LED Blink – Example Sketch

Let's examine a sample sketch to highlight the key elements for building a simple circuit that blinks an LED using the ESP32.

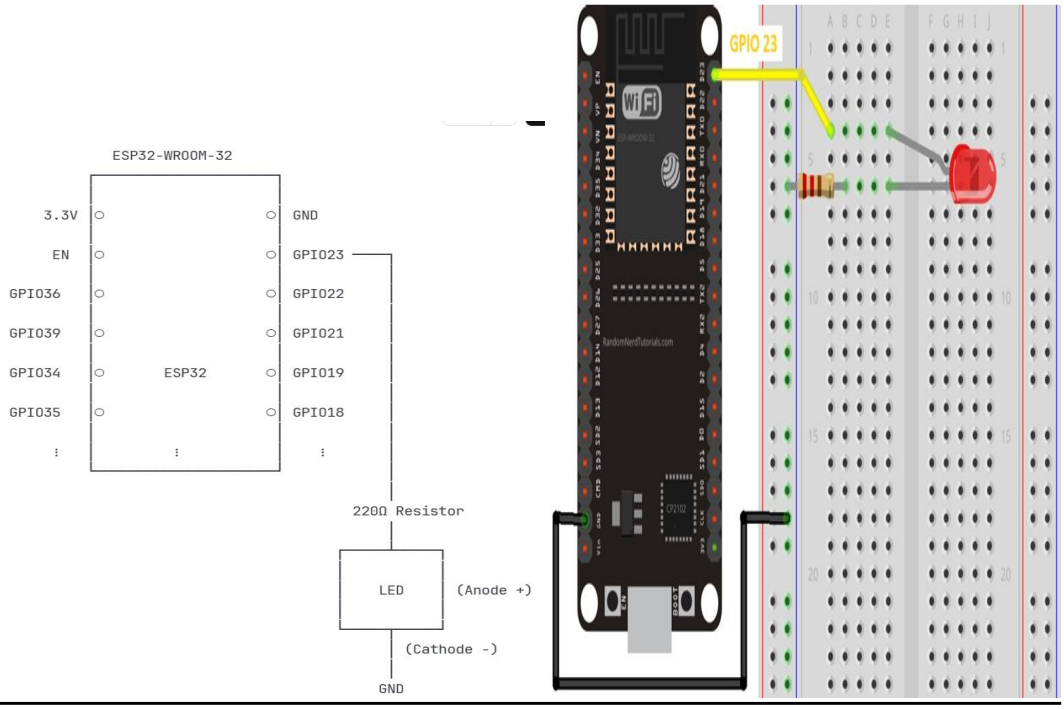
ESP32 LED Circuit - Complete Hardware Analysis

- **Required Hardware Components**

Essential Components:

1. **ESP32 Development Board** (any variant: DevKitC, WROOM, etc.)
2. **LED** - Any standard 5mm LED (Red, Green, Blue, Yellow)
3. **Current-Limiting Resistor** - 220Ω to 1kΩ (1/4W or 1/8W)
4. **Breadboard** - Half-size or full-size for prototyping
5. **Jumper Wires** - Male-to-male for connections
6. **USB Cable** - USB-A to Micro-USB or USB-C (depending on board)

Complete Circuit Diagram:



Electrical Analysis & Calculations Voltage and Current Analysis:

When LED is ON (GPIO23 = HIGH):

Circuit Parameters:

- ESP32 GPIO23 output: 3.3V
- LED forward voltage (Vf): 2.0V (typical red LED)
- Current-limiting resistor: 220Ω
- GND reference: 0V

Voltage Distribution:

- Total supply voltage: 3.3V
- Voltage drop across LED: 2.0V
- Voltage drop across resistor: 3.3V - 2.0V = 1.3V

Current Calculation (Ohm's Law):

I = V / R = 1.3V / 220Ω = 5.91mA

Power Dissipation:

- LED power: P = Vf × I = 2.0V × 5.91mA = 11.82mW
- Resistor power: P = Vr × I = 1.3V × 5.91mA = 7.68mW
- Total power: 19.5Mw

When LED is OFF (GPIO23 = LOW):

Circuit Parameters:

- ESP32 GPIO23 output: 0V
- Circuit voltage: 0V - 0V = 0V
- Circuit current: 0mA
- Power consumption: 0mW
- LED state: OFF (no light emission)

LED Forward Voltage by Color:

LED Color	Forward Voltage (Vf)	Recommended Resistor	Current
Red	1.8V - 2.2V	220Ω - 330Ω	3-6mA
Green	2.0V - 2.4V	180Ω - 270Ω	4-7mA
Blue	2.8V - 3.4V	47Ω - 100Ω	3-10mA
Yellow	1.9V - 2.3V	200Ω - 300Ω	3-6mA
White	2.8V - 3.6V	33Ω - 82Ω	5-15mA

Practical cautions & common mistakes :

- **No resistor**: Don't connect LED directly to pin without resistor — you risk damaging the pin or LED.
- **Wrong polarity**: LED anode (long leg) should go to the pin via resistor; cathode (short leg) to GND. If reversed it won't light.
- **Blocking delay()**: delay() blocks the CPU. While the sketch waits, the CPU won't be able to do other tasks (read sensors, respond to buttons, etc.). delay() is fine for simple examples.
- **Pin choice**: Some ESP32 pins are input-only (GPIO34–39) or have special boot functions. GPIO 23 is a general-purpose pin and is fine for an LED.
- **Voltage levels**: ESP32 IO = 3.3 V. Don't connect to 5 V logic without level shifting.
- **Current limits**: Avoid drawing large currents from a GPIO pin. Absolute maximum ratings are high but risky — prefer ≤ 10–20 mA per pin for safe operation and long life.

- **A — Non-blocking blink (use millis()): lets the CPU do other tasks while blinking :** Why this is better: millis() gives a timestamp; comparing timestamps avoids pausing the CPU. Great for multi-tasking.

```
// Non-blocking blink using millis()
const int ledPin = 23;
unsigned long previousMillis = 0;        // last time LED toggled
const unsigned long interval = 1000UL;   // interval at which to blink (ms)
bool ledState = false;                   // current state of LED
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  unsigned long currentMillis = millis();   // current time since board started (ms)

  // Check if enough time has passed since last toggle
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;        // remember the time of this toggle
    ledState = !ledState;                  // flip the state
    digitalWrite(ledPin, ledState ? HIGH : LOW);
  }
  // Here you can do other things (read sensors, handle buttons, send data)
  // without being blocked by delay().
}
```

Simple PWM brightness (ESP32 LEDC)

The ESP32 has hardware PWM (LEDC). This lets you vary LED brightness smoothly.

```
// Simple PWM (fade) on ESP32 using LEDC
const int ledPin = 23;
const int pwmChannel = 0;   // 0..15 channels possible on ESP32
```

```
const int freq = 5000;      // 5 kHz PWM frequency
const int resolution = 8;   // 8-bit resolution -> duty 0..255
void setup() {
  // Configure PWM channel and attach it to the pin
  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(ledPin, pwmChannel);
}
void loop() {
  // Fade up
  for (int duty = 0; duty <= 255; duty++) {
    ledcWrite(pwmChannel, duty);   // set duty cycle
    delay(5);
  }
  // Fade down
  for (int duty = 255; duty >= 0; duty--) {
    ledcWrite(pwmChannel, duty);
    delay(5);
  }
}
```