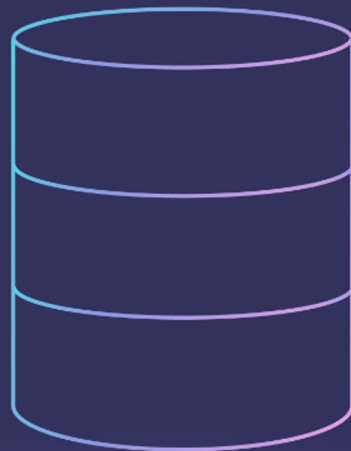


# Making things smarter



---

## GLOBAL AGENDA

---

### Embedded Programming

Lectures (1 & 2):

01/10/2021

Lab 1:

01/10/2021

### IoT communication protocols

Lecture 1:

06/10/2021

Lab 1:

15/10/2021

### Introduction to Embedded Machine Learning

Lecture 3:

12/11/2021

Lab 2:

12/11/2021

Embedded Programming 2/3

# Introduction to Real Time Operating Systems (RTOS)

## 1. Introduction

- General Purpose Operating System vs. Real-Time Operating System
- Loop vs. Multi-threaded Application
- RTOS requirements

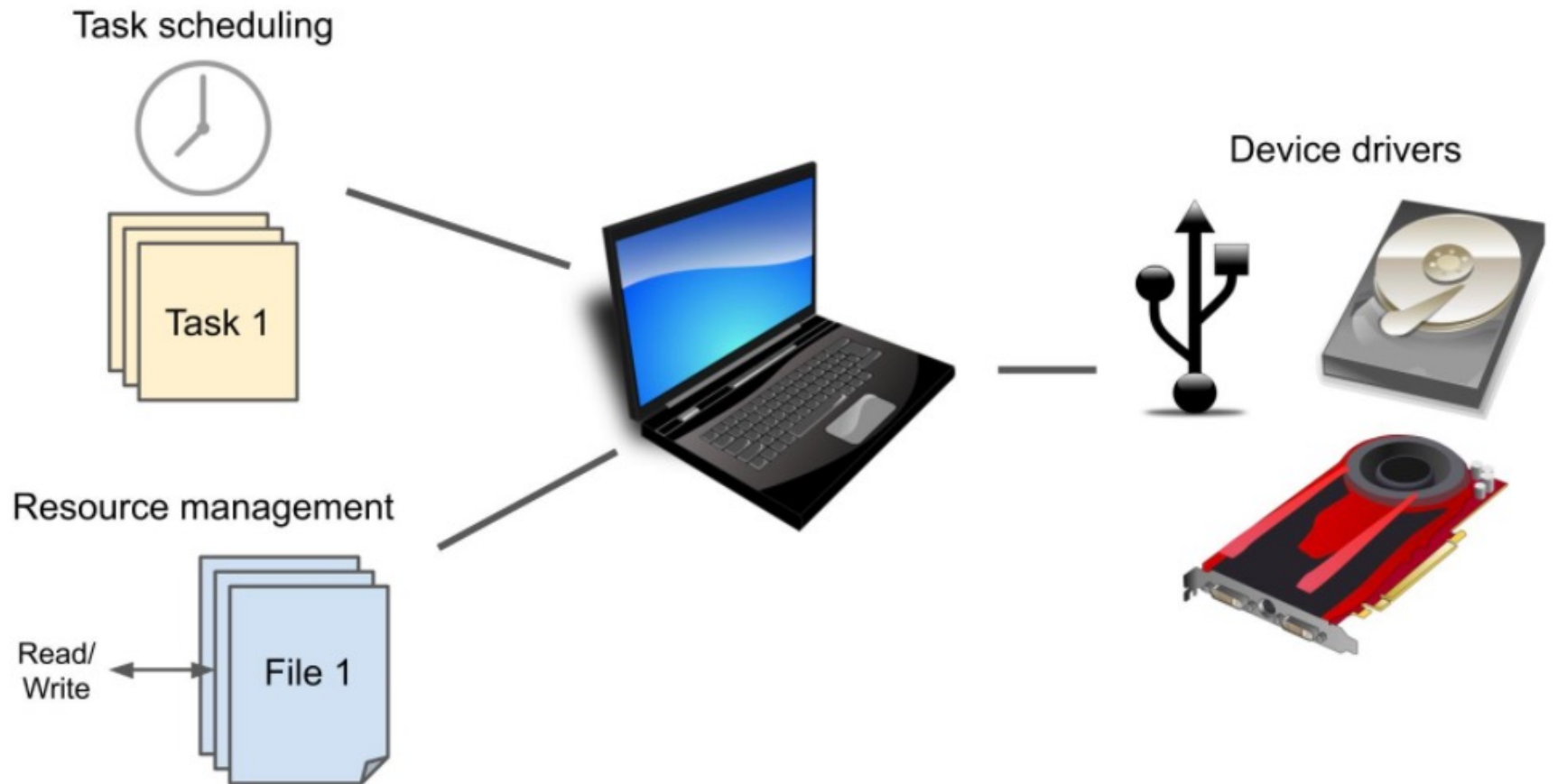
## 2. Example with FreeRTOS on an ESP32

## 3. Task Scheduling

## 1. Introduction

## Introduction

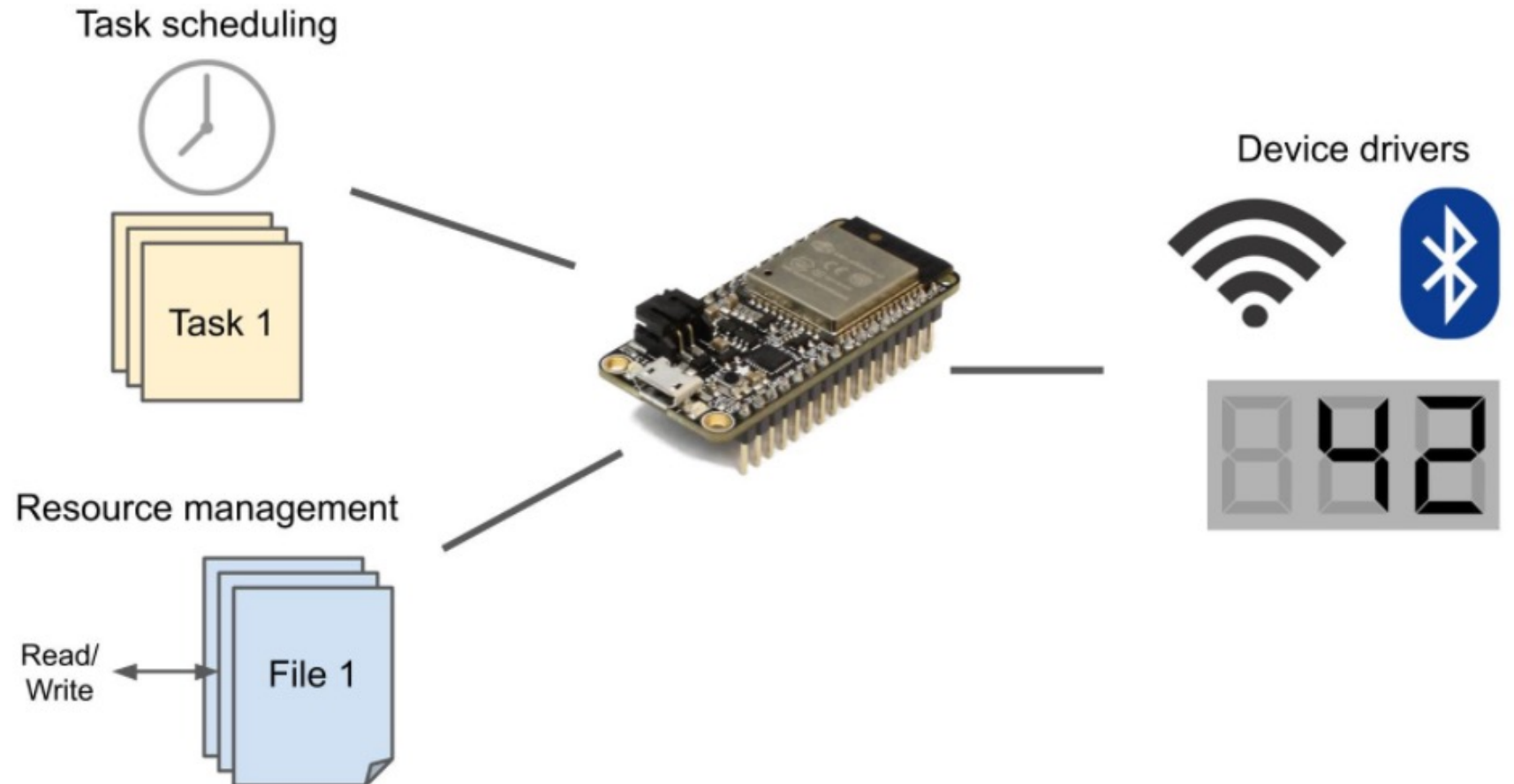
### General Purpose Operating System (GPOS)



