. The robotic arm runs through a customized drive board capable of supplying 24 V_{DC} to the components. The robotic arm features 5 servo motors & hence 5 flexible ways to move the arm. The base of the arm also features a Square Bearing Turntable for a 0° to 180° movement of the arm itself. The arm will have a claw capable of grabbing and dropping lightweight objects.
- **Use of MCU and Sensors:** The customized Arm Drive Board is similar to Arduino Uno R3. The Arm Drive Board is mainly composed of a Microcontroller Unit (MCU), a universal input/output interface, etc.



- **Power LED:** Power LED is used to indicate the power status of the system. The LED is on, indicating that the system is powered on and ready to run; the LED is off, indicating that the system is not powered on.
- **Servo:** It is the pin interface of Servo Motors.
- **V_{in} (6-24V):** It is the pin interface for external power supply. A 6-24V external power supply should be used to power the development board.
- **RESET:** Restarting the development board.
- **Switch:** When using V_{in} (6-24V) as an external power supply, Switch can turn the development board ON and OFF.
- **Micro USB:** It is used to connect the micro connector of the micro USB data cable, and the USB connector of the micro USB data cable is connected to the USB interface of the computer, uploading program and serial monitoring between the development board and the computer.
- **Potentiometer Knob:** Potentiometer knob has five buttons: A0, A1, A2, A3, and A6. By rotating these buttons, you can control the movement of the Robotic Arm.
- **ATmega328P:** The ATmega328P is a popular 8-bit microcontroller from Atmel (now part of Microchip Technology).
    - **Architecture:** The ATmega328P uses the AVR architecture, which is an 8-bit RISC architecture designed for high performance with low power consumption
    - **Clock Speed:** The ATmega328P can run at a maximum clock speed of 20 MHz.
    - **Memory:**
        - **32 KB of flash memory:** This is used for storing the program code. Of the 32 KB, 0.5 KB is reserved for the bootloader.
        - **2 KB of SRAM:** Used for runtime data storage while the program is running.
        - **1 KB of EEPROM:** Used for non-volatile data storage, meaning data persists even when the power is turned off.
    - **Input/Output Pins:** The ATmega328P has 23 general-purpose I/O pins in total, but some of these pins can serve multiple functions like digital I/O, analog input, PWM, or communication interfaces (SPI, I2C, UART). It also has 6 analog inputs.
    - **Peripherals:** The ATmega328P includes a variety of integrated peripherals, such as:
        - 3 timers/counters for generating time delays or PWM signals.
        - UART for serial communication.
        - SPI (Serial Peripheral Interface) for communication with other devices like sensors, displays, etc.
        - I2C for communication with I2C-compatible devices (although, in the Arduino ecosystem, I2C support is usually via libraries).
        - It also has PWM outputs available on certain pins.
    - **Low Power:** The ATmega328P supports several power-saving modes (such as sleep modes), making it suitable for energy-efficient applications, especially in battery-powered projects.

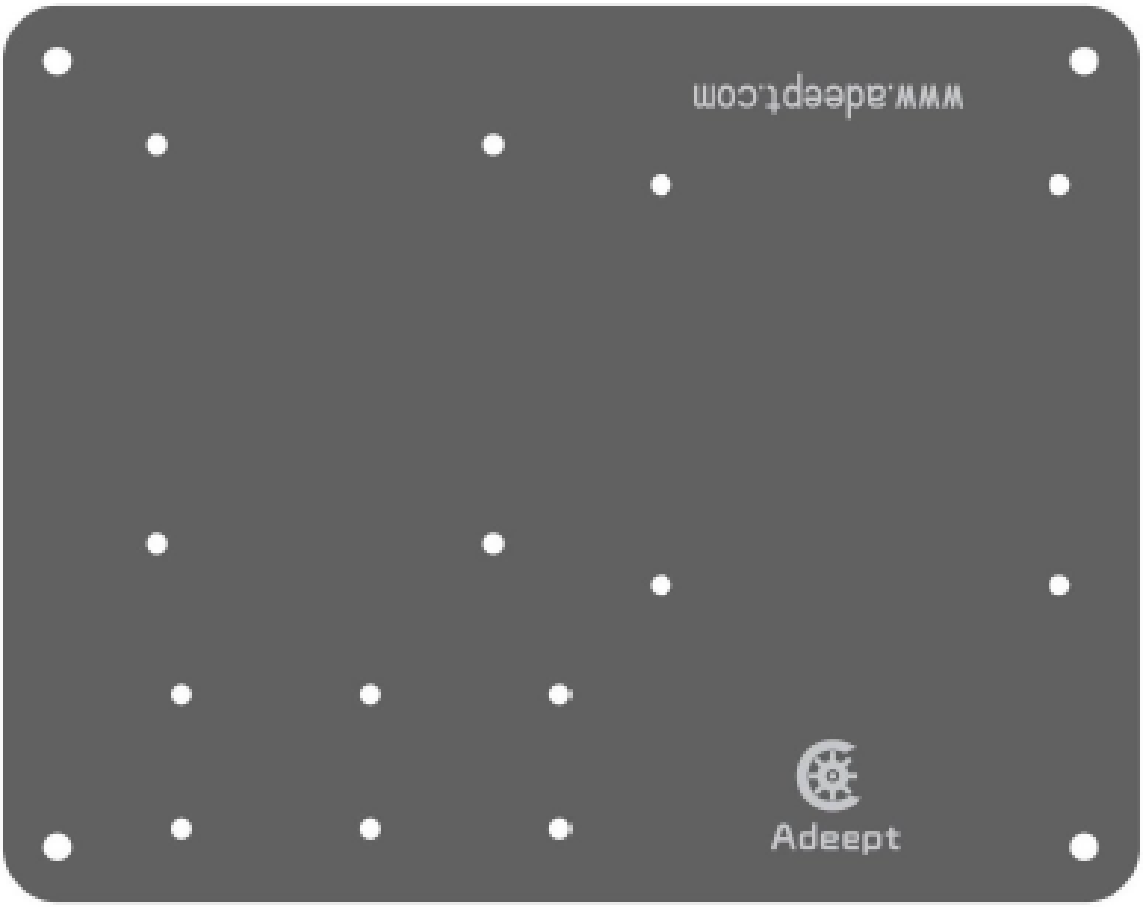
- **Code Explanation & Ownership:** All the control codes are explained in the **Control Code** section.
- **Functionality & Testing:** The project was thoroughly tested and the components worked as expected.
- **Teamwork & Contribution:** The project was divided into the Hardware & Software phases. The hardware assembly and construction were performed by Deval Malhotra, whereas the code debugging and testing were performed by Saif Chhipa.

Design & Development

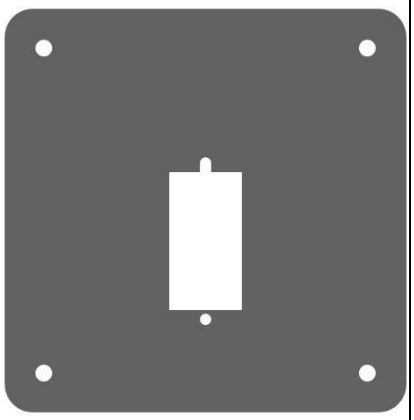
- **Team Member Contributions:**
  - **Deval Malhotra:** Responsible for the assembly, testing and troubleshooting of the hardware components, and documentation.
  - **Saif Chhipa:** Responsible for the testing and troubleshooting of the software components, including the coding and Graphical Control dashboard.

- **Circuit Design and Setup:**
  - **Component List:**

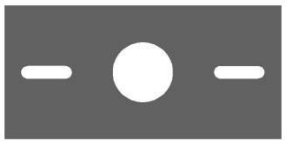
1. Acrylic Plates



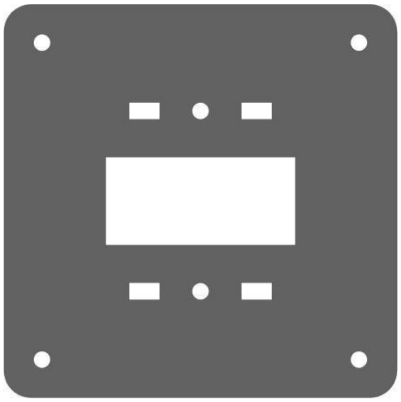
A01 1pcs



A02
1pcs



A03
1pcs



A04
1pcs

A05
1pcs

A06
1pcs

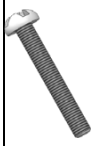
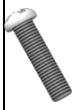
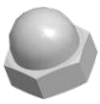
A07
1pcs

A08
1pcs

A09
3pcs

A10
1pcs

2. Machinery Parts

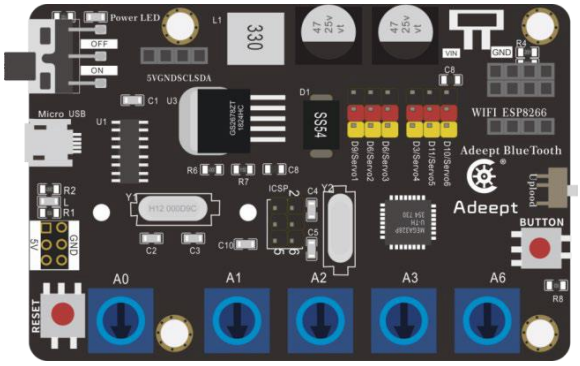
M2 Nut  ×11	M3 Nut  ×10	 M3 Lock Nut ×3	M1.4*6 Self-Tapping Screw  ×2	M2*14  Screw ×7
M2*10  Screw ×4	M2*5  Screw ×4	M2.5*7  Screw ×5	M3*8  Screw ×24	M3*12  Screw ×5
M3*18 Screw  ×1	M3*10  Countersunk Head Screw ×9	M2*11  Copper Standoff ×2	M3*8 Copper Standoff  ×1	M3*15  Nylon Standoff ×4
M3*30  Nylon Standoff ×5	M3*40  Nylon Standoff ×2	Sucking Disc Component  Cap Nut ×4  Sucking Disc ×4		

Square Bearing Turntable

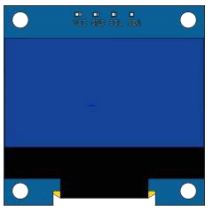
×1

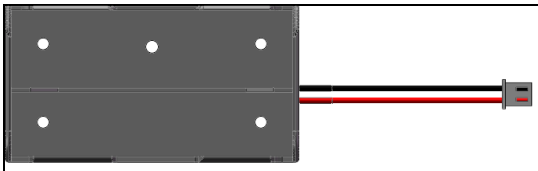
3. Electronic Parts:

Arm Drive Board ×1

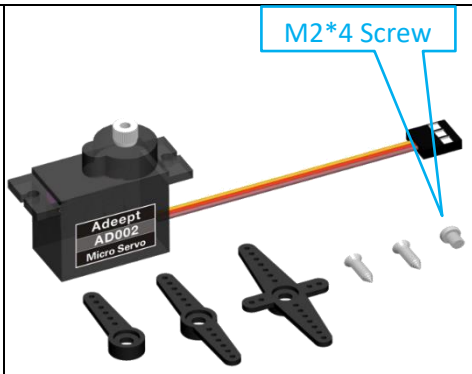


OLED ×1



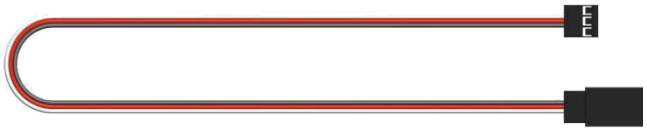


18650x2 Battery Holder ×1

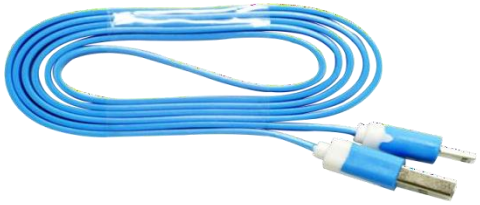


Servo ×5

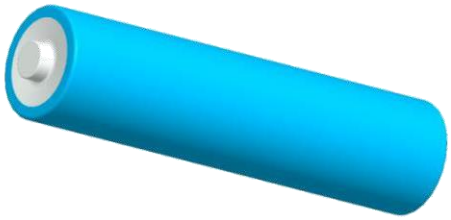
Servo Extension Cable ×1



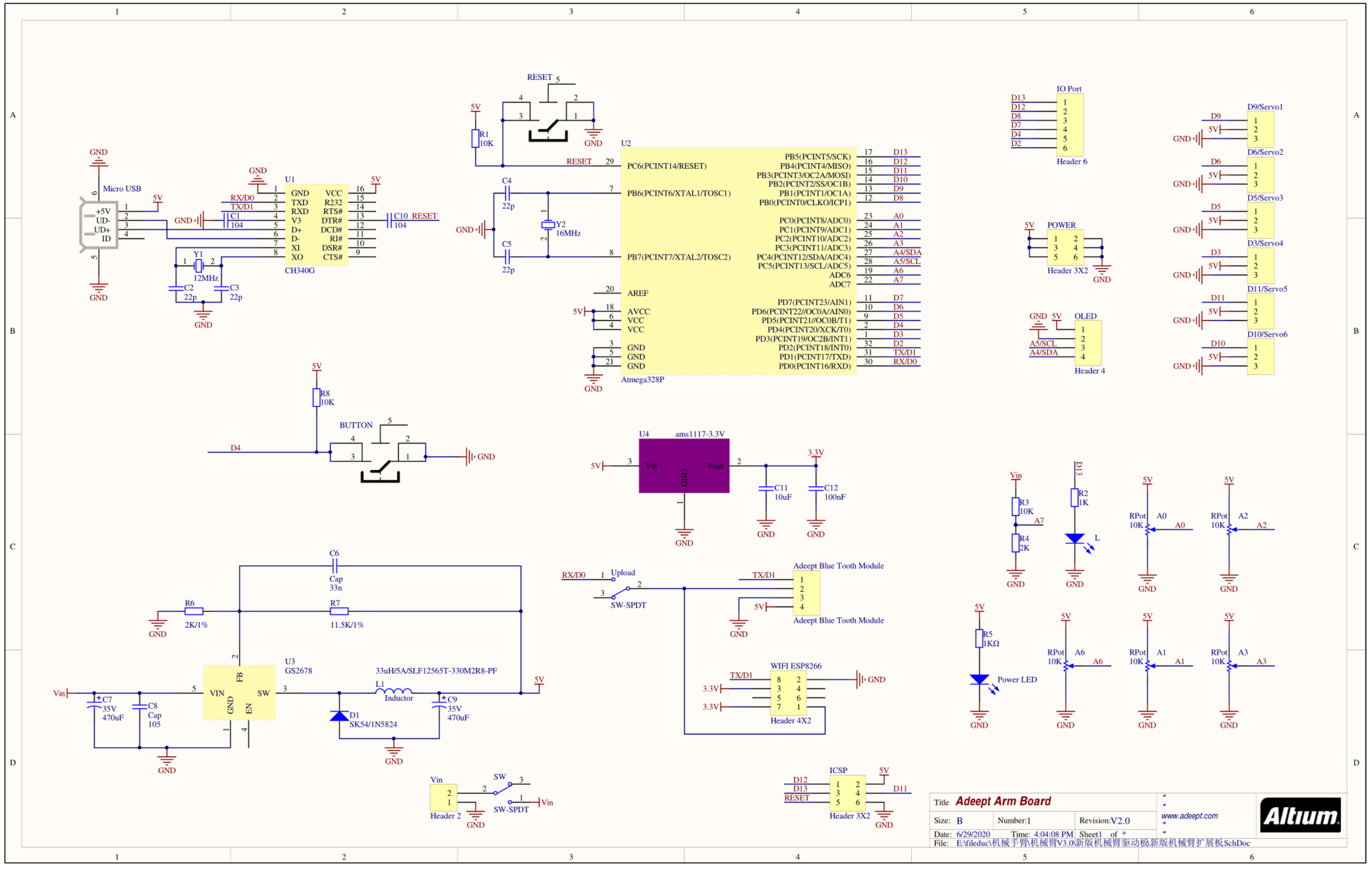
USB Cable ×1



18650 Battery ×2

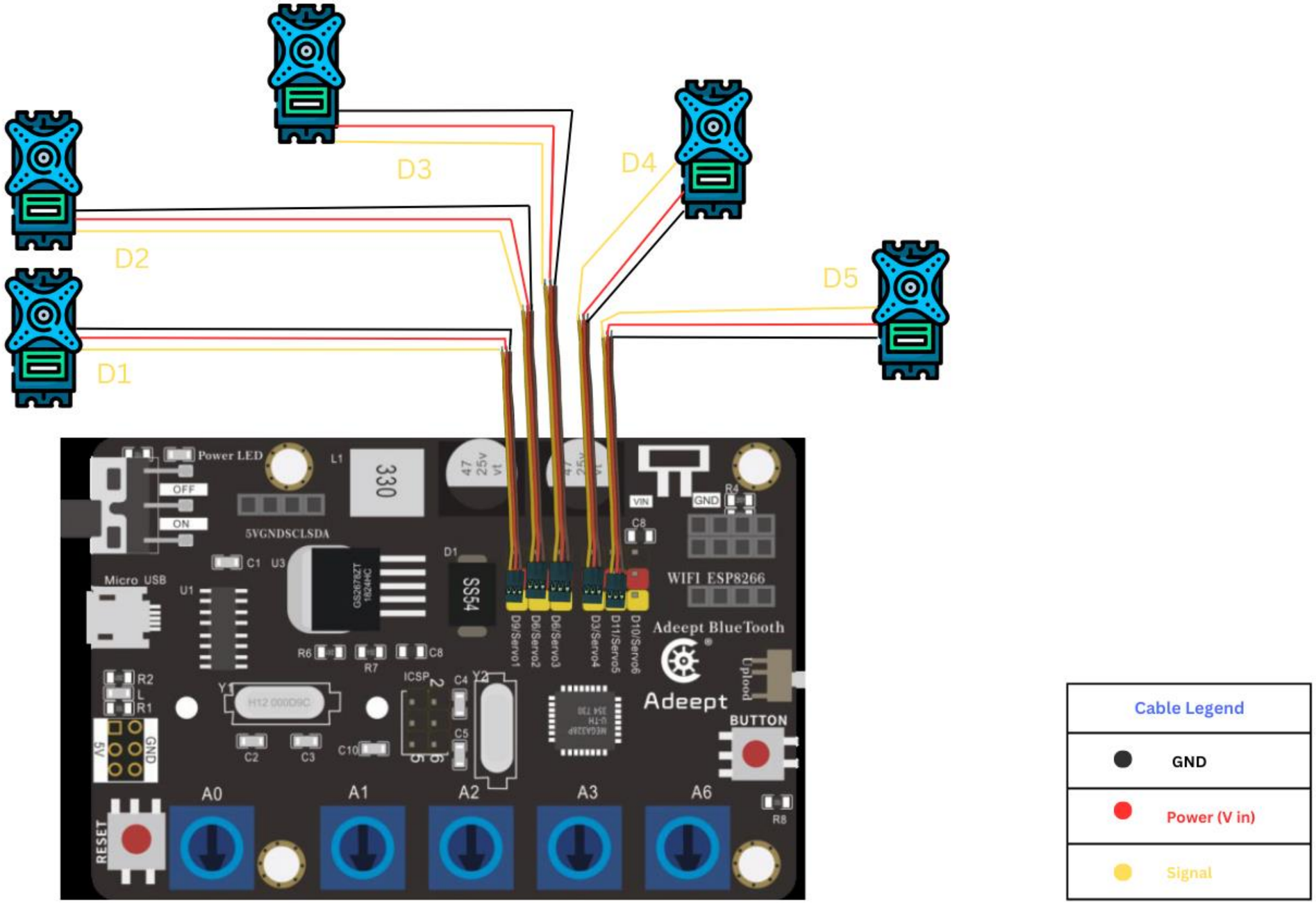


Schematic Diagram - 1:

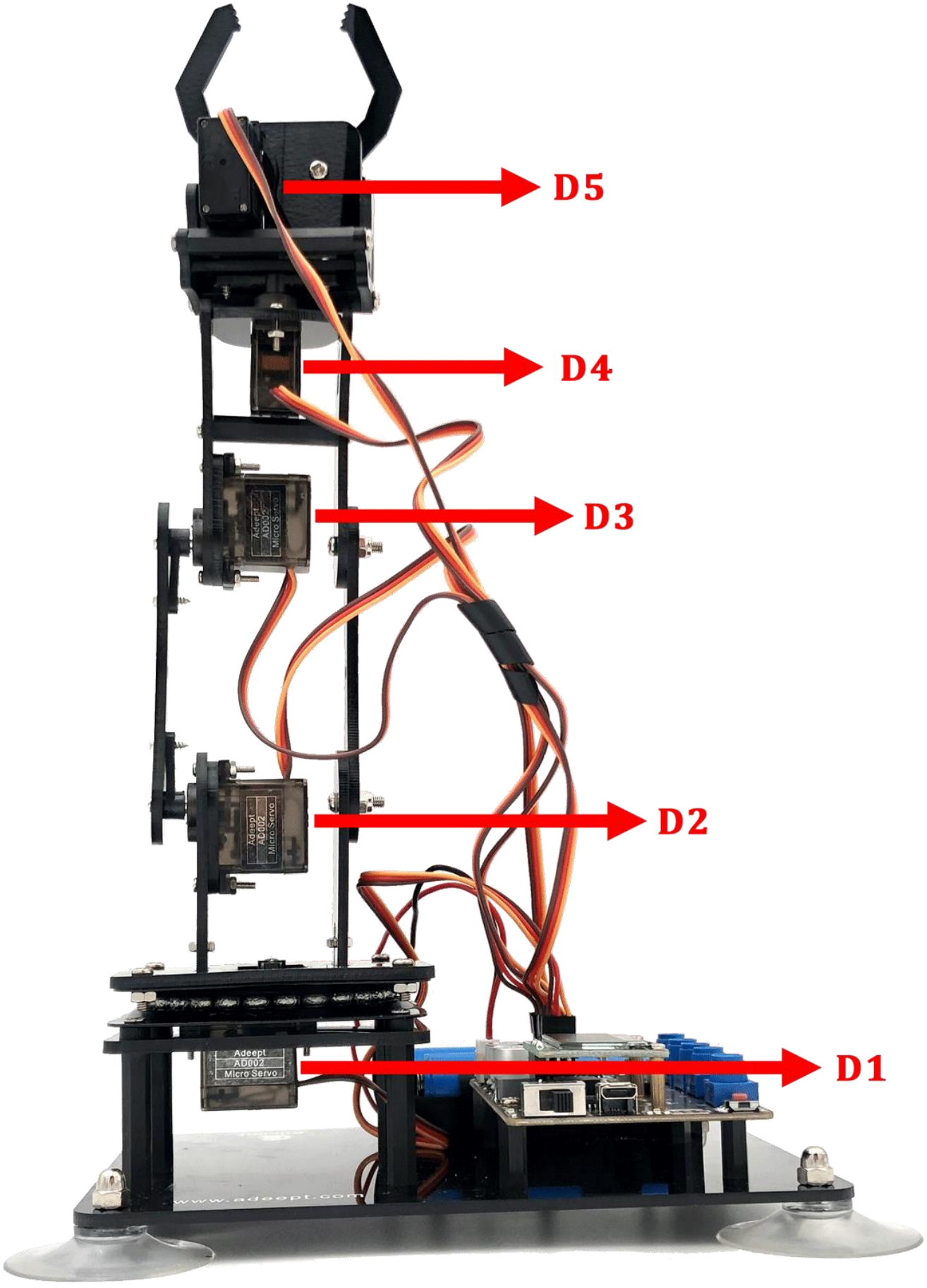


Title	**Adeept Arm Board**		
Size: B	Number:1	Revision:V2.0	www.adeept.com
Date: 6/29/2020	Time: 4:04:08 PM	Sheet1 of *	
File: E:\fileduc\机械手臂\机械臂V3.0\新版机械臂驱动板\新版机械臂扩展板.SchDoc			

- **Schematic Diagram - 2:**



Cable Legend
GND
Power (V in)
Signal


D5
D4
D3
D2
D1
Adeept
AD002
Micro Servo
www.adeept.com

- **Circuit Description:**
  - **Servo1 connected to Pin D1:** Servo1 is connected to the digital pin D1. Servo1 is attached at the base of the Robotic Arm and will provide 0° to 180° movement to move the entire robotic arm.
  - **Servo2 connected to Pin D2:** Servo2 is connected to the digital pin D2. Servo2 provides 0° to 180° UP-DOWN movements to the Robotic Arm.
  - **Servo3 connected to Pin D3:** Servo3 is connected to the digital pin D3. Servo3 provides further flexibility via 0° to 180° movements during the UP-DOWN motion.
  - **Servo4 connected to Pin D4:** Servo4 is connected to the digital pin D4. Servo4 provides 0° to 180° movement to adjust the claw angle of the Robotic Arm.
  - **Servo5 connected to Pin D5:** Servo5 is connected to the digital pin D5. Servo5 provides 35° to 90° motion for the protraction and retraction of the claw.

