CSE461

Introduction to Robotics Lab

Lab No. : 03
Group : 03
Section : 08
Semester : Summer_2025

Group members :

MD.Sohanur Rahman Shimul	22299079
Fayez Ahmad Protik	23101474
Mahibi Islam	22201828
Tithi Halder	22101406
Sharmin Jahan Ananna	22101850

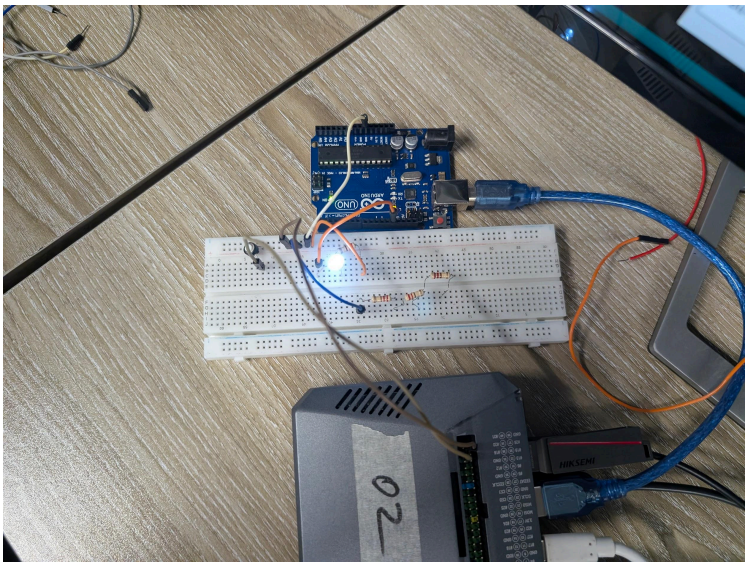
Submitted Date: 31-07-2025
Submitted to -
Utsha Kumar Roy & Sadman Sharif

1. **Objectives:** Interfacing Arduino and Raspberry PI using UART protocol to control an LED with a switch.

2. **Equipments:**

    - Arduino Uno R3
    - Raspberry PI 4
    - LED (Light Emitting Diode)
    - Push-button switch
    - Resistor (220Ω for LED)
    - Breadboard
    - Jumper wires

3. **Experimental Setup:**
   (Picture + Explanation)



Explanation:

In this experiment, first we upload Arduino code before connecting UART pins to avoid upload conflict. Then Raspberry Pi and an Arduino Uno are interfaced using the UART protocol to control an LED. A push-button is connected to the Raspberry Pi (GPIO pin 20), and an LED is connected to pin 13 of our Arduino Uno. When the button is pressed, the Raspberry Pi sends a signal through UART to the Arduino. After receiving this signal, Arduino blinks the LED

To enable UART communication, the TX pin of the Raspberry Pi (GPIO14) is connected to the RX pin of the Arduino (pin 0), and the RX pin of the Raspberry Pi (GPIO15) is connected to the TX pin of the Arduino (pin 1) through a voltage divider to step down the 5V signal from Arduino to 3.3V for the Raspberry Pi. A common GND connection is shared between both devices.

The Raspberry Pi code constantly monitors the button state and sends a UART signal when pressed. The Arduino listens for incoming serial data and activates the LED when it receives the specified signal.

4. **Code:** **(If Applicable)**

#Raspberry PI Code:

```
import RPi.GPIO as GPIO
import serial
import time

# Set up GPIO for button
button_pin = 20  # GPIO pin 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(button_pin, GPIO.IN)

# Set up UART (Serial) communication with Arduino
ser = serial.Serial('/dev/serial0', 9600)

print("Raspberry Pi is ready. Press the button to blink the LED...")

try:
    while True:
        if GPIO.input(button_pin) == GPIO.LOW:  # If button is pressed
            print("Button pressed! Sending signal to Arduino...")
            ser.write(b'1')  # Send signal to Arduino to blink LED
            time.sleep(0.2)  # Debounce delay
        else:
            print("Button not pressed. Waiting for press...")
        time.sleep(0.1)  # Short delay to avoid excessive CPU usage
except KeyboardInterrupt:
    print("Program interrupted.")
finally:
    GPIO.cleanup()
    print("GPIO cleaned up. Exiting program.")
```

#Arduino Code:

```
const int ledPin = 13;  // LED pin

void setup() {
  Serial.begin(9600);  // Initialize UART communication at 9600 baud rate
```

```
  pinMode(ledPin, OUTPUT);  // Set the LED pin as an output
  Serial.println("Arduino is ready. Waiting for command...");
}

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

    if (command == '1') {
      Serial.println("Button pressed: Blinking LED...");
      digitalWrite(ledPin, HIGH);  // Turn the LED on
      delay(1000);  // Wait for 1 second
      digitalWrite(ledPin, LOW);  // Turn the LED off
      Serial.println("LED blinked.");
    }
  }
}
```

5. Results (Output of the experiment):

When the push-button on the Raspberry Pi is pressed:

Raspberry Pi sends 1 over UART. Arduino receives the command and blinks the LED connected to pin 13.

When the button is not pressed:

No signal is sent and the LED stays off.

6. Discussions/Answers:

(May contain conclusions or learnings from the result along with problems faced and solving methodology.)

In this experiment, we learned how to make a Raspberry Pi and an Arduino work together using UART communication. When the button on the Raspberry Pi is pressed, it sends a message to the Arduino, which then turns the LED on and off. We had to be careful with the wiring because the two boards use different voltages that were fixed using a voltage divider. Also, we had to upload the Arduino code before connecting the UART wires, or it wouldn't work properly. Overall, the project helped us understand how to connect two devices and make them respond to each other in real-time.