## 1. Introduction

## Introduction

### Real-Time Operating System (RTOS)



# A Beginners Guide to RTOS

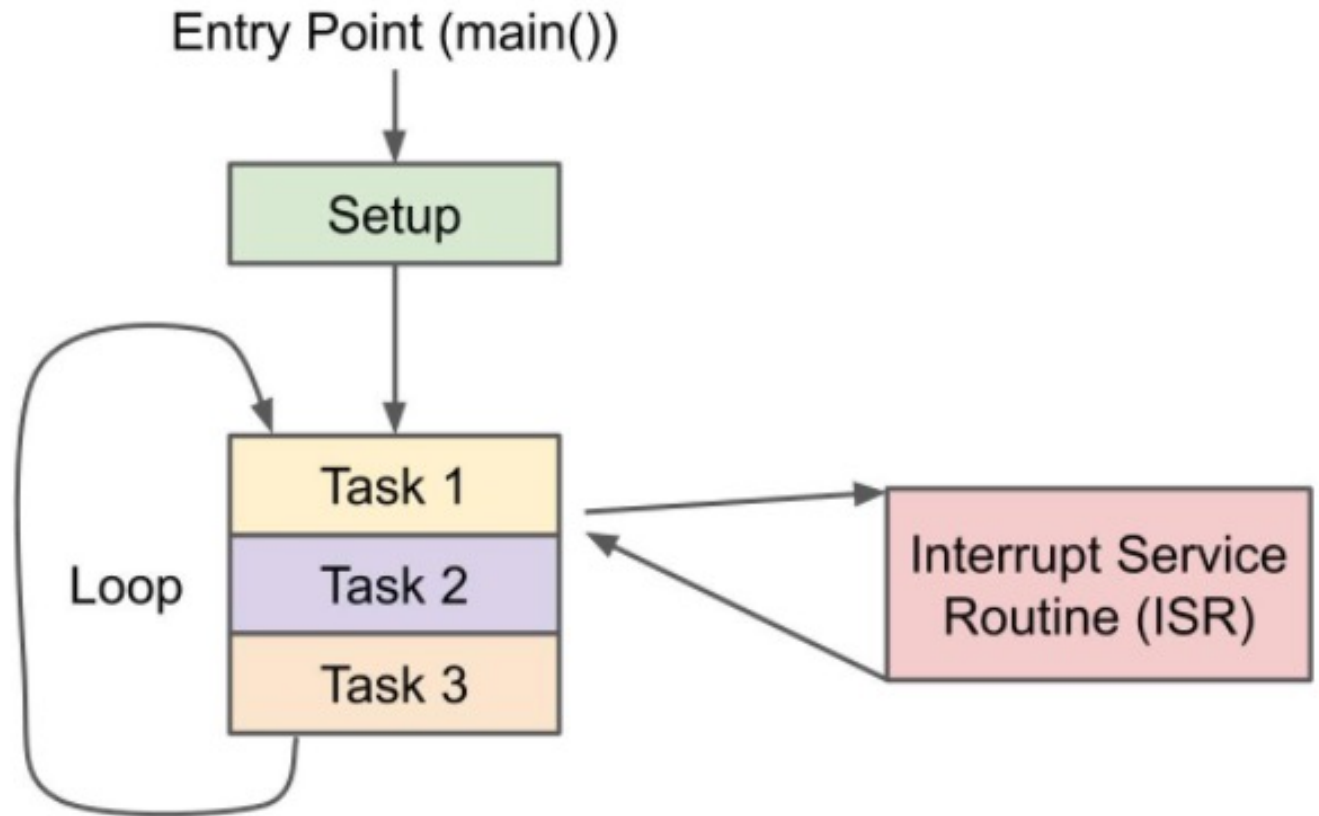
(Real Time Operating System)

