



MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SECTION 2, SEMESTER 1, 2024/2025

ACTIVITY REPORT

Week :

3b-Serial Communication

Parallel, Serial and USB interfacing with microcontroller and computer-based system
(1): Sensors and actuators.

No	Name (Group 6)	Matric No.
1	MUHAMMAD BASIL BIN ABDUL HAKIM	2215315
2	MUHAMMAD SYAFIQ BIN NOR AZMAN	2213187
3	MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
4	MUHAMMAD AMMAR ZUHAIR BIN NOR AZMAN SHAH	2110259
5	MUHAMMAD RAZIQ BIN KAHARUDDIN	2120225

Table of content

Table of content.....	2
Abstract.....	3
Material and equipment.....	4
Experimental Setup and Methodology.....	6
Results.....	7
Discussions.....	8
Question.....	15
Recommendation.....	17
Conclusion.....	17
Student Declaration.....	18

Abstract

This experiment mainly focuses on serial communication between Python and Arduino. Serial communication allows data exchange between computers and microcontrollers. The microcontroller can be controlled and monitored based on the code written on both Python and Arduino IDE. It involves the use of a laptop, Arduino board, and also coding. The other part involves controlling a servo motor through serial input from the laptop using Python.

Introduction

1. Purpose and Objectives

The main purpose of this experiment is to get a hands-on understanding of serial communication and how the theory applies to real-life situations. By setting up the physical components required and coding on both Arduino and Python, we can:

- Gain a deeper understanding of how serial communication works practically
- Learn how to control different electrical components using Python
- Develop coding skills in Arduino and gain exposure to Python language

2. Background and Theory

Serial communication is used for communication between a microcontroller and a PC or other devices. Communication here means exchanging data from one device to another. All Arduino boards have at least one serial port, known as USART or UART, to carry out serial communication. In this experiment, Arduino Uno is used and it consists of 2 serial ports. USB connects the Arduino board to the PC, allowing data transfer. In order to use Python in serial communication, PySerial needs to be installed.

3. Hypothesis/ Expected Outcomes

We as a group will carry out the experiment successfully. The lab activities should enhance our practical skills and coding skills when dealing with serial communication involving Python and Arduino. This should result in a better understanding of transferring data from Python to Arduino. We are expected to design more complex digital systems in the future.

Material and Equipment

1. Arduino (Uno)



2. Servo motor



3. Wires/jumper (male-to-male)



Experimental Setup and Methodology

1. Servo's signal, power and ground wires are made sure to be connected properly to an Arduino.
2. The Arduino is connected to a computer via a USB cable.
3. Codes that have been written in Arduino IDE are uploaded into the Arduino
4. Arduino IDE is closed to make sure its serial monitor does not disturb Python or any other programme.
5. Python is booted up and codes to read data from the Arduino are written.
6. After making sure the serial port and baud rate are the same as in Arduino IDE, the programme is run.
7. When Python's terminal prompts an angle, an angle between 0° and 180° is entered to make the servo motor rotate to the desired angle.

Results

We were successfully able to rotate the servo with the potentiometer. On the Python terminal, we could also observe the current position of the servo and the current potentiometer value. However, the program to control the servo using the Potentiometer was coded on the Arduino directly and thus the program doesn't stop even if we press 'q' on the keyboard and stop the Python program.

Discussions

HARDWARE

The hardware setup for this experiment is quite simple. Here is what we can say about, how the instruments (hardware) used in the experiment are connected to each other:

❖ Connecting the Servo's Signal Wire.

Usually, you connect the servo's signal wire to a PWM-capable pin on the Arduino. In this experiment, we used pin D9.

❖ Power up the servo using the Arduino's 5V and GND pin.

Servos typically require a supply voltage of more or less 5V so, we connect the servo's power wire (ours is red) to the 5V output on the Arduino board.

❖ Connecting the Servo's Ground Wire.

We connect the servo's ground wire (ours is brown) to one of the ground (GND) pins on the Arduino.

