MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

WEEK 3:

SERIAL COMMUNICATION

SECTION 1

SEMESTER 1, 2025/20256

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

DATE OF SUBMISSION: 28 OCTOBER 2025

GROUP 8

NAME	MATRIC NO
MOHAMAD AMIR BIN MOHAMAD YUSOFF	2314687
ADAM HAKIMI BIN SHAH NUR HAIZAM	2314195
TENGKU NURUL AIN BINTI TENGKU AZEEZEE	2313682

Table Of Contents

Abstract

This project investigated using Python to establish serial connection between an Arduino microcontroller and a PC. A potentiometer was utilized as an analog input device in Experiment 1, and Python was used to display the real-time readings. Furthermore, threshold sensor results were used to drive an LED. In Experiment 2, angle data transmitted from Python to the Arduino were used to operate a servo motor. The experiment showed how serial transmission, actuator control, and data collecting may be combined. The outcomes demonstrated precise actuator control and seamless data transfer. This project serves to emphasize basic mechatronics concepts related to actuation, processing, and sensing.

Introduction

The main objective of the experiment is to show how an Arduino microcontroller and a computer may exchange data via serial communication. Because it enables real-time collaboration between sensors, controllers, and actuators, serial communication is crucial in mechatronic systems.

There were two phases to the experiment. A potentiometer was employed as an analog input sensor in Experiment 1. The potentiometer value was read by the Arduino and transmitted to Python, where it was shown on the computer. Whether the sensor value was above or below a predetermined threshold also determined the control of an LED. This illustrated how basic control decisions can be made using input signals.

In Experiment 2, the direction of communication was reversed. A servo motor was controlled by the Arduino using angle directives that Python gave to it. This demonstrated how serial connectivity can be used to control physical motion from a computer.

A fundamental idea in mechatronics and embedded systems, the experiment illustrated how sensor data may be collected, transferred, analyzed, and utilized to drive actuators.

TASK 1

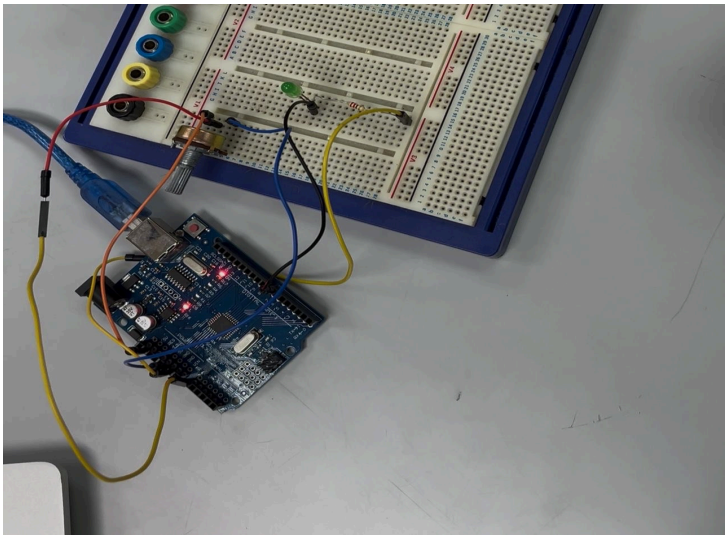
Materials and Equipment

1. Arduino board
2. Potentiometer
3. LED
4. 220 Ω resistor
5. Jumper wires
6. Breadboard

Experimental Setup

1. The potentiometer was connected to the Arduino with its left pin connected to the 5V supply, its right pin connected to GND, and the middle wiper pin connected to the analog input A0. This configuration allows the Arduino to read variable voltage values corresponding to the potentiometer's position.
2. An LED was connected to digital pin 9 of the Arduino through a 220 Ω resistor in series to limit the current and protect both the LED and the microcontroller. The LED's anode (long leg) was connected to pin 9, and the cathode (short leg) was connected to GND via the resistor.
3. The Arduino was connected to a computer via a USB cable to allow programming and serial communication. The uploaded Arduino sketch continuously read the potentiometer's analog value and sent it to the serial port.

4. The Python script was executed on the computer to read the potentiometer values from the Arduino's serial port. The script displayed the readings in real time in the Python terminal and logged the values for further analysis.
5. The Python script was programmed to monitor the potentiometer value and send commands to the Arduino to turn the LED ON when the reading exceeded half of its maximum range and OFF when below the threshold. This allowed automatic LED control based on potentiometer input.
6. Care was taken to ensure the Arduino Serial Plotter was closed while Python accessed the serial port, to prevent interference in communication.
7. The breadboard's power rails were used to distribute 5V and GND from the Arduino. All components shared a common ground to ensure consistent voltage references across the circuit.
8. After wiring and uploading the program, the potentiometer knob was adjusted to observe the real-time change in readings on the Python terminal and the LED response according to the threshold.