# Embedded Programming 2/3

## 1. Introduction

## Loop vs. Multi-threaded Application

### Super Loop

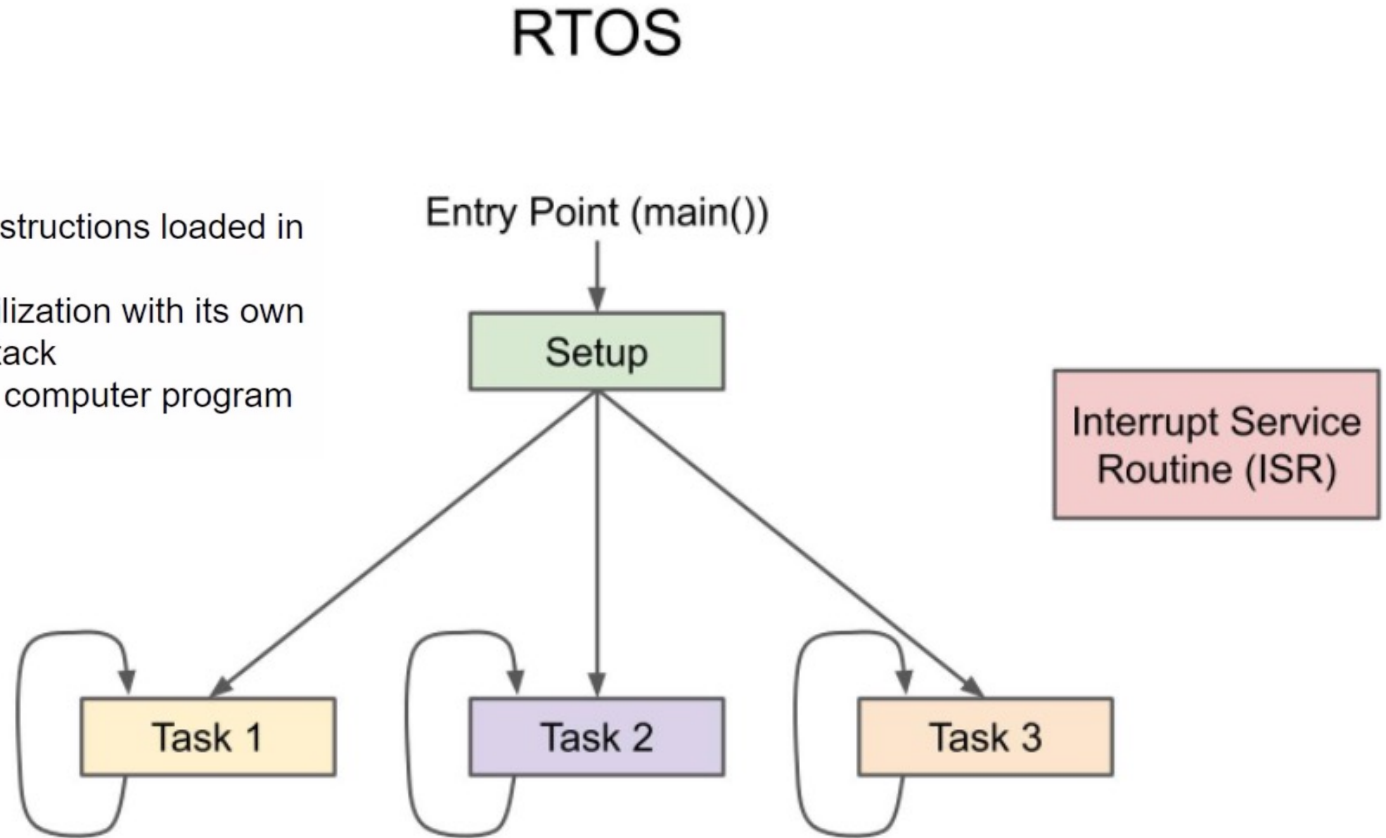


# Embedded Programming 2/3

## 1. Introduction

- **Task:** set of program instructions loaded in memory
- **Thread:** unit of CPU utilization with its own program counter and stack
- **Process:** instance of a computer program

