LAB REPORT 3b :
SERIAL COMMUNICATION (SERVO MOTOR)

GROUP 5

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 23 OCTOBER 2024

NO	NAME	MATRIC NUMBER
1.	MUHAMMAD AFIQ ADHAM BIN MOHD NADZRI	2227531
2.	MUHAMMAD AMIN BIN MOHAMAD RIZAL	2217535
3.	MUHAMMAD AFIQ BIN MOHD ASRI	2212541
4.	MUHAMMAD AKMAL BIN MOHD FAUZI	2214077
5.	MUHAMMAD AFIQ IKHWAN BIN NOR SHAHRIZAL	2215897

TABLE OF CONTENT

INTRODUCTION

The objective of this experiment is to establish and demonstrate effective serial communication between a microcontroller and a servo motor, allowing precise control over the motor's position through serial commands. Additionally, it involves implementing code to control the servo’s position via serial commands, evaluating the motor’s response time, accuracy, and range, and analyzing the data transmission quality to identify any issues related to latency, interference, or signal integrity.

ABSTRACT

This experiment explores the use of serial communication for controlling a servo motor, focusing on transmitting position data from a microcontroller to a servo motor. The setup enables precise servo positioning via serial commands, facilitating applications where exact movement control is essential. By implementing code to communicate positional commands, the experiment examines the servo's response accuracy, timing, and range of motion under various conditions. Additionally, the study analyzes the quality of serial data transmission, identifying potential issues like latency, interference, and signal degradation that may impact performance. This experiment provides insights into the practical considerations of using serial communication in motor control applications.

MATERIALS AND EQUIPMENT

- Arduino Uno Board
- Servo Motor
- Potentiometer
- Jumper Wires
- USB cable for Arduino
- Computer with Arduino IDE and Python installed

EXPERIMENTAL SETUP

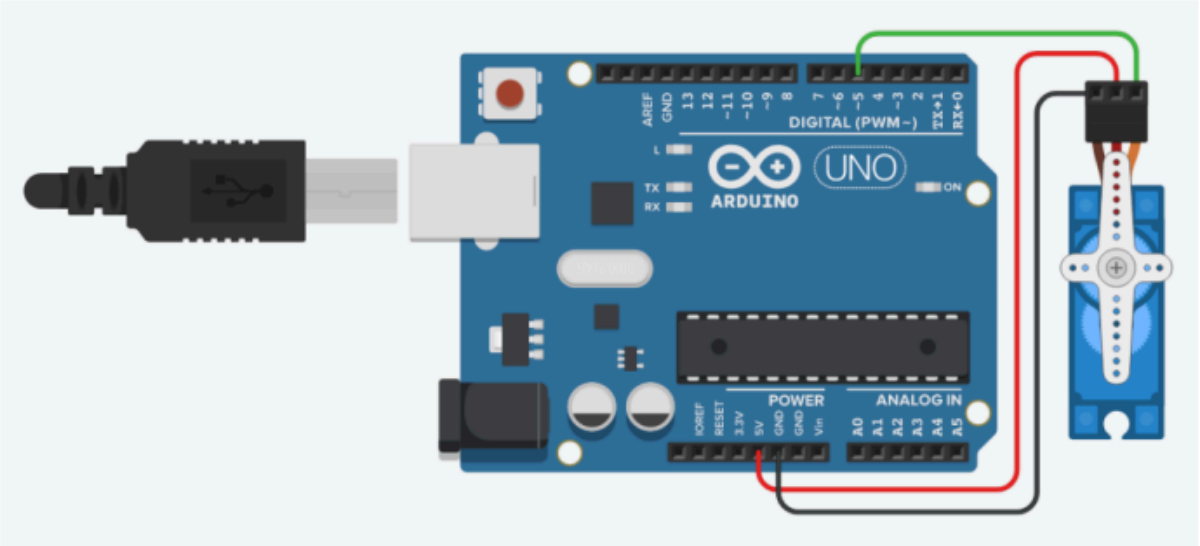


Fig. 1

1. Connect one leg of the servo motor to 5V on the Arduino.
2. Connect the other leg of the servo motor to GND on the Arduino.
3. Connect the servo's signal wire to a PWM capable pin on the Arduino, which is pin 5. An example of the circuit setup is shown in Fig 1.

