# ESP-IDF FreeRTOS Example for ESP32

## Step 1: Setup ESP-IDF Environment

Make sure you have ESP-IDF installed and set up. You can follow the official ESP-IDF setup guide for your operating system: [ESP-IDF Setup Guide](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html).

## Step 2: Create a New Project

Create a new project using the ESP-IDF tools:

--------BASH----------

$ idf.py create-project freeRTOS\_ESP  
$ cd freeRTOS\_ESP

Replace `freeROTS\_ESP` with your Project Directory.

## Step 3: Edit CMakeLists.txt

* Modify the `CMakeLists.txt` file in your project directory (freeRTOS\_ESP) to include FreeRTOS and set up your source files:

$ vim CMakeList.txt

cmake\_minimum\_required(VERSION 3.5)  
include($ENV{IDF\_PATH}/tools/cmake/project.cmake)  
project(freertos\_blink)  
  
# Add esp-idf FreeRTOS component  
set(EXTRA\_COMPONENT\_DIRS $ENV{IDF\_PATH}/components/freertos)

## Step 4: Create the Application Code

Create a `main.c` file in your project **main** directory with the following content:

#include <stdio.h>

#include "freertos/FreeRTOS.h"

#include "freertos/FreeRTOSConfig\_arch.h"

#include "freertos/task.h"

#include "driver/gpio.h"

// Define the LED pin

#define LED\_PIN GPIO\_NUM\_2

// Task to blink the LED

void blink\_led\_task(void \*pvParameter) {

while (1) {

// Turn the LED on

gpio\_set\_level(LED\_PIN, 1);

vTaskDelay(1000 / portTICK\_PERIOD\_MS);

// Turn the LED off

gpio\_set\_level(LED\_PIN, 0);

vTaskDelay(1000 / portTICK\_PERIOD\_MS);

}

}

// Task to print a message

void print\_task(void \*pvParameter) {

while (1) {

printf("Hello from FreeRTOS!\n");

vTaskDelay(2000 / portTICK\_PERIOD\_MS);

}

}

void app\_main() {

// Init LED BUILTIN for blink

// gpio\_reset\_pin(LED\_PIN);

gpio\_set\_direction(LED\_PIN, GPIO\_MODE\_OUTPUT);

// Create the LED blink task

xTaskCreate(&blink\_led\_task, "blink\_led\_task", configMINIMAL\_STACK\_SIZE, NULL, 5, NULL);

// Create the print task

xTaskCreate(&print\_task, "print\_task", configMINIMAL\_STACK\_SIZE, NULL, 5, NULL);

}

## Step 5: Build and Flash the Project

After Creating your program come back to your parent directory (freeRTOS\_ESP) folder.

Build and flash the project to your ESP32 using `idf.py`:

idf.py build  
idf.py -p <port> flash

Replace `<port>` with your ESP32's serial port. Eg. /dev/ttyUSB0

## Step 6: Monitor the Output

Monitor the output to see the blinking LED:

idf.py -p <port> monitor

Replace `<port>` with your ESP32's serial port. Eg. /dev/ttyUSB0