Complete beginner IoT project: sensor + actuator + Wi-Fi + remote control:-

Here's a **complete ESP32 IoT project code** combining:

- Pushbutton input
- LED output
- Internal pull-down resistor
- Wi-Fi connection
- Web server to monitor and control the LED remotely

This code is **ready to upload** to your ESP32.

```
#include <WiFi.h>
#include <WebServer.h>

// ----- Wi-Fi Credentials -----
const char* ssid = "Your_WiFi_Name";
const char* password = "Your_WiFi_Password";

// ----- Pin Definitions -----
#define LED_PIN 2
#define BUTTON_PIN 4

// ----- Web Server -----
WebServer server(80);

// ----- Function to handle root page -----
void handleRoot() {
  String html = "<h1>ESP32 LED Control</h1>";

  // Display current LED status
  html += "<p>LED is ";
  html += digitalRead(LED_PIN) ? "ON" : "OFF";
  html += "</p>";

  // Add buttons to control LED remotely
  html += "<p><a href=\"/on\">Turn ON</a></p>";
  html += "<p><a href=\"/off\">Turn OFF</a></p>";

  server.send(200, "text/html", html);
}

// ----- Functions to handle LED via web -----
void handleLEDOn() {
  digitalWrite(LED_PIN, HIGH);
  server.sendHeader("Location","/"); // redirect to home
  server.send(303);
}

void handleLEDOff() {
  digitalWrite(LED_PIN, LOW);
```

```
  server.sendHeader("Location","/"); // redirect to home
  server.send(303);
}

// ----- Setup -----
void setup() {
  Serial.begin(115200);

  // Initialize pins
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLDOWN); // use internal pull-down

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected! ESP32 IP: ");
  Serial.println(WiFi.localIP());

  // Setup web server routes
  server.on("/", handleRoot);
  server.on("/on", handleLEDOn);
  server.on("/off", handleLEDOff);

  server.begin();
  Serial.println("Web server started");
}

// ----- Main Loop -----
void loop() {
  // Handle web client requests
  server.handleClient();

  // Read button and control LED
  int buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == HIGH) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
}
```

□ How to Run

1. Replace `"Your_WiFi_Name"` and `"Your_WiFi_Password"` with your Wi-Fi credentials.
2. Upload the code to the **ESP32**.
3. Open **Serial Monitor** at 115200 baud → ESP32 will print its **IP address**.
4. Open a browser on the same Wi-Fi → type the IP address → you'll see the LED control page.

5. You can **press the physical button** or **use the web buttons** to turn the LED ON/OFF.

Here's a clear **breadboard diagram for your ESP32 IoT project** with LED and pushbutton using the internal pull-down resistor.

□ Components

- ESP32 Dev Board
- LED + 220Ω resistor
- Pushbutton
- Jumper wires
- USB cable for power

□ Wiring

1. **LED Wiring**
   - **Anode (+ long leg)** → GPIO 2 through **220Ω resistor**
   - **Cathode (- short leg)** → GND
2. **Pushbutton Wiring**
   - **One side of button** → 3.3V
   - **Other side of button** → GPIO 4

□ No external resistor needed because we use `INPUT_PULLDOWN` in code.

3. **Power**
   - ESP32 powered via USB cable to computer

□ Visual Text Diagram

```
ESP32 Board:

   3.3V  --------+
                 |
                 |
              [Button]
                 |
   GPIO 4 -------+
                 |
               (internal pull-down)
```

```
GPIO 2 -------[220Ω]----> |LED| ---- GND
```

□ How It Works

1. **Physical Button:**
    - Press → GPIO 4 reads HIGH → LED turns ON
    - Release → GPIO 4 reads LOW → LED turns OFF
2. **Web Control:**
    - Open ESP32 IP in browser → web page shows LED status
    - Use **Turn ON / Turn OFF buttons** on web page → control LED remotely