MCTA 3203

MECHATRONIC SYSTEM INTEGRATION

SECTION 1

SEMESTER 2 2024/2025

LECTURER:

SIR WAHJU SEDIONO

SIR ZULKIFLI BIN ZAINAL

LAB REPORT 3B:

SERIAL COMMUNICATION (SERVO MOTOR)

GROUP 2

DATE OF SUBMISSION:

24 MARCH 2025

GROUP MEMBERS	MATRIC NUMBERS
AMIRUL IMAN BIN MAT KHIR	2316633
AHMAD IRFAN FAHMI BIN AHMAD USRI	2318631
MUHAMAD NUR HAFIZ BIN ROSLY	2317001
FAKHRULLAH BIN YUZAIRI	2220289

TABLE OF CONTENTS

INTRODUCTION

Serial communication plays a crucial role in mechatronic systems, enabling seamless data transfer between microcontrollers and computers. In this experiment, we focus on using serial communication to control a servo motor via an Arduino board. By sending angle data from a Python script to the Arduino, the servo motor can be actuated to specific positions. This experiment highlights the practical application of parallel, serial, and USB interfacing in mechatronic system integration.

ABSTRACT

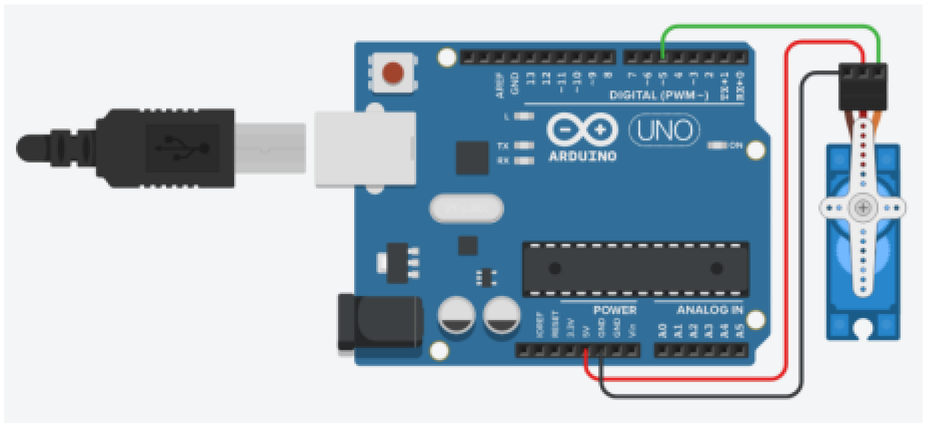
This experiment explores the integration of a servo motor with an Arduino microcontroller and a Python-based computer system to control the servo's position using serial communication. The setup involves connecting a servo motor to an Arduino board, writing Arduino code to interpret angle data received via serial communication, and developing a Python script to send angle commands to the Arduino. The experiment demonstrates how serial communication facilitates real-time control of hardware components, such as servo motors, from a computer. Additionally, the experiment encourages further enhancement by incorporating a potentiometer for manual angle adjustments and integrating keyboard input to halt the system, showcasing the versatility of microcontroller-computer interactions in mechatronic systems.

MATERIALS AND EQUIPMENT

The following components are required for the experiment:

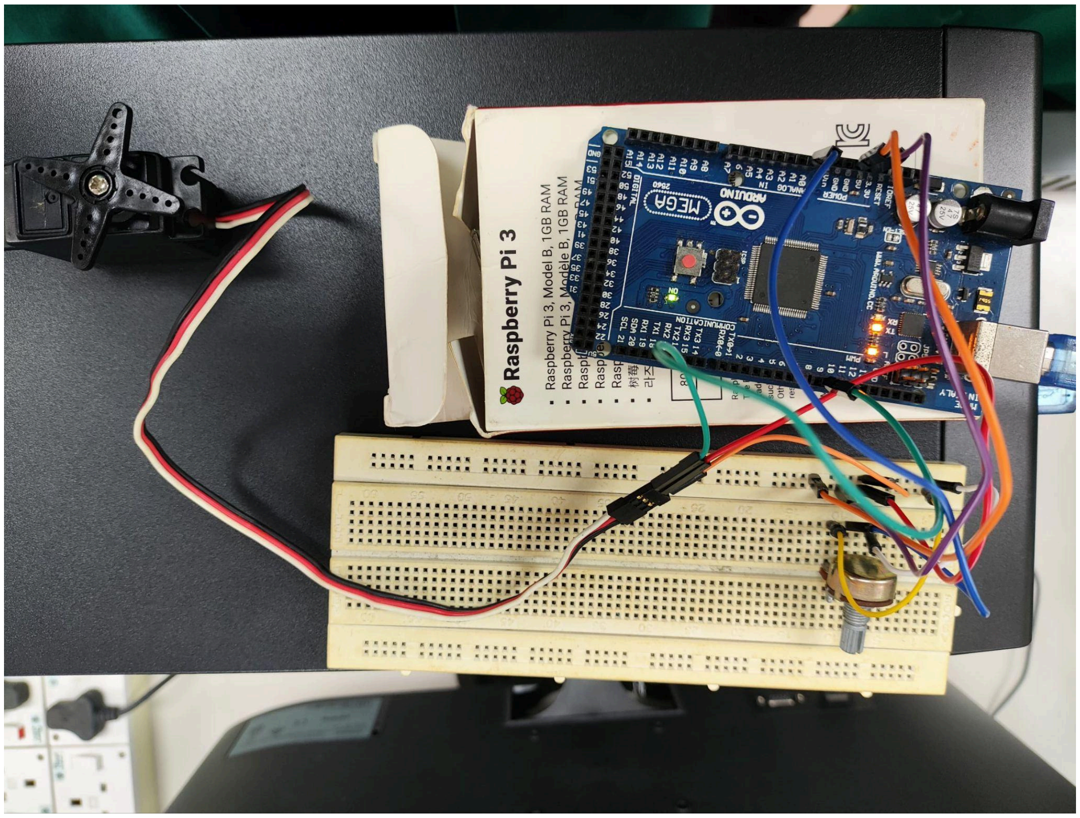
- Arduino board (e.g., Arduino Uno)
- Servo motor
- Jumper wires
- Potentiometer (for manual angle input)
- USB cable for Arduino
- Computer with Arduino IDE and Python installed

EXPERIMENTAL SETUP AND METHODOLOGY



- Hardware Setup:
    - Connect the servo motor’s signal wire to a PWM-capable pin on the Arduino (e.g., digital pin 9).
    - Power the servo using the Arduino’s 5V and GND pins.
    - Connect the servo’s ground wire to one of the Arduino’s GND pins.
- Software Setup:
    - Install the necessary Arduino libraries:
        - Open Arduino IDE.
        - Navigate to "Sketch" > "Include Library" > "Servo" to install the Servo library if not already installed.
    - Write and upload the Arduino code to read angle data from the serial port and move the servo accordingly.
    - Install the required Python library:
        - Use the command pip install pyserial to install the pyserial package.
    - Write a Python script to send angle values to the Arduino via serial communication.
- Execution:
    - Run the Python script and input an angle between 0-180 degrees.
    - Observe the servo motor’s response based on the provided angle.
    - Exit the script by entering ‘q’, which terminates the serial connection.

RESULT



During the experiment, the servo motor successfully responded to the angle values sent from the Python script via serial communication. The key observations were:

1. Servo Movement Accuracy:
    - When an angle between 0 and 180 degrees was entered in the Python script, the servo motor adjusted accordingly.
    - The movement was smooth, with minimal delay between input and response.
2. Serial Communication Performance:
    - The data transmission between the computer and Arduino was stable, with no noticeable lag or errors.
    - The serial port correctly processed numerical inputs, but invalid values (e.g., out-of-range numbers or non-numeric characters) were ignored.

