

Introduction to RTOS - Solution to Part 2 (FreeRTOS)

[By ShawnHymel](#)

A real-time operating system (RTOS) allows programmers and firmware developers to create multi-threaded programs on microcontrollers that meet real-time deadlines. Note that meeting a deadline does not mean “fast.” Rather, it means that task execution time is deterministic, so developers can calculate ahead of time if their tasks can meet deadlines.

In this series, we will go over various RTOS concepts. At the end of each video, we will provide a challenge that we encourage you to try. Possible solutions to these challenges will be covered in these project pages.

Required Hardware

To complete the series, we recommend an ESP32 development board. Any development board should work, so long as it's supported in the Arduino IDE. [See here](#) for a list of supported ESP32 boards.

This solution uses the [Adafruit Feather HUZZAH32](#).

Note that you can also use almost any development board as long as it is capable of running FreeRTOS. There are some differences between the modified FreeRTOS in ESP-IDF and vanilla FreeRTOS. I will attempt to note those differences in the code comments. If your board does not support FreeRTOS out of the box, please refer to the [FreeRTOS porting guide](#).

Video

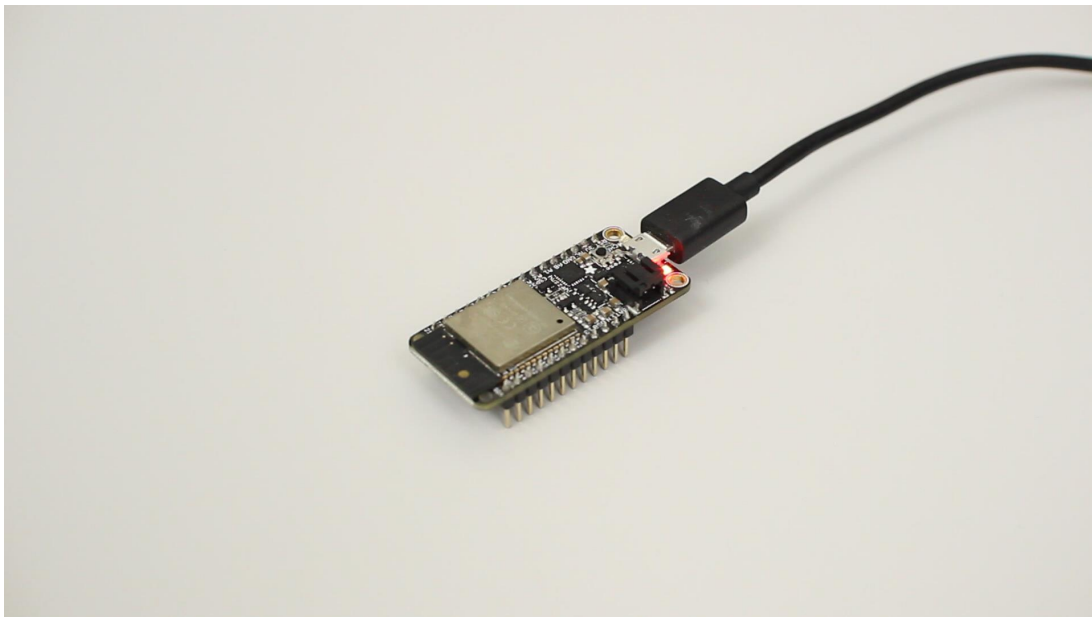
If you have not done so, please watch the following video, which provides the steps necessary to get started with FreeRTOS and demonstrates a working version of the challenge:

Introduction to RTOS Part 2 - Getting Started with FreeRTOS | Digi-Key Elec...



Challenge

Using FreeRTOS, create two separate tasks that blink the same LED at two different rates. That means controlling 1 LED with two different delay times.



Solution

Spoilers below! I highly encourage you to try the challenge on your own before comparing your answer to mine. Note that my solution may not be the only way to solve the challenge.

Copy Code

```
/**
 * Solution to 02 - Blinky Challenge
 *
 * Toggles LED at different rates using separate tasks.
 *
 * Date: December 3, 2020
```

```

* Author: Shawn Hymel
* License: 0BSD
*/

// Use only core 1 for demo purposes
#if CONFIG_FREERTOS_UNICORE
static const BaseType_t app_cpu = 0;
#else
static const BaseType_t app_cpu = 1;
#endif

// LED rates
static const int rate_1 = 500; // ms
static const int rate_2 = 323; // ms

// Pins
static const int led_pin = LED_BUILTIN;

// Our task: blink an LED at one rate
void toggleLED_1(void *parameter) {
    while(1) {
        digitalWrite(led_pin, HIGH);
        vTaskDelay(rate_1 / portTICK_PERIOD_MS);
        digitalWrite(led_pin, LOW);
        vTaskDelay(rate_1 / portTICK_PERIOD_MS);
    }
}

// Our task: blink an LED at another rate
void toggleLED_2(void *parameter) {
    while(1) {
        digitalWrite(led_pin, HIGH);
        vTaskDelay(rate_2 / portTICK_PERIOD_MS);
        digitalWrite(led_pin, LOW);
        vTaskDelay(rate_2 / portTICK_PERIOD_MS);
    }
}

void setup() {
    // Configure pin
    pinMode(led_pin, OUTPUT);

    // Task to run forever
    xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS
        toggleLED_1, // Function to be called
        "Toggle 1", // Name of task
        1024, // Stack size (bytes in ESP32, words in FreeRTOS)
        NULL, // Parameter to pass to function
        1, // Task priority (0 to configMAX_PRIORITIES - 1)
        NULL, // Task handle
        app_cpu); // Run on one core for demo purposes (ESP32 only)

    // Task to run forever
    xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS
        toggleLED_2, // Function to be called
        "Toggle 2", // Name of task
        1024, // Stack size (bytes in ESP32, words in FreeRTOS)
        NULL, // Parameter to pass to function
        1, // Task priority (0 to configMAX_PRIORITIES - 1)
        NULL, // Task handle
        app_cpu); // Run on one core for demo purposes (ESP32 only)

    // If this was vanilla FreeRTOS, you'd want to call vTaskStartScheduler() in
    // main after setting up your tasks.
}

void loop() {
    // Do nothing
    // setup() and loop() run in their own task with priority 1 in core 1
    // on ESP32
}

