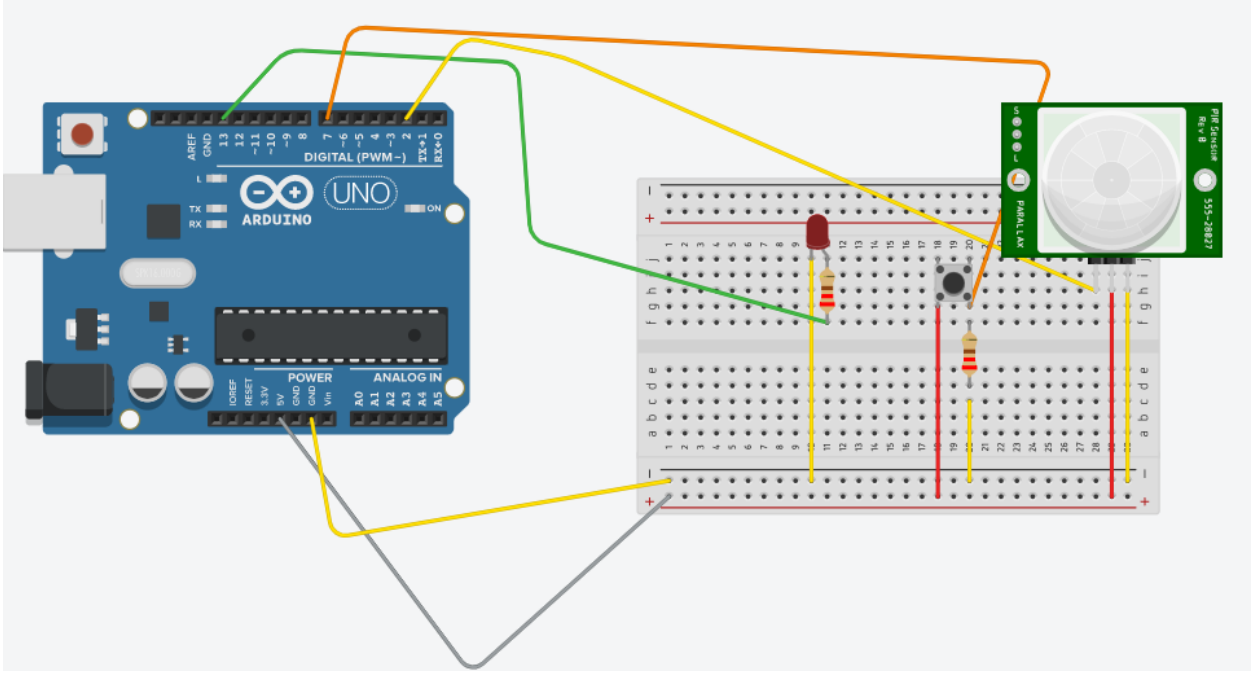
Program to demonstrate working with LED, Button, PIR(Passive Infrared Sensor) sensor

To demonstrate how to work with an LED, a button, and a PIR (Passive Infrared) sensor using an Arduino, you can create a simple project that combines these components. Here's an example of how you might set up a system where the LED is turned on when motion is detected by the PIR sensor and the button can be used to reset or toggle the system.



```
// Define pin numbers
const int pirPin = 2;        // Pin connected to PIR sensor OUT
const int ledPin = 13;       // Pin connected to LED
const int buttonPin = 7;     // Pin connected to pushbutton

// Variables
bool motionDetected = false;
bool buttonPressed = false;

void setup() {
  pinMode(pirPin, INPUT);     // Set PIR pin as input
  pinMode(ledPin, OUTPUT);    // Set LED pin as output
  pinMode(buttonPin, INPUT_PULLUP);  // Set button pin as input with pull-up resistor

  Serial.begin(9600);         // Initialize serial communication
}

void loop() {
  int pirState = digitalRead(pirPin);    // Read PIR sensor state
  buttonPressed = (digitalRead(buttonPin) == LOW); // Check if button is pressed (assuming active LOW)
```

```
  // Check if motion is detected
  if (pirState == HIGH) {
    motionDetected = true;
    Serial.println("Motion Detected!");
  } else {
    motionDetected = false;
    Serial.println("No Motion");
  }

  // Control LED based on PIR sensor and button state
  if (motionDetected || buttonPressed) {
    digitalWrite(ledPin, HIGH); // Turn on LED
  } else {
    digitalWrite(ledPin, LOW);  // Turn off LED
  }

  delay(100); // Small delay to avoid bouncing issues
}
```