3. Potentiometer Integration:
    - When the potentiometer was used, the servo motor dynamically adjusted its position in real-time.
    - The serial monitor displayed the changing angle values, confirming proper data transmission.
4. Script Termination:
    - Entering ‘q’ successfully closed the serial connection, preventing further communication errors.

QUESTION

Arduino code:

```
 #include <Servo.h>
 Servo myServo; // Create a servo object
 int potPin = A0; // Pin where the potentiometer is connected
 int potValue = 0; // Value read from the potentiometer
 int angle = 0; // Servo angle (0- 180)
 bool isRunning = true; // Flag to control program execution
 8
void setup() {
 Serial.begin(9600); // Start serial communication
 myServo.attach(9); // Attach the servo to pin 9
 Serial.println("Servo Control Started");
 Serial.println("Enter 's' to stop the program");
 }
 void loop() {
 if (Serial.available() > 0) {
```

```
char input = Serial.read();
if (input == 's' | input == 'S') { // Accept both lower and uppercase 's'
isRunning = false;
myServo.write(90); // Move servo to neutral position when stopping
Serial.println("Program stopped. Reset Arduino to restart.");
while(true) { // Hold program in infinite loop
delay(1000);
}
}
}
if (isRunning) {
potValue = analogRead(potPin);
// Read the potentiometer value (0-1023)
angle = map(potValue, 0, 1023, 0, 180); // Map the pot value to a range for the servo (0-180)
myServo.write(angle);
// Set the servo angle
// Print formatted output
Serial.print("Servo Angle: ");
Serial.println(angle);
delay(100); // Small delay for stability
}
}
```

This Arduino code sets up a basic servo control system using a potentiometer and serial communication. It starts by creating a Servo object called myServo, which is then attached to pin 9 on the Arduino board, and connects a potentiometer to analog pin A0 to read its analog input values. The code initializes serial communication at a baud rate of 9600, allowing messages to be sent between the Arduino and a computer via the serial monitor. Upon startup, the program displays a message indicating that the servo control has begun and instructs the user to enter 's' to stop the program.

In the main program loop, the code listens for serial input. If it detects the letter 's' (in either lowercase or uppercase), it stops the program by setting the isRunning flag to false, moves the servo to a neutral position (90°), and displays a message indicating that the program has stopped. It then enters an infinite loop, effectively halting further execution until the Arduino is reset. This pause feature allows the user to easily stop the servo's movement through a simple serial command.

When the program is running (isRunning is true), it reads the potentiometer's analog value (ranging from 0 to 1023), maps this value to a corresponding servo angle (from 0 to 180°), and adjusts the servo's position accordingly. The current angle is also printed to the serial monitor, providing real-time feedback to the user. By turning the potentiometer, the user can control the servo's position dynamically, while the serial monitor displays the servo angle. This setup offers a straightforward and interactive way to manage servo motion with both physical and serial controls.

DISCUSSION

In this experiment, we explored the control of a servo motor through serial communication between an Arduino and a computer. The experiment was divided into two stages. In the first stage, the Arduino received angular data from a Python script, which controlled the servo motor's movement to a specified angle.

The second stage introduced a potentiometer to manually adjust the servo angle. These two methods illustrated key aspects of microcontroller-based hardware management, highlighting the differences between dynamic, sensor-based adjustments and programmed control inputs.

To allow user control over the servo's position between 0 and 180 degrees, we initialized serial communication in the first stage. The Python script transmitted angle data to the Arduino, demonstrating how external data inputs from a computer interface can precisely control hardware components. Maintaining a continuous connection was crucial, which was achieved by synchronizing the baud rates of Python and Arduino to ensure uninterrupted data flow. While this method showcased automation potential, it required constant user interaction, making it less suitable for applications demanding autonomous responses to environmental changes.

In the second stage, a potentiometer served as an analog input, enabling direct control of the servo without additional computer input. By converting potentiometer readings into angle values, we created a continuous, real-time feedback loop that adjusted the servo position in response to the potentiometer's movement. This approach provided dynamic, manual control, which is beneficial for user-operated robotic systems. However, while the potentiometer allowed real-time responsiveness, it was less programmable than Python, making it more suited for hands-on regulation rather than automated tasks requiring external command integration.

