



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُونِيسَيْتِي إِسْلَامُ إِنْتَارَايَغُسِيَا مِلَيْسِيَا
Garden of Knowledge and Virtue

REPORT 3B : SERIAL COMMUNICATION

GROUP 4

MCTA 3203

SEMESTER 2 2023/2024

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION:

	NAME	MATRIC NUMBER
1.	NUR SHAHEERA BINTI KAMIS	2214040
2.	NORAMIRA HUSNA BT NORKHAIRANI	2214496
3.	MUHAMMAD EIQBAL BIN HASBOLLAH	2216911

TABLE OF CONTENT

Introduction.....	3
Abstract.....	3
Materials and Equipment.....	3
Experimental Setup.....	4
Methodology.....	4
Procedure.....	4
Results.....	5
Discussion.....	6
Question.....	11
Conclusion.....	16
Recommendation.....	16
References.....	17
Acknowledgement.....	17
Student's Declaration.....	18

INTRODUCTION

Python and Arduino boards are a common combination for controlling servo motors in robotics and other fields. Because they are capable of controlling an angular position, servo motors are a crucial component of many electromechanical systems. This report explains how to connect an Arduino board with Python to drive a servo motor step-by-step. A variety of applications are made possible by the ability for users to direct the servo motor to move to precise angles by sending angle data via a Python script to the Arduino.

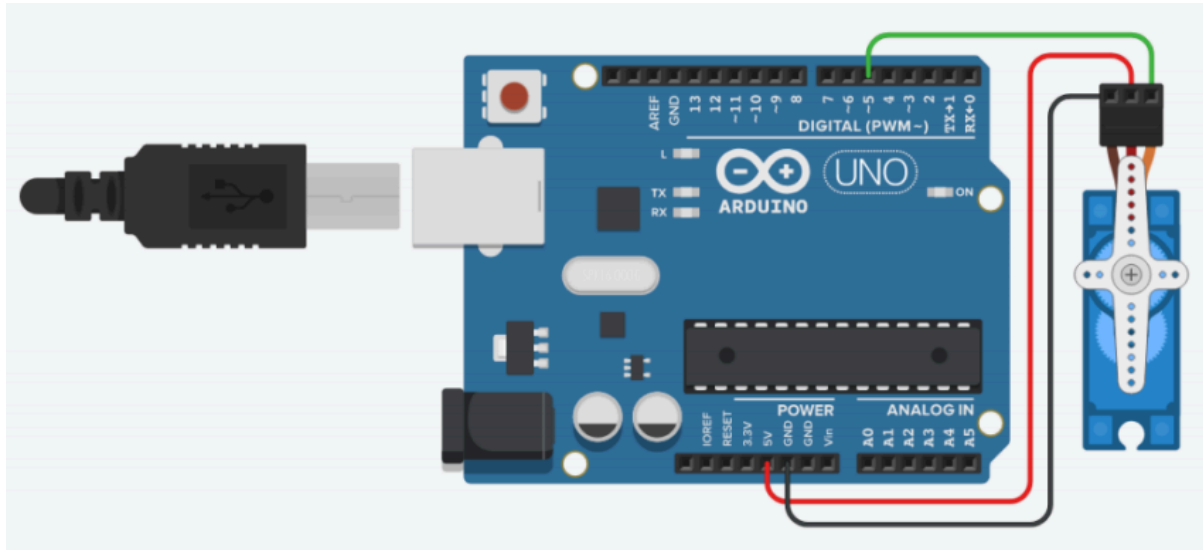
ABSTRACT

This paper provides an explanation of how to work with Python and Arduino to control a servo motor. accurate angular control is frequently required to use servo motors. According to the report's approach, angle data is transmitted via a Python script to an Arduino board, which then understands it and uses it to operate the servo motor appropriately. With the help of full instructions, users are able to combine servo motor control with Python, making it an appropriate choice for a range of projects and uses.

MATERIALS AND EQUIPMENT

- Arduino Uno board
- Servo motor
- Jumper wires
- Potentiometer
- USB cable for Arduino

EXPERIMENTAL SETUP



METHODOLOGY

Connect the servo motor to the Arduino board.

