BRAC UNIVERSITY

Inspiring Excellence

CSE461

Introduction to Robotics Lab

Lab No. : 02
Group : 06
Section : 03
Semester : Fall_2025

Group members :

Muaz Mohammad Zimam	23101529
Abdullah Al Mahmud	22201502
Safin Ahamed Ifty	22299283
Razia Marzan Mou	22299123
Quazi Sumaya Sultana Prionti	22101454

Submitted Date: 02-11-2025
Submitted to - Mahadi Ibne Bakar & Aabrar Islam

1. **Objectives:**
   To become familiar with the Raspberry Pi and its GPIO pin layout. To understand how to interface simple I/O components like LEDs and switches with the Raspberry Pi. To write and execute Python code to control external components. To gain hands-on experience in circuit setup, coding, and debugging.

2. **Equipments:**
   Raspberry Pi 4 Model B

   Breadboard

   Jumper wires (Male-to-Female)

   220Ω resistor (for current limiting)

   LED (Light Emitting Diode)

   Push-button switch

   USB or microSD card with Raspberry Pi OS

   Display monitor, keyboard, and mouse

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

   The LED's anode was connected to GPIO pin 4 through a 220Ω resistor.

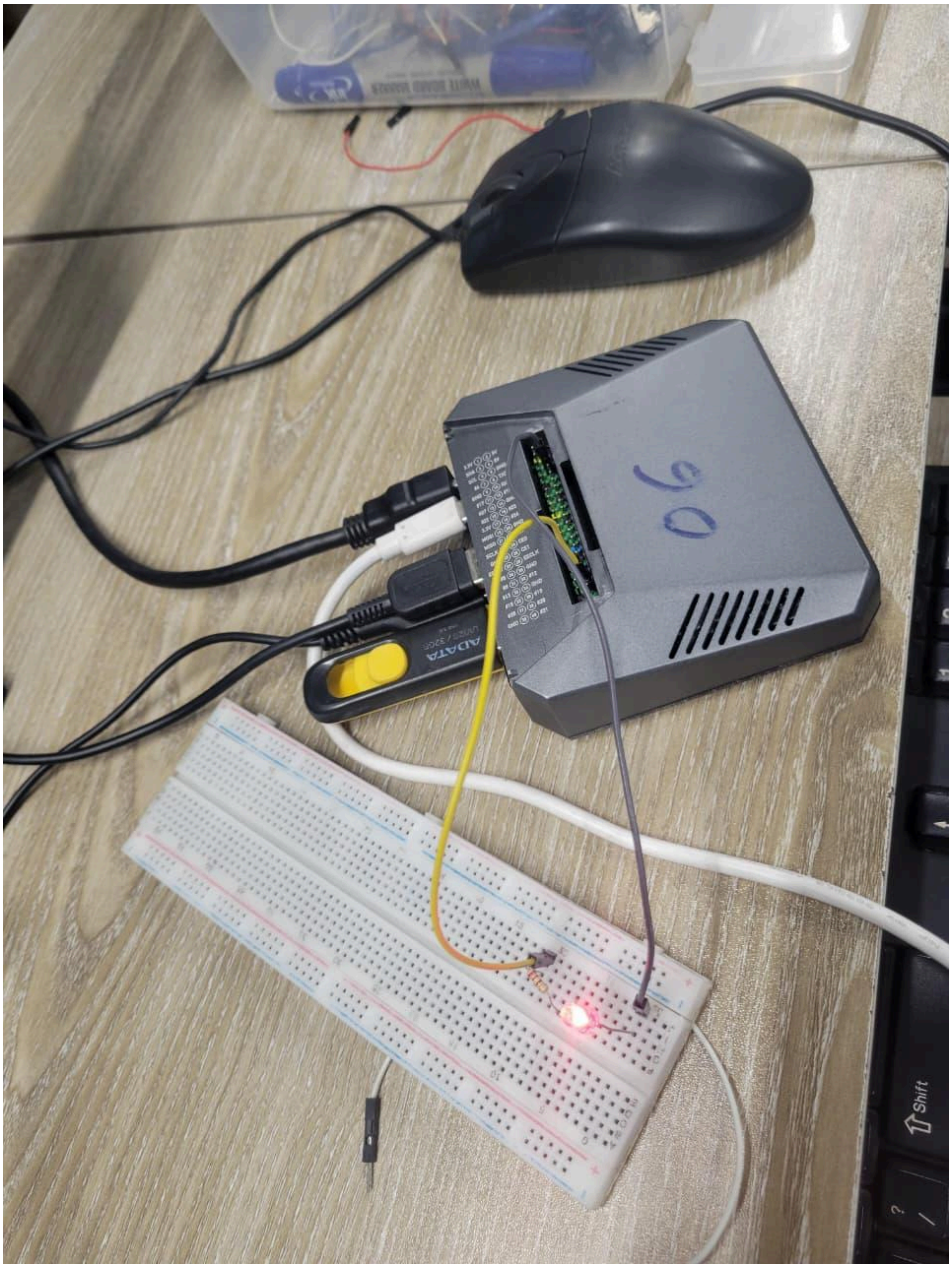
   The LED's cathode was connected to the ground (GND) pin.

   The push-button switch was connected between GPIO pin 14 and GND.

   The Raspberry Pi was powered through its standard power supply and connected to a monitor, keyboard, and mouse for code execution.

   The experiment used the `gpiozero` library in Python for easy GPIO control.

   Experiment 1 (LED blinking):Make an LED blink using Raspberry Pi GPIO output.

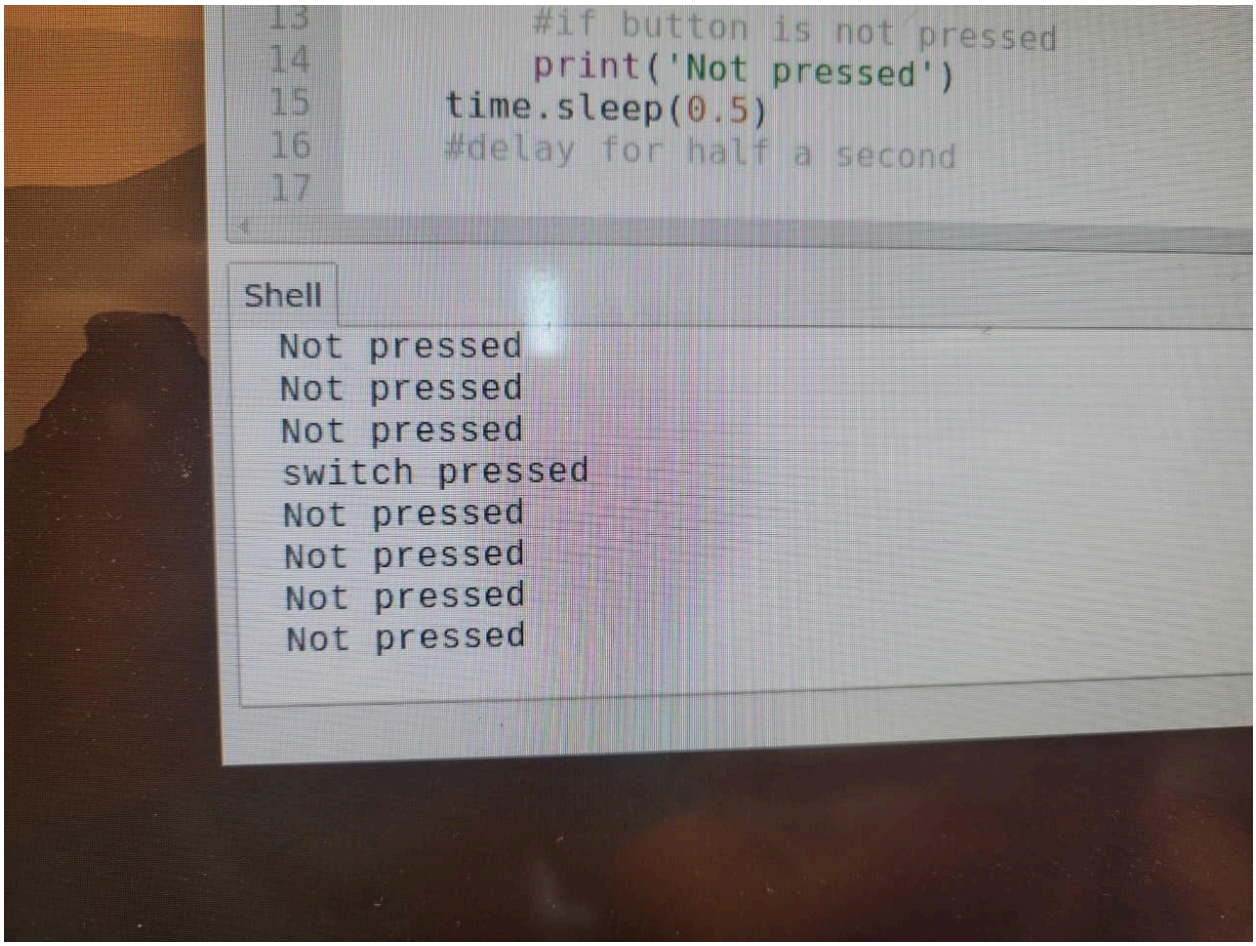


Explanation:

- GPIO pin 4 was declared as an output pin controlling the LED.
- The `led.on()` and `led.off()` commands turned the LED on and off alternately.
- `time.sleep(1)` created a 1-second delay between on/off cycles.

Result: The LED blinked continuously with a one-second interval.

Experiment 2 (Switch-activated serial messaging): Detect button press and print a message in the terminal.

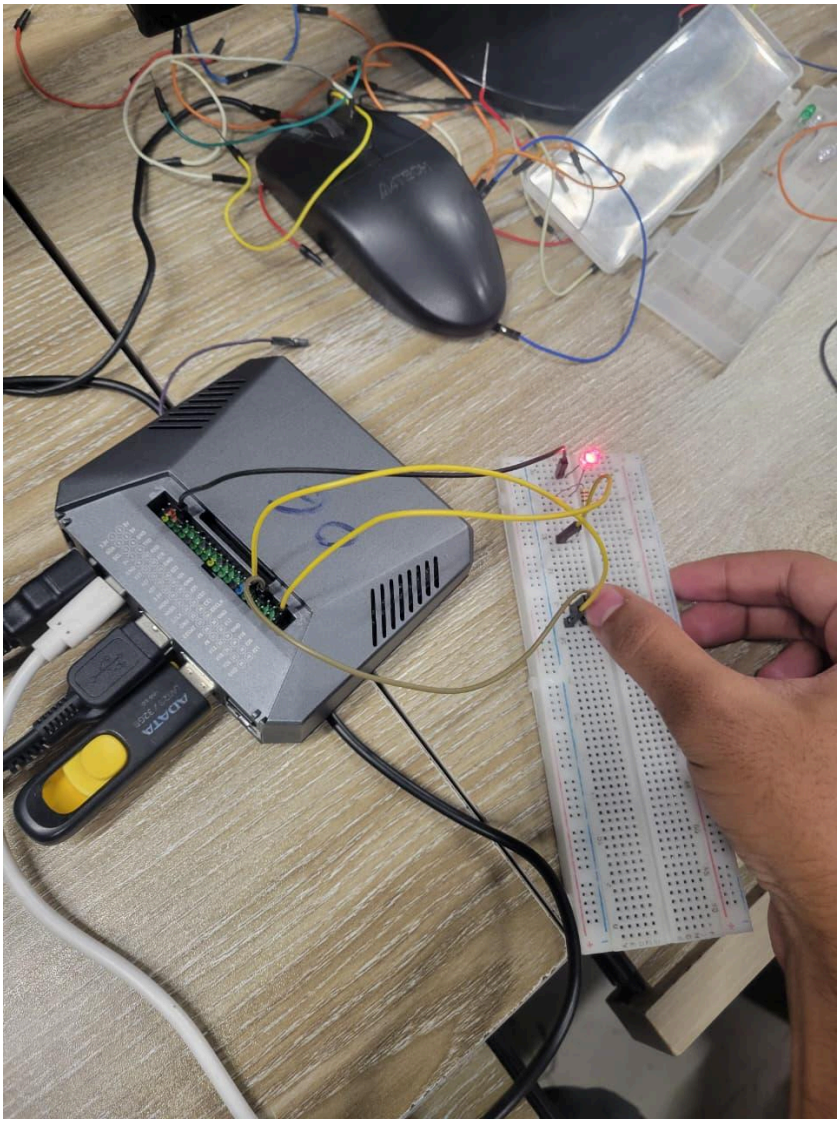


Explanation:

- GPIO pin 14 was set as input for the button.
- When the button was pressed, the terminal displayed “Switch pressed”; otherwise, “Not pressed.”

Result: The terminal correctly showed the switch state when pressed and released.

Lab Task (Switch-controlled LED): Combine Experiments 1 and 2 so the LED turns on when the switch is pressed.



Explanation:

- The LED (output) and button (input) were linked in code.
- When the button was pressed, the LED turned on.

- When the button was released, the LED turned off.

Result: The LED responded accurately to the button press.

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

```
# 1
from gpiozero import LED
import time

led = LED(4)

while True:
    led.on()
    time.sleep(1)
    led.off()
    time.sleep(1)


#2
from gpiozero import Button
import time

button = Button(14)

while True:
    if button.is_pressed:
        print("Switch pressed")
    else:
        print("Not pressed")
    time.sleep(0.5)

#Lab_ Task

from gpiozero import LED, Button
import time

led = LED(4)
button = Button(14)

while True:
    if button.is_pressed:
        led.on()
    else:
        led.off()
    time.sleep(0.1)
```

5. Results (Output of the experiment):

Experiment 1: LED blinked at regular one-second intervals.

Experiment 2: The terminal displayed “Switch pressed” when the button was pressed and “Not pressed” when released.

Lab Task: The LED illuminated only when the button was pressed and turned off immediately upon release.

6. Discussions/Answers:

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

Learned how to use GPIO pins as digital inputs and outputs.

Understood how to interface LEDs and switches with the Raspberry Pi.

Gained practical experience in writing and running Python code on Raspberry Pi OS.

Faced issues such as incorrect GPIO numbering and unstable switch readings; resolved them by verifying BCM pin mode and enabling pull-up resistors.

Recognized the importance of current-limiting resistors to protect the LED from overcurrent.