❖ Control Code:

The program operates with three distinct scripts: **block_py.ino**, **OLED.ino**, and **servosGUI.py**, each responsible for a specific task in the robotic arm system.

- **block_py.ino:** This code is embedded into the microcontroller. It handles various components (peripherals) of the driver board, including the OLED display, servo motors, and IR receiver.
  It achieves this by continuously receiving and processing commands from the Python GUI via serial communication. The commands are serialized into JSON format, which is then parsed (reserialized) by the **my_cmp** function to determine the appropriate action.

- **servosGUI.py:** The Python script runs on the computer and provides a graphical user interface (GUI) for controlling the robotic arm. Once the user selects the appropriate COM port, the Python GUI serializes commands into **JSON format** and sends them to the Arduino over the selected serial connection. Serialization here refers to converting the commands (including parameters like servo positions or IR data) into a structured format that can be transmitted over serial communication.
  The embedded code in the driver board (block_py.ino) continuously listens for incoming **JSON** data from the Python GUI. A 100-byte buffer is allocated in the driver board's memory to temporarily store the incoming data. The data is then parsed to extract keys and values, which represent specific commands and parameters. For example, a JSON object may look like this: { "servo": "position A", "position B" }, which is similar to how dictionaries are used in Python.
  Once the data is parsed, the **strcmp function** matches the extracted keys with predefined actions. These actions can include:
    - Displaying messages on the OLED screen
    - Processing data from the IR sensor

- Moving the servo to a specified position, thereby controlling the robotic arm's movement.

After each command is processed, the buffered data is cleared, preventing data overflow and ensuring the system continues to operate smoothly. This mechanism helps maintain stability by ensuring that the buffer doesn't exceed its allocated size, which would otherwise cause the program to malfunction.

To carry out this task effectively and efficiently, libraries such as **ArduinoJson.h** and **Servo.h** were included in the embedded code (block_py). The JSON library facilitates the deserialization of the received JSON data, making it easier to extract keys and their associated values to perform actions, such as changing the servo's position.

The Python script plays a crucial role in collecting the servo's position through a Graphical User Interface (GUI) and then sending the collected data to the board using the **Adeept** Python library. This extensive library, provided by the manufacturer, allows for comprehensive control of servos, sensors, and various other hardware components. It enables easy serialization of data in real-time, which is highly effective for moving the arm via the GUI.

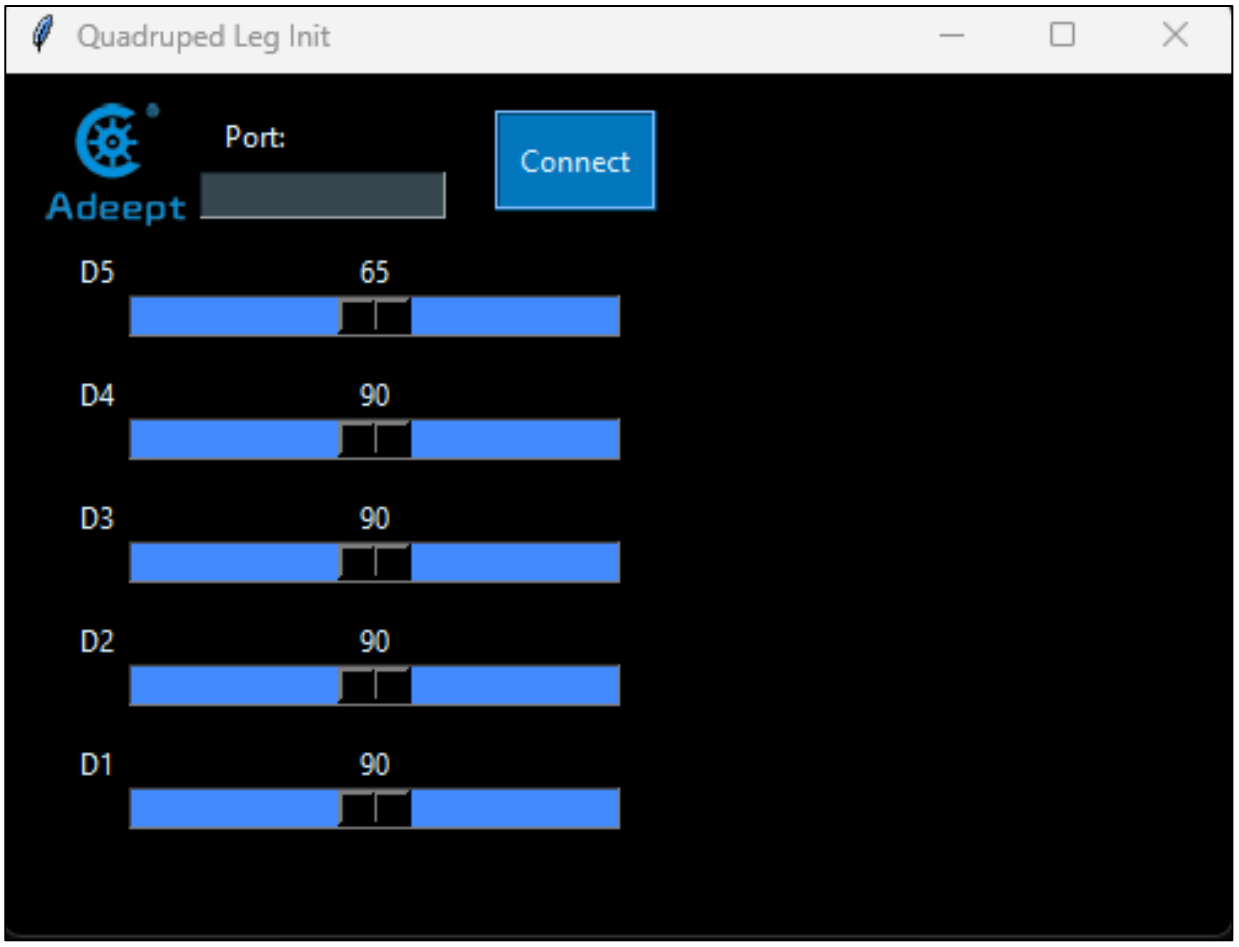
The Python script "servoGUI" includes several libraries, each serving different purposes. For example, the "**socket**" library is used for configuring web sockets, while "**threading**" allows a function to run multiple times before waiting to complete it once. The "**Tkinter**" module is essential for providing the **GUI** and collecting user input. The "**JSON**" library is utilized to write JSON data into a .json file (note that this is different from serializing JSON data to the board). Additionally, the "**os**" and "**sys**" libraries are used for system configurations, such as locating and writing files.

In this project, our primary focus is on libraries like **Tkinter** for the GUI, **Adeept** for communication, and **time** for delays.

- **OLED.ino:** The ***setup()* function** sets the display colour of the font to white using *display.setTextColor(WHITE)*. In the ***loop()* function**, *display.setTextSize(2)* sets the display font size to 2. The *display.setCursor(x,y)* sets the position of the text displayed on the OLED screen, and *display.println("MESSAGE")* prints out the message that needs to be displayed on the OLED screen.

❖ Dashboard Design:

- **Dashboard Layout:**
  - **Port:** Inputting the desired COM port.
  - **Connect:** Connecting to the inputted COM port.
  - **D1:** Controlling the D1 pin (Servo1) and hence controlling the movement of the base of the Robotic Arm.
  - **D2:** Controlling the D2 pin (Servo2) and hence controlling the UP-DOWN movement of the Robotic Arm.
  - **D3:** Controlling the D3 pin (Servo3) and hence providing flexibility in the UP-DOWN movement of the Robotic Arm.
  - **D4:** Controlling the D4 pin (Servo4) and hence controlling the claw angle of the Robotic Arm.
  - **D5:** Controlling the D5 pin (Servo5) and hence controlling the protraction & retraction of the Robotic Arm Claw.
  - **Buttons:** The black buttons act similarly to potentiometer knobs and can be slid left or right to control the movement of the servo motor. The number at the top of the buttons represents the current angle of the respective servo motor.
- **User Interaction:** After inputting and connecting to the desired COM port that the Robotic Arm is connected to, the user can slide the black buttons left or right to control the respective movement of the Robotic Arm. The number at the top of the black buttons represents the current angle of the respective servo motor.



Testing and Debugging:

- **Testing Process:** The Robotic Arm was controlled through the GUI dashboard, and the tester tried to lift a small, empty plastic box.
- **Results:** The Robotic Arm Claw kept getting stuck at the 90° angle, thus rendering it useless to grab any objects.
- **Issues & Fixes:** The tester performed a troubleshooting of the software code and discovered no errors. The tester performed hardware troubleshooting, and the source of the error was found to be a combination of a faulty servo motor (Servo5) and a non-linear rocker arm, attached to Servo5. The servo motor attached at the Servo5 position was replaced and a linear rocker arm was attached to the new Servo5. The Robotic Arm Claw was tested again and worked as expected, eliminating the previously encountered errors.