METHODOLOGY

1. Setup for Arduino Board
    - Connect the Servo Motor as mentioned in experimental setup.
    - Connect the Arduino Board to the computer via USB cable

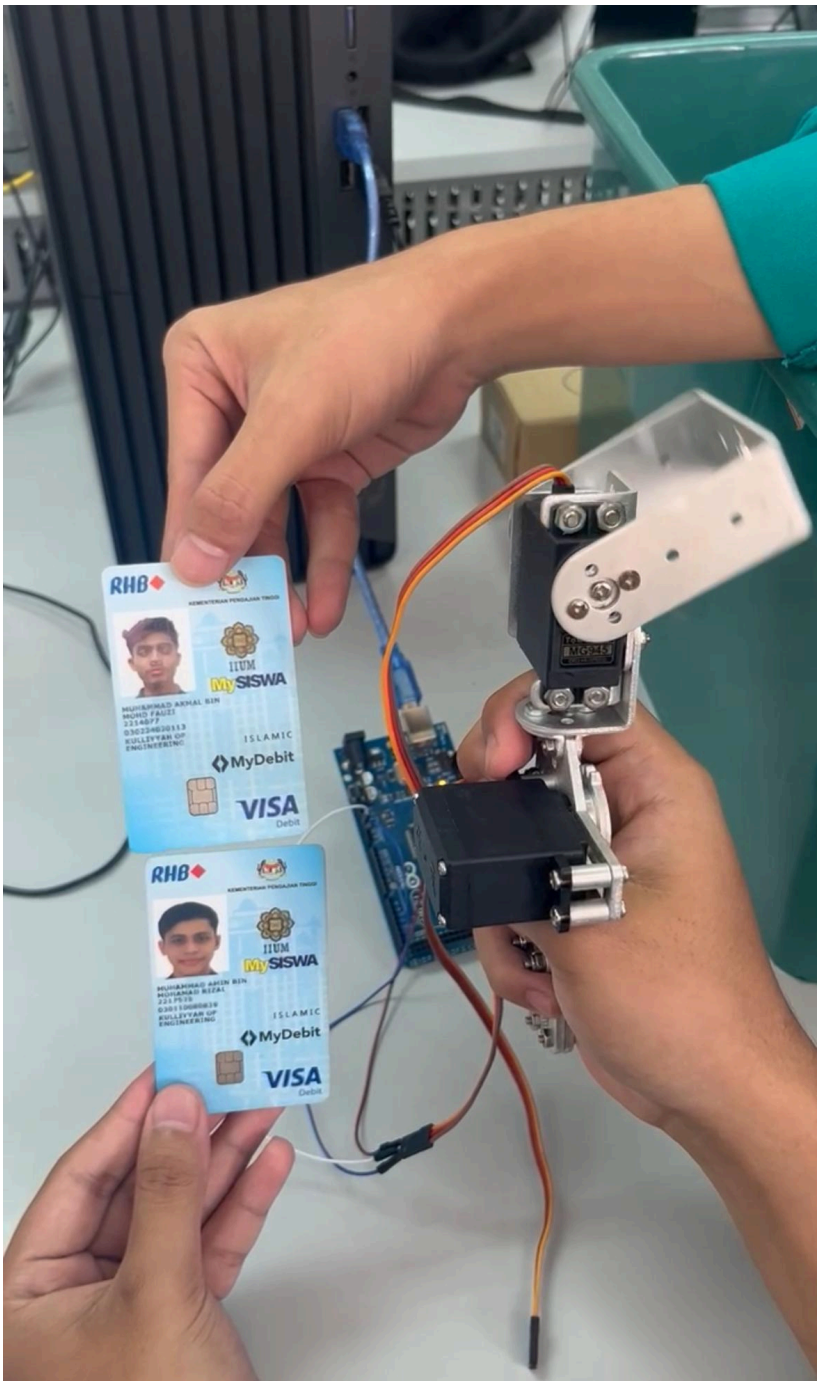
2. Software Programming Arduino IDE
    - Open a new sketch in the Arduino IDE.