## Loop vs. Multi-threaded Application





# Embedded Programming 2/3

## 1. Introduction

## RTOS Requirements



ATmega 328p

- 16 MHz
- 32 kB flash
- 2 kB RAM



STM32L476RG

- 80 MHz
- 1 MB flash
- 128 kB RAM



ESP-WROOM-32

- 240 MHz (dual core)
- 4 MB flash
- 520 kB RAM

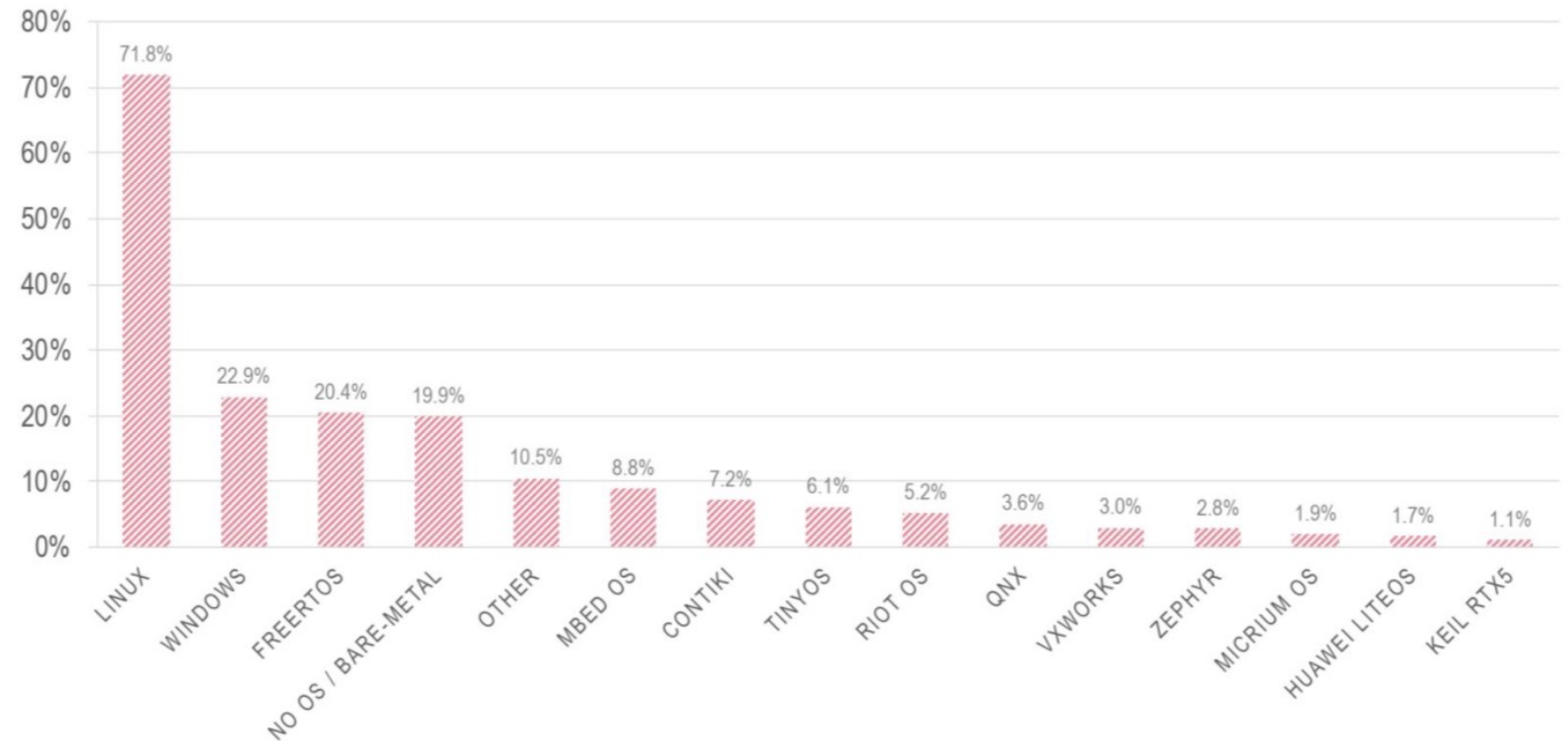
Super Loop



RTOS

### IoT OPERATING SYSTEMS

*Which operating system(s) do you use for your IoT devices?*



Copyright (c) 2018, Eclipse Foundation, Inc. | Made available under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/) (CC BY 4.0).

# Embedded Programming 2/3

1. Introduction
2. Example with FreeRTOS

## Example with FreeRTOS on ESP32

```
FreeRTOS_Example | Arduino 1.8.15

FreeRTOS_Example
1 #define LED_BUILTIN 4
2
3 // Use only core 1 for demo purposes
4 static const BaseType_t app_cpu = 1;
5
6 // LED rates
7 static const int rate_1 = 500; // ms
8 static const int rate_2 = 2000; // ms
9
10 // Pins
11 static const int led_pin = LED_BUILTIN;
12
13 // Our task: blink an LED at one rate
14 void toggleLED(void *parameter) {
15     while(1) {
16         digitalWrite(led_pin, HIGH);
17         vTaskDelay(rate_1 / portTICK_PERIOD_MS);
18         digitalWrite(led_pin, LOW);
19         vTaskDelay(rate_1 / portTICK_PERIOD_MS);
20     }
21 }
22
23 // Our task: blink an LED at another rate
24 void say_hello(void *parameter) {
25     while(1) {
26         vTaskDelay(rate_2 / portTICK_PERIOD_MS);
27         Serial.println("Hello HEI");
28     }
29 }
30
31 void setup() {
32     // Configure pin
33     pinMode(led_pin, OUTPUT);
34     Serial.begin(115200);
35
36     // Task to run forever
37     xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS
38         toggleLED,          // Function to be called
39         "Blink",            // Name of task
40         2048,               // Stack size (bytes in ESP32, words in FreeRTOS)
41         NULL,               // Parameter to pass to function
42         0,                  // Task priority (0 to configMAX_PRIORITIES - 1)
43         NULL,               // Task handle
44         app_cpu);           // Run on one core for demo purposes (ESP32 only)
45
46     // Task to run forever
47     xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS
48         say_hello,          // Function to be called
49         "Hello",            // Name of task
50         2048,               // Stack size (bytes in ESP32, words in FreeRTOS)
51         NULL,               // Parameter to pass to function
52         0,                  // Task priority (0 to configMAX_PRIORITIES - 1)
53         NULL,               // Task handle
54         app_cpu);           // Run on one core for demo purposes (ESP32 only)
55
56     // If this was vanilla FreeRTOS, you'd want to call vTaskStartScheduler() in
57     // main after setting up your tasks.
58 }
59
60 void loop() {
61     // Do nothing
62     // setup() and loop() run in their own task with priority 1 in core 1
63     // on ESP32
64 }
65 }

Done Saving.
Writing at 0x00000000... (100 %)
Wrote 267440 bytes (125388 compressed) at 0x00010000 in 3.4 seconds (effective 627.3 kbit/s)...
Flash of data verified.
Compressed 3072 bytes to 119...
61
```