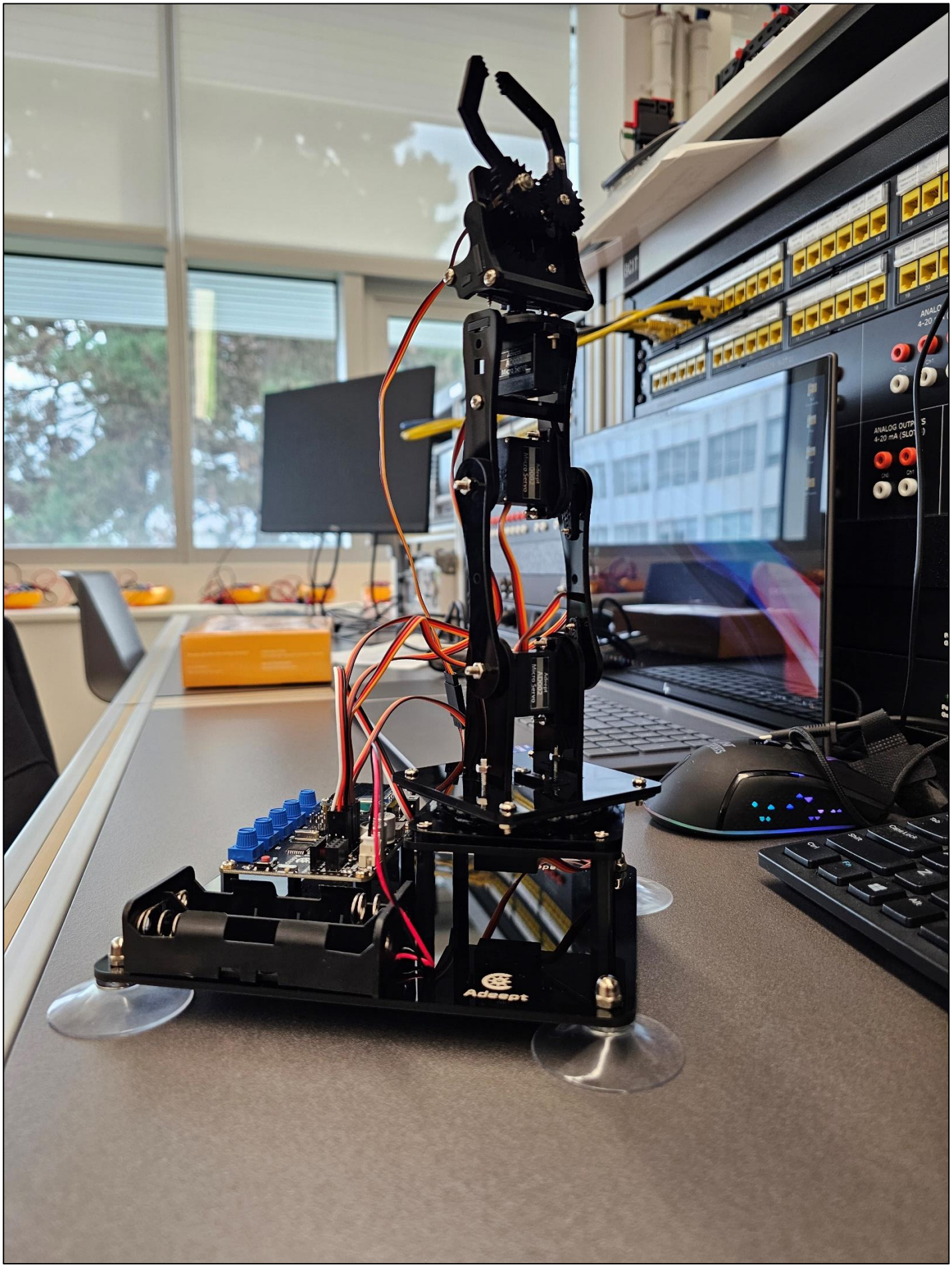
Linear Rocker Arm

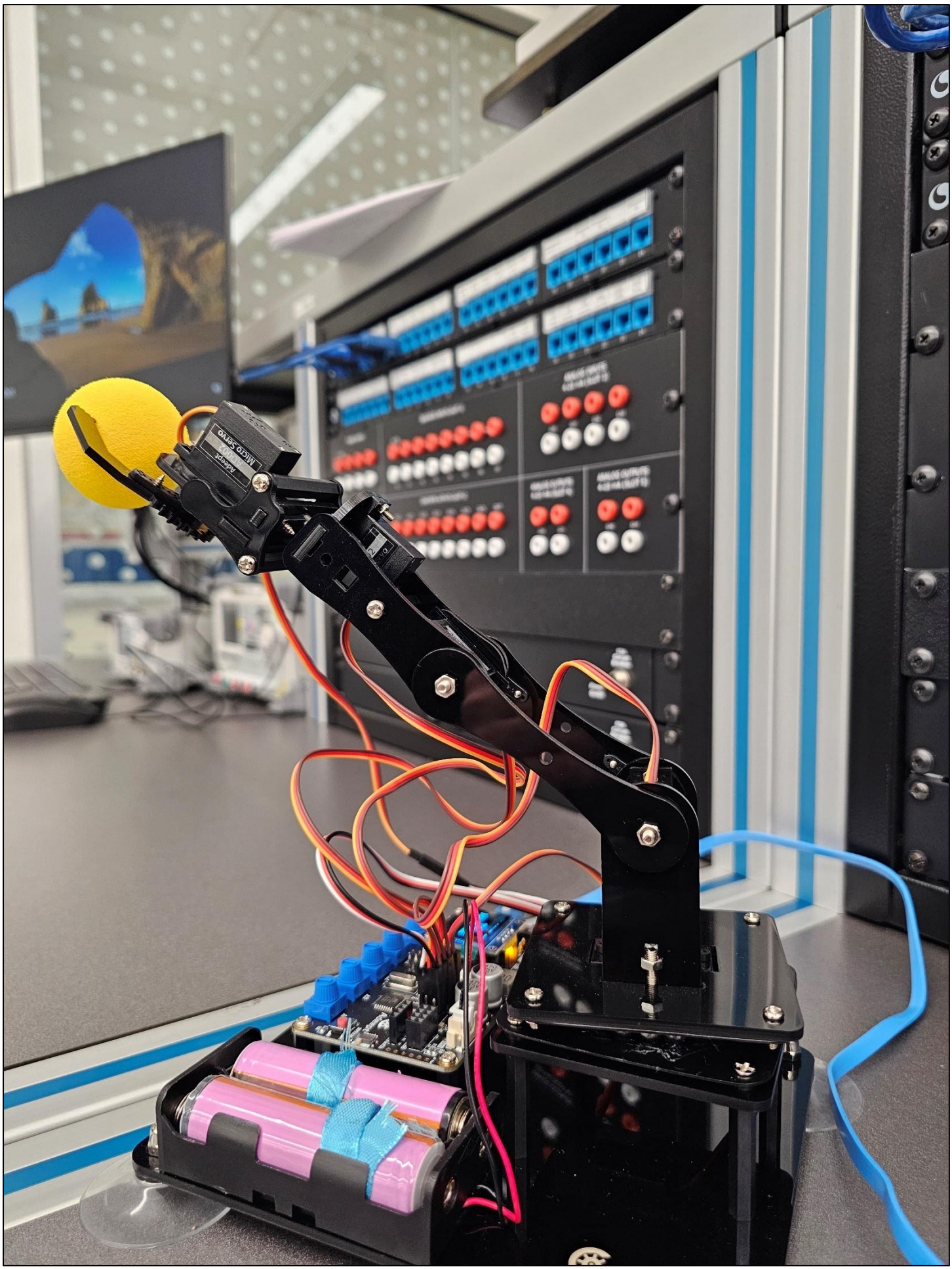
Non-Linear Rocker Arm (Source of Error)

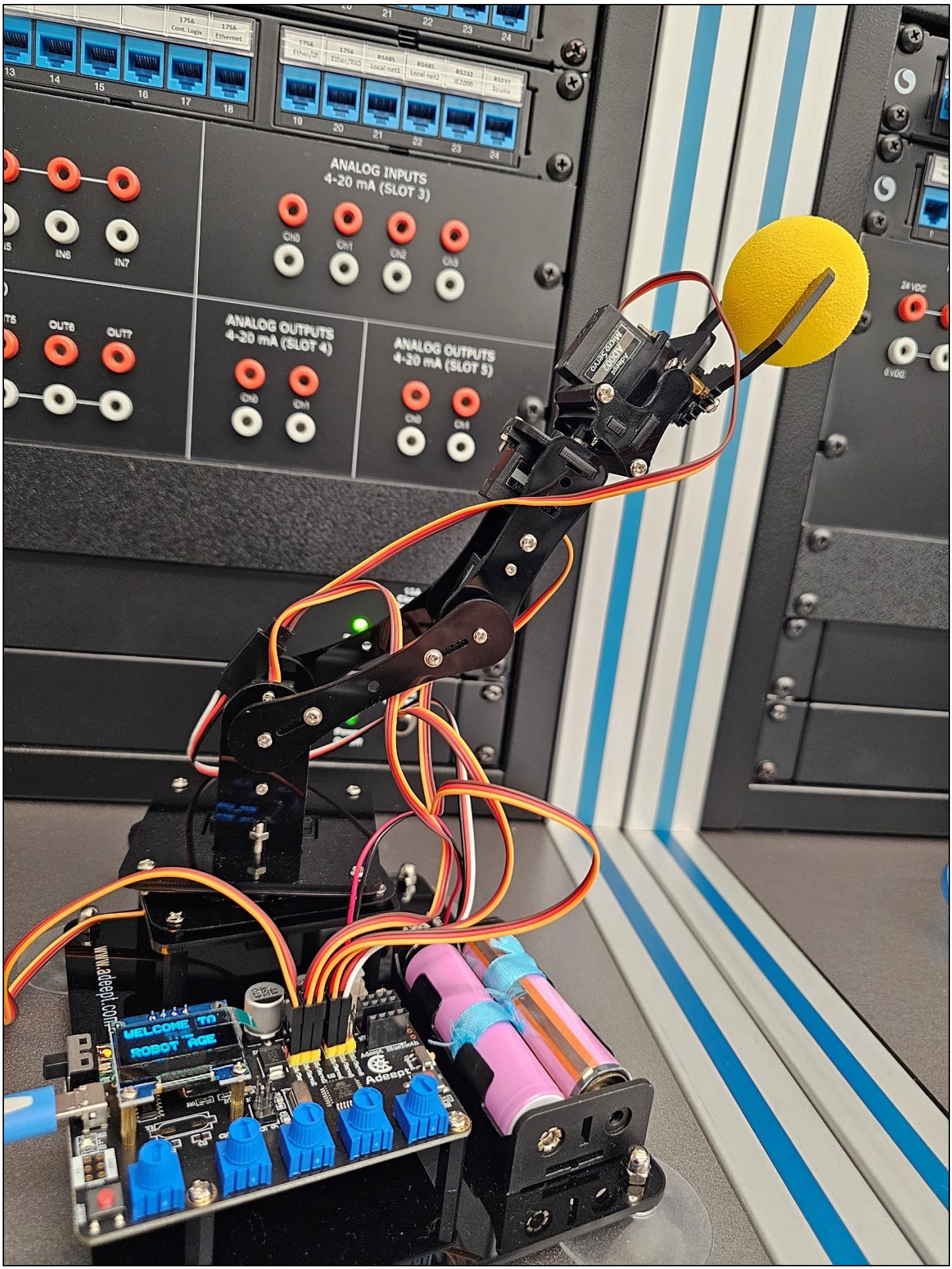
Final Outcomes:

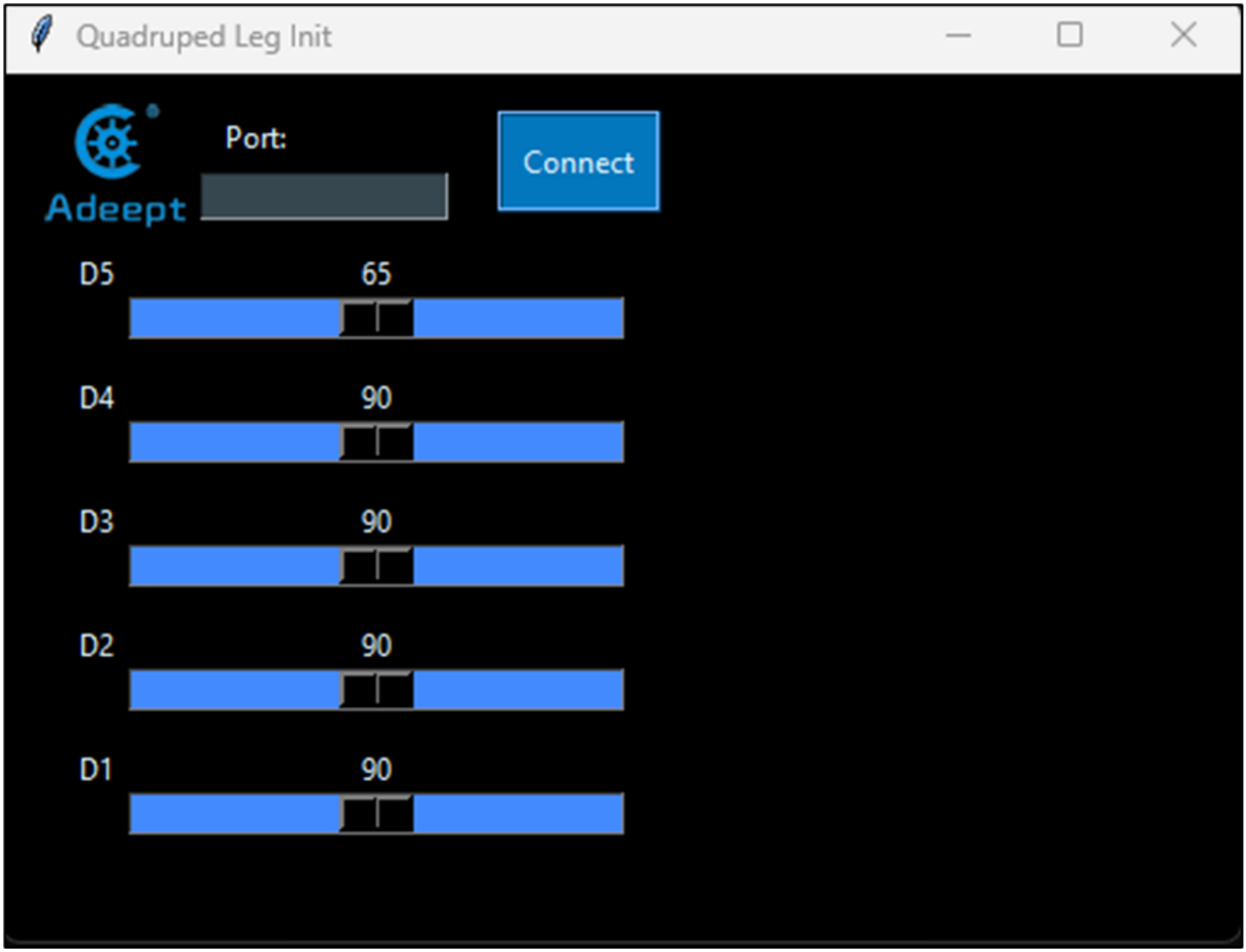
- **Project Results:** The final project is a functional Robotic Arm that is capable of various movements, similar to a human arm. The five servo motors provide extensive flexibility and can be compared to the joints in a human arm. The Robotic Arm can be controlled through the potentiometers on the Arm Drive Board, through a Graphical Dashboard, or the web. However, for the purposes of the presentation, only the Graphical Dashboard control will be used. The Robotic Arm is capable of grabbing, lifting, transporting & dropping lightweight objects. The OLED screen attached to the Robotic Arm is used to display "**WELCOME TO THE ROBOT AGE**". The Robotic Arm is designed to prototype heavy-duty and more robust Industrial Robotic Arm Systems.

- **Photographs & Screenshots:**


Adeept


MICRO SERVO


ANALOG INPUTS
4-20 mA (SLOT 3)
CH0
CH1
CH2
CH3
IN6
IN7
OUT6
OUT7
ANALOG OUTPUTS
4-20 mA (SLOT 4)
ANALOG OUTPUTS
4-20 mA (SLOT 5)
24 VDC
0 VDC
WELCOME TO
ROBOT AGE
www.adeept.com
Adeept



Screenshot of the Graphical Dashboard

Conclusion & Reflections:

- **Challenges Faced:** The major challenge in the project was troubleshooting. Since the Robotic Arm contains a lot of failure points, it was challenging to narrow down and locate the sources of error. The Robotic Arm is also made of several small parts which are fragile, and hence it was challenging to fix the error-inducing parts after the construction was completed. Moreover, considering the complexity of the code that was used for the project, troubleshooting and debugging the software code came off as a challenge.
- **Lessons Learned:** Through the project process, we have learnt about the operation of a Drive Board, and how it differs from the standard Arduino MCU. In summary, an Arduino Board can only supply small amounts of current, as opposed to a Drive Board. Arduino generally represents 5 V_{DC} when the pin is set to HIGH, whereas the Drive Board used in this project can go up to 24 V_{DC} when the pin is set to HIGH. Moreover, we learnt about the operation of an OLED and how the Drive Board communicates with the I/O pins and hence the I/O modules.
- **Future Enhancements:** We look forward to allowing Remote Access Control to our Robotic Arm so that it can function remotely and wirelessly, with the operator accessing and controlling the machine through a web interface or an application via the Internet.

Appendix:

- **OLED Code (OLED.ino):**

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET     4
Adafruit_SSD1306 display(128, 64, &Wire, OLED_RESET);


void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.setTextColor(WHITE);//Sets the font display color
  display.clearDisplay();//cls
}

void loop() {
  //Set the font size
  display.setTextSize(2);
  //Set the display location
  display.setCursor(0,0);
  //String displayed
  display.println("WELCOME TO");
  display.setCursor(50,20);
  display.println("THE");
   display.setCursor(5,40);
  display.println("ROBOT AGE");
```

```
  //Began to show
  display.display();
}
```

❖ Drive Board Code (block_py.ino):

```
/*
 * When BUZZER is 1, the infrared sensor cannot be used, but the BUZZER can be used
 * When BUZZER is 0, the infrared sensor can be used, but the BUZZER cannot be used
 *(you can change these values as you apply them)
*/

#define BUZZER   1
#include <ArduinoJson.h>
#if !BUZZER
#include <IRremote.h>
#endif
#include <Servo.h>

// OLED
#include <SSD1306Ascii.h>
#include <SSD1306AsciiWire.h>
// #define OLED_RESET     4
SSD1306AsciiWire display;

#if !BUZZER
char RECV_PIN = 11;//The definition of the infrared receiver pin 11
IRrecv irrecv(RECV_PIN);
decode_results results;
#endif
Servo myservo[5];

char line[60] = ""; // Serial data
int ret = 0;
char pingPin,trigPin;

void setup() {
  memset(line, 0, sizeof(line));
Serial.begin(115200); // Open the serial port and set the data transmission rate of 115200

}

void loop() {
  DynamicJsonDocument  jsonBuffer(100);
// Operate when the serial port is available
if (Serial.available() > 0) {

//  read incoming data: read up to \n, or up to 500 characters
ret = Serial.readBytesUntil('\n', line, 100);
//Serial.println(line);
```

```
  deserializeJson(jsonBuffer, line);
JsonObject root = jsonBuffer.as<JsonObject>();

  my_cmp(root);

memset(line, 0, sizeof(line));
}

}
void my_cmp(JsonObject root)
{
  /* parses the JSon data and executes the corresponding function */
  const char *python_char =  root["start"][0];
  const char *python_one =  root["start"][1];
  long one = root["start"][1],two = root["start"][2];
  long three = root["start"][3],four = root["start"][4];

  const char *Oledone = root["start"][1];

  //Serial.println(one);Serial.println(two);
  //Serial.println(one);
  if(strcmp(python_char,"setup")==0)
  {
      Serial.println("succes");
      memset(line, 0, sizeof(line));
  }

//OLED
  else if(strcmp(python_char,"OLED_init")==0)
  {
    Wire.begin();
    Wire.setClock(400000L);
    display.begin(&Adafruit128x64, 0x3C);
    display.setFont(Adafruit5x7);
  }
  else if(strcmp(python_char,"OLED_Ts")==0)
  {
    if (one == 1){
      display.set1X();
    }else if (one == 2){
      display.set2X();
    }else {
      display.set2X();
    }
  }

  else if(strcmp(python_char,"OLED_Cursor")==0)
  {
    display.setCursor(one, two); // (x,y)
    memset(line, 0, sizeof(line));
```

```
}
else if(strcmp(python_char,"OLED_Show")==0)
{
  display.print(Oledone);
  memset(line, 0, sizeof(line));
}
else if(strcmp(python_char,"OLED_Clear")==0)
{
  display.clear();
  memset(line, 0, sizeof(line));
}
else if(strcmp(python_char,"lcd_print")==0)
{
  display.print(Oledone);
  memset(line, 0, sizeof(line));
}


else if(strcmp(python_char,"pinmode")==0)
{
    int q,w;
    q = root["start"][1];
    w = root["start"][2];
    //input 0,output 1
    pinMode(one, two);
    memset(line, 0, sizeof(line));

}
 else if(strcmp(python_char,"pulseIn")==0)
{
    int q,w;
    q = root["start"][1];
    w = root["start"][2];
    //input 0,output 1
    Serial.println(pulseIn(one, two));
    memset(line, 0, sizeof(line));

}
else if(strcmp(python_char,"digitalWrite")==0)
{
    int q,w;
    q = root["start"][1];
    w = root["start"][2];
    //input 0,output 1
    digitalWrite(one, two);
    memset(line, 0, sizeof(line));

}
else if(strcmp(python_char,"digitalRead")==0)
{
```

```
    int q,w;
    q = root["start"][1];

    //input 0,output 1
    w = digitalRead(one);
    Serial.println(w);
    memset(line, 0, sizeof(line));

}
else if(strcmp(python_char,"analogRead")==0)
{
    int q,w;
    q = root["start"][1];

    //input 0,output 1
    w = analogRead(one);
    //Serial.print("text:");
    Serial.println(w);
    memset(line, 0, sizeof(line));

}
else if(strcmp(python_char,"analogWrite")==0)
{

    int q,w;
    //q = root["start"][1];
    //w = root["start"][2];
    //Serial.println(q);
    //input 0,output 1
    analogWrite(one, two);
    memset(line, 0, sizeof(line));

}
#if BUZZER
else if(strcmp(python_char,"tone")==0)
{

    int q,w;
    //q = root["start"][1];
    //w = root["start"][2];
    //Serial.println(q);
    //input 0,output 1
    tone(one, two);
    memset(line, 0, sizeof(line));

}
else if(strcmp(python_char,"noTone")==0)
{

    int q,w;
```

```
    //q = root["start"][1];
    //w = root["start"][2];
    //Serial.println(q);
    //input 0,output 1
    noTone(one);
    memset(line, 0, sizeof(line));

}
#endif

else if(strcmp(python_char,"servo_attach")==0)
{

  myservo[one].attach(two);
  memset(line, 0, sizeof(line));
}
else if(strcmp(python_char,"servo_write")==0)
{
  int q,w;



  myservo[one].write(two);

  memset(line, 0, sizeof(line));
}
#if !BUZZER
else if(strcmp(python_char,"irrecv_init")==0)
{
  int q,w;
  q = root["start"][1];
  w = root["start"][2];
  irrecv.enableIRIn();
  memset(line, 0, sizeof(line));
}
else if(strcmp(python_char,"irrecv_recv")==0)
{
    if (irrecv.decode(&results))
    {
      int irr_data = results.value;
      int irr_recv = switch_irr(irr_data);
      Serial.println(irr_recv);
      //Serial.println(results.value, HEX);//Wrap output in hex receive code
      irrecv.resume(); //Receiving the next value
    }
    else
    {
      Serial.println(-1);
    }
    memset(line, 0, sizeof(line));
```

```
  }
  #endif

  else if(strcmp(python_char,"ultra_init")==0)
  {
     pingPin = one;
     trigPin = two;
     pinMode(pingPin, INPUT); //Set the connection pin output mode Echo pin
     pinMode(trigPin, OUTPUT);//Set the connection pin output mode trog pin
    memset(line, 0, sizeof(line));
  }