- Write Arduino code that reads angle data (90) from the serial port and moves the servo accordingly.
- Include a function that ask the servo to reverse the angle data.
- Upload the code to the Arduino Board.

3. Arduino Execution
    - Record the movement of Servo Motor, make sure the movement follows the angle that has been set.
    - Change the angle for a better result.

4. Python Execution
    - Write a program using python that includes input of an angle (0 to 180 degrees).
    - Press Enter so that the Python script will send the angle to the Arduino over the serial port.
    - Record the angle of servo that will be displayed in the python script.
    - Put multiple angles to see the servo move accordingly and compare it with Arduino IDE.

RESULT

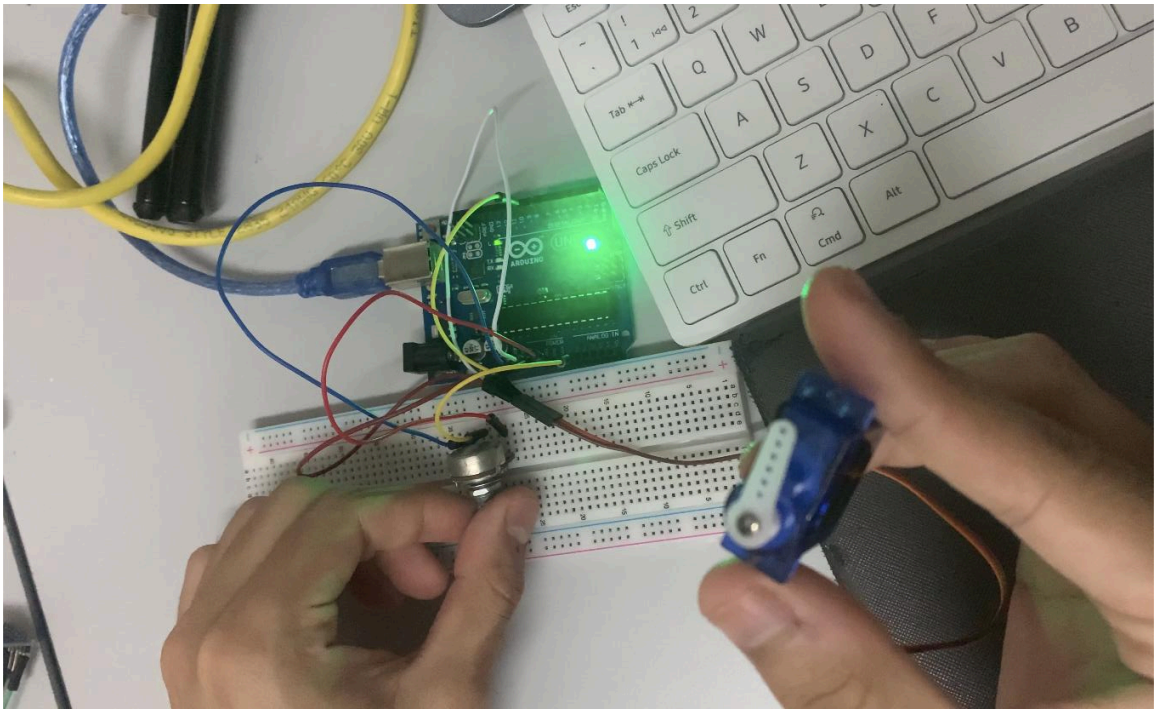


Link Video:

https://github.com/GROUP5-MSI/WEEK-3-Serial-Communication/blob/main/Video%20W3%20Servo.mp4

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.