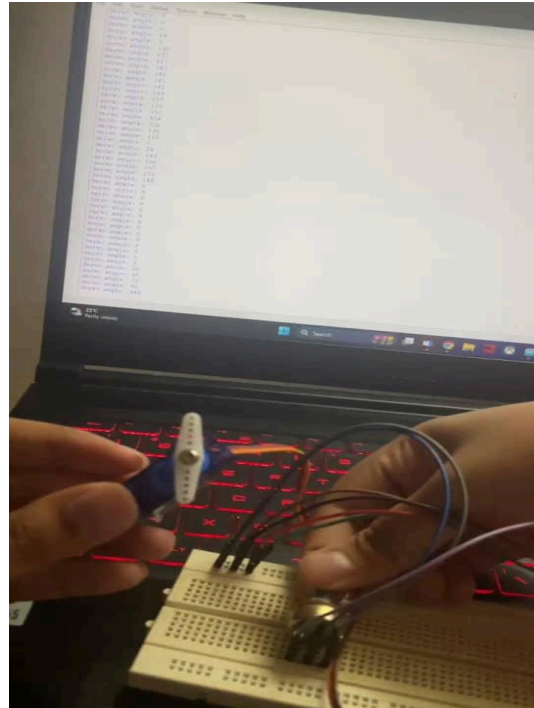
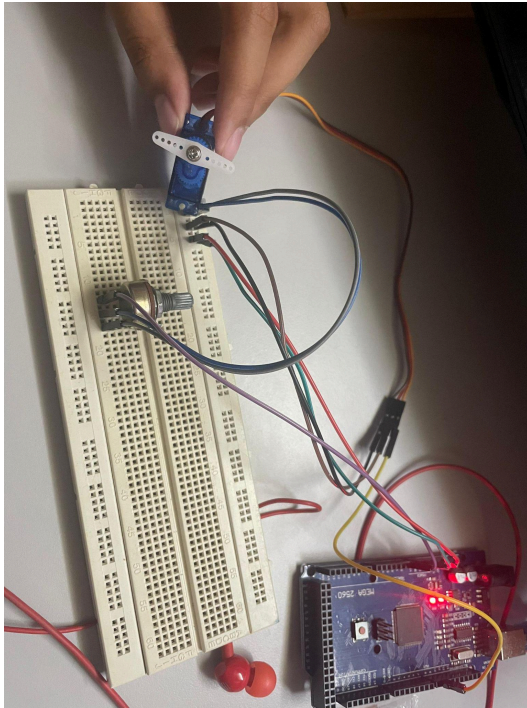
- Connect the power (red wires) to 5v in Arduino
- Connect the ground (brown wires) to GND in Arduino
- Connect the signal (yellow wires) to PWM in the Arduino. In this experiment we use pin

9

PROCEDURE

1. Built the circuit per the setup instructions provided.
2. Uploaded the provided Arduino code to your Arduino Uno and Python.
3. Control the servo by inserting the value on python that we already run the code.

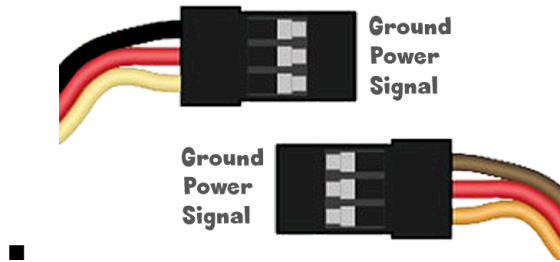
RESULTS



DISCUSSION

- **Hardware**

- Connect the Servo's Signal Wire:
 - The servo motor normally has three wires: red, brown, and orange/yellow (but colours may differ). The orange/yellow wire is the signal wire.
 - The signal line must be attached to a PWM (Pulse Width Modulation) pin on the Arduino board. PWM pins allow the Arduino to control the servo's position by changing the width of the electrical pulses transmitted to it.



- Power the servo using the Arduino's 5V and GND pins:
 - Servo motors typically require a supply voltage of +5V.
 - Connect the servo's red (or power) wire to the Arduino board's 5V output pin. This delivers the required power to the servo.
 - Connect the servo's brown wire (or ground) to one of the Arduino's ground (GND) pins. This completes the circuit and establishes the servo's ground connection.

- Connect the Servo's Ground Wire:
 - Servo motors normally require a voltage of +5V.
 - Connect the servo's red (power) wire to the Arduino's 5V output pin. This provides the appropriate power to the servo.
 - Connect the servo's brown wire (or ground) to an Arduino ground (GND) pin. This completes the circuit and connects the servo to the ground.