Methodology

1. Overview

   This experiment involved using a potentiometer as an analog input to the Arduino, which sent the voltage readings to a computer via serial communication. A Python program was used to display the values in real time, while an LED connected to the Arduino indicated changes in the potentiometer output.

2. Hardware Setup

Arduino Uno	Main controller to read data and send it to the computer.
Potentiometer	Provides variable voltage input.
LED	Lights up based on potentiometer value.
Resistor	Limits current to the LED.
Breadboard	Platform to connect all components.
Jumper Wires	Connect components to the Arduino.
USB Cable	Powers Arduino and enables serial communication.

3. Circuit Design

   The potentiometer's middle pin was connected to analog input A0, with the outer pins connected to 5V and GND. An LED was connected to digital pin 8 through a 220 Ω resistor. The Arduino Uno was powered via USB, which also provided serial communication with the computer.

4. Program Development

Program Code for Real-Time Graph Display of Potentiometer Readings

Arduino IDE code :

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

void loop() {
  int potValue = analogRead(A0);
  Serial.println(potValue);
  delay(1000);
}
```

PyCharm code :

```
# --- Read and Plot Data ---
sample = 0
try:
    while True:
        pot_value = ser.readline().decode().strip()
        if pot_value.isdigit():  # ensure it's numeric
            pot_value = int(pot_value)
            y_data.append(pot_value)
            x_data.append(sample)
            sample += 1

            line.set_xdata(x_data)
            line.set_ydata(y_data)
            ax.relim()
            ax.autoscale_view()
            plt.pause(0.01)  # update plot
except KeyboardInterrupt:
    print("Closing connection...")
    ser.close()
    plt.ioff()
    plt.show()
```

```
import serial
import time
import matplotlib
matplotlib.use('TkAgg')  # or 'Qt5Agg' if you prefer
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

# --- Setup Serial Connection ---
ser = serial.Serial( port: 'COM5',  baudrate: 9600)  # change COM3 to your port
time.sleep(2)  # allow time for Arduino to reset

# --- Initialize plot ---
plt.ion()  # turn on interactive mode
fig, ax = plt.subplots()
y_data = []
x_data = []
line, = ax.plot(x_data, y_data, 'b-', label="Potentiometer Value")