# Embedded Programming 2/3

1. Introduction
2. Example with FreeRTOS

## Example with FreeRTOS on ESP32

```
#define LED_BUILTIN 4

// Use only core 1 for demo purposes
static const BaseType_t app_cpu = 1;

// LED rates
static const int rate_1 = 500; // ms
static const int rate_2 = 2000; // ms

// Pins
static const int led_pin = LED_BUILTIN;
```

# Embedded Programming 2/3

1. Introduction
2. Example with FreeRTOS

## Example with FreeRTOS on ESP32

```
// Our task: blink an LED at one rate
void toggleLED(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 say_hello(void *parameter) {
    while(1) {
        vTaskDelay(rate_2 / portTICK_PERIOD_MS);
        Serial.println("Hello HEI");
    }
}
```

# Embedded Programming 2/3

## 1. Introduction

## 2. Example with FreeRTOS

## Example with FreeRTOS on ESP32

```
void setup() {  
  
    // Configure pin  
    pinMode(led_pin, OUTPUT);  
    Serial.begin(115200);  
  
    // Task to run forever  
    xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS  
        toggleLED, // Function to be called  
        "Blink", // Name of task  
        2048, // Stack size (bytes in ESP32, words in FreeRTOS)  
        NULL, // Parameter to pass to function  
        0, // 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  
        say_hello, // Function to be called  
        "Hello", // Name of task  
        2048, // Stack size (bytes in ESP32, words in FreeRTOS)  
        NULL, // Parameter to pass to function  
        0, // 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.  
}
```