To demonstrate how to work with an LED, a button, and a PIR (Passive Infrared) sensor using an Arduino, you can create a simple project that combines these components. Here's an example of

how you might set up a system where the LED is turned on when motion is detected by the PIR sensor and the button can be used to reset or toggle the system.

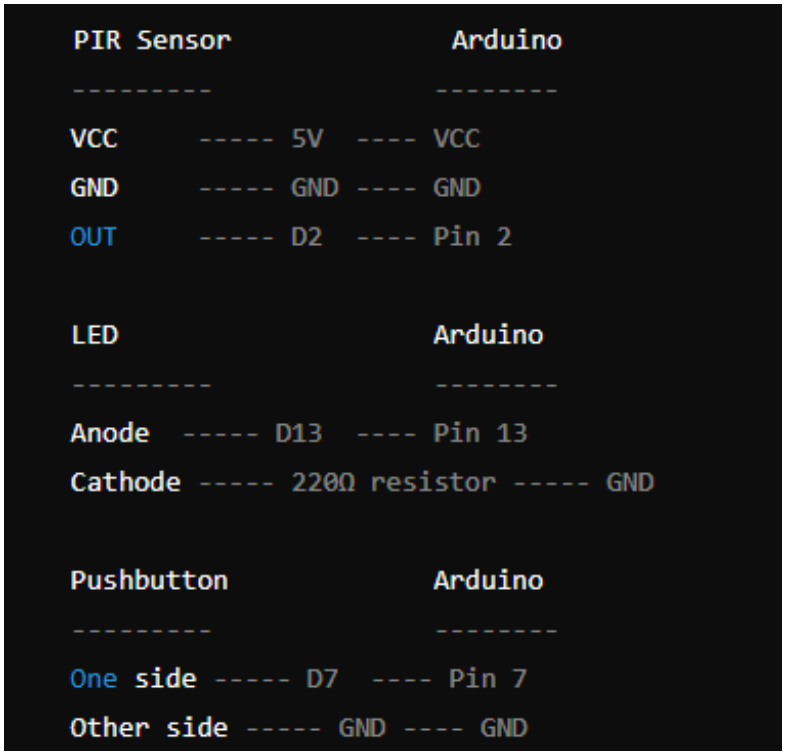
Components Required:

1. **Arduino Board** (e.g., Arduino Uno)
2. **PIR Sensor**
3. **LED**
4. **Resistor** (220 ohms for the LED) and **Resistor** (10k ohms for the LED)
5. **Pushbutton**
6. **Breadboard and Jumper Wires**

Wiring Diagram:

1. **PIR Sensor Connections:**
    - **VCC** to **Arduino 5V**
    - **GND** to **Arduino GND**
    - **OUT** to **Arduino Digital Pin 2**
2. **LED Connections:**
    - **Long Leg (Anode)** to **Arduino Digital Pin 13**
    - **Short Leg (Cathode)** to **one end of the 220-ohm resistor**
    - **Other end of the resistor** to **Arduino GND**
3. **Pushbutton Connections:**
    - **One side of the button** to **Arduino Digital Pin 7**
    - **Other side of the button** to **Arduino GND**
    - **Optional:** You might use a pull-up resistor (10k ohms) between the button pin and 5V for more reliable operation.

Circuit Schematic:



Arduino Code:

This code will turn on the LED when motion is detected by the PIR sensor. Pressing the button will reset the system or change its behavior, depending on your requirements.