The hardware connection showed no issues as it functioned properly and was able to serve us with the desired outputs.

SOFTWARE

Arduino code

```
#include <Servo.h>
```

```
Servo myServo;
```

```
void setup() {
```

```
  Serial.begin(9600); // Start serial communication at 9600 baud
```

```
  myServo.attach(9); // Attach the servo to pin 9
```

```
}
```



```

void loop() {

  if (Serial.available() > 0) {

    int angle = Serial.parseInt(); // Read the incoming angle

    if (angle >= 0 && angle <= 180) {

      myServo.write(angle); // Set the servo to the specified angle

    }

  }

}

```

Python code

```
import serial
```

```
import time
```

Define the serial port and baud rate (adjust the port as per your Arduino)

```
ser = serial.Serial('COM4', 9600, timeout=1)
```

```
try:
```

```
    time.sleep(2) # Wait for the serial connection to initialize
```

```
    while True:
```

```
        angle = input("Enter servo angle (0-180 degrees) or 'q' to quit: ")
```

```
        if angle.lower() == 'q':
```

```
            break
```

```
    try:
```

```
        angle = int(angle)
```

```

if 0 <= angle <= 180:

    # Send the servo's angle to the Arduino

    ser.write(str(angle).encode())

    time.sleep(0.1) # Optional delay to allow processing

else:

    print("Angle must be between 0 and 180 degrees.")

except ValueError:

    print("Invalid input. Please enter a number between 0 and 180.")

except KeyboardInterrupt:

    print("Program interrupted. Exiting...")

finally:

    ser.close() # Close the serial connection

    print("Serial connection closed.")

```

The execution of both Arduino code in *Arduino IDE* and Python code in *Visual Studio Code* run successfully. The process involves uploading code to the Arduino using the Arduino IDE, closing it to avoid conflicts with Python code, and then running a Python program that reads data from the Arduino in *Visual Studio Code*. By ensuring the correct serial port and baud rate settings, we, as a user, can enter an angle between 0° and 180° to control the servo motor's rotation accurately.

Question

We are required to enhance our Arduino and Python code to incorporate a potentiometer for real-time adjustments of the servo motor's angle. In the updated Arduino code, we ensured the ability to halt execution by pressing a designated key on the computer's keyboard. After completing the modification, we restarted the Python script to receive and display servo position data from the Arduino over the serial connection. While experimenting with the potentiometer, we observed the corresponding changes in the servo motor's position.

Here is the code that we managed to write for the enhancement:

Arduino code (enhanced)

```
#include <Servo.h>
Servo servo;
int angle = 90;

void setup() {
  servo.attach(9); // Attach the servo to digital pin 9
  Serial.begin(9600); // Initialize serial communication
}

void loop() {
  if (Serial.available() > 0) {
    char key = Serial.read(); // Read the incoming key

    if (key == 'q') { // Halt if 'q' is pressed
      while (true) {} // Halt the program
    }
  }
}
```

```

int potVal = analogRead(A0); // Read potentiometer value
int mappedVal = map(potVal, 0, 1023, 0, 180); // Map potentiometer value to servo angle range
servo.write(mappedVal); // Set servo angle
delay(50); // Delay for stability
}

```

Python code (enhanced)

```

import matplotlib.pyplot as plt
import serial
import time
import keyboard
# Open serial port

ser = serial.Serial('COM3', 9600) # Update 'COM3' with your Arduino's port

time.sleep(2) # Wait for serial connection to establish

# Initialize empty lists to store data
x_data = []
y_data_pot = []
y_data_servo = []

# Set up the plot

plt.ion() # Turn on interactive mode
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
line_pot, = ax1.plot(x_data, y_data_pot, 'b-', label='Potentiometer Reading')
line_servo, = ax2.plot(x_data, y_data_servo, 'r-', label='Servo Position')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Potentiometer Reading', colour='b')
ax2.set_ylabel('Servo Position', colour='r')

# Main loop
try:
    while True:

# Read potentiometer value and servo position from Arduino
data = ser.readline().decode().strip()
if data:
    values = data.split(',')

