KULLIYYAH OF ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

Laboratory Report

Mechatronics Interfacing Lab MCTA 3203

Week 3: Software–Hardware UART Communication

GROUP NUMBER : E

Muhamad Haziq Iskandar bin Hassan Nordin	2119327
Muhammad Aliff Iqmal bin Zainun @ Zainuddin	2114805
Muhammad Nazim bin Akhmar	2114551
Muhammad Arfan bin Mohd. Zulkifli	2112945

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 physical components required and coding on both Arduino and Python, we can:
   - Gain 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

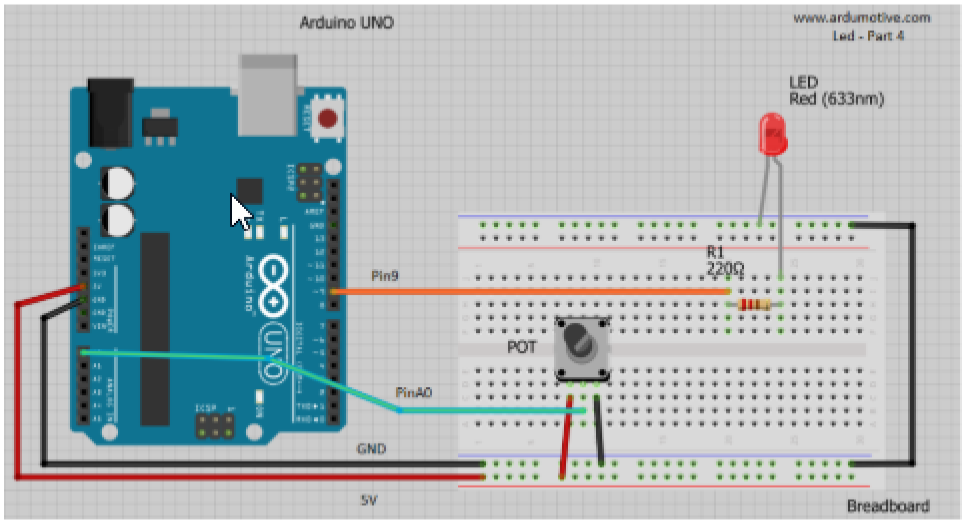
   We as a group will carry out the experiment successfully. The lab activities should enhance our practical skills and coding skills when dealing with serial communication involving Python and Arduino. This should result in better understanding of transferring data from Python to Arduino. We are expected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

<u>Lab Activity 1</u>

Material used

- Potentiometer
- Arduino Uno Board
- Light emitting diode (LED)
- 1k-ohm resistor
- 2 breadboards
- Jump wires

Experimental Setup



Methodology

This lab activity involves setting up physical components and coding on both Arduino and Python. First, all the components were set up according to the image above. Serial communication was then set up between Arduino and Python. Arduino code was written first. The code was then uploaded to the Arduino Uno board by connecting the board to a laptop using a USB. Next, pySerial was installed to carry out serial communication. The coding was written in Python then debugged. The port and board rate of Arduino were checked to make sure there is no issue during serial communication. The serial port is COM 5 with a baud rate of 9600.

<u>Arduino Code:</u>

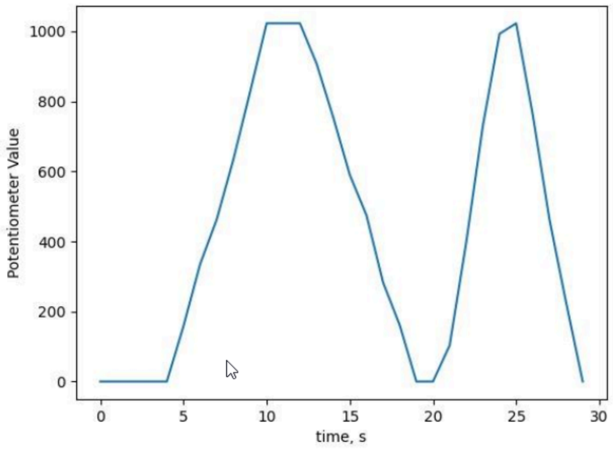
```
const int ledPin = 8; int incomingByte;
void setup() {
```