- **Electrical**

- The Arduino Board
 - Responsible for controlling the movement of the servo motor.
 - Simple to program and offers fine control through built-in features.
- Servo Motor:
 - Performs physical movement at particular angles.
 - Designed for precision and provides feedback to help maintain position.
- Jumper Wires:
 - Connect components without soldering.
 - Ensures electrical connections are solid and reliable.
- Potentiometer (optional):
 - When used, allows for manual angle input.
 - Convenient interface for altering servo position.
- USB Cable:
 - Connects Arduino to PC for programming and powering.
 - Supports real-time monitoring and modifications.
- Power Supply and Ground Connection:
 - Powers and completes circuits.
 - Provides steady and dependable performance.

- **Software**

- **Arduino**

■ Include the Servo Library:

- **#include**

This line includes the Servo library, which contains functions for controlling servo motors.

■ Define Pin and Initialise Servo.

- **#Define SER 9:**

This line specifies the pin number (9 in this example), to which the servo motor's signal wire is linked.

- **Servo servo; :**

This line sets up a Servo object named servo.

■ Global Variables:

- **int mssg; :**

This variable holds the angle data obtained via serial transmission.

■ Setup Function:

- **void setup():**

This is the setup function, which is called once when the Arduino is turned on or reset.

- **servo.attach(SER); :**

Attaches the servo object to the Arduino's specified pin (SER).

- **Serial.begin(9600); :**

This line sets up serial communication at a baud rate of 9600 bits per second.

- Loop Function:

- **void loop()**

- : This is the loop function, which is called repeatedly following the setup function.

- **If (Serial.available()>0)**

- : This condition determines if there is data available to be read from the serial port.

- **mssg = Serial.parseInt();**

- : This line receives an integer value from the serial port and assigns it to the mssg variable.

- **servo.write(mssg);**

- : This line changes the position of the servo motor to the value contained in the mssg variable.

- **delay(50);**

- : This line adds a slight delay (50 milliseconds) to help stabilise the servo motor's movement.

- **Python**

- Import Serial Library and Time Module:

- **import serial:**

- This line loads the serial library, which contains facilities for serial communications.

- **from time import sleep:**

This line imports the sleep function from the time module, which is used to create delays in the program.

■ Open Serial Port:

- **Ser = serial.Serial('COM3', 9600):**

This line establishes a serial connection with the Arduino board via the COM3 port at a baud rate of 9600 bits/second.

- **sleep(5):**

This line adds a 5-second wait to make sure the serial connection is properly established.

■ User Input and Serial Transmission:

- **print("Enter a number between 0-180, or 'e' to exit the program\n"):**

This line produces a prompt asking the user to enter a number between 0 and 180, or 'e' to quit the programme.

- **Enter = input().**

This line reads user input from the console.

■ Loop for User Interaction:

- **while enter!= 'e'::**

This line initiates a loop that will run until the user enters 'e' to exit.

- **If int(enter) is more than 0 and less than 180, then:**

This condition determines if the user input is an integer between 0 and 180.

- **`ser.write(str(enter).encode()):`**

If the condition is fulfilled, this line delivers the user's input (converted to a string and encoded) to the Arduino board over the serial port.

- **`print("I have sent: " + str(enter));`**

This line produces a confirmation message that includes the angle data given to the Arduino.

- **`else:`**

If the user input is not an integer between 0 and 180:

- **`print("Enter a number between 0-180, or 'e' to exit");`**

This line displays an error notice, requesting the user to provide a correct number.

- **`Print("Enter a number between 0-180:");`**

This line encourages the user to enter another number.

- **`Enter = input();`**

This line reads user input for the loop's next iteration.

QUESTION

Enhance your Arduino and Python code to incorporate a potentiometer for real-time adjustments of the servo motor's angle. Ensure that, in the updated Arduino code, you have the ability to halt its execution by pressing a designated key on your computer's keyboard. Following the modification, restart the Python script to receive and display servo position data from the Arduino over the serial connection. While experimenting with the potentiometer, observe the corresponding changes in the servo motor's position.