# *Embedded Programming 2/3*

1. Introduction
2. Example with FreeRTOS

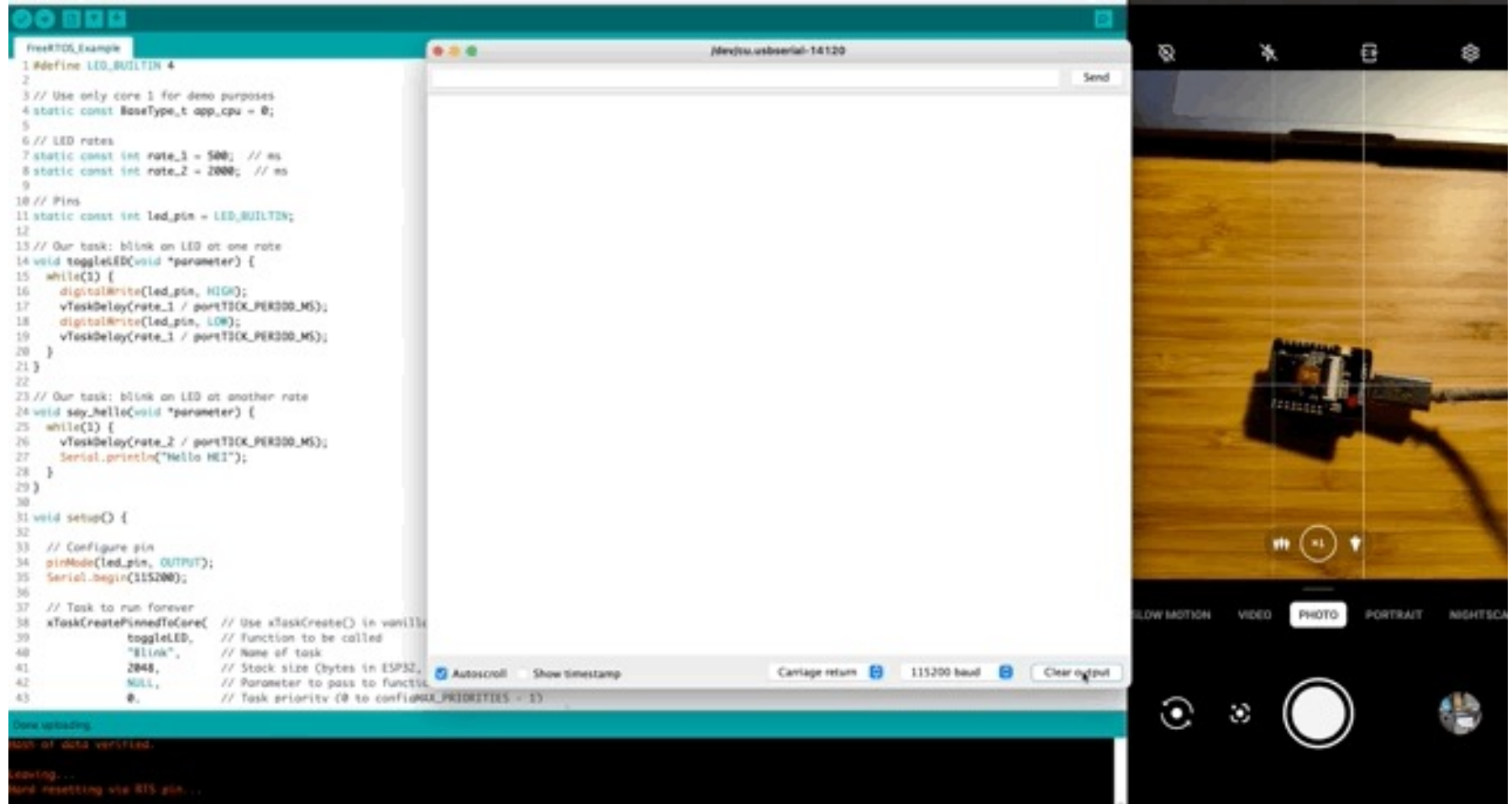
## Example with FreeRTOS on ESP32

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

# Embedded Programming 2/3

1. Introduction
2. Example with FreeRTOS

## Example with FreeRTOS on ESP32





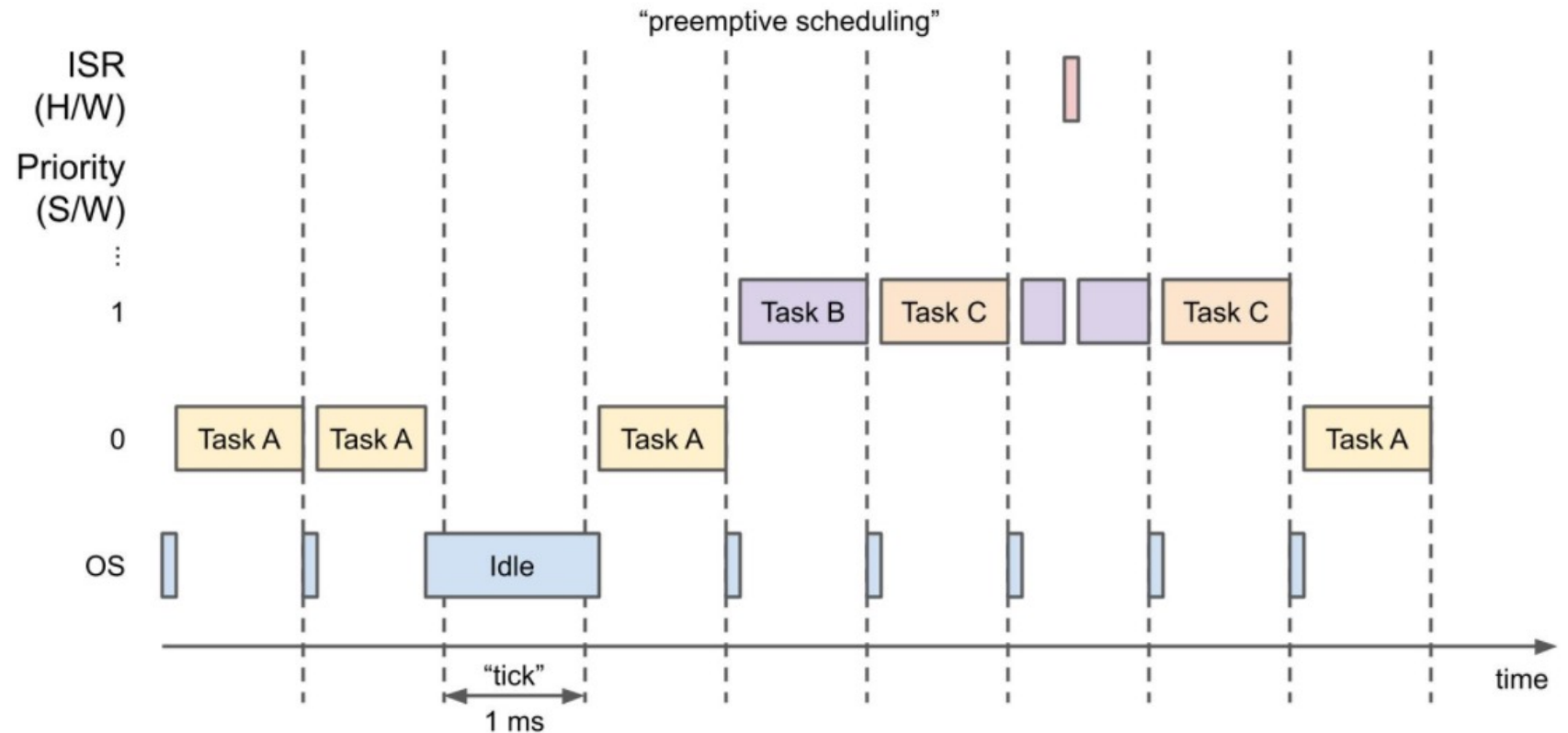
# Embedded Programming 2/3

1. Introduction
