Arduino LED Control with Push Buttons

---

1. Objective

The goal of this project is to use three push buttons to individually control three LEDs using an ESP32 microcontroller. When a button is pressed, its corresponding LED will turn ON. When the button is released, the LED will turn OFF.

2. Required Components

Component	Quantity
ESP32 Development Board	1
Push Buttons (4-pin type)	3
LEDs	3
220–330Ω Resistors (for LEDs)	3
10kΩ Resistors (pull-down)	3
Breadboard and Jumper Wires	1 set

3. Code – Full Listing

```
#include <Arduino.h>
#define led1 12
#define led2 14
#define led3 27
#define sw1 5
#define sw2 18
#define sw3 19


void setup() {
  // put your setup code here, to run once:
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
```

```
  pinMode(sw1,INPUT);
  pinMode(sw2,INPUT);
  pinMode(sw3,INPUT);
  //Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(digitalRead(sw1) == HIGH)
  {
    //Serial.println("BUTTON1 HIGH");
    digitalWrite(led1,HIGH);
  }
  if(digitalRead(sw1) == LOW)
  {
    //Serial.println("BUTTON1 LOW");
    digitalWrite(led1,LOW);
  }

  if(digitalRead(sw2) == HIGH)
  {
    digitalWrite(led2,HIGH);
  }
  if(digitalRead(sw2) == LOW)
  {
    digitalWrite(led2,LOW);
  }

  if(digitalRead(sw3) == HIGH)
  {
    digitalWrite(led3,HIGH);
  }
  if(digitalRead(sw3) == LOW)
  {
    digitalWrite(led3,LOW);
  }
}
```

4. Code Explanation – Line by Line

🔹 Header and Pin Definitions

```
#include <Arduino.h>
```

- Includes the core Arduino functionality such as `pinMode()`, `digitalRead()`, `digitalWrite()`.

```
#define led1 12
#define led2 14
#define led3 27
#define sw1 5
#define sw2 18
#define sw3 19
```

- Creates aliases for each GPIO pin for clarity and ease of updates.

🔹 Setup Function

```
void setup() {
  // put your setup code here, to run once:
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(sw1,INPUT);
  pinMode(sw2,INPUT);
  pinMode(sw3,INPUT);
  //Serial.begin(115200);
}
```

- Configures GPIO pins as inputs for buttons and outputs for LEDs.

🔹 Loop Function

```
void loop() {
  // put your main code here, to run repeatedly:
  if(digitalRead(sw1) == HIGH)
  {
    //Serial.println("BUTTON1 HIGH");
    digitalWrite(led1,HIGH);
  }
  if(digitalRead(sw1) == LOW)
  {
    //Serial.println("BUTTON1 LOW");
    digitalWrite(led1,LOW);
  }
```

```
 if(digitalRead(sw2) == HIGH)
 {
   digitalWrite(led2,HIGH);
 }
 if(digitalRead(sw2) == LOW)
 {
   digitalWrite(led2,LOW);
 }

 if(digitalRead(sw3) == HIGH)
 {
   digitalWrite(led3,HIGH);
 }
 if(digitalRead(sw3) == LOW)
 {
   digitalWrite(led3,LOW);
 }
```

- Constantly checks the state of each button.
- If the button is pressed (`HIGH`), the corresponding LED is turned ON.
- If not pressed (`LOW`), the LED is turned OFF.

5. Hardware & Wiring Explanation

🔹 LED Wiring

Each LED should be connected as follows:

- Anode (+) → GPIO pin through a 220–330Ω resistor
- Cathode (–) → GND

This limits the current and prevents damage to both the LED and ESP32.

🔹 Button Wiring with Pull-Down Resistors

You're using external pull-down resistors, which is correct. The wiring for each button is:

```
VCC (3.3V)
   |
 [ Button ]
   |______ GPIO (e.g., GPIO5)
   |
 [ 10kΩ resistor ]
   |
  GND
```

🔸 How It Works:

- When the button is not pressed, GPIO is pulled to GND through the 10kΩ resistor → reads LOW.
- When the button is pressed, it connects GPIO directly to VCC (3.3V) → reads HIGH.

This avoids “floating” input states.

6. Electrical Considerations

- Debouncing: Physical buttons may bounce and cause rapid toggling. In simple projects like this, it's usually acceptable, but for production systems, consider software debouncing using delay or filters.
- Power: Ensure the ESP32 is powered with at least 500mA to handle LED current and GPIO logic.

7. Behavior Summary

Button	LED	Button Pressed (HIGH)	Button Released (LOW)
sw1	led1	LED1 ON	LED1 OFF
sw2	led2	LED2 ON	LED2 OFF
sw3	led3	LED3 ON	LED3 OFF

8. Tips & Optional Enhancements

Add `Serial.print()` statements for debugging:

```
Serial.println("Button 1 Pressed");
```

- Add LED blink or toggle functionality instead of continuous ON/OFF.
- Replace external pull-downs with internal pull-ups (`INPUT_PULLUP`) and change logic if needed.

9. Conclusion

This project demonstrates a fundamental technique in embedded systems: digital input/output control. You’ve learned how to:

- Read button states via digital inputs
- Control LEDs with outputs
- Wire external components safely using pull-down resistors
- Avoid floating pins and unstable behavior