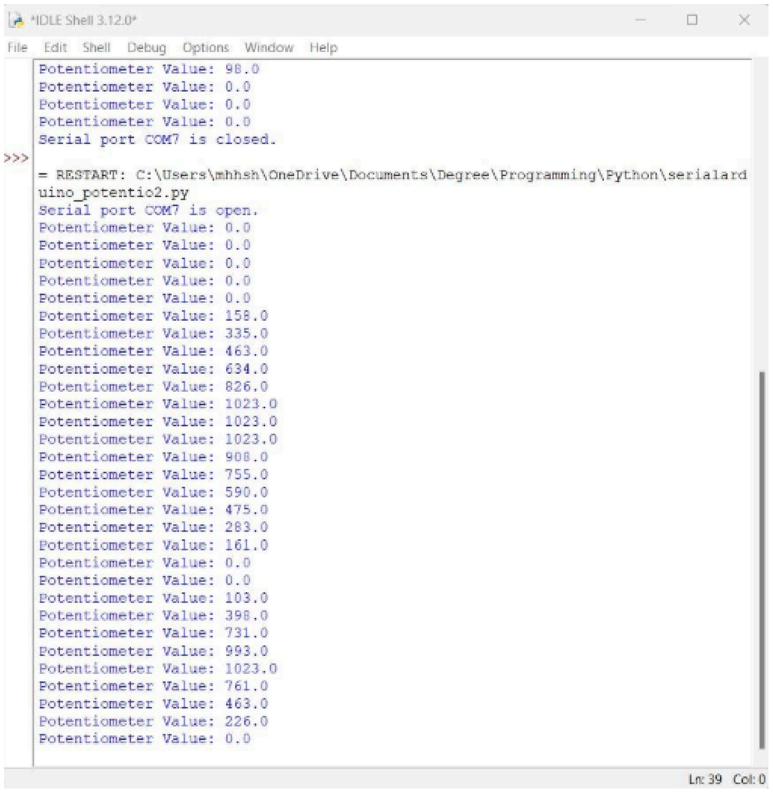
```
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
}
void loop() {
if (Serial.available() > 0) {
incomingByte = Serial.read(); }
if (incomingByte == 'H'){
digitalWrite(ledPin, HIGH);
}
if (incomingByte == 'L') {
digitalWrite(ledPin, LOW);
}
}
```

Python Code:

```
import serial
import time

ser = serial.Serial('COM5', 9600, timeout=1)

user_input = 'L'
while user_input != 'q':
        user_input = input('H = on, L = off, q = quit:')
        byte_command = user_input.encode()
        ser.write(byte_command)
        Time.sleep(0.5)

print('q entered. Exiting the program')
ser.close()
```

Result





```
    Potentiometer Value: 98.0
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Serial port COM7 is closed.
>>>
    = RESTART: C:\Users\mhhsh\OneDrive\Documents\Degree\Programming\Python\serialard
    uino_potentio2.py
    Serial port COM7 is open.
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Potentiometer Value: 158.0
    Potentiometer Value: 335.0
    Potentiometer Value: 463.0
    Potentiometer Value: 634.0
    Potentiometer Value: 826.0
    Potentiometer Value: 1023.0
    Potentiometer Value: 1023.0
    Potentiometer Value: 1023.0
    Potentiometer Value: 908.0
    Potentiometer Value: 755.0
    Potentiometer Value: 590.0
    Potentiometer Value: 475.0
    Potentiometer Value: 283.0
    Potentiometer Value: 161.0
    Potentiometer Value: 0.0
    Potentiometer Value: 0.0
    Potentiometer Value: 103.0
    Potentiometer Value: 398.0
    Potentiometer Value: 731.0
    Potentiometer Value: 993.0
    Potentiometer Value: 1023.0
    Potentiometer Value: 761.0
    Potentiometer Value: 463.0
    Potentiometer Value: 226.0
    Potentiometer Value: 0.0
```

