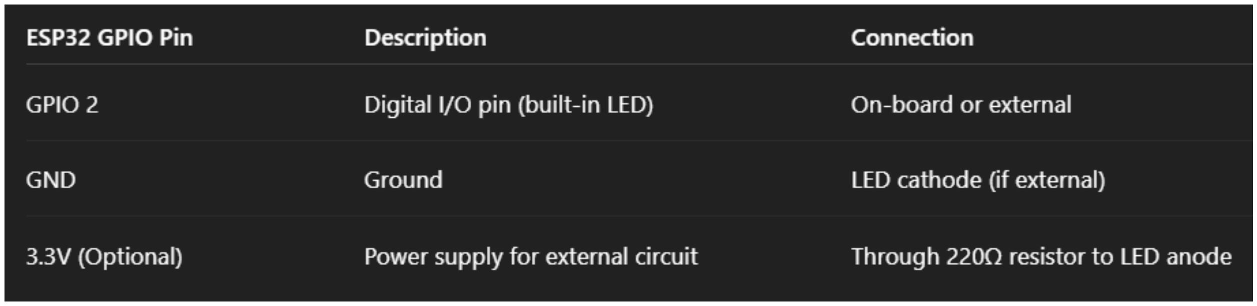
# ESP32 LED Blinking Code – Detailed Explanation and Documentation

---

## Hardware Requirements

- ESP32 Development Board (e.g., DevKit v1)
- USB Cable for programming and power
- On-board LED (typically connected to **GPIO 2** on most ESP32 dev boards)
- Optional: External LED with 220Ω resistor for testing GPIO control

## Pin Connection



| ESP32 GPIO Pin | Description | Connection |
|---|---|---|
| GPIO 2 | Digital I/O pin (built-in LED) | On-board or external |
| GND | Ground | LED cathode (if external) |
| 3.3V (Optional) | Power supply for external circuit | Through 220Ω resistor to LED anode |

If using an external LED:

ESP32 GPIO2 --> 220Ω resistor --> LED Anode
LED Cathode --> GND

## Code: LED Blinking with FreeRTOS

**Complete code**

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

#define LED_PIN GPIO_NUM_2  // Built-in LED on many ESP32 dev boards
void app_main(void)
{
    gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);  // Set as output

    while (1) {
        gpio_set_level(LED_PIN, 1);  // LED ON
        vTaskDelay(pdMS_TO_TICKS(500));
        gpio_set_level(LED_PIN, 0);  // LED OFF
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

**Line by line explanation**

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
```

**Explanation:**

- #include <stdio.h>:
  Includes the standard input/output header for general-purpose functions like printf.
- #include "freertos/FreeRTOS.h":
  Includes the core definitions of FreeRTOS used in ESP-IDF. Provides access to RTOS APIs, constants, and macros.
- #include "freertos/task.h":
  Contains APIs related to task creation, delays, and management (e.g., vTaskDelay).
- #include "driver/gpio.h":
  This header provides functions to control GPIO pins — essential for setting pin mode and writing digital levels.

```
#define LED_PIN GPIO_NUM_2  // Built-in LED on many ESP32 dev boards
```

**Explanation:**

Defines a macro LED_PIN and assigns it GPIO_NUM_2.
GPIO_NUM_2 is a constant defined in the ESP-IDF for GPIO pin 2, typically connected to the onboard LED.

```
void app_main(void)
```

**Explanation:**

This is the entry point of the application in ESP-IDF.
Unlike standard C main(), in ESP-IDF app_main() is automatically called after the system has initialized (FreeRTOS scheduler is running).

```
  gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);  // Set as output
```

**Explanation:**

Configures GPIO2 as an **output pin**.

- gpio_set_direction() is used to set the direction of the GPIO (input/output).
- GPIO_MODE_OUTPUT tells the microcontroller that the pin will **send** signals rather than read them.

```
  while (1) {
```

**Explanation:**

Starts an infinite loop that runs continuously throughout the operation of the device.
This is necessary in embedded systems to keep tasks running.

```
    gpio_set_level(LED_PIN, 1);  // LED ON
```

**Explanation:**

Sets GPIO2 **high** (logic 1 = 3.3V), which turns the LED **ON**.

- For most ESP32 boards with active-high LEDs, writing 1 enables the LED.

```
    vTaskDelay(pdMS_TO_TICKS(500));
```

**Explanation:**

- vTaskDelay() is a FreeRTOS function that delays a task for a number of **ticks**.
- pdMS_TO_TICKS(500) converts **500 milliseconds** into **RTOS ticks**.

- This creates a half-second delay while the LED remains ON.

```
    gpio_set_level(LED_PIN, 0);  // LED OFF
```

**Explanation:**

Sets GPIO2 **low** (logic 0 = 0V), which turns the LED **OFF**.

```
    vTaskDelay(pdMS_TO_TICKS(500));
  }
```

**Explanation:**

Delays the task again for 500 milliseconds with the LED OFF.

## Complete Flow of Operation

1. The ESP32 boots and runs app_main().
2. GPIO 2 is configured as an output pin.
3. The system enters an infinite loop.
4. In each loop:
    - GPIO2 is set HIGH → LED turns ON.
    - Waits for 500 ms.
    - GPIO2 is set LOW → LED turns OFF.
    - Waits for 500 ms.
5. This loop continues indefinitely, resulting in a **1 Hz blink frequency** (LED toggles every 0.5s).

## Timing Diagram

```
Time (ms):  0    500   1000  1500  ...
LED:       ON   OFF   ON    OFF   ...
```

Each cycle (ON + OFF) lasts 1000 milliseconds (1 second).

## Additional Notes

- **GPIO Safety:** Avoid using GPIOs reserved for flash, boot, or internal functions. GPIO2 is safe and commonly used.
- **Power Considerations:** The onboard LED draws minimal current, but external LEDs should be current-limited with a resistor (~220Ω).

- **RTOS Scheduling:** vTaskDelay yields CPU control to allow other tasks to run, essential in multitasking systems.

**Conclusion**

This basic ESP32 application demonstrates how to blink an LED using FreeRTOS in the ESP-IDF environment. Understanding the configuration and control of GPIOs is essential for embedded development, and this serves as a foundation for more advanced hardware control.