RECOMMENDATION

The experiment on serial communication between a computer and an Arduino microcontroller for servo motor control has proven to be effective and practical. However, improvements can be made to enhance its functionality, accuracy, and efficiency.

One key improvement is integrating a potentiometer for real-time adjustments. This allows users to manually control the servo's movement, making the system more interactive. Additionally, implementing two-way communication will enable the Arduino to send feedback on the servo's position, improving monitoring and accuracy.

A graphical user interface (GUI) could make the system more user-friendly by providing sliders or buttons to adjust the servo position instead of using a command-line interface. Furthermore, incorporating a safety mechanism, such as predefined angle limits or an emergency stop function, can prevent damage to the servo motor.

Enhancing the system with wireless communication, using Bluetooth or Wi-Fi modules, would allow remote control, making the experiment more flexible. Expanding the setup to control multiple servos simultaneously could also be useful for advanced applications like robotic arms.

Adding data logging features can help track performance and identify inconsistencies over time. Moreover, integrating machine learning algorithms could optimize servo movements based on previous patterns, making the system more adaptive and efficient.

By implementing these recommendations, the experiment can be significantly improved, making it more reliable and suitable for real-world applications. These enhancements will provide a strong foundation for more advanced mechatronics projects and automation systems.

CONCLUSION

The practical use of serial communication between a computer and a microcontroller to operate a servo motor is effectively illustrated by this example. The demonstration demonstrates the smooth integration of software and hardware in mechatronic systems by using Python to transmit angle commands and Arduino to decipher and carry out these commands. The fact that a basic Python script can drive the servo motor in real-time highlights the value of serial connection in contemporary engineering applications where automation and precise control are crucial.

By including a potentiometer for manual changes and keyboard input to stop the system, the experiment also promotes more research. This improvement highlights the adaptability of such systems in practical situations in addition to adding an element of interactivity. For example, comparable configurations can be used to precisely operate robotic arms or other mechanical components in industrial automation. The use of potentiometer-based manual control further emphasises the value of human-machine interaction by keeping systems flexible and user-friendly in the face of changing needs.

From an Islamic perspective, this experiment aligns with the concept of *Tawhid* (the Oneness of Allah) by showcasing the harmony and balance between different components—software, hardware, and human input—working together to achieve a common goal. It reflects the idea that all creations, including technological systems, are interconnected and function under the divine order of Allah. Additionally, the experiment embodies the Islamic principle of *Ihsan* (excellence) by encouraging continuous improvement and precision in engineering practices. By striving to enhance the system's functionality and usability, engineers and developers fulfill their role as stewards (*Khalifah*) of technology, utilizing it responsibly and ethically to benefit society. This experiment serves as a reminder that technological advancements should always be guided by ethical principles and a sense of responsibility toward humanity and the environment.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Sir Wahju Sediono for their guidance and support throughout this project. Our thanks also extend to the teaching assistants, for their constructive feedback and assistance,which greatly contributed to the completion of this work.

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 no further improvement on the reports is needed from any of the individual's 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.**

Name:Amirul Iman Bin Mat Khir — Read [✓]

Matric Number: 2316633 — Understand [✓]

Signature: [illegible] — Agree [✓]

Contribution:Title,abstract,introduction,methodology,acknowledgement

Name:Ahmad Irfan Fahmi Bin Ahmad Usri Read [✓]

Matric Number: 2319531 Understand [✓]

Signature: [illegible] Agree [✓]

Contribution:Result,Discussion,Material and Equipment

Name:Muhamad Nur Hafiz Bin Rosly Read [✓]

Matric Number: 2317001 Understand [✓]

Signature: [illegible] Agree [✓]

Contribution:Recommendation,Conclusion

Name: Fakhrulah Bin Yuzairi Read [✓]

Matric Number: 2220289 Understand [✓]

Signature: [illegible] Agree [✓]

Contribution:Intro,Materials,Experimental setup,Result