```

Explanation

The code is very similar to the demonstration we saw in the video. The difference is that instead of 1 task, we create 2. At the top of the code, we define our two separate delay rates:

Copy Code

```
// LED rates
static const int rate_1 = 500; // ms
static const int rate_2 = 323; // ms
```

I recommend playing around with these numbers to achieve an effect that makes you happy.

We define our LED pin number and then define our tasks. These should be separate functions, each with their own while(true) loop. Each loop has a simple turn LED on, delay, turn LED off, delay set of instructions.

Note that the pin that's being toggled is the same in both tasks. However, the delay used is different. This allows the LED to toggle at different rates, creating odd patterns.

In setup, we create our two tasks. Each call to xTaskCreatePinnedToCore() provides slightly different arguments. The first call passes in the toggleLED_1 function whereas the second call passes in the toggleLED_2 function.

If you are using vanilla FreeRTOS on a single-core system, you would want to use xTaskCreate() (the *PinnedToCore functions were added in ESP-IDF to allow you to run a task in a specific processor core). Here is how you would start 2 tasks in vanilla FreeRTOS (note that the minimum stack size might be different for your system):

Copy Code

```
// Task to run forever
xTaskCreate(toggleLED_1, "Toggle 1", 1024, NULL, 1, NULL);
xTaskCreate(toggleLED_2, "Toggle 2", 1024, NULL, 1, NULL);
```

Also note that if you're using vanilla FreeRTOS, you would want to call the following function somewhere in your setup code:

Copy Code

```
vTaskStartScheduler()
```

This will actually start the scheduler in the background. Your tasks won't run without it!

We leave loop() empty. Note that both setup() and loop() functions run in their own task for the Arduino version of ESP-IDF. Because all three tasks (Toggle 1, Toggle 2, and "Setup and Loop") are priority 1, the processor will split its time equally among them. We will get into prioritization in a future video.

Feel free to try creating other tasks that run concurrently that do things like read from sensors or print to the Serial Monitor. What happens if you change the priority on the toggle LED tasks to be 0? Why doesn't the LED blink? How might you fix that?

Recommended Reading

Example code from the video and the solution can also be found in the following repository: <https://github.com/ShawnHymel/introduction-to-rtos>.


If you'd like to dig deeper into FreeRTOS, ESP32, or ESP-IDF, I recommend checking out these great articles:

- FreeRTOS Book and Reference Guide: https://www.freertos.org/Documentation/RTOS_book.html
- Differences between FreeRTOS and ESP-IDF: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/freertos-smp.html>
- Multitasking on ESP32 with Arduino and FreeRTOS: <https://savjee.be/2020/01/multitasking-esp32-arduino-freertos/>
- [What is a Real-Time Operating System Part 1\(RTOS\)?](#)
- [Introduction to RTOS - Solution to Part 3 \(Task Scheduling\)](#)

- [Introduction to RTOS - Solution to Part 4 \(Memory Management\)](#)
- [Introduction to RTOS - Solution to Part 5 \(FreeRTOS Queue Example\)](#)
- [Introduction to RTOS - Solution to Part 6 \(FreeRTOS Mutex Example\)](#)
- [Introduction to RTOS - Solution to Part 7 \(FreeRTOS Semaphore Example\)](#)
- [Introduction to RTOS - Solution to Part 8 \(Software Timers\)](#)
- [Introduction to RTOS - Solution to Part 9 \(Hardware Interrupts\)](#)
- [Introduction to RTOS - Solution to Part 10 \(Deadlock and Starvation\)](#)
- [Introduction to RTOS - Solution to Part 11 \(Priority Inversion\)](#)
- [Introduction to RTOS - Solution to Part 12 \(Multicore Systems\)](#)

Key Parts and Components

1 Items



Mfr Part # 3405

HUZZAH32 ESP32 FEATHER LOOSE HDR


Adafruit Industries LLC

\$19.95

Details



[Add all Digi-Key Parts to Cart](#)

 [TechForum](#)

Have questions or comments? Continue the conversation on [TechForum](#), DigiKey's online community and technical resource.

Visit [TechForum](#)

[Arduino](#) | [3D Printing](#) | [Raspberry Pi](#)

Project Details

Development

C

Tags

Arduino

RTOS

License

Attribution

Get Involved

Like

Save



1-800-344-4539


218-681-6674



sales@digikkey.com



218-681-3380

 United States | Copyright © 1995-2023, DigiKey. | All Rights Reserved.

Local Support: 701 Brooks Avenue South, Thief River Falls, MN 56701 USA

Do Not Sell / Do Not Share My Personal Information