ANS:

Here's a step-by-step explanation of our enhanced code

Arduino Code:

- Declare Variables:

```
Servo myservo;  
  
int potValue = 0;  
  
int angle = 0;
```

- Define Potentiometer Pin:

```
int potPin = A0;
```

- Setup Function:

```
void setup() {  
  
    Serial.begin(9600);  
  
    myservo.attach(9);  
  
}
```

- Loop Function:

```
void loop() {  
  
    // Read the potentiometer value  
  
    potValue = analogRead(potPin);
```

```

// Map the potentiometer value to servo angle (0 to 180)

angle = map(potValue, 0, 1023, 0, 180);

// Send servo angle to Python over serial

Serial.println(angle);

// Read from serial for halt command

if (Serial.available() > 0) {

    char command = Serial.read();

    if (command == 'H' || command == 'h') {

        while (Serial.available() > 0) Serial.read(); // Clear serial buffer

        while (true) {} // Halting execution

    }

}

// Set servo angle

myservo.write(angle);

// Some delay for stability

delay(1000);

}

```

- Python Code:

1. Import Necessary Libraries:

```
import serial
```

```
import time

import sys

import threading
```

2. Define Serial Reader:

```
# Function to read from serial

def serial_read(ser):

    while True:

        try:

            # Read from serial and decode bytes to string

            data = ser.readline().decode('utf-8').strip()

            print("Servo angle:", data)

        except UnicodeDecodeError:

            pass
```

3. Main Function:

```
def main():

    try:

        # Establish serial connection

        ser = serial.Serial('COM3', 9600, timeout=1)

        print("Serial port opened successfully")
```

```

# Start thread to continuously read from serial

serial_thread = threading.Thread(target=serial_read, args=(ser,),
daemon=True)

serial_thread.start()


# Main loop to check for keyboard input
while True:

    # Check if keyboard input is available

    if sys.stdin in select.select([sys.stdin], [], [], 0)[0]:

        key = sys.stdin.read(1)

        if key.lower() == 'q':

            break # Quit if 'q' is pressed

        elif key.lower() == 'h':

            ser.write(b'H') # Send halt command to Arduino


# Close serial connection

ser.close()

print("Serial port closed")


except serial.SerialException as e:

    print("Error opening serial port:", e)

```

4. Establish Entry-point:

```
if __name__ == "__main__":  
    main()
```

CONCLUSION

In conclusion, the experiment demonstrated a solid structure for real-time data gathering and visualisation that smoothly integrated Arduino and Python technologies. The implementation of serial communication and the use of matplotlib for graphing provided a realistic avenue for monitoring dynamic events and conducting meaningful analysis. The installation of a graphical user interface (GUI) improved user involvement by providing easy control over experiment parameters and data visualisation settings. Furthermore, the addition of data recording and exporting features assured data retention and permitted offline analysis, emphasising the experiment's importance in scientific research, technical applications, and instructional settings.

RECOMMENDATION

- Instead of Matplotlib, we could create a basic graphical interface with Python modules such as Tkinter or PyQt. This allows us to simply change experiment settings, start/stop data collecting, and tweak plot parameters.
- Implement a functionality that allows you to log data to a file (for example, CSV format) straight from your Python script. Also, we may export plotted data or snapshots for offline analysis. This assures data protection and allows users to undertake detailed analyses using other tools.

- Use checksums or CRC to ensure data integrity, and build handshake procedures for reliable transfer. This guarantees precise data transport, particularly in loud surroundings.

REFERENCES

[1]“Servo Motor Basics with Arduino | Arduino Documentation,” *docs.arduino.cc*.

<https://docs.arduino.cc/learn/electronics/servo-motors/>

[2]“Using Servos With CircuitPython and Arduino Created by Tony DiCola.” Accessed: Mar. 25, 2024. [Online]. Available:

<https://cdn-learn.adafruit.com/downloads/pdf/using-servos-with-circuitpython.pdf>

[3]“13.6 KB file on MEGA,” mega.nz.

<https://mega.nz/file/yugWkJDJ#TTDOamAu27zf70pivSavlGCzs1X6RHp2RhBMdYQ0F5A>

(accessed Mar. 25, 2024).

ACKNOWLEDGEMENT

A special thanks goes out to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this report. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work. Their time, patience, and commitment to supporting my academic success are greatly appreciated.

STUDENT'S DECLARATION


Certificate of Originality and Authenticity


This is to certify that we are responsible for the work submitted in this report, that **the original work** is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.


We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and that no further improvement on the reports is needed from any of the individual contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us**.

Signature: 	Read	/
Name: Noramira Husna Bt Norkhairani	Understand	/
Matric No: 2214496	Agree	/

Signature: 	Read	/
Name: Nur Shaheera Bt Kamis	Understand	/
Matric No: 2214040	Agree	/

Signature: 	Read	/
Name: Muhammad Eiqbal bin Hasbollah	Understand	/
Matric No: 2216911	Agree	/