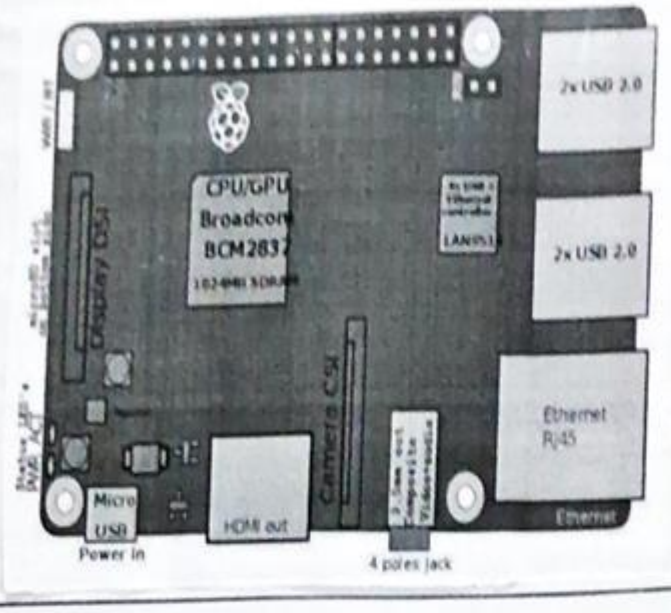
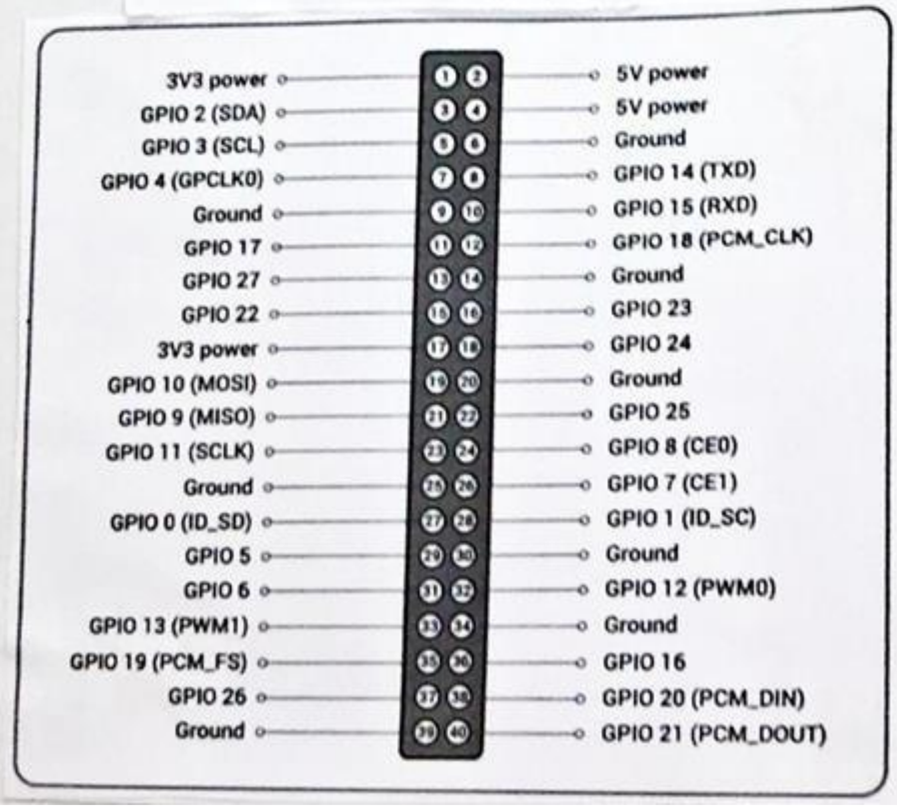
ASSIGNMENT-01

RASPBERRY PI KIT

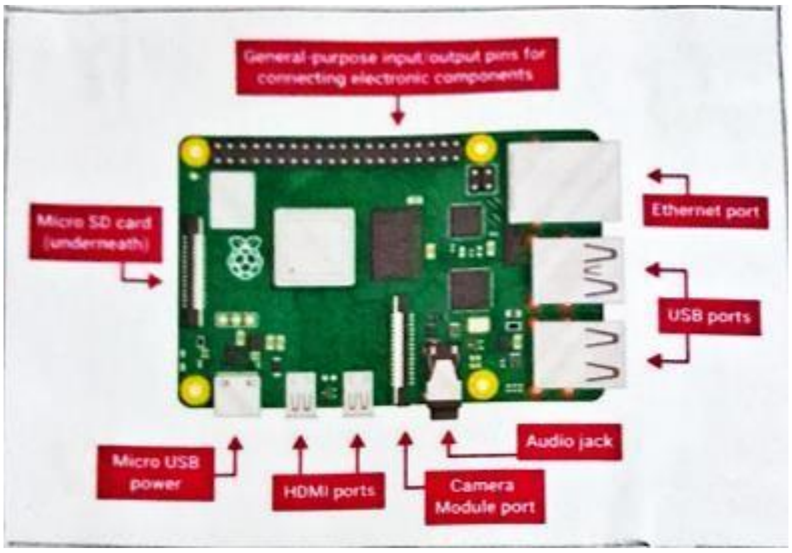




Aim of the Experiment:
To study the Raspberry Pi components and configure Raspberry Pi and its OS.

Components:

1. **System-on-Chip (SoC):**
   The heart of the Raspberry Pi is its Broadcom BCM2711 SoC, which features a quad-core Cortex-A72 (ARM v8) 64-bit processor running at up to 1.5 GHz. This provides significant processing power for a wide range of tasks.
2. **Memory (RAM):**
   The Raspberry Pi is available with different RAM options, including 2GB, 4GB, and 8GB LPDDR4 SDRAM. This memory is shared between the CPU and GPU and plays a crucial role in the overall performance of the system.
3. **Graphics Processing Unit (GPU):**
   The Raspberry Pi utilizes a video-core VI GPU, which supports OpenGL ES 3.x and can drive dual 4K displays at 60Hz. This GPU enables smooth graphics rendering and multimedia processing.
4. **Connectivity:**
   - **Ethernet:** The Pi includes a Gigabit Ethernet port, providing fast wired network connectivity.
   - **Wi-Fi and Bluetooth:** It features built-in dual-band 802.11ac Wi-Fi and Bluetooth 5.0, allowing wireless communication with other devices and networks.


General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Audio jack
Micro USB power
HDMI ports
Camera Module port

PART OF RASHBERRY PI

5. **USB Ports:**
   There are two USB 2.0 ports and two USB 3.0 ports, providing ample connectivity for peripherals such as keyboards, mouse, storage devices, and more.

6. **Storage:**
   The Raspberry Pi does not have built-in storage, but it includes a microSD card slot for storage expansion. The OS and other files are booted and run from the microSD card.

7. **Video Outputs:**
   It supports dual micro HDMI ports, each capable of outputting up to 4K resolution at 60Hz. This allows users to connect multiple displays for enhanced productivity or multimedia applications.

8. **Audio:**
   The Raspberry Pi includes a 3.5mm audio jack for connecting headphones or speakers. It also supports audio output over HDMI.

9. **GPIO (General Purpose Input/Output):**
   The Raspberry Pi has a 40-pin GPIO header, allowing users to interface with various external devices and components such as sensors, LEDs, motors, etc.

10. **Power:**
    Pi is powered via a USB-C port, providing a reliable and standardized power source. The Pi requires more power (especially under load), so it is essential to use an adequate power supply.

11. **Cooling:**
    Due to the increased performance, the Raspberry Pi may generate more heat. To ensure optimal performance and longevity, users may opt to use passive cooling solutions like heat sinks or active cooling methods (e.g., fans).

Raspberry Pi Pins:

1. **Pin Numbering:**
   The GPIO pins are numbered from 1 to 40, starting from one corner of the Raspberry Pi board and progressing in a consistent sequence. Pin numbering can vary slightly between different models of Raspberry Pi.

2. **Voltage Levels:**
   The GPIO pins support both digital inputs and outputs operations. They operate at 3.3V (3V3), which means that any signal or voltage connected to the GPIO pins should not exceed this level to prevent damage to the Raspberry Pi.

3. **Types of Pins:**

   - **Power Pins:** These pins provide power to external components. They include +3.3V, 5V, and ground (GND) pins.
   - **General Purpose Input/Output (GPIO Pins):** These pins are configured as either inputs or outputs, allowing the Raspberry Pi to read data from sensors or control external devices such as LEDs, motors, and relays.
   - **Special Function Pins:** These pins have specific functions, such as providing hardware PWM (pulse-width modulation) signals, SPI (Serial Peripheral Interface) communication, I2C (Inter-Integrated Circuit) communication, UART (Universal Asynchronous Receiver-Transmitter) serial communication, and more.

4. **Functionality:**
   Each GPIO pin can be programmed to perform various functions, including:
   - **Digital Input:** Reading digital signals (high or low) from external sensors or switches.
   - **Digital Output:** Controlling the state (high or low) of external devices such as LEDs, relays, and motors.
   - **Analog Input:** Reading analog signals from sensors that provide continuous voltage levels.
   - **PWM Output:** Generating pulse-width modulation signals for controlling the brightness of LEDs, the speed of motors, or other applications that require variable output levels.
   - **Communication:** Sending and receiving data over serial interfaces such as SPI, I2C, or UART.
5. **Initial Setup:**
   - Follow the on-screen instructions to complete the initial setup of Raspberry Pi OS.
   - You will be prompted to set a new password for the default "pi" user.
   - You can also expand the file system to use the full capacity of the MicroSD card.
6. **Update Software:**
   Once the initial setup is completed, it's a good idea to update the software packages on your Raspberry Pi to ensure you have the latest security patches and features.

7. **Explore and Customise:**
   Now that our Raspberry Pi is set up and running, we can explore the pre-installed applications, install additional software, and customize it according to our needs.

Conclusion:

We learnt about different components of a Raspberry Pi in detail. We also learnt about the pin configurations and its uses in detail. We setup our Raspberry Pi device and loaded its OS in MicroSD card and used that card to boot the Pi.

NAME: MD SHAYAHN SIDDIQUI

ROLL-A2(10)

SIC:22BCSC21

ASSIGNMENT - 02

AIM:
Raspberry Pi: Configuration and control a LED light (ON/OFF) using Raspberry Pi.

Components:

1. Raspberry Pi model 4B
2. Power adapter, keyboard, mouse and monitor
3. Breadboard
4. LED Light
5. Resistor
6. Male to female connector (2x)

Connections s Configurations:

Raspberry Pi Configuration and its OS:

1. **Get the necessary hardware:**
   - Raspberry Pi Model 4B
   - Micro SD card (at least 8GB)
   - Power adapter (USB-C for model 4B)
   - HDMI Cable
   - Keyboard and mouse (USB or Bluetooth)
   - Monitor or TV with HDMI Input
2. **Prepare the MicroSD card:**
   - Insert the MicroSD card into your computer using a card reader.
   - Download the latest version of Raspberry Pi OS from the official Raspberry Pi Website.
3. **Flash the OS image onto the MicroSD card:**
   - Use a tool like Etcher to flash the downloaded Raspberry Pi OS image onto the MicroSD card.
   - Open Etcher and select the downloaded OS image file.
   - Select the MicroSD card as the target drive.

    - Click on 'Flash' and wait for the process to complete.

4. **Insert the MicroSD card and Boot Raspberry Pi:**
    - Safely eject the MicroSD card from your computer and insert it into the Raspberry Pi's MicroSD slot.
    - Connect the Raspberry Pi to a monitor using an HDMI cable.
    - Connect a keyboard and mouse to the USB port.
    - Finally, connect the power adapter to the Raspberry Pi to boot it up.
5. **Initial Setup:**
    - Follow the on-screen instructions to complete the initial setup of Raspberry Pi OS.
    - We'll be prompted to set a new password for the default "pi" user.
    - We can also expand the file system to use the full capacity of the MicroSD card.
6. **Update Software:**
    - Once the initial setup is complete, it's a good idea to update the software packages on our Raspberry Pi to ensure we have the latest security patches C features.
7. **Explore and Customise:**
    - Now that our Raspberry Pi is setup and running, we can explore the pre-installed applications, install additional software and customize it according to our needs.

Connection Details:

- Locate the GPIO pins on our Raspberry Pi. Here, we use **GPIO2 (PIN-3)** and ground (**GND**) pin.
- Place the LED on the breadboard. Put the resistor in the breadboard.
- Connect the longer (+ve) leg of the LED with the resistor and the small (-ve) leg elsewhere.
- Using jumper wires, we will connect the GPIO 2 pin with the resistor's other leg and ground pin with the (-ve) leg of the LED.
- After the setup, we run the Python code we wrote for the LED to turn ON/OFF.

CODE:

1) **For turning ON the LED:**

python

Copy code

```
import RPi.GPIO as rasp
rasp.setwarnings(False)
rasp.setmode(rasp.BOARD)
rasp.setup(3, rasp.OUT)
rasp.output(3, 1)
```

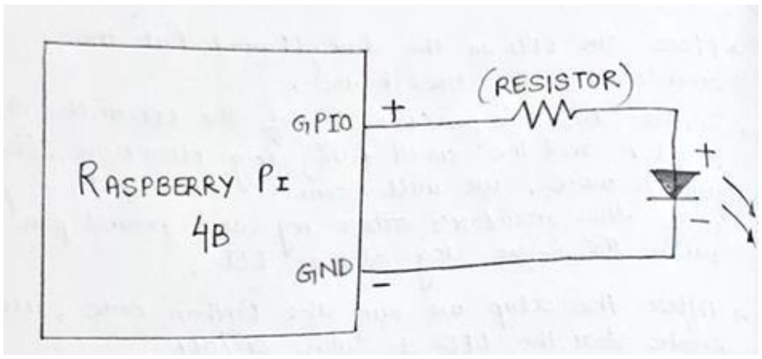
2) **For turning OFF the LED:**

python

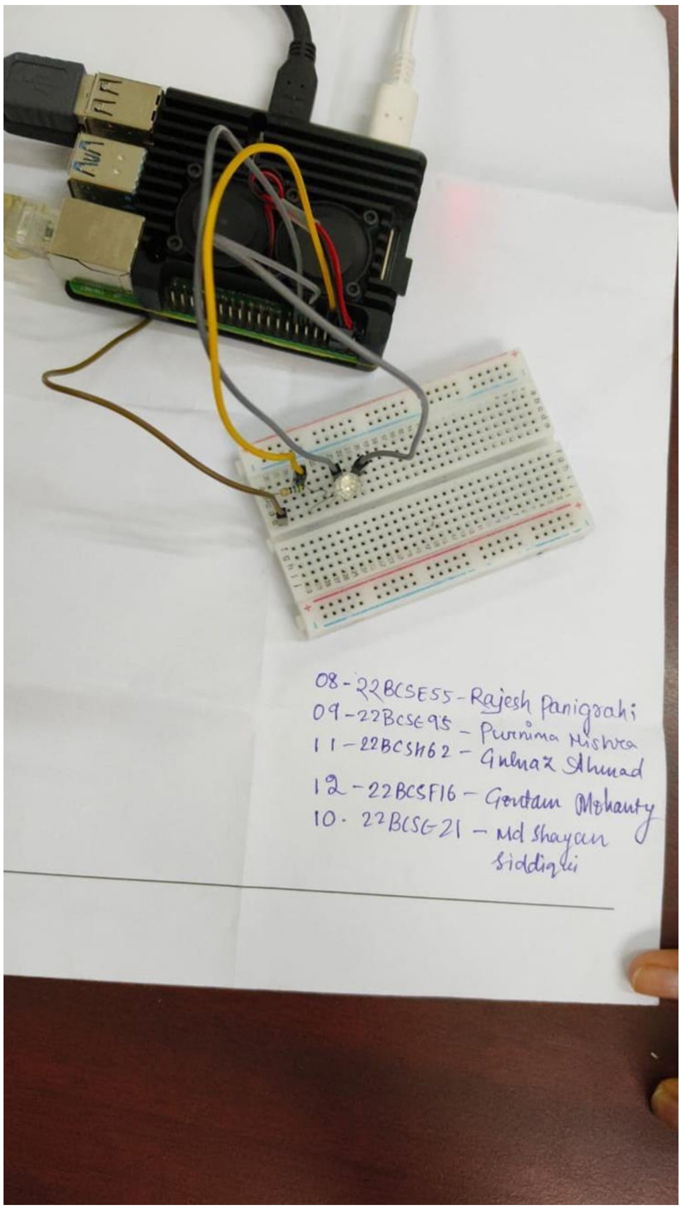
Copy code

```
import RPi.GPIO as rasp
rasp.setwarnings(False)
rasp.setmode(rasp.BOARD)
rasp.setup(3, rasp.OUT)
rasp.output(3, 0)
```

OUTPUT:



CIRCUIT DIAGRAM


08-22BCSE55-Rajesh Panigrahi
09-22BCSE95 - Purnima Mishra
11-22BCSH62 - Gulnaz Ahmad
12-22BCSF16 - Goutam Mohanty
10. 22BCSG21 - Md Shayan Siddiqui

- When we set rasp.OUT as 1, then the LED will turn **ON**.
- When we set rasp.OUT as 0, then the LED will turn **OFF**.

CONCLUSION:

We learnt about the Raspberry Pi configurations and its use in detail. We have also setup our Raspberry Pi device and loaded its OS in MicroSD card. Finally, we also performed how to turn ON/OFF a LED signal using Raspberry Pi.

NAME: MD SHAYAHN SIDDIQUI

ROLL; A2(10)

SIC:22BCSC21

ASSIGNMENT – 03

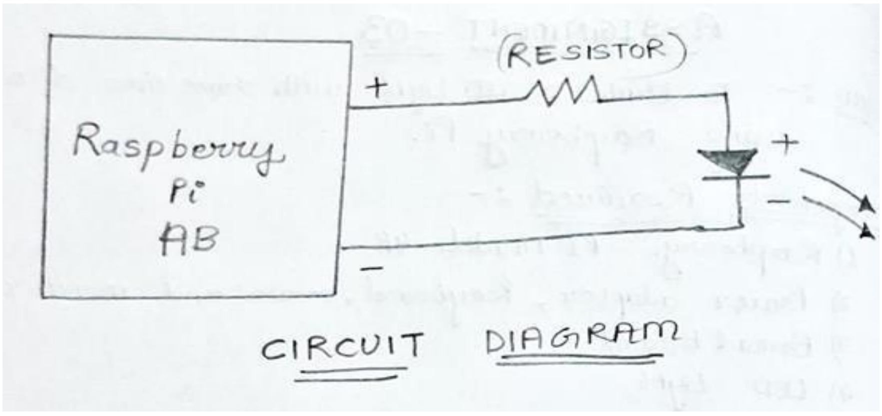
AIM:
To blink a LED light with some time slice using Raspberry Pi.

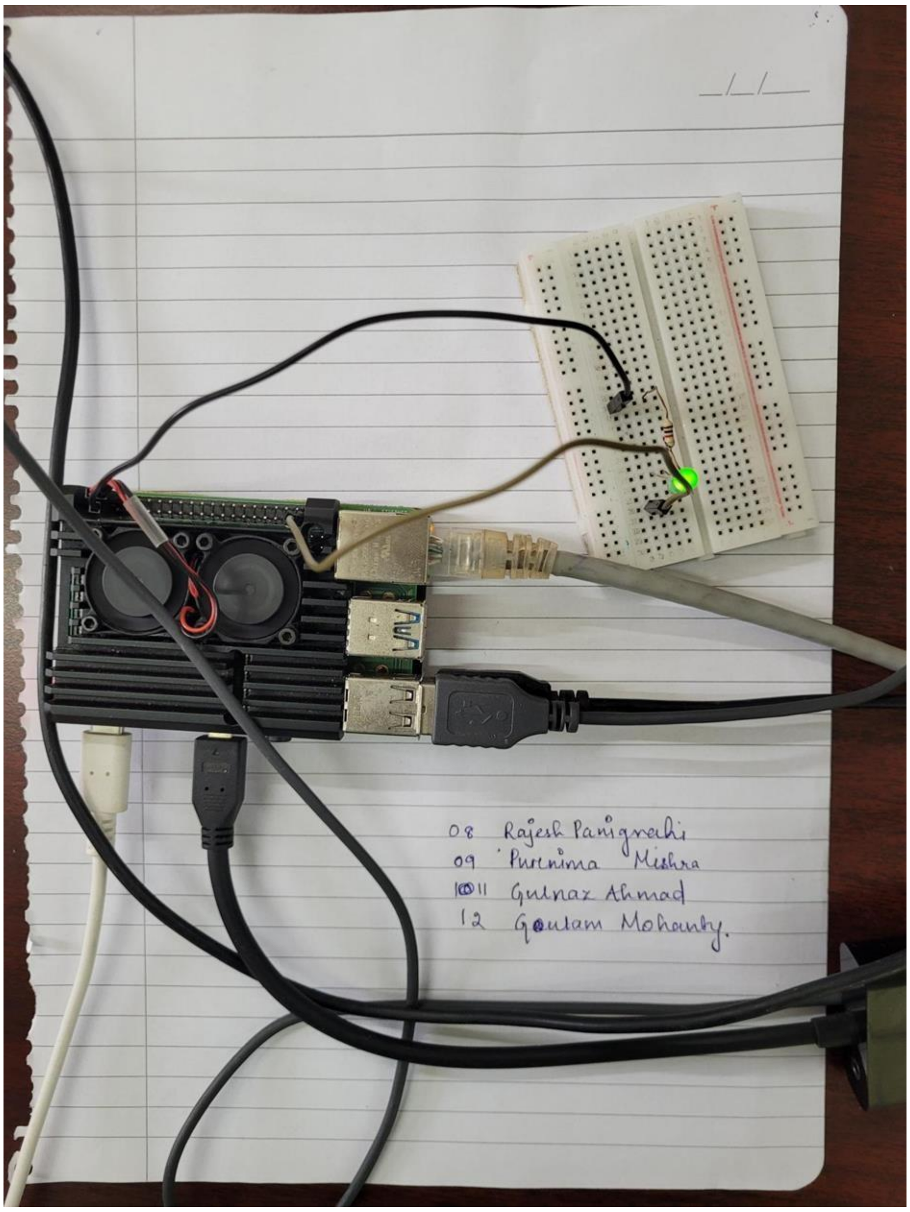
Components Required:

1. Raspberry Pi Model 4B
2. Power adapter, keyboard, mouse, and monitor
3. Breadboard
4. LED Light
5. Resistor
6. Male to female connector (2×)

Connection Details:

- Locate the GPIO pins on our Raspberry Pi. Here, we use GPIO2 (pin 3) and ground (GND) pin.
- Place the LED on the breadboard. Put the resistor in the breadboard.
- Connect the longer (+)ve leg of the LED with the resistor and the smaller (–)ve leg elsewhere.
  Using jumper wire, we will connect the GPIO2 pin with the resistor's other leg and ground pin with the (–)ve leg of the LED.
- After the setup, we run the Python code we write for the LED to blink with some time slice.





CODE:

```
import RPi.GPIO as rasp

import time

rasp.setwarnings(False)

rasp.setmode(rasp.BCM)

rasp.setup(2, rasp.OUT)

for i in range(1, 11):

   rasp.output(2, rasp.HIGH)
```

```
time.sleep(2)

rasp.output(2, rasp.LOW)

time.sleep(2)
```

Controlled:

- When we set rasp.HIGH, then the LED will ON.
- When we set rasp.LOW, then the LED will OFF.
- Here, we set sleep as 2, so after every 2 seconds, the LED will turn ON or OFF.
- Since we wrote a for loop from 1 to 11, so the light will blink for 10 times with 2 seconds gap for sleep.

Conclusion:

We performed how to blink a LED light using Raspberry Pi.

NAME: MD SHAYAHN SIDDIQUI

SIC:22BCSC21
ROLL:A2(10)

ASSIGNMENT- 04

AIM OF THE EXPERIMENT:
To control two LEDs connected to a Raspberry Pi, toggling them ON and OFF according to a predefined binary pattern.

COMPONENTS REQUIRED:

1. Raspberry Pi Model 4B
2. Power adapter, keyboard, mouse, and monitor
3. Breadboard
4. 2 LEDs
5. 2 Resistors
6. Male to female wires (4x)

CONNECTION DETAILS:

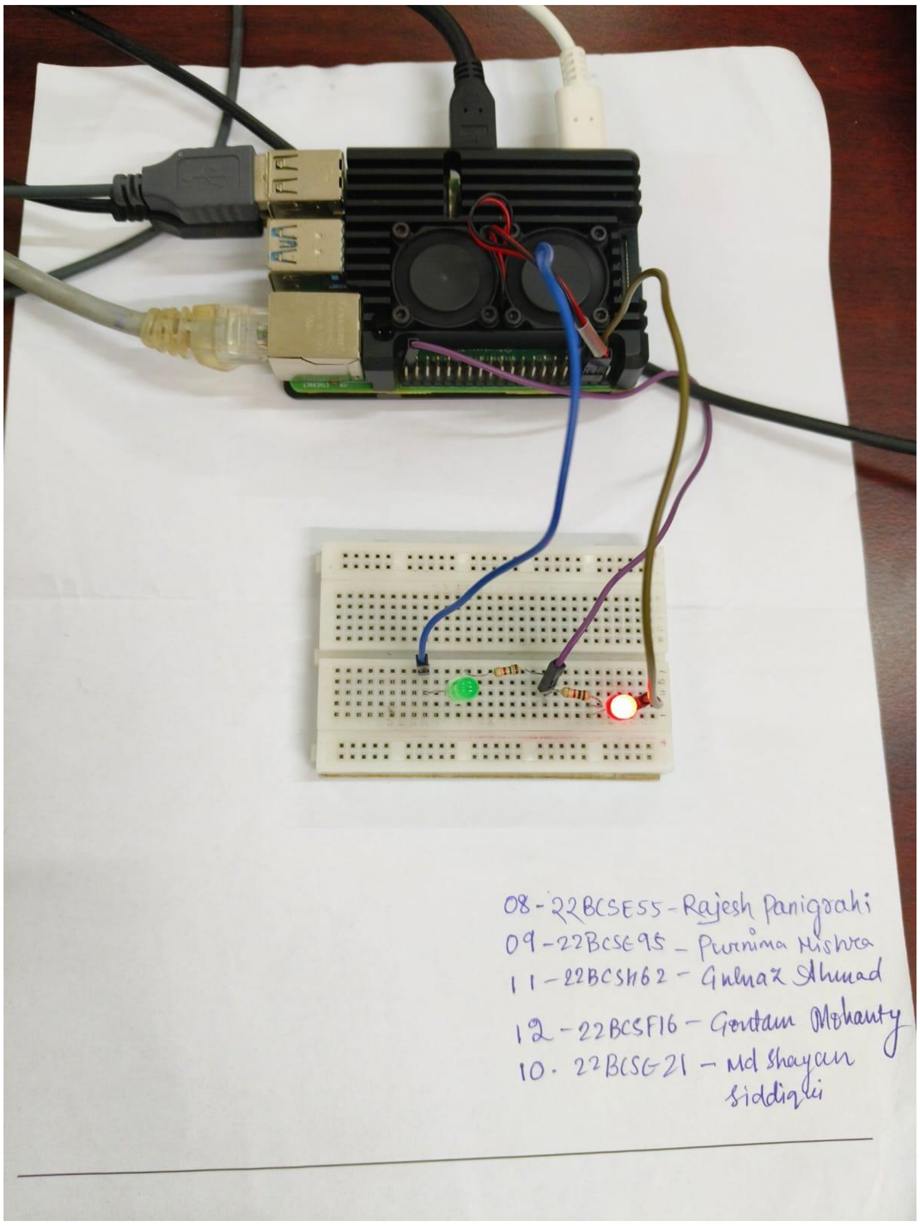
1. Locate the GPIO pins on our Raspberry Pi. Here, we use GPIO 2 and GPIO 3 as output pins and ground (GND) pins.
2. Identify the two LEDs and connect their positive terminals to GPIO 2 and GPIO 3. Put a resistor in series with each LED.
3. Connect the negative terminals of the LEDs to the ground (GND) of the Raspberry Pi via resistors, and longer (+) leg of LED with resistor.
4. Use Python to control the LEDs by turning them ON and OFF in a sequence based on binary values.

PYTHON CODE:

```
import RPi.GPIO as g

import time as t

g.setmode(g.BCM)

g.setwarnings(False)

g.setup(2, g.OUT)

g.setup(3, g.OUT)


x, y = "101101", "010110"
```

```
for i in range(6):
   g.output(2, int(x[i]))
   g.output(3, int(y[i]))
   t.sleep(2)
g.output(2, 0)
g.output(3, 0)
```

OUTPUT:


08-22BCSE55-Rajesh Panigrahi
09-22BCSE95 _ Purnima Mishra
11-22BCSH62 - Gulnaz Ahmad
12-22BCSF16 - Goutam Mohanty
10. 22BCSG21 - Md Shayan Siddiqui

CONTROLLED BEHAVIOUR:

- Setting GPIO HIGH turns ON the respective LED.
- Setting GPIO LOW turns OFF the respective LED.
- The sleep function delays the transition for 2 seconds.
- The loop runs through a predefined binary sequence, toggling the LEDs accordingly.

CONCLUSION:
We have successfully demonstrated how to control two LEDs using a Raspberry Pi by toggling them in a sequential pattern based on binary values.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT - 05

AIM OF THE EXPERIMENT:
To control multiple LEDs in a specific sequence using Raspberry Pi.

COMPONENTS REQUIRED:

1. Raspberry Pi Model 4B
2. Power adapter, keyboard, mouse, and monitor
3. Breadboard
4. 3 LED Lights
5. Resistors (×3)
6. Male to female connectors (6×)

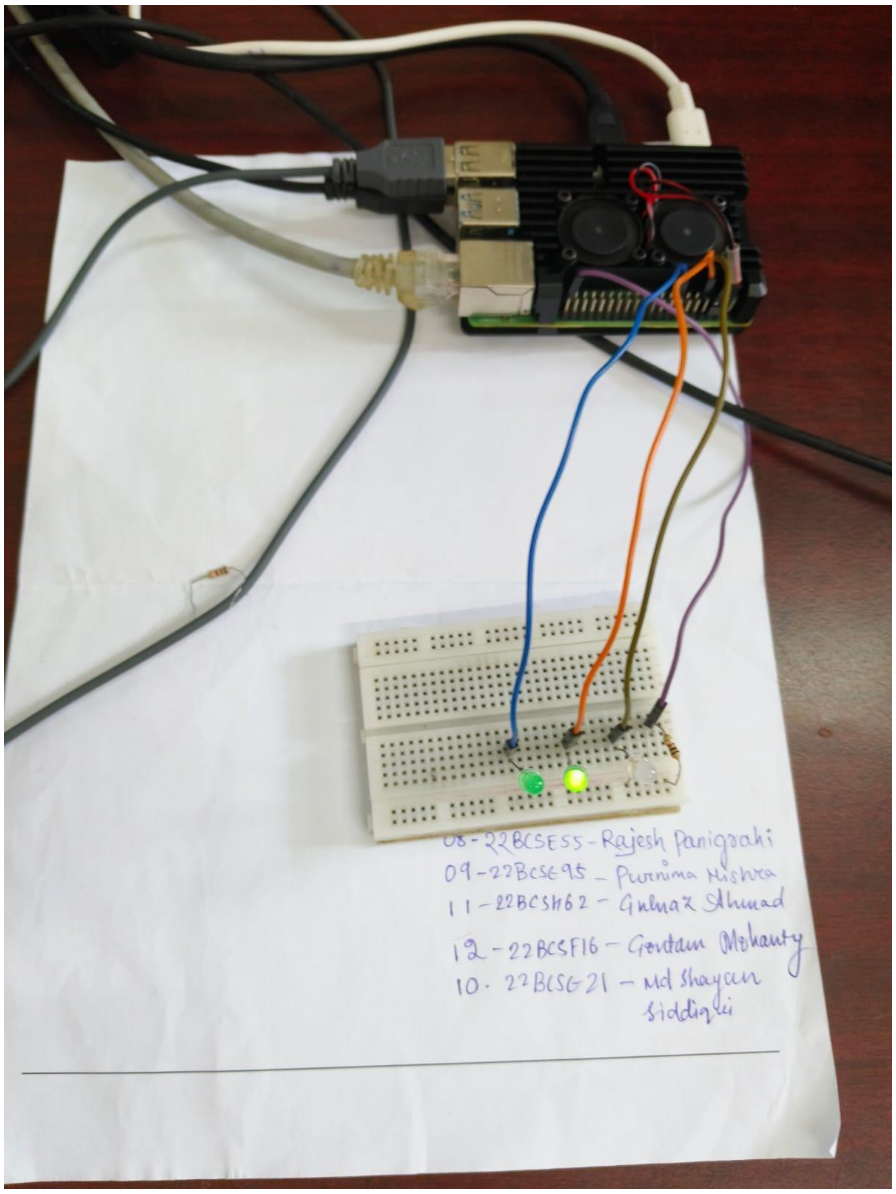
CONNECTION DETAILS:

1. Locate the GPIO pins on our Raspberry Pi. Here, we use GPIO 17, GPIO 27, and GPIO 22 as output pins and ground (GND) pins.
2. Place the LEDs on the breadboard. Put a resistor in series with each LED.
3. Connect the longer (+) leg of each LED with a resistor and the smaller (–) leg elsewhere.
4. Using jumper wires, connect GPIO 17, GPIO 27, and GPIO 22 to the resistors' other legs. Connect the ground pin (GND) to the (–) legs of the LEDs.
5. After the setup, we run the Python code to turn the LEDs ON and OFF in a pre-defined sequence.

PYTHON CODE:

```
import RPi.GPIO as f

import time as t

f.setwarnings(False)

f.setmode(f.BOARD)

f.setup(3, f.IN)

f.setup(5, f.OUT)

try:

  while True:
        flame = f.input(3)
        if flame == f.LOW:
            f.output(5, f.HIGH)  # Turn LED ON
        else:
            f.output(5, f.LOW)  # Turn LED OFF
        t.sleep(2)
except KeyboardInterrupt:
    f.cleanup()
```

OUTPUT:


08-22BCSE55-Rajesh Panigrahi
09-22BCSE95 _ Purnima Mishra
11-22BCSH62 - Gulnaz Ahmad
12-22BCSF16 - Goutam Mohanty
10. 22BCSG21 - Md Shayan Siddiqui

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT - 06

AIM OF THE EXPERIMENT:
To control a single LED to glow in red, blue and green using Raspberry Pi.

COMPONENTS REQUIRED:

1. Raspberry Pi Model 4B
2. Power adapter, keyboard, mouse and monitor
3. Breadboard
4. 1 RGB LED, Relaya
5. 1 Resistor LED
6. Male to female jumper wires (4x)

CONNECTION DETAILS:

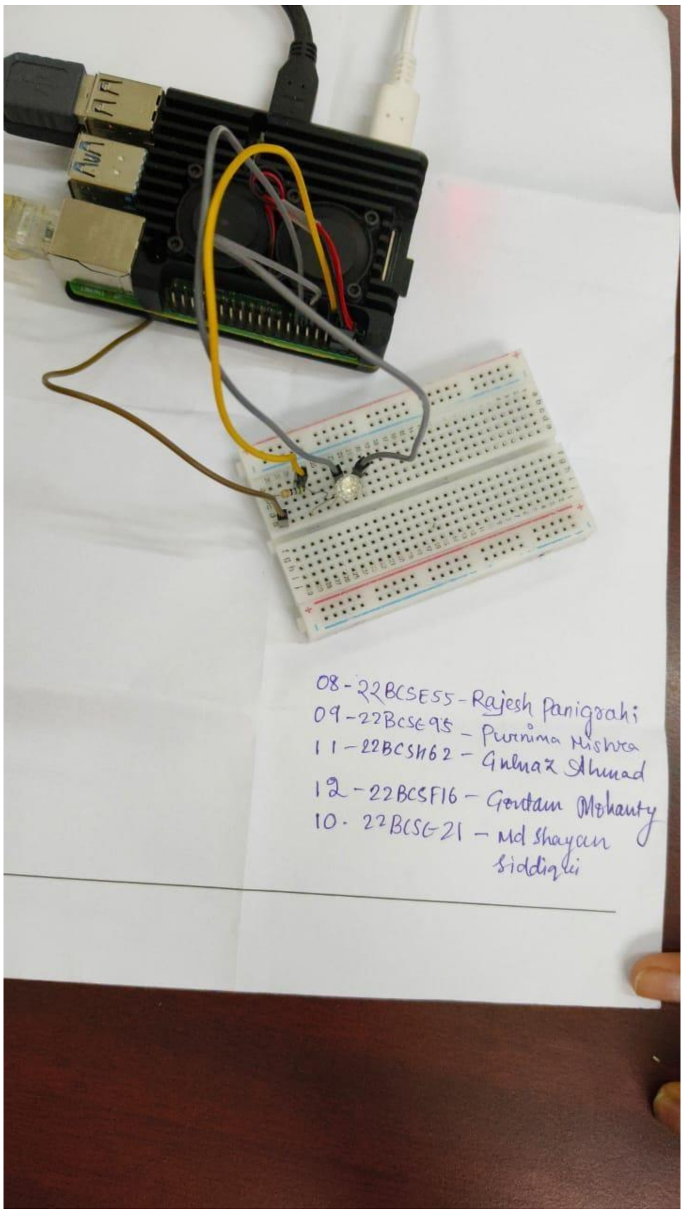
1. Locate the GPIO pins on the Raspberry Pi.
2. Identify the four pins of the RGB LED: Red (R), Green (G), Blue (B), and the common cathode (–) i.e.
3. Connect the common negative leg (–) ve of the LED to the ground (GND) of the Raspberry Pi.
4. Connect the Red, Green, and Blue pins of the LED to GPIO 17, GPIO 27 and GPIO 22 respectively through a single resistor.
5. After setup, run the Python code to turn the LED ON in red, blue and green.

PYTHON CODE:

```
import RPi.GPIO as g

import time as t

g.setmode(g.BCM)

g.setwarnings(False)

g.setup(17, g.OUT)  # RED

g.setup(27, g.OUT)  # GREEN

g.setup(22, g.OUT)  # BLUE

colors = [(1,0,0), (0,1,0), (0,0,1)]

for i in colors:
```

```
g.output(17, i[0])
  g.output(27, i[1])
  g.output(22, i[2])
  t.sleep(2)
```

OUTPUT


08-22BCSE55-Rajesh Panigrahi
09-22BCSE95 - Purnima Mishra
11-22BCSH62 - Gulnaz Ahmad
12-22BCSF16 - Goutam Mohanty
10. 22BCSG21 - Md Shayan Siddiqui

CONTROLLED BEHAVIOR:

- Setting GPIO HIGH turns on the respective LED color.
- Setting GPIO LOW turns off the respective LED color.
- The sleep function delays the colour transition for 2 seconds.

- The loop runs through red, green and blue colors in sequence.

CONCLUSION:

We have successfully demonstrated how to control an RGB LED using a Raspberry Pi by turning it ON and OFF in different colours sequentially.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT-7

Aim of the Experiment:

To detect the presence of flame using a **flame sensor** and trigger an LED using **Raspberry Pi GPIO** based on the sensor's output.

Components Required:

- Raspberry Pi
- Breadboard
- Flame sensor module
- 1x LED
- 1x Resistor (220Ω)
- Jumper wires

Connection Details:

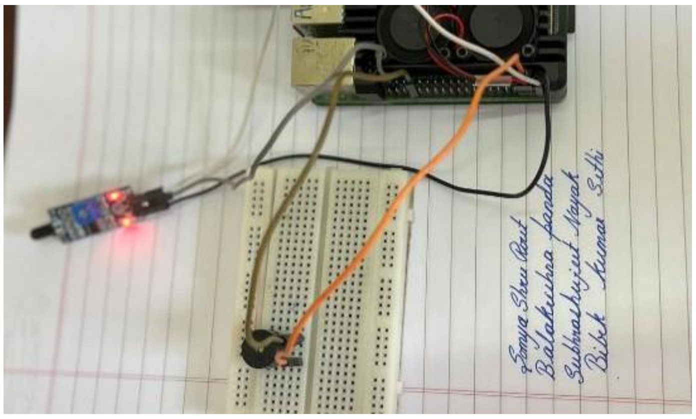
From the image:

- The **flame sensor module** has three pins: VCC, GND, and D0 (Digital Output).
    - **VCC** → 3.3V or 5V pin of Raspberry Pi
    - **GND** → Ground pin of Raspberry Pi
    - **D0** → GPIO pin 3 (used as input)
- The LED's **anode** is connected to **GPIO pin 5** via a resistor.
- The **cathode** is connected to **GND**.

Code (converted from image to text):

```
import RPi.GPIO as f
import time as t
f.setwarnings(False)
f.setmode(f.BOARD)
f.setup(3, f.IN)
f.setup(5, f.OUT)
while True:
  flame = f.input(3)
  if (flame == f.LOW):
    f.output(5, f.HIGH)
  else:
    f.output(5, f.LOW)
  t.sleep(2)
f.cleanup()
```

OUTPUT:



Conclusion:

The experiment demonstrates **flame detection** using a digital flame sensor with Raspberry Pi. When the flame is detected (sensor output goes LOW), the LED turns ON as an alert signal. This project exemplifies basic **digital sensor interfacing**, **condition-based control**, and **GPIO usage** in real-time monitoring applications.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT-8

Aim of the Experiment:

To control the brightness of an LED using **PWM (Pulse Width Modulation)** with a Raspberry Pi.

Components Required:

- Raspberry Pi (with GPIO pins)
- Breadboard
- 1x LED
- 1x Resistor (220Ω or 330Ω)
- Jumper wires

Connection Details:

From the image:

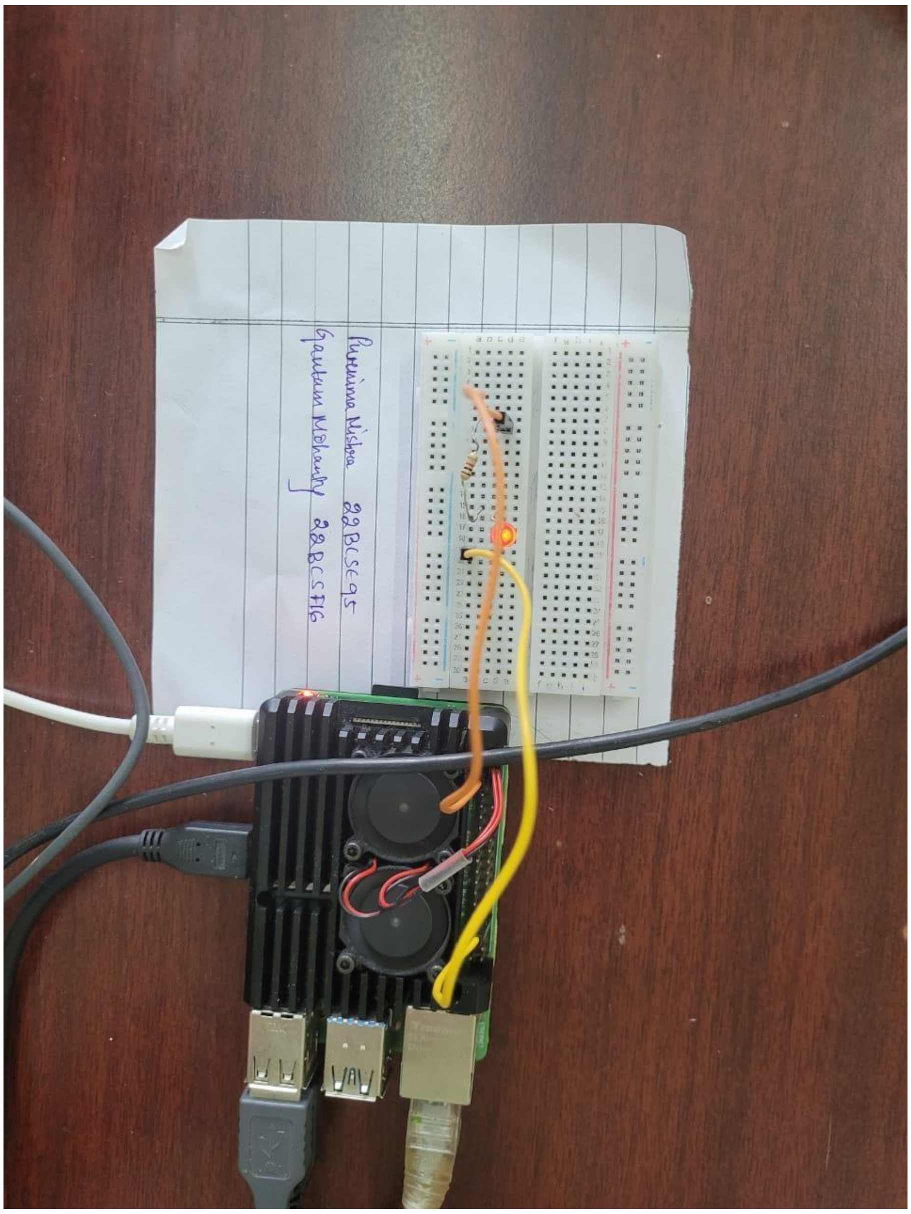
- The **long leg (anode)** of the LED is connected to **GPIO pin 5** of the Raspberry Pi via a **current-limiting resistor**.
- The **short leg (cathode)** of the LED is connected to the **GND** rail of the breadboard.
- GND of the breadboard is connected to a **GND pin** on the Raspberry Pi.
- GPIO pin 5 is used for **PWM signal** output to control LED brightness.

Code (converted from image to text):

```
import RPi.GPIO as f

import time as t

f.setwarnings(False)

f.setmode(f.BOARD)

f.setup(5, f.OUT)

p = f.PWM(5, 100)

p.start(100)

for i in range(100):

  p.ChangeDutyCycle(100 - i)
```

```
    t.sleep(0.05)
p.ChangeDutyCycle(0)
t.sleep(1)
for i in range(100):
    p.ChangeDutyCycle(i)
    t.sleep(0.05)
f.cleanup()
```

OUTPUT:



Conclusion:

The experiment successfully demonstrates **PWM control of LED brightness** using a Raspberry Pi. The LED gradually dims from full brightness to off and then gradually brightens back up. This showcases how **duty cycle manipulation** in PWM can be used to achieve analog-like behavior using digital outputs.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT-G

Aim of the Experiment:

To control an LED connected to a Raspberry Pi **remotely** through the **Adafruit IO dashboard**, using Python and GPIO.

Components Required:

- Raspberry Pi (with internet access)
- Breadboard
- 1x LED
- 1x Resistor (220Ω)
- Jumper wires
- Adafruit IO account

Connection Details:

From the breadboard image:

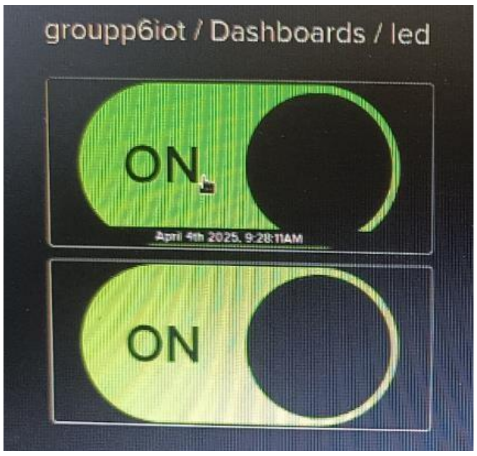
- The **LED** is connected in series with a **resistor**.
  - **Anode (long leg)** of LED → GPIO pin of Raspberry Pi
  - **Cathode (short leg)** of LED → GND (via resistor)

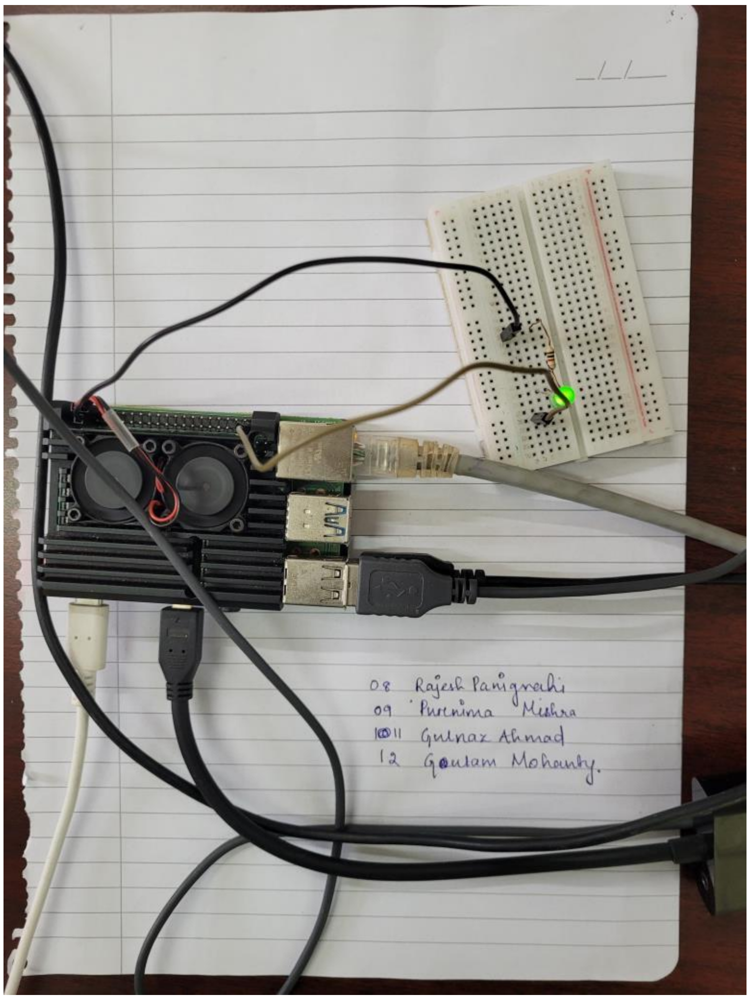
The control is achieved remotely via the **Adafruit IO dashboard** where a toggle switch sends commands to turn the LED ON or OFF.

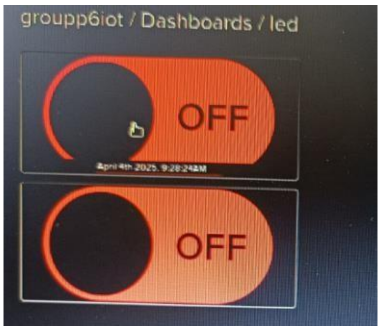
Python Code (from image, converted to text):

```
import time

import board

import digitalio

from Adafruit_IO import MQTTClient

AIO_FEED = "led"

led = digitalio.DigitalInOut(board.D18)

led.direction = digitalio.Direction.OUTPUT

def connected(client):

  print("Connected to Adafruit IO!")
  client.subscribe(AIO_FEED)
def disconnected(client):
  print("Disconnected from Adafruit IO!")
  sys.exit(1)
def message(client, feed_id, payload):
  print(f"Received: {payload}")
  if payload == "ON":
    led.value = True
  else:
    led.value = False
client = MQTTClient("groupp6iot ", " aio_mgzX59FXpRTgSHM6hzKcf0Hkj9jd ")
client.on_connect = connected
client.on_disconnect = disconnected
client.on_message = message
client.connect()
client.loop_blocking()
```

OUTPUT:




08 Rajesh Panigrahi
09 Purnima Mishra
11 Gulnaz Ahmad
12 Goutam Mohanty.


groupp6iot / Dashboards / led
OFF
April 4th 2025, 9:28:24AM
OFF

Conclusion:

This experiment shows how **IoT** (Internet of Things) can be used to **remotely control hardware** components using cloud platforms like **Adafruit IO**. The Raspberry Pi listens to feed updates and controls the LED in real time. This introduces fundamental concepts of **cloud integration**, **MQTT protocol**, and **remote device control**.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

<u>ASSIGNMENT -10</u>

<u>OBJECTIVE</u>: Blinking LED Bulb using Raspberry Pi and relay module

<u>COMPONENTS:</u>

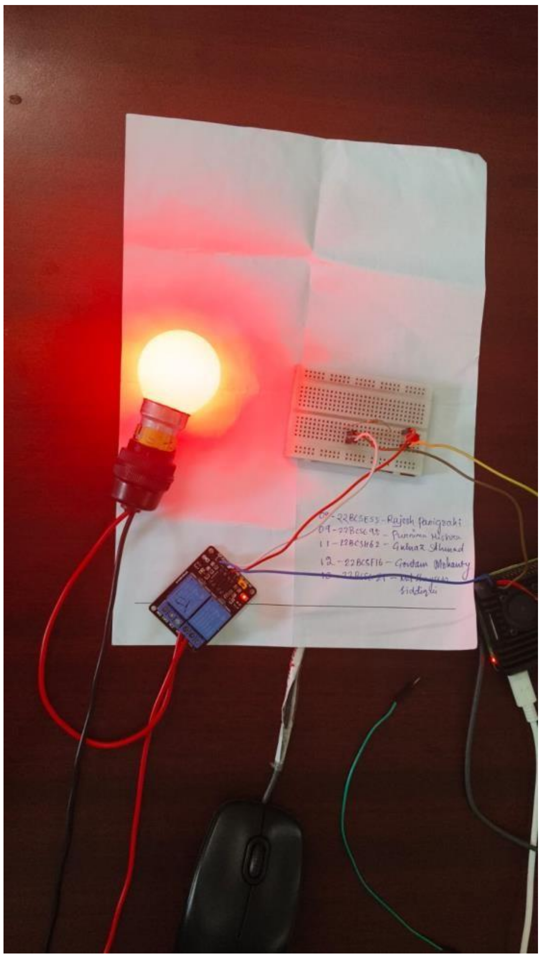
1. Raspberry Pi Model 4B
2. Power Adapter, Keyboard, Mouse, and Monitor
3. Breadboard
4. Single LED Bulb
5. Relay Module
6. Connecting Wires

<u>Code for Blinking LED Bulb:</u>

```
import RPi.GPIO as g from
time import sleep
g.setwarnings(False)
g.setmode(g.BCM)
g.setup(3, g.OUT)
while True:
   g.output(3, 1)
   sleep(0.5)
   g.output(3, 0)
   sleep(0.5)
```

<u>EXPERIMENT SETUP:</u> The circuit consists of an LED bulb connected to a relay module, which is further connected to the Raspberry Pi's GPIO pin. The LED alternates between ON and OFF states at a fixed interval, creating a blinking effect.

<u>IMAGE OF THE EXPERIMENT:</u>



OUTPUT:

The LED bulb blinks continuously with a 0.5-second interval, demonstrating the control of external components using Raspberry Pi GPIO pins .

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

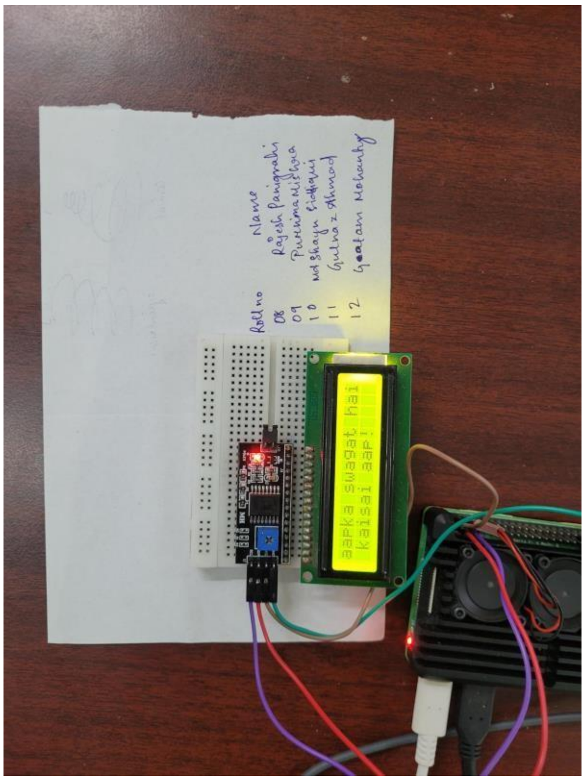
ASSIGNMENT-11

OBJECTIVE: Display text on LCD using raspberry pi.

Code for Display text on LCD:

```
From RPLCD.i2c import charLCD
lcd=charLCD('PCF8574',0x74)
lcd.write_string("Kris,Biv,Prat")
```

IMAGE OF THIS EXPERIMENT:



OUTPUT:

The text was displayed on the LCD display panel.

Name: MD SHAYAHN
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT -12

OBJECTIVE: To detect fire using smoke detector and buzzer.

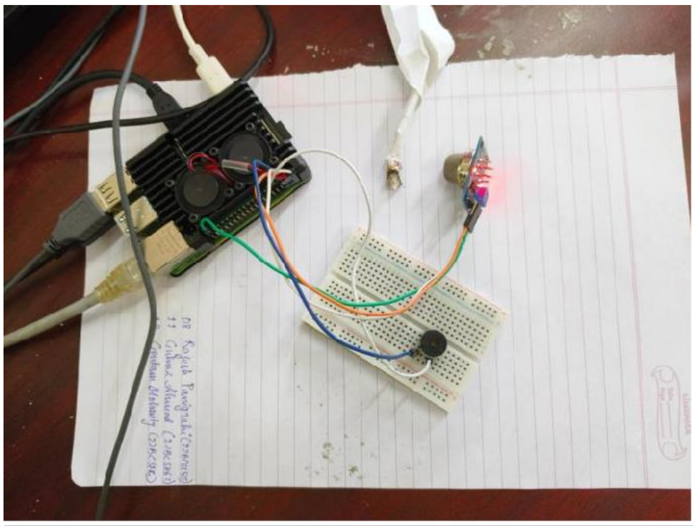
COMPONENTS:

- Raspberry Pi model 4B
- Power Adapter,Keyboard,Mouse and Monitor
- Smoke Detector
- Breadboard
- Buzzer
- Male to Female connectors

Code for Flame sensor

```
import RPi.GPIO as g
from time import sleep
g.setwarnings(False)
g.setmode(g.BOARD)
g.setup(3, g.IN)
g.setup(5, g.OUT)
while True:
   x = g.input(3)
   if x == g.LOW:
      g.output(5, g.HIGH)
   else:
      g.output(5, g.LOW)
   sleep(2)
```

IMAGE OF THE EXPERIMENT:



OUTPUT:

When smoke is detected, the sensor sends a LOW signal,
and the buzzer turns ON. If no smoke is detected,
the sensor sends a HIGH signal,
and the buzzer stays OFF.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT-13

OBJECTIVE: To detect alcohol using buzzer.

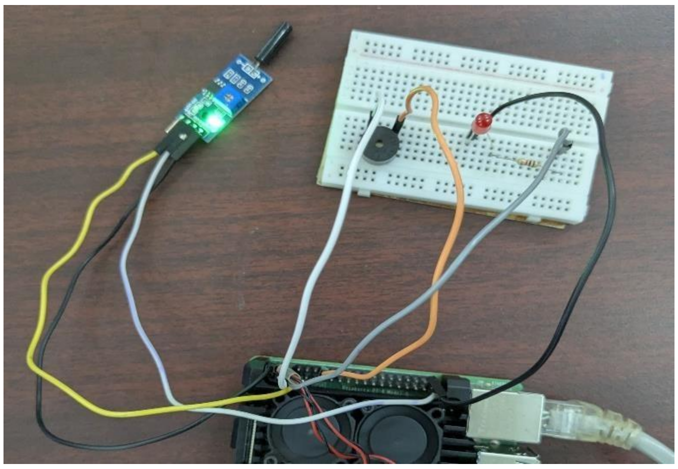
COMPONENTS:

- Raspberry Pi model 4B
- Power Adapter,Keyboard,Mouse and Monitor
- Flame sensor
- Breadboard
- Buzzer
- LED
- Male to Female connectors

Code for Alcohol detector:

```
import RPi.GPIO as gp
import time
gp.setwarnings(False)
gp.setmode(gp.BOARD)
gp.setup(3, gp.IN)
gp.setup(5, gp.OUT)
gp.setup(11, gp.OUT)
while(True):
    flame_detected = gp.input(3)
    if flame_detected == gp.HIGH:
        print("Flame Detected! Buzzer ON")
        gp.output(5, gp.HIGH)
        gp.output(11, gp.HIGH)
    else:
        print("No Flame Detected. Buzzer OFF")
        gp.output(5, gp.LOW)
        gp.output(11, gp.LOW)
    time.sleep(1)
gp.cleanup()
```

IMAGE OF THE EXPERIMENT:



OUTPUT:

The code turns on a buzzer and LED when flame is detected, prints a message, and turns them off
when no flame is present. It runs in a loop until stopped.

Name: MD SHAYAHN
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT-14

OBJECTIVE: To detect alcohol using buzzer.

COMPONENTS:

- Raspberry Pi model 4B
- Power Adapter,Keyboard,Mouse and Monitor
- Alcohol Detector
- Breadboard
- Buzzer
- LED
- Male to Female connectors

Code for Alcohol detector:

```
import RPi.GPIO as gp
import time
gp.setwarnings(False)
gp.setmode(gp.BOARD)
gp.setup(3, gp.IN)
gp.setup(5, gp.OUT)
gp.setup(11, gp.OUT)
while(True):
    alcohol_detected = gp.input(3)
    if alcohol_detected == gp.HIGH:
        print("Alcohol Detected! Buzzer ON")
        gp.output(5, gp.HIGH)
        gp.output(11, gp.HIGH)
    else:
        print("No Alcohol Detected. Buzzer OFF")
        gp.output(5, gp.LOW)
        gp.output(11, gp.LOW)
    time.sleep(1)
gp.cleanup()
```

OUTPUT:

The code turns on a buzzer and LED when alcohol is detected, prints a message, and turns them off when no alcohol is present. It runs in a loop until stopped.

Name: MD SHAYAHN
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)

ASSIGNMENT -15

OBJECTIVE: To detect presence of raindrops using raindrop module sensor.

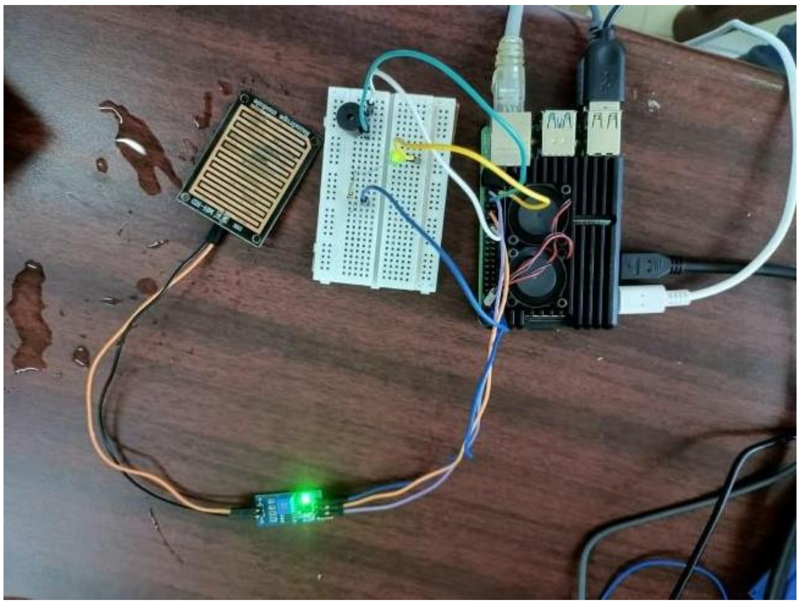
COMPONENTS:

- Raindrop Sensor Module
- Raspberry Pi 4B
- Breadboard
- LED
- Buzzer
- Resistor
- Male to Female Connectors

Code for raindrop sensor

```
import RPi.GPIO as GPIO
import time

# GPIO pin setup
RAIN_SENSOR_PIN = 17
LED_PIN = 27
BUZZER_PIN = 22

GPIO.setmode(GPIO.BCM)
GPIO.setup(RAIN_SENSOR_PIN,
GPIO.IN)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(BUZZER_PIN,
GPIO.OUT)

try:
  while True:
    if
GPIO.input(RAIN_SENSOR_PIN) ==
GPIO.LOW:
      print("Rain detected!")
      GPIO.output(LED_PIN,
GPIO.HIGH) # Turn on LED
      GPIO.output(BUZZER_PIN,
GPIO.HIGH) # Turn on Buzzer
    else:
      print("No rain.")
      GPIO.output(LED_PIN,
GPIO.LOW)
      GPIO.output(BUZZER_PIN,
GPIO.LOW) # Turn off Buzzer
    time.sleep(1)

except KeyboardInterrupt:
  print("Program stopped")
  GPIO.cleanup()
```

IMAGE OF THIS EXPERIMENT:



OUTPUT:

When the sensor detects water (rain), the LED and buzzer turn ON and “Rain detected!”is printed. If no water is detected, the LED and buzzer remain OFF and “No rain.” is printed.

Name: MD SHAYAHN SIDDIQUI
SIC: 22BCSC21
Roll No: 10
Branch: CSE - A (2)