Explanation of the Code:

1. **Setup Function:**
    - Initializes the pins for the PIR sensor, LED, and button.
    - Begins serial communication for debugging purposes.
2. **Loop Function:**
    - Reads the state of the PIR sensor and button.
    - Updates the motionDetected variable based on PIR sensor input.
    - Updates the buttonPressed variable based on button input.
    - Turns the LED on if motion is detected or if the button is pressed; otherwise, turns it off.

- Prints status messages to the serial monitor for debugging.

Additional Notes:

- **Button Behavior:** The button is set up with a pull-up resistor to simplify wiring and avoid floating inputs. The code checks if the button is pressed (active LOW), which will change the LED's behavior.
- **PIR Sensor Adjustment:** The PIR sensor might need a few seconds to warm up after powering on before it starts detecting motion reliably.
- **LED Indication:** The LED will turn on if either motion is detected or the button is pressed, which helps visualize the sensor and button states.

This project combines basic input (button) and sensor (PIR) functionality to control an output (LED), providing a straightforward demonstration of how to use these components with an Arduino.

Here's a concise summary of the **why** and **how** of using a **button**, **PIR sensor**, and **LED** in an electronic project:

Why Use Each Component:

1. **Button:**
   - **Purpose:** Allows user interaction with the system. Buttons can be used to trigger actions, input data, or toggle states.
   - **Application:** Commonly used in user interfaces for initiating actions like starting or stopping processes, or in combination with other sensors for controlling devices.
2. **PIR Sensor:**
   - **Purpose:** Detects motion by sensing changes in infrared radiation (heat) emitted by moving objects, typically humans or animals.

- **Application:** Often used in security systems to detect intruders, in automated lighting systems to turn on lights when someone enters a room, or in energy management systems to save power by controlling devices based on occupancy.

3. **LED:**
   - **Purpose:** Provides visual feedback or indication. LEDs are used to show the status of a system or component, or to signal events such as notifications or alerts.
   - **Application:** Commonly used in projects to display whether a system is active, indicate a detected event, or serve as a status indicator.

How They Are Used Together:

1. **Button:**
   - **Usage:** The button is wired to a digital input pin on a microcontroller (e.g., Arduino). When pressed, it sends a signal (often LOW) to the microcontroller. This input can be used to trigger or modify the behavior of the system.
2. **PIR Sensor:**
   - **Usage:** The PIR sensor is connected to a digital input pin on the microcontroller. It detects motion by outputting a HIGH signal when motion is detected and a LOW signal when no motion is detected. The microcontroller reads this signal to determine the presence of motion.
3. **LED:**
   - **Usage:** The LED is connected to a digital output pin on the microcontroller through a current-limiting resistor. The microcontroller can turn the LED on or off based on inputs from the PIR sensor or button, providing visual feedback or signaling.

Example Project Integration:

- **Objective:** Create a system where an LED lights up when motion is detected by the PIR sensor, and the button can be used to toggle the LED on or off regardless of motion.
- **How It Works:**

1. **Motion Detection:** The PIR sensor detects motion and sends a HIGH signal to the microcontroller when motion is detected.
2. **Button Interaction:** The button can be pressed to override or reset the LED's state.
3. **LED Control:** The microcontroller uses the PIR sensor's signal and the button's state to control the LED. The LED turns on if motion is detected or if the button is pressed, providing clear visual feedback about the system's state.

Summary:

- **Button:** Allows user input or control.
- **PIR Sensor:** Detects motion by sensing infrared radiation changes.
- **LED:** Provides visual feedback or indication of system status.

By combining these components, you can create interactive and responsive systems where the button provides user control, the PIR sensor detects environmental changes, and the LED provides clear visual signals based on the inputs from the other components.

 Default Operation: When motion is detected by the PIR sensor, the LED lights up.

 Button Functionality:

- **Pressing the Button:** Toggles the LED on or off, regardless of the PIR sensor's state. This allows users to manually control the LED without relying solely on motion detection.
- **Extended Use Case:** If you want to debug or change the system's sensitivity, pressing the button could switch the system into a different mode or adjust settings.

Incorporating a button into your project with a PIR sensor and LED provides additional control and functionality. The button can be used to manually control the LED, reset the system, change modes, or enable/disable the PIR sensor. This enhances the versatility of the system and allows for more flexible interactions and user control.