```

```

    pot_val = int(values[0])
    servo_pos = int(values[1])
    print("Potentiometer Reading:", pot_val, "Servo Position:", servo_pos)

# Append data to lists
x_data.append(time.time())

# Use time as x-axis
y_data_pot.append(pot_val)
y_data_servo.append(servo_pos)

# Update plot
line_pot.set_xdata(x_data)
line_pot.set_ydata(y_data_pot)
line_servo.set_xdata(x_data)
line_servo.set_ydata(y_data_servo)
ax1.relim()
ax1.autoscale_view()
ax2.relim()
ax2.autoscale_view()
fig.canvas.draw()
fig.canvas.flush_events()

# Check for keyboard input to halt
if keyboard.is_pressed('q'):
    break

except KeyboardInterrupt:
    ser.close()
    print("Serial connection closed")

```

The Arduino and Python code work together to show real-time control and display with a servo motor and potentiometer. The Arduino code changes the servo motor's angle based on the potentiometer's position, allowing for easy adjustments. It also has a feature to stop running when a certain key is pressed, making it more flexible and user-friendly.

On the other hand, the Python code connects to the Arduino to get and display real-time data from the potentiometer and servo motor. It uses 'matplotlib' to create an interactive plot that makes it easy for users to see how the system is working.

From the instruction of the enhancement (referring to the lab manual), if it is to be understood in a simpler we find that the objective was to update the Arduino and Python code to incorporate a potentiometer for controlling the servo motor's angle in real-time, add a key press feature to stop the Arduino code and restart the Python script to display the servo's position data as we adjusted the potentiometer.

We successfully completed these updates, allowing the potentiometer to control the servo angle in real-time. We added a function to stop the Arduino with a key press, then restarted the Python script to view the servo's position data. During testing, we clearly observed how the servo's position changed with the potentiometer adjustments, achieving all intended goals.

Recommendation

Based on the results of this experiment, it is recommended to enhance control and flexibility based on the results, it is recommended to integrate two-way communication between Python and Arduino, allowing Python to send commands that the Arduino can recognize and respond to, such as stopping the servo motor when prompted. This could be achieved by modifying the Arduino code to check for specific commands in the serial data continuously, enabling more flexible control options. Additionally, improving the Python interface with simple controls, like buttons or prompts, would make it easier to start, stop, or reset the servo motor directly, facilitating smoother real-time management without manual code changes.


Conclusion


This experiment successfully showed how Python and Arduino can communicate through serial data, meeting our main goal of learning to control hardware using software. We were able to rotate the servo motor based on potentiometer input and see its position in real-time on the Python terminal, proving that data can be reliably sent from Python to Arduino. Although we couldn't fully stop the servo program from Python, this highlighted an area for improvement in controlling Arduino systems directly through serial commands. Overall, this experiment provided a strong base in serial communication, allowing us to explore more complex projects where multiple devices can work together through similar data exchanges.


Student Declaration


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 taken or done by unspecified sources or person. 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 have been verified by us.

NAME: Muhammad Basil bin Abdul Hakim	READ	✓
MATRIC NO: 2215315	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: Muhammad Syafiq Bin Nor Azman	READ	✓
MATRIC NO: 2213187	UNDERSTAND	✓
SIGNATURE 	AGREE	✓

NAME:Muhammad Hafiz Hamzah Bin Fansuri	READ	✓
MATRIC NO:2212803	UNDERSTAND	✓
SIGNATURE : 	AGREE	✓

NAME: Muhammad Ammar Zuhair Bon Nor Azman Shah	READ	✓
MATRIC NO: 2110259	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: MUHAMMAD RAZIQ BIN KAHARUDDIN	READ	✓
MATRIC NO: 2120225	UNDERSTAND	✓
SIGNATURE	AGREE	✓