}
```

- **Graphical Dashboard Code (servosGUI.py):**

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from socket import *
import sys
import time
import threading as thread
import tkinter as tk
import json
import os

import Adeept

ser = 0


ip_stu = 1

init_1 = 90;
init_2 = 90;
init_3 = 90;
init_4 = 90;
init_5 = 90;
init_6 = 90;
init_7 = 90;
init_8 = 90;
init_9 = 90;
init_10 = 90;
init_11 = 90;
init_12 = 90;

servoSpeed = 500
```

```
def global_init():
    global color_bg, color_text, color_btn, color_line, color_can, color_oval
    color_bg='#000000'      #Set background color
    color_text='#E1F5FE'      #Set text color
    color_btn='#0277BD'    #Set button color
    color_line='#01579B'      #Set line color
    color_can='#212121'    #Set canvas color
    color_oval='#2196F3'      #Set oval color

global_init()


def replace_num(initial,new_num):   #Call this function to replace data in '.txt' file
    newline=""
    str_num=str(new_num)
    try:
        with open("ip.txt","r") as f:
            for line in f.readlines():
                if(line.find(initial) == 0):
                    line = initial+"%s" %(str_num)
                newline += line
        with open("ip.txt","w") as f:
            f.writelines(newline)   #Call this function to replace data in '.txt' file
    except:
        pass


def replace_init(initial,new_num):   #Call this function to replace data in '.txt' file
    newline=""
    # try:
    # os.rename("AlterGo\\AlterGo.ino","AlterGo\\AlterGo.txt")
    with open("../block_py/angle.h","r") as f:
        for line in f.readlines():
            if(line.find(initial) == 0):
                line = initial+"%d;\n"%(new_num)
            newline += line
    with open("../block_py/angle.h","w") as f:
        f.writelines(newline)   #Call this function to replace data in '.txt' file
    # os.rename("AlterGo\\AlterGo.txt","AlterGo\\AlterGo.ino")


def my_replace_init(initial,new_num):   #Call this function to replace data in '.txt' file
    newline=""
    # try:
    # os.rename("AlterGo\\AlterGo.ino","AlterGo\\AlterGo.txt")
    with open("servosGUI.py","r") as f:
        for line in f.readlines():
            if(line.find(initial) == 0):
                line = initial+"%d;\n"%(new_num)
```

```
            newline += line
    with open("servosGUI.py","w") as f:
        f.writelines(newline)   #Call this function to replace data in '.txt' file
    # os.rename("AlterGo\\AlterGo.txt","AlterGo\\AlterGo.ino")

def num_import(initial):            #Call this function to import data from '.txt' file
    with open("ip.txt") as f:
        for line in f.readlines():
            if(line.find(initial) == 0):
                r=line
    begin=len(list(initial))
    snum=r[begin:]
    n=snum
    return n


def pwm_show():
    # show_input = show_input.split()
    # L0.config(text=show_input[1])
    L1.config(text=init_1)
    L2.config(text=init_2)
    L3.config(text=init_3)
    L4.config(text=init_4)
    L5.config(text=init_5)
    L6.config(text=init_6)
    L7.config(text=init_7)
    L8.config(text=init_8)
    L9.config(text=init_9)
    L10.config(text=init_10)
    L11.config(text=init_11)
    L12.config(text=init_12)
    # L13.config(text=init_13)
    # L14.config(text=init_14)
    # L15.config(text=init_15)


def normal_function_button():
    Btn_function_1.config(bg=color_btn)
    Btn_function_2.config(bg=color_btn)
    Btn_function_3.config(bg=color_btn)
    Btn_function_4.config(bg=color_btn)
    Btn_function_5.config(bg=color_btn)
    Btn_function_6.config(bg=color_btn)
    Btn_function_7.config(bg=color_btn)


def connection_thread():
    while 1:
        car_info = (tcpClicSock.recv(BUFSIZ)).decode()
        if not car_info:
            continue
```

```
        elif 'PWM' in car_info:
            pwm_show(car_info)


        elif '1 Selected' == car_info:
            normal_select_button()
            Btn_S1.config(bg='#4CAF50')

        elif '2 Selected' == car_info:
            normal_select_button()
            Btn_S2.config(bg='#4CAF50')

        elif '3 Selected' == car_info:
            normal_select_button()
            Btn_S3.config(bg='#4CAF50')

        elif '4 Selected' == car_info:
            normal_select_button()
            Btn_S4.config(bg='#4CAF50')

        elif '5 Selected' == car_info:
            normal_select_button()
            Btn_S5.config(bg='#4CAF50')

        elif '6 Selected' == car_info:
            normal_select_button()
            Btn_S6.config(bg='#4CAF50')

        elif '7 Selected' == car_info:
            normal_select_button()
            Btn_S7.config(bg='#4CAF50')

        elif '8 Selected' == car_info:
            normal_select_button()
            Btn_S8.config(bg='#4CAF50')

        elif '9 Selected' == car_info:
            normal_select_button()
            Btn_S9.config(bg='#4CAF50')

        elif '10 Selected' == car_info:
            normal_select_button()
            Btn_S10.config(bg='#4CAF50')

        elif '11 Selected' == car_info:
            normal_select_button()
            Btn_S11.config(bg='#4CAF50')

        elif '12 Selected' == car_info:
            normal_select_button()
```

```
            Btn_S12.config(bg='#4CAF50')

        elif '13 Selected' == car_info:
            normal_select_button()
            Btn_S13.config(bg='#4CAF50')

        elif '14 Selected' == car_info:
            normal_select_button()
            Btn_S14.config(bg='#4CAF50')

        elif '15 Selected' == car_info:
            normal_select_button()
            Btn_S15.config(bg='#4CAF50')

        elif '16 Selected' == car_info:
            normal_select_button()
            Btn_S16.config(bg='#4CAF50')


        elif 'function_1_on' in car_info:
            normal_function_button()
            Btn_function_1.config(bg='#4CAF50')

        elif 'function_2_on' in car_info:
            normal_function_button()
            Btn_function_2.config(bg='#4CAF50')

        elif 'function_3_on' in car_info:
            normal_function_button()
            Btn_function_3.config(bg='#4CAF50')

        elif 'function_4_on' in car_info:
            normal_function_button()
            Btn_function_4.config(bg='#4CAF50')

        elif 'function_5_on' in car_info:
            normal_function_button()
            Btn_function_5.config(bg='#4CAF50')

        elif 'function_6_on' in car_info:
            normal_function_button()
            Btn_function_6.config(bg='#4CAF50')

        elif 'function_7_on' in car_info:
            normal_function_button()
            Btn_function_7.config(bg='#4CAF50')

        elif 'perform_1_on' in car_info:
            normal_function_button()
            normal_select_button()
```

```
            Btn_P1.config(bg='#4CAF50')
        elif 'perform_2_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P2.config(bg='#4CAF50')
        elif 'perform_3_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P3.config(bg='#4CAF50')
        elif 'perform_4_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P4.config(bg='#4CAF50')
        elif 'perform_5_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P5.config(bg='#4CAF50')
        elif 'perform_6_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P6.config(bg='#4CAF50')
        elif 'perform_7_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P7.config(bg='#4CAF50')
        elif 'perform_8_on' in car_info:
            normal_function_button()
            normal_select_button()
            Btn_P8.config(bg='#4CAF50')

        elif 'perform_end' in car_info:
            normal_perform_button()

        print(car_info)

def wiat_connect():
    global ser
    while 1:
        ser.write("{'start':['setup']}\n".encode("gbk"))
        #print(11)
        line = ser.readline()
        if line:
            break

def serial_connect():    #Call this function to connect with the server
    global ADDR,tcpClicSock,BUFSIZ,ip_stu,ipaddr,ser
    com=E1.get()       #Get the IP address from Entry
    Adeept.com_init(com,115200,1)
    Adeept.wiat_connect()
    Adeept.three_function("'servo_attach'",0,9)
```

```
    Adeept.three_function("'servo_attach'",1,6)
    Adeept.three_function("'servo_attach'",2,5)
    Adeept.three_function("'servo_attach'",3,3)
    Adeept.three_function("'servo_attach'",4,11)
    E1.config(state='disabled')   #Disable the Entry
    Btn14.config(state='disabled')   #Disable the Entry
    print(com+':Succes')


def connect(event):    #Call this function to connect with the server
    if ip_stu == 1:
        sc=thread.Thread(target=serial_connect) #Define a thread for connection
        sc.setDaemon(True)                    #'True' means it is a front thread,it would
close when the mainloop() closes
        sc.start()                            #Thread starts


def three_function(a,b,c):
    global ser
    a = str(a)
    b = str(b)
    c = str(c)
    p1 = "{'start':"+'['+a+','+b+','+c+']'+'}'+'\n'
    #print(p1)
    ser.write(p1.encode("gbk"))

def jsonDS(numInput, adjustInput):
    global init_1, init_2, init_3, init_4, init_5, init_6, init_7, init_8, init_9, init_10,
init_11, init_12,ser
    if numInput == 1:
        init_1 += adjustInput
        posInput = init_1
    elif numInput == 2:
        init_2 += adjustInput
        posInput = init_2
    elif numInput == 3:
        init_3 += adjustInput
        posInput = init_3
    elif numInput == 4:
        init_7 += adjustInput
        posInput = init_4
    elif numInput == 5:
        init_8 += adjustInput
        posInput = init_5
    elif numInput == 6:
        init_9 += adjustInput
        posInput = init_6
    elif numInput == 7:
        init_4 += adjustInput
        posInput = init_7
```

```
    elif numInput == 8:
        init_5 += adjustInput
        posInput = init_8
    elif numInput == 9:
        init_6 += adjustInput
        posInput = init_9
    elif numInput == 10:
        init_10 += adjustInput
        posInput = init_10
    elif numInput == 11:
        init_11 += adjustInput
        posInput = init_11
    elif numInput == 12:
        init_12 += adjustInput
        posInput = init_12

    dumpsInput = {'angle':[numInput, posInput]}
    c2json = json.dumps(dumpsInput)
    three_function("'angle'",numInput-1,posInput);
    pwm_show()


def servo_buttons(x,y):
    global L0, L1, L2, L3, L4, L5, L6, L7, L8, L9, L10, L11, L12, L13, L14, L15
    # def call_pwm0_up(event):
    #   tcpClicSock.send(('0+').encode())

    # def call_pwm0_down(event):
    #   tcpClicSock.send(('0-').encode())