ax.set_xlabel("Samples")
ax.set_ylabel("Value")
ax.set_title("Live Potentiometer Data")
ax.legend()
```

Data Collection.

1. A real-time graph plotted using matplotlib showed smooth variation of potentiometer readings as the knob was rotated.
2. The potentiometer output increased smoothly with rotation, indicating stable analog-to-digital conversion.
3. The LED threshold functioned correctly.
4. The serial communication between Arduino and Python was reliable and consistent, with minimal data delay.

Potentiometer Rotation Level (approximate) (%)	LED status
0	OFF
50	DIMLY ON
100	ON

Data Analysis

The collected data confirms that:

1. The potentiometer provided a proportional increase in analog reading as the knob was rotated clockwise.
2. The LED turned ON when the potentiometer reading exceeded half of the maximum range.
3. At mid-rotation, the LED was dimly ON, indicating the reading was near the threshold level.
4. Serial communication between Arduino and Python functioned correctly, transmitting data without delay or error.

Analysis:

1. The potentiometer output increased linearly with rotation, demonstrating accurate analog-to-digital conversion by the Arduino.
2. The data showed stable communication between Arduino and Python, with no missing or noisy readings.
3. No irregularities were observed in the readings, indicating proper wiring and reliable sensor performance.

Results

The experiment successfully demonstrated the control of a servo motor and LED through serial communication between Arduino and Python.

1. The potentiometer provided smooth and proportional analog readings as the knob was rotated.
2. The real-time graph displayed on the Python matplotlib plot showed stable and continuous variation of the potentiometer output.
3. The LED responded accurately to the potentiometer input—remaining OFF at low levels, dimly ON at mid-rotation, and fully ON at maximum rotation.
4. The servo motor responded correctly to angle inputs received from the Python program, rotating precisely according to the specified values.
5. Serial communication between Arduino and Python was stable and reliable, with minimal delay and no data loss observed during operation.

The experiment objectives were fully achieved without any errors or malfunctions. The potentiometer, LED, and servo motor operated smoothly, demonstrating successful interfacing between the Arduino and Python for real-time control and data visualization.

TASK 2

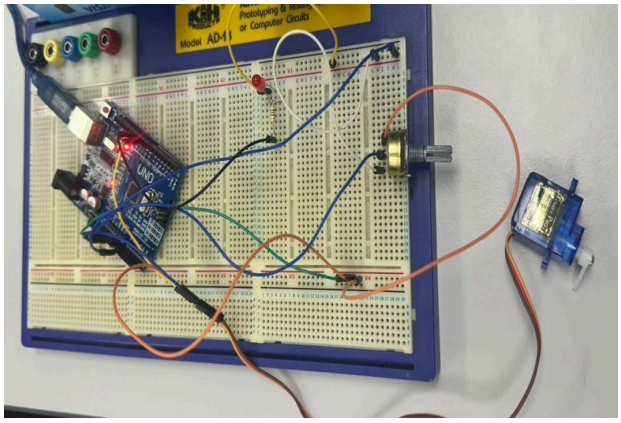
Materials and Equipment

1. Arduino Board
2. USB cable
3. Servo Motor
4. Jumper Wires
5. Potentiometer
6. LED
7. 220 Ω resistor
8. Breadboard

Experimental Setup

1. The servo motor was connected to the Arduino with its signal pin connected to digital pin 9, VCC connected to 5V, and GND connected to GND on the Arduino. This configuration allows the Arduino to control the servo's position by sending PWM signals through pin 9.
2. The potentiometer was connected to the Arduino with its left pin connected to 5V, right pin connected to GND, and middle wiper pin connected to analog input A0. This setup enables the Arduino to read variable voltage values corresponding to the potentiometer's position, allowing for real-time angle adjustments.
3. An LED was added to the circuit with its anode connected to a digital output pin (e.g., pin 10) through a 220 Ω resistor in series and its cathode connected to GND. This LED provides visual feedback based on potentiometer readings or servo position.

4. The Arduino was connected to a computer via USB to allow programming and serial communication. The Arduino sketch used the Servo library to control the servo motor and read potentiometer values. It also received angle commands from Python via the serial port.
5. The Python script was executed on the computer to send servo angle commands to the Arduino, read potentiometer values in real time, and optionally control the LED. It provided a user interface to manually input angles (0–180°) and visualized potentiometer readings using real-time plots via matplotlib.
6. The breadboard's power rails were used to distribute 5V and GND from the Arduino. All components shared a common ground to ensure consistent voltage references across the circuit.
7. Care was taken to ensure that only one program accessed the serial port at a time. The Arduino Serial Plotter was closed when the Python script was running to prevent communication conflicts.
8. After wiring and uploading the Arduino sketch, the Python script was run. Users could enter angle values to control the servo manually, or adjust the potentiometer to change the servo's position in real time. Observations of the servo movement, LED response, and live potentiometer readings were recorded throughout the experiment.



Methodology

1. Overview

   This experiment focused on controlling a servo motor through serial communication between Python and the Arduino. The servo angle was adjusted based on user input and later through a potentiometer for real-time control, with an LED used to indicate the servo's position or active operation.

2. Hardware Setup

Arduino Uno	Main controller to read data and send it to the computer.
Servo Motor	Rotates according to the control signal.
Potentiometer	Provides variable voltage input.
LED	Indicates servo activity or angle position.
Resistor	Limits current to the LED.
Breadboard	Platform to connect all components
Jumper Wires	Connect components to the Arduino.
USB Cable	Powers Arduino and enables serial communication.

3. Circuit Design

   The servo motor's signal wire was connected to digital pin 9, while the power and ground wires were connected to 5V and GND. The potentiometer was connected to A0 as in Task 1, and an LED with a 220 Ω resistor was connected to pin 8. The Arduino was powered and linked to the computer through a USB cable for control and data transfer.

4. Program Development

   Arduino IDE code :

```
#include <Servo.h>

Servo myservo;
int potPin = A0;    // Potentiometer connected to A0
int servoPin = 9;   // Servo control pin
int ledPin = 6;     // LED pin (PWM capable)

void setup() {
  Serial.begin(9600);
  myservo.attach(servoPin);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int potValue = analogRead(potPin);           // Read potentiometer (0-1023)
  int angle = map(potValue, 0, 1023, 0, 180);   // Map to servo angle
  int brightness = map(potValue, 0, 1023, 0, 255); // Map to LED brightness

  myservo.write(angle);         // Move servo
  analogWrite(ledPin, brightness); // Adjust LED brightness