Link Video:https://github.com/GROUP5-MSI/WEEK-3-Serial-Communication/blob/main/Video%20Potentiometer%20Control%20the%20Servo.mov

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
```

```
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 lab, we investigated controlling a servo motor via serial communication between an Arduino and a computer. We divided the experiment into two phases. In the first step, the Arduino received angle data from a Python script and used it to command the servo motor to rotate at the specified angle. The second component had a potentiometer for controlling the servo angle in real-time. Both approaches illustrated different features of hardware control using microcontroller-based systems, highlighting the distinction between dynamic, sensor-based changes and programmed control inputs.

To allow the user to regulate the servo's position between 0 and 180 degrees, we established serial communication in the first experiment to transfer angle data from the Python script to the Arduino. This technology revealed how external data inputs from a computer interface can be utilized to influence hardware components by providing the servo with precise, programmable control. Maintaining consistent connection was critical, and this was achieved by balancing Python and Arduino baud rates, allowing for continuous data flow. This approach demonstrated how automation might be developed, but it was limited to applications that needed to react quickly to changes in the environment due to the necessity for constant user input.

The experiment's second section incorporated a potentiometer as an analog input, enabling real-time control of the servo without the need for further computer input. We may establish a continuous and immediate feedback loop by mapping potentiometer readings to angle values and then adjusting the servo position in response to changes in the potentiometer's position. This technique enabled dynamic, manual control in situations that need responsive alterations, such as user-controlled robotic systems. Despite its ability to respond in real-time, the potentiometer was less programmable than Python, making it better suited for manual control but less flexible for automated activities that required an interface with external orders.

CONCLUSION

In conclusion, this set of experiments demonstrates the concepts of serial communication between a computer and a microcontroller using Python and Arduino. By connecting a potentiometer and a servo motor to the Arduino, we investigated two fundamental characteristics of sensors and actuators in embedded systems.

The second experiment used servo motor control, which was based on angle input from a Python script to the Arduino. This conversation revealed an important aspect of PWM signals and how software can modify physical elements. Changing the angle using a potentiometer demonstrated how input devices may be used to influence outputs in a controlled way.

These experiments, in general, show the value of serial communication in bridging the hardware-software gap. After the projects are completed successfully, we will be prepared for advanced-level applications. This improves our understanding of microcontroller programming and data interchange in embedded devices.

RECOMMENDATION

Several proposals would improve the experiment's reliability and effectiveness. First, double-check the connections before powering on the circuit, as incorrect wiring might result in poor functioning or damage. Next, the lab must provide a multimeter to examine voltage levels at various locations, which could reveal the source of problems in the power supply or improper resistor values. It is critical to ensure that the resistor values are accurate. Erroneous values will result in a dull display or damage. Use the resistor required to prevent the LED from being overpowered.

Once the fundamental functionality is built, testing various scenarios can identify potential issues far earlier. Following these criteria results in a much more minimal, productive, and successful experimental experience, as well as significant learning opportunities.

ACKNOWLEDGEMENTS

A special thank you to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and peers, for their outstanding assistance and support in completing this report. Their advice, input, and expertise have had a significant impact on the quality and comprehension of this work. Their time, patience, and dedication to my academic progress are deeply 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 acknowledgment, and that the original work contained herein has 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:
Name: MUHAMMAD AFIQ ADHAM BIN MOHD NADZRI
Matric Number: 2227531

Read ☑
Understand /
Agree ☑

Signature:
Name: MUHAMMAD AMIN BIN MOHAMAD RIZAL
Matric Number: 2217535

Read ☑
Understand ☑
Agree ☑

Signature:
Name: MUHAMMAD AFIQ BIN MOHD ASRI
Matric Number: 2212541

Read ☑
Understand ☑
Agree ☑

Signature:
Name: MUHAMMAD AKMAL BIN MOHD FAUZI
Matric Number: 2214077

Read ☑
Understand ☑
Agree ☑

Signature: Afiq
Name: MUHAMMAD AFIQ IKHWAN BIN NOR SHAHRIZAL
Matric Number: 2215897

Read ☑
Understand ☑
Agree ☑