Raspberry Pi code

---

step-by-step guide for working with an LED, a button, and a PIR sensor using a Raspberry Pi:

Materials Needed:

- Raspberry Pi (with Raspbian OS installed)
- LED
- 220-ohm resistor
- Push button
- PIR motion sensor
- Breadboard and jumper wires

Wiring Diagram:

1. **LED:**
    - **Anode (longer leg)** to **GPIO pin** (e.g., GPIO 17).
    - **Cathode (shorter leg)** to one end of a **220-ohm resistor**.
    - Other end of the resistor to **GND (Ground)** pin on the Raspberry Pi.
2. **Button:**
    - One terminal to **GPIO pin** (e.g., GPIO 27).

- The other terminal to **GND (Ground)** on the Raspberry Pi.

3. **PIR Sensor:**
   - **VCC** to **5V** (or **3.3V** if your sensor is rated for that).
   - **GND** to **GND (Ground)**.
   - **OUT** to a **GPIO pin** (e.g., GPIO 22).

Step-by-Step Procedure:

1. **Set Up Your Raspberry Pi:**
   - **Install Raspbian:** Ensure Raspbian OS is installed on your Raspberry Pi.
   - **Update Your System:**

     ```
     sudo apt-get update
     sudo apt-get upgrade
     ```

2. **Wiring:**
   - **Connect the LED:**
     1. Place the LED on the breadboard.
     2. Connect the anode (long leg) to GPIO 17.
     3. Connect the cathode (short leg) through a 220-ohm resistor to GND.
   - **Connect the Button:**
     1. Place the button on the breadboard.
     2. Connect one terminal to GPIO 27.
     3. Connect the other terminal to GND.
   - **Connect the PIR Sensor:**
     1. Connect the VCC pin to the 5V pin on the Raspberry Pi.
     2. Connect the GND pin to a GND pin on the Raspberry Pi.
     3. Connect the OUT pin to GPIO 22.
3. **Install Required Libraries:**
   - Ensure you have the GPIO library installed:

     ```
     sudo apt-get install python3-rpi.gpio
     ```

4. **Python Script:**
   - Create a Python script to control the LED with the button and PIR sensor:

     ```
     import RPi.GPIO as GPIO
     import time

     # Set up GPIO mode
     GPIO.setmode(GPIO.BCM)

     # Define GPIO pins
     LED_PIN = 17
     BUTTON_PIN = 27
     PIR_PIN = 22

     # Set up GPIO pins
     GPIO.setup(LED_PIN, GPIO.OUT)
     ```

```
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(PIR_PIN, GPIO.IN)

try:
    # Main loop
    while True:
        # Read PIR sensor
        if GPIO.input(PIR_PIN):
            # Motion detected
            GPIO.output(LED_PIN, GPIO.HIGH)
            print("Motion Detected!")
        else:
            # No motion detected
            GPIO.output(LED_PIN, GPIO.LOW)

        # Read button state
        button_state = GPIO.input(BUTTON_PIN)
        if button_state == GPIO.LOW:
            # Button pressed
            GPIO.output(LED_PIN, GPIO.HIGH)
            print("Button Pressed!")
        else:
            # Button not pressed
            GPIO.output(LED_PIN, GPIO.LOW)

        # Add a small delay
        time.sleep(0.1)

except KeyboardInterrupt:
    print("Exiting program.")

finally:
    # Clean up GPIO settings
    GPIO.cleanup()
```

5. **Run Your Script:**
    - Save the Python script as `led_button_pir.py` and run it with:

```
python3 led_button_pir.py
```

Explanation:

- **Button Control:** When the button is pressed, the script turns the LED on. If the button is not pressed, the LED remains off.
- **PIR Sensor Control:** When the PIR sensor detects motion, it turns the LED on. If no motion is detected, the LED stays off.

This setup demonstrates basic GPIO interaction on a Raspberry Pi. You can modify the script and wiring for more complex behaviors or additional sensors and components.