  Serial.println(potValue);     // Send pot value to Python for plotting
  delay(100);                   // Short delay for smooth movement
}
```

PyCharm code :

```python
import serial
import matplotlib
matplotlib.use('TkAgg')  # Use interactive backend for PyCharm
import matplotlib.pyplot as plt

# --- Serial connection ---
ser = serial.Serial( port: 'COM5',  baudrate: 9600)  # Change COM port to match your Arduino

# --- Live plot setup ---
plt.ion()
fig, ax = plt.subplots()
x_vals, y_vals = [], []

try:
    while True:
        line = ser.readline().decode().strip()
        if line.isdigit():  # Make sure we got a number
            pot_value = int(line)
            print("Potentiometer Value:", pot_value)
            x_vals.append(len(x_vals))
            y_vals.append(pot_value)

            ax.clear()
            ax.plot(x_vals, y_vals, color='green', linewidth=2)
            ax.set_xlabel("Sample Number")
            ax.set_ylabel("Potentiometer Value (0-1023)")
            ax.set_title("Real-Time Potentiometer, Servo, and LED Control")
            ax.grid(True)
```

```python
            plt.pause(0.1)
except KeyboardInterrupt:
    print("\nStopped by user.")
finally:
    ser.close()
    plt.ioff()
    plt.show()
    print("Serial connection closed.")
```

Data Collection

1. The servo rotated smoothly between 0°–180° as the potentiometer value increased.
2. The LED began to glow dimly near mid-rotation and fully on beyond 90°, confirming the threshold condition.
3. Data transmission between Arduino and Python was stable and continuous, allowing accurate real-time control and plotting.
4. The experiment successfully demonstrated bidirectional interaction hardware input from the potentiometer influencing actuator movement, and Python software providing visualization and command control.

Potentiometer Rotation Level (approximate) (%)	Servo Angle (°)	LED Status
0	0	OFF
25	45	DIMLY ON
50	90	DIMLY ON
75	135	ON
100	180	ON

Data Analysis

The collected data confirms that:

1. The servo motor successfully responded to angle values transmitted from Python through serial communication.
2. When integrated with the potentiometer, the servo angle adjusted in real time according to the analog reading.
3. The data transmission between Arduino and Python was stable, showing no delay or loss during continuous operation.

Analysis:

1. The servo responded linearly to both manual Python inputs and real-time potentiometer control, confirming accurate signal conversion and transmission.
2. The LED activation correctly reflected the threshold condition, turning dimly on at approximately half rotation and fully ON beyond 90°.

3. The relationship between potentiometer reading and servo position demonstrated a near-linear correlation, validating the mapping function.
4. Serial communication between Arduino and Python operated consistently, with smooth servo motion and no noticeable jitter or lag.
5. The experiment successfully integrated sensor input, actuator control, and visualization fulfilling the objective of real-time servo manipulation through serial communication.

Results

The experiment successfully demonstrated real-time control of a servo motor and LED through serial communication between Arduino and Python, using a potentiometer as the input device.

1. The servo motor rotated smoothly between 0° and 180° as the potentiometer value increased.
2. The LED responded accurately to the potentiometer's rotation, beginning to glow dimly near mid-range and fully lighting beyond 90°, confirming the programmed threshold condition.
3. The real-time graph generated using Python's matplotlib library showed a continuous and stable variation of potentiometer readings without noise or delay.
4. Data transmission between Arduino and Python remained stable and consistent throughout the experiment, enabling accurate real-time control and visualization.
5. The system demonstrated bidirectional communication—hardware input from the potentiometer influenced servo movement, while Python software provided visualization and command control.