    def call_pwm1_up(event):
        jsonDS(1, 3)

    def call_pwm1_down(event):
        jsonDS(1, -3)

    def call_pwm2_up(event):
        jsonDS(2, 3)

    def call_pwm2_down(event):
        jsonDS(2, -3)

    def call_pwm3_up(event):
        jsonDS(3, 3)

    def call_pwm3_down(event):
        jsonDS(3, -3)

    def call_pwm4_up(event):
        jsonDS(4, 3)
```

```
def call_pwm4_down(event):
    jsonDS(4, -3)

def call_pwm5_up(event):
    jsonDS(5, 3)

def call_pwm5_down(event):
    jsonDS(5, -3)

def call_pwm6_up(event):
    jsonDS(6, 3)

def call_pwm6_down(event):
    jsonDS(6, -3)

def call_pwm7_up(event):
    jsonDS(7, 3)

def call_pwm7_down(event):
    jsonDS(7, -3)

def call_pwm8_up(event):
    jsonDS(8, 3)

def call_pwm8_down(event):
    jsonDS(8, -3)

def call_pwm9_up(event):
    jsonDS(9, 3)

def call_pwm9_down(event):
    jsonDS(9, -3)

def call_pwm10_up(event):
    jsonDS(10, 3)

def call_pwm10_down(event):
    jsonDS(10, -3)

def call_pwm11_up(event):
    jsonDS(11, 3)

def call_pwm11_down(event):
    jsonDS(11, -3)

def call_pwm12_up(event):
    jsonDS(12, 3)

def call_pwm12_down(event):
```

```
        jsonDS(12, -3)

    # def call_pwm13_up(event):
    #   tcpClicSock.send(('13+').encode())

    # def call_pwm13_down(event):
    #   tcpClicSock.send(('13-').encode())

    # def call_pwm14_up(event):
    #   tcpClicSock.send(('14+').encode())

    # def call_pwm14_down(event):
    #   tcpClicSock.send(('14-').encode())

    # def call_pwm15_up(event):
    #   tcpClicSock.send(('15+').encode())

    # def call_pwm15_down(event):
    #   tcpClicSock.send(('15-').encode())

    # L0 = tk.Label(root,width=8,text='PWM0',fg=color_text,bg='#212121')
    # L0.place(x=x,y=y-25)
    # Btn_0i = tk.Button(root, width=8,
text='PWM0+',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_0i.place(x=x,y=y)
    # Btn_0i.bind('<ButtonPress-1>', call_pwm0_up)

    # Btn_0d = tk.Button(root, width=8, text='PWM0-
',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_0d.place(x=x,y=y+35)
    # Btn_0d.bind('<ButtonPress-1>', call_pwm0_down)

    # L13 = tk.Label(root,width=8,text='PWM13',fg=color_text,bg='#212121')
    # L13.place(x=x+500,y=y+75)
    # Btn_13i = tk.Button(root, width=8,
text='PWM13+',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_13i.place(x=x+500,y=y+100)
    # Btn_13i.bind('<ButtonPress-1>', call_pwm13_up)

    # Btn_13d = tk.Button(root, width=8, text='PWM13-
',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_13d.place(x=x+500,y=y+135)
    # Btn_13d.bind('<ButtonPress-1>', call_pwm13_down)

    # L14 = tk.Label(root,width=8,text='PWM14',fg=color_text,bg='#212121')
    # L14.place(x=x+600,y=y+75)
    # Btn_14i = tk.Button(root, width=8,
text='PWM14+',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_14i.place(x=x+600,y=y+100)
    # Btn_14i.bind('<ButtonPress-1>', call_pwm14_up)
```

```
    # Btn_14d = tk.Button(root, width=8, text='PWM14-
',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_14d.place(x=x+600,y=y+135)
    # Btn_14d.bind('<ButtonPress-1>', call_pwm14_down)


    # L15 = tk.Label(root,width=8,text='PWM15',fg=color_text,bg='#212121')
    # L15.place(x=x+700,y=y+75)
    # Btn_15i = tk.Button(root, width=8,
text='PWM15+',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_15i.place(x=x+700,y=y+100)
    # Btn_15i.bind('<ButtonPress-1>', call_pwm15_up)

    # Btn_15d = tk.Button(root, width=8, text='PWM15-
',fg=color_text,bg=color_btn,relief='ridge')
    # Btn_15d.place(x=x+700,y=y+135)
    # Btn_15d.bind('<ButtonPress-1>', call_pwm15_down)


def posSelect_buttons(x,y):
    global Btn_S1, Btn_S2, Btn_S3, Btn_S4, Btn_S5, Btn_S6, Btn_S7, Btn_S8, Btn_S9, Btn_S10,
Btn_S11, Btn_S12, Btn_S13, Btn_S14, Btn_S15, Btn_S16
    def select_1(event):
        tcpClicSock.send(('Select 1').encode())

    def select_2(event):
        tcpClicSock.send(('Select 2').encode())

    def select_3(event):
        tcpClicSock.send(('Select 3').encode())

    def select_4(event):
        tcpClicSock.send(('Select 4').encode())

    def select_5(event):
        tcpClicSock.send(('Select 5').encode())

    def select_6(event):
        tcpClicSock.send(('Select 6').encode())

    def select_7(event):
        tcpClicSock.send(('Select 7').encode())

    def select_8(event):
        tcpClicSock.send(('Select 8').encode())


    def select_9(event):
        tcpClicSock.send(('Select 9').encode())
```

```
    def select_10(event):
        tcpClicSock.send(('Select 10').encode())

    def select_11(event):
        tcpClicSock.send(('Select 11').encode())

    def select_12(event):
        tcpClicSock.send(('Select 12').encode())

    def select_13(event):
        tcpClicSock.send(('Select 13').encode())

    def select_14(event):
        tcpClicSock.send(('Select 14').encode())

    def select_15(event):
        tcpClicSock.send(('Select 15').encode())

    def select_16(event):
        tcpClicSock.send(('Select 16').encode())


    Btn_S1 = tk.Button(root, width=8, text='Q',fg=color_text,bg=color_btn,relief='ridge')
    Btn_S1.place(x=x,y=y)
    Btn_S1.bind('<ButtonPress-1>', select_1)
    root.bind('<KeyPress-q>', select_1)

    Btn_S2 = tk.Button(root, width=8, text='W',fg=color_text,bg=color_btn,relief='ridge')
    Btn_S2.place(x=x+100,y=y)
    Btn_S2.bind('<ButtonPress-1>', select_2)
    root.bind('<KeyPress-w>', select_2)

    Btn_S3 = tk.Button(root, width=8, text='E',fg=color_text,bg=color_btn,relief='ridge')
    Btn_S3.place(x=x+200,y=y)
    Btn_S3.bind('<ButtonPress-1>', select_3)
    root.bind('<KeyPress-e>', select_3)

    Btn_S4 = tk.Button(root, width=8, text='R',fg=color_text,bg=color_btn,relief='ridge')
    Btn_S4.place(x=x+300,y=y)
    Btn_S4.bind('<ButtonPress-1>', select_4)
    root.bind('<KeyPress-r>', select_4)

    Btn_S5 = tk.Button(root, width=8, text='T',fg=color_text,bg=color_btn,relief='ridge')
    Btn_S5.place(x=x+400,y=y)
    Btn_S5.bind('<ButtonPress-1>', select_5)
    root.bind('<KeyPress-t>', select_5)

    Btn_S6 = tk.Button(root, width=8, text='Y',fg=color_text,bg=color_btn,relief='ridge')
    Btn_S6.place(x=x+500,y=y)
    Btn_S6.bind('<ButtonPress-1>', select_6)
```

```
root.bind('<KeyPress-y>', select_6)

Btn_S7 = tk.Button(root, width=8, text='U',fg=color_text,bg=color_btn,relief='ridge')
Btn_S7.place(x=x+600,y=y)
Btn_S7.bind('<ButtonPress-1>', select_7)
root.bind('<KeyPress-u>', select_7)

Btn_S8 = tk.Button(root, width=8, text='I',fg=color_text,bg=color_btn,relief='ridge')
Btn_S8.place(x=x+700,y=y)
Btn_S8.bind('<ButtonPress-1>', select_8)
root.bind('<KeyPress-i>', select_8)


Btn_S9 = tk.Button(root, width=8, text='A',fg=color_text,bg=color_btn,relief='ridge')
Btn_S9.place(x=x,y=y+35)
Btn_S9.bind('<ButtonPress-1>', select_9)
root.bind('<KeyPress-a>', select_9)

Btn_S10 = tk.Button(root, width=8, text='S',fg=color_text,bg=color_btn,relief='ridge')
Btn_S10.place(x=x+100,y=y+35)
Btn_S10.bind('<ButtonPress-1>', select_10)
root.bind('<KeyPress-s>', select_10)

Btn_S11 = tk.Button(root, width=8, text='D',fg=color_text,bg=color_btn,relief='ridge')
Btn_S11.place(x=x+200,y=y+35)
Btn_S11.bind('<ButtonPress-1>', select_11)
root.bind('<KeyPress-d>', select_11)

Btn_S12 = tk.Button(root, width=8, text='F',fg=color_text,bg=color_btn,relief='ridge')
Btn_S12.place(x=x+300,y=y+35)
Btn_S12.bind('<ButtonPress-1>', select_12)
root.bind('<KeyPress-f>', select_12)

Btn_S13 = tk.Button(root, width=8, text='G',fg=color_text,bg=color_btn,relief='ridge')
Btn_S13.place(x=x+400,y=y+35)
Btn_S13.bind('<ButtonPress-1>', select_13)
root.bind('<KeyPress-g>', select_13)

Btn_S14 = tk.Button(root, width=8, text='H',fg=color_text,bg=color_btn,relief='ridge')
Btn_S14.place(x=x+500,y=y+35)
Btn_S14.bind('<ButtonPress-1>', select_14)
root.bind('<KeyPress-h>', select_14)

Btn_S15 = tk.Button(root, width=8, text='J',fg=color_text,bg=color_btn,relief='ridge')
Btn_S15.place(x=x+600,y=y+35)
Btn_S15.bind('<ButtonPress-1>', select_15)
root.bind('<KeyPress-j>', select_15)

Btn_S16 = tk.Button(root, width=8, text='K',fg=color_text,bg=color_btn,relief='ridge')
Btn_S16.place(x=x+700,y=y+35)
```

```
    Btn_S16.bind('<ButtonPress-1>', select_16)
    root.bind('<KeyPress-k>', select_16)


def perform_buttons(x,y):
    global Btn_P1, Btn_P2, Btn_P3, Btn_P4, Btn_P5, Btn_P6, Btn_P7, Btn_P8
    def perform_1(event):
        tcpClicSock.send(('perform_1').encode())
    def perform_2(event):
        tcpClicSock.send(('perform_2').encode())
    def perform_3(event):
        tcpClicSock.send(('perform_3').encode())
    def perform_4(event):
        tcpClicSock.send(('perform_4').encode())
    def perform_5(event):
        tcpClicSock.send(('perform_5').encode())
    def perform_6(event):
        tcpClicSock.send(('perform_6').encode())
    def perform_7(event):
        tcpClicSock.send(('perform_7').encode())
    def perform_8(event):
        tcpClicSock.send(('perform_8').encode())

    Btn_P1 = tk.Button(root, width=8,
text='Perform_1',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P1.place(x=x,y=y)
    Btn_P1.bind('<ButtonPress-1>', perform_1)
    root.bind('<KeyPress-z>', perform_1)

    Btn_P2 = tk.Button(root, width=8,
text='Perform_2',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P2.place(x=x+100,y=y)
    Btn_P2.bind('<ButtonPress-1>', perform_2)
    root.bind('<KeyPress-x>', perform_2)

    Btn_P3 = tk.Button(root, width=8,
text='Perform_3',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P3.place(x=x+200,y=y)
    Btn_P3.bind('<ButtonPress-1>', perform_3)
    root.bind('<KeyPress-c>', perform_3)

    Btn_P4 = tk.Button(root, width=8,
text='Perform_4',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P4.place(x=x+300,y=y)
    Btn_P4.bind('<ButtonPress-1>', perform_4)
    root.bind('<KeyPress-v>', perform_4)

    Btn_P5 = tk.Button(root, width=8,
text='Perform_5',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P5.place(x=x+400,y=y)
```

```
    Btn_P5.bind('<ButtonPress-1>', perform_5)
    root.bind('<KeyPress-b>', perform_5)

    Btn_P6 = tk.Button(root, width=8,
text='Perform_6',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P6.place(x=x+500,y=y)
    Btn_P6.bind('<ButtonPress-1>', perform_6)
    root.bind('<KeyPress-n>', perform_6)

    Btn_P7 = tk.Button(root, width=8,
text='Perform_7',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P7.place(x=x+600,y=y)
    Btn_P7.bind('<ButtonPress-1>', perform_7)
    root.bind('<KeyPress-m>', perform_7)

    Btn_P8 = tk.Button(root, width=8,
text='Perform_8',fg=color_text,bg=color_btn,relief='ridge')
    Btn_P8.place(x=x+700,y=y)
    Btn_P8.bind('<ButtonPress-1>', perform_8)
    root.bind('<KeyPress-,>', perform_8)


def normal_perform_button():
    Btn_P1.config(bg=color_btn)
    Btn_P2.config(bg=color_btn)
    Btn_P3.config(bg=color_btn)
    Btn_P4.config(bg=color_btn)
    Btn_P5.config(bg=color_btn)
    Btn_P6.config(bg=color_btn)
    Btn_P7.config(bg=color_btn)
    Btn_P8.config(bg=color_btn)


def normal_select_button():
    Btn_S1.config(bg=color_btn)
    Btn_S2.config(bg=color_btn)
    Btn_S3.config(bg=color_btn)
    Btn_S4.config(bg=color_btn)
    Btn_S5.config(bg=color_btn)
    Btn_S6.config(bg=color_btn)
    Btn_S7.config(bg=color_btn)
    Btn_S8.config(bg=color_btn)
    Btn_S9.config(bg=color_btn)
    Btn_S10.config(bg=color_btn)
    Btn_S11.config(bg=color_btn)
    Btn_S12.config(bg=color_btn)
    Btn_S13.config(bg=color_btn)
    Btn_S14.config(bg=color_btn)
    Btn_S15.config(bg=color_btn)
    Btn_S16.config(bg=color_btn)
```

```
def connent_input(x,y):
    global E1, Btn14
    E1 = tk.Entry(root,show=None,width=16,bg="#37474F",fg='#eceff1')
    E1.place(x=x+30,y=y+25)                          #Define a Entry and put it in position

    l_ip_3=tk.Label(root,width=10,text='Port:',fg=color_text,bg='#000000')
    l_ip_3.place(x=x+15,y=y)                       #Define a Label and put it in position

    Btn14= tk.Button(root, width=8,height=2,
text='Connect',fg=color_text,bg=color_btn,relief='ridge')
    Btn14.place(x=x+150,y=y)                         #Define a Button and put it in position

    root.bind('<Return>', connect)
    Btn14.bind('<ButtonPress-1>', connect)


def initSet(event):
    replace_init("int angle0 = ", init_1)
    replace_init("int angle1 = ", init_2)
    replace_init("int angle2 = ", init_3)
    replace_init("int angle3 = ", init_4)
    replace_init("int angle4 = ", init_5)

    replace_init("int angle5 = ", init_6)
    replace_init("int angle6 = ", init_7)
    replace_init("int angle7 = ", init_8)
    replace_init("int angle8 = ", init_9)
    replace_init("int angle9 = ", init_10)

    replace_init("int angle10 = ", init_11)
    replace_init("int angle11 = ", init_12)


    my_replace_init("init_1 = ", init_1)
    my_replace_init("init_2 = ", init_2)
    my_replace_init("init_3 = ", init_3)

    my_replace_init("init_4 = ", init_4)
    my_replace_init("init_5 = ", init_5)
    my_replace_init("init_6 = ", init_6)

    my_replace_init("init_7 = ", init_7)
    my_replace_init("init_8 = ", init_8)
    my_replace_init("init_9 = ", init_9)

    my_replace_init("init_10 = ", init_10)
    my_replace_init("init_11 = ", init_11)
    my_replace_init("init_12 = ", init_12)
```

```
def set_button(x,y):
    global L_select_feedback
    L_select_feedback = tk.Label(root,width=23,text='Click here after
debugging',fg=color_text,bg='#000000')
    L_select_feedback.place(x=x-90,y=y-25)

    Btn_Switch_1 = tk.Button(root, width=8,
text='SET',fg=color_text,bg=color_btn,relief='ridge')
    Btn_Switch_1.place(x=x,y=y)
    Btn_Switch_1.bind('<ButtonPress-1>', initSet)


def scale(x,y,w):
    global var_A,var_B, var_C,var_D,var_E
    def pix_send1(event):
        time.sleep(0.03)
        Adeept.three_function("'servo_write'",0,var_A.get())
        #print(var_A.get())
    def pix_send2(event):
        time.sleep(0.03)
        Adeept.three_function("'servo_write'",1,var_B.get())
        #print(var_B.get())
    def pix_send3(event):
        time.sleep(0.03)
        Adeept.three_function("'servo_write'",2,var_C.get())
        #print(var_C.get())
    def pix_send4(event):
        time.sleep(0.03)
        Adeept.three_function("'servo_write'",3,var_D.get())
        #print(var_D.get())
    def pix_send5(event):
        time.sleep(0.03)
        Adeept.three_function("'servo_write'",4,var_E.get())
        #print(var_E.get())


    #tcpClicSock.send(('pix %s'%var_B.get()).encode())


    def time_send(event):
        time.sleep(0.03)
        #tcpClicSock.send(('time %s'%var_C.get()).encode())



    var_A = tk.StringVar()
    var_A.set(90)
    var_B = tk.StringVar()
    var_B.set(90)
```

```
    var_C = tk.StringVar()
    var_C.set(90)
    var_D = tk.StringVar()
    var_D.set(90)
    var_E = tk.StringVar()
    var_E.set(65)




    Scale_B = tk.Scale(root,label=None,
    from_=30,to=100,orient=tk.HORIZONTAL,length=w,
    showvalue=1,tickinterval=None,resolution=1,variable=var_E,troughcolor='#448AFF',command=p
ix_send5,fg=color_text,bg=color_bg,highlightthickness=0)
    Scale_B.place(x=x,y=y)                          #Define a Scale and put it in position




    Scale_B = tk.Scale(root,label=None,
    from_=0,to=180,orient=tk.HORIZONTAL,length=w,
    showvalue=1,tickinterval=None,resolution=1,variable=var_D,troughcolor='#448AFF',command=p
ix_send4,fg=color_text,bg=color_bg,highlightthickness=0)
    Scale_B.place(x=x,y=y+50)                         #Define a Scale and put it in position




    Scale_B = tk.Scale(root,label=None,
    from_=0,to=180,orient=tk.HORIZONTAL,length=w,
    showvalue=1,tickinterval=None,resolution=1,variable=var_C,troughcolor='#448AFF',command=p
ix_send3,fg=color_text,bg=color_bg,highlightthickness=0)
    Scale_B.place(x=x,y=y+100)                         #Define a Scale and put it in
position




    Scale_B = tk.Scale(root,label=None,
    from_=0,to=180,orient=tk.HORIZONTAL,length=w,
    showvalue=1,tickinterval=None,resolution=1,variable=var_B,troughcolor='#448AFF',command=p
ix_send2,fg=color_text,bg=color_bg,highlightthickness=0)
    Scale_B.place(x=x,y=y+150)                         #Define a Scale and put it in
position



    Scale_B = tk.Scale(root,label=None,
    from_=0,to=180,orient=tk.HORIZONTAL,length=w,
```

```
    showvalue=1,tickinterval=None,resolution=1,variable=var_A,troughcolor='#448AFF',command=p
ix_send1,fg=color_text,bg=color_bg,highlightthickness=0)
    Scale_B.place(x=x,y=y+200)                          #Define a Scale and put it in
position

def speed_set(x,y,w):
    def speed_send(event):
        time.sleep(0.03)
        tcpClicSock.send(('speed %s'%var_speed.get()).encode())
    global var_speed
    var_speed = tk.StringVar()
    var_speed.set(1)

    Scale_speed = tk.Scale(root,label=None,
    from_=1,to=50,orient=tk.HORIZONTAL,length=w,
    showvalue=1,tickinterval=None,resolution=1,variable=var_speed,troughcolor='#448AFF',comma
nd=speed_send,fg=color_text,bg=color_bg,highlightthickness=0)
    Scale_speed.place(x=x,y=y)                          #Define a Scale and put it in
position


def function_buttons(x,y):
    global function_stu, Btn_function_1, Btn_function_2, Btn_function_3, Btn_function_4,
Btn_function_5, Btn_function_6, Btn_function_7
    def call_function_1(event):
        tcpClicSock.send(('function_1_on').encode())

    def call_function_2(event):
        tcpClicSock.send(('function_2_on').encode())

    def call_function_3(event):
        tcpClicSock.send(('function_3_on').encode())

    def call_function_4(event):
        tcpClicSock.send(('function_4_on').encode())

    def call_function_5(event):
        tcpClicSock.send(('function_5_on').encode())

    def call_function_6(event):
        tcpClicSock.send(('function_6_on').encode())

    def call_function_7(event):
        tcpClicSock.send(('function_7_on').encode())

    Btn_function_1 = tk.Button(root, width=8,
text='FastMove',fg=color_text,bg=color_btn,relief='ridge')
    Btn_function_2 = tk.Button(root, width=8,
text='Smooth',fg=color_text,bg=color_btn,relief='ridge')
```

```
    Btn_function_3 = tk.Button(root, width=8,
text='Bézier',fg=color_text,bg=color_btn,relief='ridge')
    Btn_function_4 = tk.Button(root, width=8,
text='Config',fg=color_text,bg=color_btn,relief='ridge')
    Btn_function_5 = tk.Button(root, width=8, text='-
',fg=color_text,bg=color_btn,relief='ridge')
    Btn_function_6 = tk.Button(root, width=8, text='-
',fg=color_text,bg=color_btn,relief='ridge')
    Btn_function_7 = tk.Button(root, width=8, text='-
',fg=color_text,bg=color_btn,relief='ridge')

    Btn_function_1.place(x=x,y=y)
    Btn_function_2.place(x=x+70,y=y)
    Btn_function_3.place(x=x+140,y=y)
    Btn_function_4.place(x=x+210,y=y)
    Btn_function_5.place(x=x+280,y=y)
    Btn_function_6.place(x=x+350,y=y)
    Btn_function_7.place(x=x+420,y=y)

    Btn_function_1.bind('<ButtonPress-1>', call_function_1)
    Btn_function_2.bind('<ButtonPress-1>', call_function_2)
    Btn_function_3.bind('<ButtonPress-1>', call_function_3)
    Btn_function_4.bind('<ButtonPress-1>', call_function_4)
    Btn_function_5.bind('<ButtonPress-1>', call_function_5)
    Btn_function_6.bind('<ButtonPress-1>', call_function_6)
    Btn_function_7.bind('<ButtonPress-1>', call_function_7)


def loop():
    global root
    root = tk.Tk()
    root.title('Quadruped Leg Init')
    root.geometry('500x350')
    root.config(bg=color_bg)

    try:
        logo =tk.PhotoImage(file = 'logo.png')
        l_logo=tk.Label(root,image = logo,bg=color_bg)
        l_logo.place(x=280,y=42)
        logo1 =tk.PhotoImage(file = 'logo1.png')
        l_logo1=tk.Label(root,image = logo1,bg=color_bg)
        l_logo1.place(x=15,y=10)
    except:
        pass

    l_ip_3=tk.Label(root,width=10,text='D5',fg=color_text,bg='#000000')
    l_ip_3.place(x=0,y=70)                          #Define a Label and put it in position
    l_ip_3=tk.Label(root,width=10,text='D4',fg=color_text,bg='#000000')
    l_ip_3.place(x=0,y=120)                         #Define a Label and put it in position
    l_ip_3=tk.Label(root,width=10,text='D3',fg=color_text,bg='#000000')
```

```
    l_ip_3.place(x=0,y=170)                     #Define a Label and put it in position
    l_ip_3=tk.Label(root,width=10,text='D2',fg=color_text,bg='#000000')
    l_ip_3.place(x=0,y=220)                     #Define a Label and put it in position
    l_ip_3=tk.Label(root,width=10,text='D1',fg=color_text,bg='#000000')
    l_ip_3.place(x=0,y=270)                     #Define a Label and put it in position
    connent_input(50,15)
    #set_button(630,35)
    #set_button1(630,35)
    #speed_set(350,25,280)

    #servo_buttons(-70,120)

    # posSelect_buttons(30,385)

    # perform_buttons(30,465)

    scale(50,70,200)

    # function_buttons(30,320)

    root.mainloop()


if __name__ == '__main__':
    loop()
```

References:

https://www.microchip.com/en-us/product/atmega328p

https://www.intel.com/content/www/us/en/robotics/robotic-arm.html

https://www.adeept.com/learn/detail-64.html

https://www.reddit.com/r/arduino/comments/ruc4f1/what_are_driver_boards_for/?rdt=58354