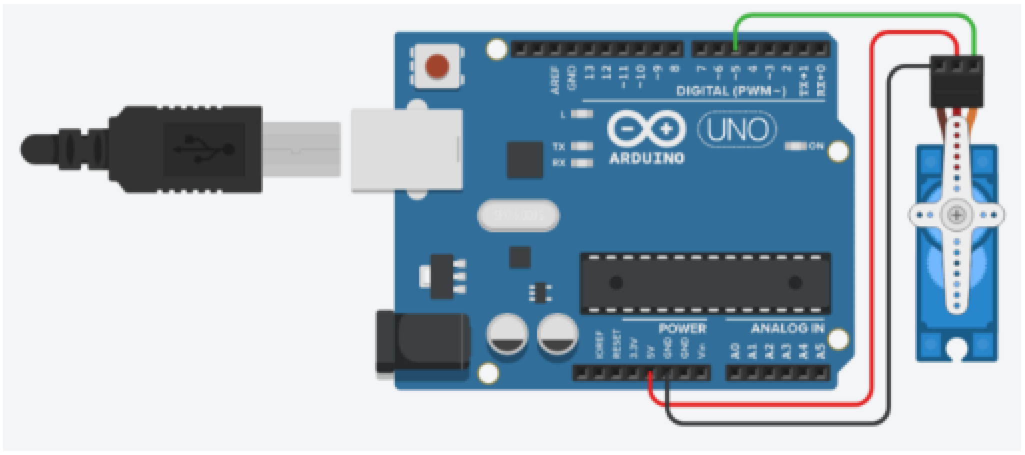
The experiment went smoothly. We managed to control the Led by running the code in Python. The Led lit up whenever ‘H’ was entered, it turned off whenever ‘L’ was entered. We also managed to exit the program by entering ‘q’. We can conclude from the result that the data exchange between the Arduino board and laptop worked successfully.

Lab Activity 2

Material used

- Arduino Uno
- Jump wires
- Servo module

Experimental Setup



Methodology

This activity involves connecting a servo motor to an Arduino Uno and controlling it using the laptop the arduino is connected to. This is done through serial communication. A servo motor is connected to an Arduino Uno in the standard configuration as in the image above. At first an Arduino code is loaded onto the Uno. Next, a script is written in python that connects to serial port COM6 with a baud rate of 9600. The codes are given below:

Arduino Code:

```
#include <Servo.h>

Servo servo;
int angle = 0;
```

```
void setup() {
  servo.attach(9); // Attach the servo to dig. pin 9
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    int angle = Serial.read();
    servo.write(angle); // Set to the desired angle
    delay(1000); // Wait for 1 second
    angle = 180 - angle; // Reverse the angle (e.g., 90 to 90 degrees)
  }
}
```

Python Code:

```
import serial
import time

 ser = serial.Serial('COM6', 9600)

 try:
      while True:
      angle = input("Enter servo angle (0-180 degrees): ")
      if angle.lower() == 'q':
            break
      angle = int(angle)
      if 0 <= angle <= 180:
            # Send the servo’s angle to the Arduino
            ser.write(bytes([angle]))
            time.sleep(0.1)  # Add a small delay for the servo to move
      else:
            print("Angle must be between 0 and 180 degrees.")


 except KeyboardInterrupt:
      pass  # Handle keyboard interrupt

 finally:
      ser.close() # Close the serial connection
      print("Serial connection closed.")
```

Result

Once the python code was run, it prompted the user to input an angle between 0 to 180 degrees. After the user input, the python script was able to use serial communication and rotate the servo as defined by the user input.

Lab Activity 3 (3b Question)

Material used

- Arduino Uno
- Jump wires
- Servo module
- Potentiometer

Experimental Setup

Methodology

This activity involves rotating the servo with a potentiometer. The servo is connected in the standard configuration and a potentiometer is connected with its middle pin connected to pin A0 on the Arduino. The value of the potentiometer is read and then mapped from 0 to 180 degrees so that it can be used to rotate the servo. The Arduino code is loaded and then Python is used to facilitate the task.

Arduino Code:

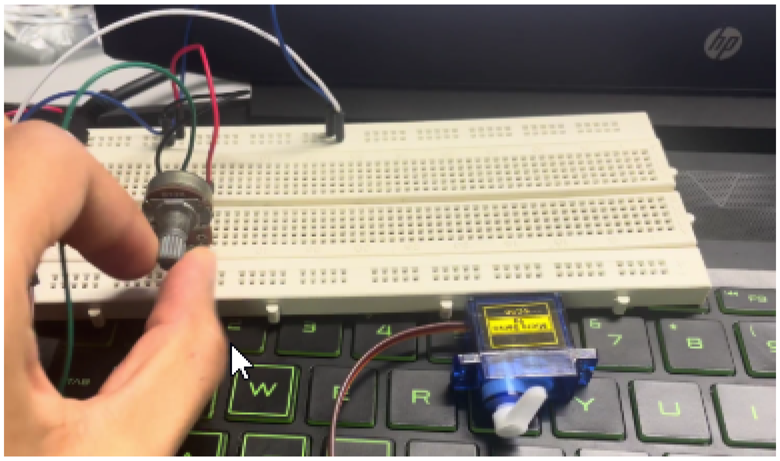
```
#include <Servo.h>
```

```
Servo myservo;

int potPin = A0;

int servoPin = 9;

int potValue;

int servoPos = 90; // Initial servo position


void setup() {

  myservo.attach(servoPin);

  Serial.begin(9600);

}


void loop() {

  // Read potentiometer value

  potValue = analogRead(potPin);

  // Map the potentiometer value to the servo position (0-180)

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


  // Send both values to the computer

  Serial.print(potValue);

  Serial.print(',');

  Serial.println(servoPos);
  // Move the servo to the new position

  myservo.write(servoPos);



  // Adjust the delay as needed

}
```

Python Code:

```
import serial
 import keyboard  # Import the keyboard library

 # Define the serial port and baud rate
 ser = serial.Serial('COM6', 9600)  # Use the correct serial
port for your Arduino

 while True:
     try:
         # Read data from the Arduino
         data = ser.readline().decode().strip()
         pot_value, servo_pos = map(int, data.split(','))

         # Print or use the values as needed
         print(f"Potentiometer Value: {pot_value}")
         print(f"Servo Position: {servo_pos}")

         # Check if 'q' is pressed to exit the program
         if keyboard.is_pressed('q'):
         print("Exiting the program.")
         break

     except KeyboardInterrupt:
         break

 ser.close()
```

Result:



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 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.

Discussion

Upon initiation, the Python script establishes a bidirectional communication interface with the Arduino through the PySerial library, ensuring seamless data exchange between the two systems. This script not only initiates the connection but is also intricately designed to meticulously govern the servo motor's angle upon activation. Within the Python script's logic, there is a structured mechanism for accessing the serial port, patiently awaiting user-defined inputs to regulate the servo motor's angle. This interactive feature empowers users to command specific angles for the servo motor by inputting numerical values. Upon receiving these angle instructions, the Arduino interprets and translates them into tangible movements, effectively adjusting the servo's position as directed. The servo motor, upon receiving angle commands, promptly executes the prescribed movements and maintains its position until new directives are relayed by the user. This approach ensures a step-by-step control, allowing for the precise positioning and regulated movements of the servo motor. The Python program communicates with the Arduino through a USB serial connection, transmitting instructions in a defined format:

"str(angle) + 'A'", with 'A' acting as a delimiter, denoting the end of one angle command and the commencement of the next angle for controlling the servo's position.

Conclusion

The culmination of a serial data experiment from Python to Arduino hinges on the specific objectives and results sought in the experiment. Yet, in a broad context, one might deduce the successful establishment of a communication link between Python and Arduino through serial communication. This accomplishment enables the transmission of data from Python to Arduino, serving diverse applications such as sensor data transfer or remote control. The success of the experiment is contingent upon factors like the reliability of data transfer, data processing on the Arduino side, and the realization of intended goals. In essence, this experiment not only proved successful but also paved the way for further exploration, leveraging the foundational understanding of serial communication. This knowledge can be applied in intricate systems that extensively utilize multiple microcontrollers and computers. It signifies the commencement of a broader field of communication possibilities.

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 contributor 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

Signature: haziqiskandar	Read ✓
Name: Muhamad Haziq Iskandar bin Hassan Nordin	Understand ✓
Matric Number:2119327	Agree ✓

Signature: aliffiqmal	Read ✓
Name: Muhammad Aliff Iqmal bin Zainun @ Zainuddin	Understand ✓
Matric Number: 2114805	Agree ✓

Signature: nazim	Read ✓
Name: Muhammad Nazim Bin Akhmar	Understand ✓
Matric Number: 2114551	Agree ✓

Signature: arfan	Read ✓
Name: Muhammad Arfan bin Mohd. Zulkifli	Understand ✓
Matric Number: 2112945	Agree ✓