The experiment objectives were fully achieved without any errors or malfunctions. The servo motor, potentiometer, and LED operated in synchronization, and the data visualization in Python accurately represented real-time changes. The results confirmed successful integration of sensor input, actuator response, and serial communication between hardware and software

Discussion

1. System Performance and Observations

   The system operated as intended, establishing effective communication between the potentiometer, microcontroller, and output components. Minor discrepancies were observed, such as fluctuating potentiometer readings due to electrical noise and slight servo deviations from power fluctuations and mechanical backlash. Despite these issues, the LED and servo responded reliably, meeting the expected functional objectives.

2. Sources of Error and Limitations

   The main errors stemmed from signal noise, shared power lines, and limited hardware precision. The USB serial communication added a delay of about 25–30 ms, producing minor control lag.

3. Analysis and Improvements

   Improvements such as signal averaging, dedicated servo power, and binary data transmission could enhance accuracy and responsiveness. Upgrading to a faster microcontroller would further reduce noise, lag, and instability.

Conclusion

The experiment demonstrated that a closed-loop control system can be effectively established using a potentiometer, microcontroller, and servo motor integrated through Python communication. The main findings showed that the system accurately translated analog input into corresponding actuation, with minor noise and delay that did not significantly affect performance. These results support the hypothesis that real-time interaction between hardware and software can be achieved through serial communication. The experiment highlights the practical application of feedback control in mechatronic systems and reinforces its relevance to automated systems, robotics, and sensor-based control processes.

Recommendations

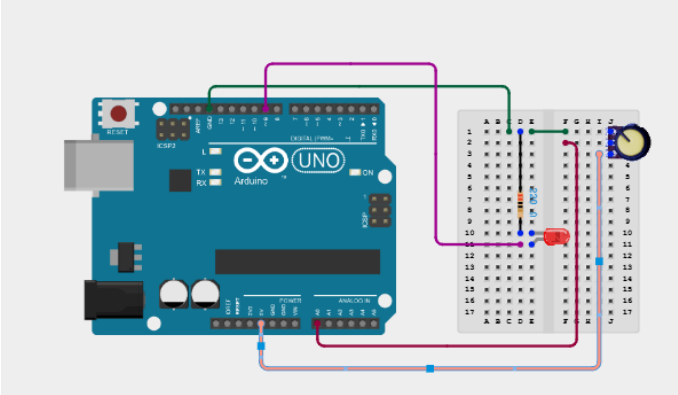
1. Provide a dedicated power supply for the servo motor to prevent voltage drops and reduce movement instability.
2. Ensure proper grounding of the serial connection at startup to allow reliable program execution.
3. Refine circuit layout and wiring to minimize electrical interference and signal loss.
4. Focus on systematic testing and code optimization to enhance accuracy, responsiveness, and understanding of control behavior in future experiments.

References

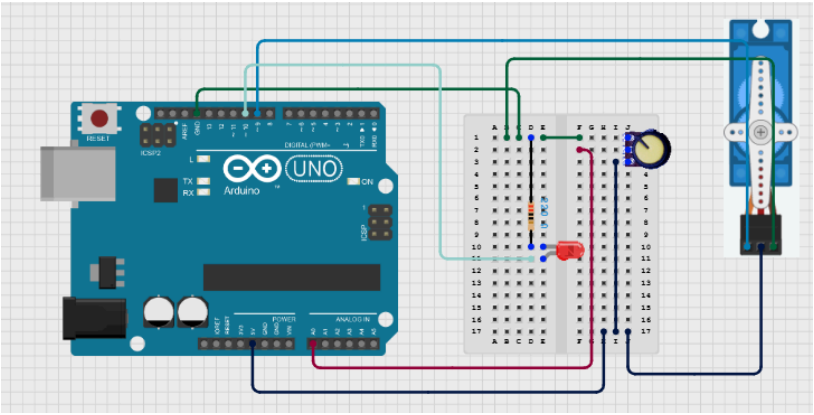
1. Arduino Reference. (n.d.). *digitalWrite() – Arduino Reference*. Retrieved from https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/
2. PySerial contributors. (2023). pySerial API overview. https://pyserial.readthedocs.io/en/latest/
3. Arduino. (2023). Arduino Uno Rev3 documentation. https://docs.arduino.cc/hardware/uno-rev3

Appendices

CirKit Design for Task 1 :



CirKit Design for Task 2 :



Acknowledgments

Alhamdulillah, all praise and gratitude be to Allah SWT for His blessings, guidance, and mercy that enabled us to successfully complete this project. Throughout the development process, we faced several challenges, but with His will and guidance, we managed to overcome them and complete the project successfully.

We would also like to express our heartfelt appreciation to our lecturer for their continuous guidance, support, and valuable feedback throughout this project. Their advice has greatly helped us in understanding both the theoretical and practical aspects of microcontroller systems and digital display design.

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.**

Signature: *Amir* Read [/]
Name: Mohamad Amir Bin Mohamad Yusoff Understand [/]
Matric Number: 2314687 Agree [/]
Contribution:

Signature: *Adam* Read [/]
Name: Adam Hakimi Bin Shah Nur Haizam Understand [/]
Matric Number: 2314195 Agree [/]
Contribution:

Signature: *Ain* Read [/]
Name: Tengku Nurul Ain Binti Tengku Azeezee Understand [/]
Matric Number: 2313682 Agree [/]
Contribution: