**Relay Control using Raspberry Pi**

**Introduction:**

A relay is an electrical switch that allows you to control high-power devices like home appliances, lights, or motors using low-power signals from a Raspberry Pi. Since the Raspberry Pi operates at 3.3V logic, it cannot directly control high-power devices. A relay acts as a bridge between the Raspberry Pi and these devices, making automation possible.

**Components Required:**

- Raspberry Pi (any model with GPIO pins)
- Relay Module (5V or 12V, depending on your application)
- Jumper Wires
- Power Supply (as per relay module specifications)
- Load (such as a bulb, fan, or motor)

**Circuit Connection:**

1. Power Connection – Connect the VCC pin of the relay module to the 5V pin of the Raspberry Pi. Connect the GND of the relay to the GND pin of the Raspberry Pi.
2. Control Signal – Connect the IN pin of the relay module to a GPIO pin on the Raspberry Pi (e.g., GPIO17).
3. Load Connection – Connect one end of the electrical device (bulb, motor, etc.) to the COM (Common) terminal of the relay and the other to the power source. Connect the NO (Normally Open) terminal of the relay to complete the circuit.
4. Testing – When the Raspberry Pi sends a signal to the relay, the switch will activate, allowing current to flow through the load.

**Applications:**

- Home Automation – Controlling lights, fans, and appliances remotely.
- Industrial Automation – Automating machines and conveyor belts.
- Security Systems – Controlling alarms and door locks.
- Smart Irrigation – Turning water pumps on and off based on sensor data.

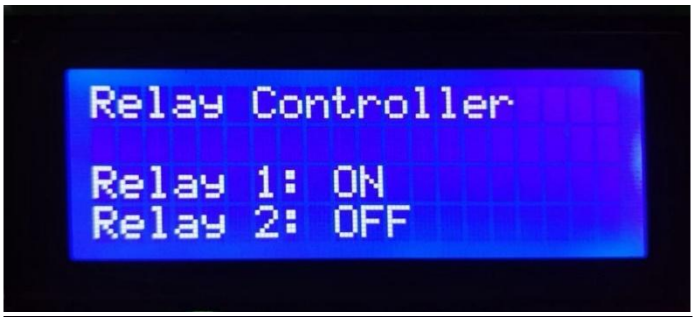
**Learnings:**

- Understand how a relay module works and how it interacts with a microcontroller.
- Learn to interface hardware components with a Raspberry Pi.
- Gain insights into automation and IoT applications.
- Develop skills in handling electrical circuits safely.

**Conclusion:**

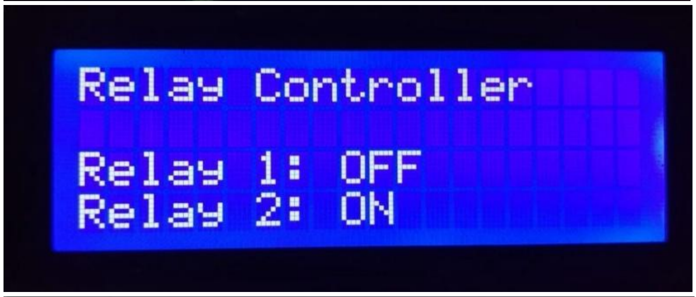
Relay control using Raspberry Pi is a simple yet powerful technique that enables automation and remote control of electrical devices. By integrating relays with sensors and programming logic, you can create smart home solutions, industrial automation systems, and much more. This project provides a foundation for working with IoT-based applications and embedded systems.

**Outcome:**



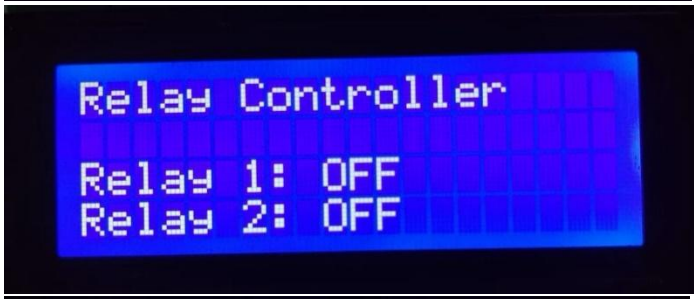
```
Relay Controller

Relay 1: ON
Relay 2: OFF
```



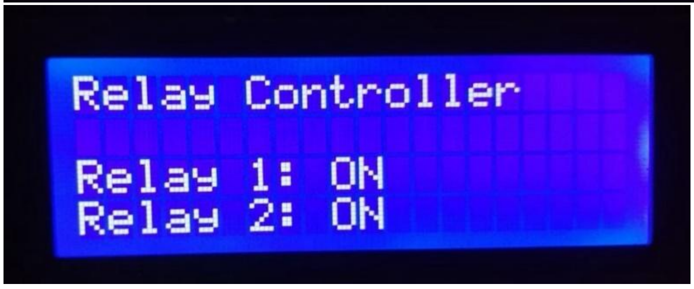
```
Relay Controller

Relay 1: OFF
Relay 2: ON
```



```
Relay Controller

Relay 1: OFF
Relay 2: OFF
```



```
Relay Controller

Relay 1: ON
Relay 2: ON
```

**Program:**

```
import RPi.GPIO as GPIO

import time

from RPLCD.i2c import CharLCD


# GPIO Pin Definitions

RELAY1 = 26  # Relay 1 Control

RELAY2 = 27  # Relay 2 Control

BUTTON1 = 23  # Button 1 (Toggle Relay 1)
```

```
BUTTON2 = 24  # Button 2 (Toggle Relay 2)

GPIO.setwarnings(False)

# Setup GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(RELAY1, GPIO.OUT, initial=GPIO.LOW)  # Start with relays OFF

GPIO.setup(RELAY2, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(BUTTON1, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # Pull-up resistor

GPIO.setup(BUTTON2, GPIO.IN, pull_up_down=GPIO.PUD_UP)


# Initialize LCD

lcd = CharLCD('PCF8574', 0x27)


# Relay states

relay1_state = False

relay2_state = False


def update_display():

    """Update the LCD with the relay states."""

    lcd.clear()

    lcd.cursor_pos = (0, 0)

    lcd.write_string("Relay Controller\n")

    lcd.cursor_pos = (2, 0)

    lcd.write_string(f"Relay 1: {'ON' if relay1_state else 'OFF'}\n")

    lcd.cursor_pos = (3, 0)

    lcd.write_string(f"Relay 2: {'ON' if relay2_state else 'OFF'}")


try:

    update_display()

    while True:

        if GPIO.input(BUTTON1) == GPIO.LOW:  # Button 1 Pressed

            relay1_state = not relay1_state  # Toggle state
            GPIO.output(RELAY1, relay1_state)  # Update relay
            update_display()
            time.sleep(0.3)  # Debounce delay

        if GPIO.input(BUTTON2) == GPIO.LOW:  # Button 2 Pressed
            relay2_state = not relay2_state  # Toggle state
            GPIO.output(RELAY2, relay2_state)  # Update relay
            update_display()
            time.sleep(0.3)  # Debounce delay

except KeyboardInterrupt:
    GPIO.cleanup()
    lcd.clear()
    print("Program Stopped")
```