2. Example with FreeRTOS
3. Task Scheduling

## Example with FreeRTOS on ESP32

### What actually happens\*

\*assuming single-core processor

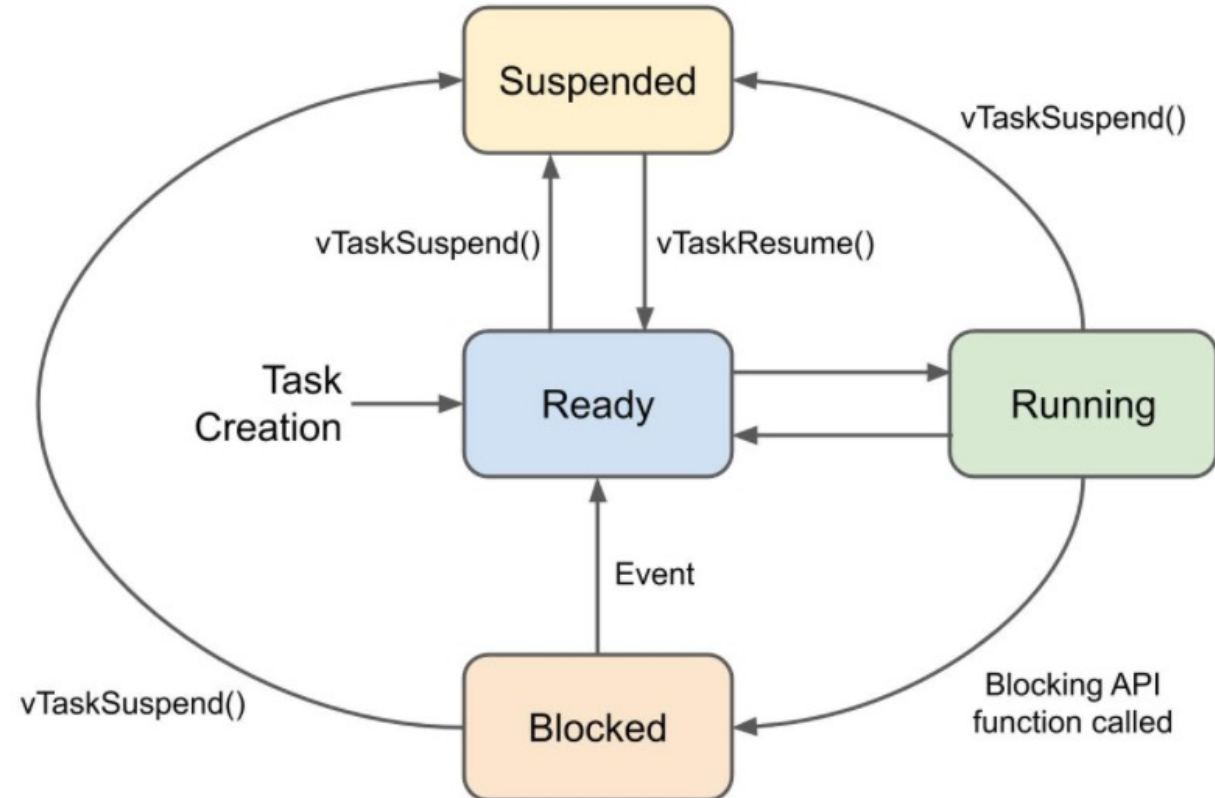


# Embedded Programming 2/3

1. Introduction
2. Example with FreeRTOS
3. Task Scheduling

## Example with FreeRTOS on ESP32

### Task States



# Thank you!



[hello@edgeimpulse.com](mailto:hello@edgeimpulse.com)

Edge Impulse  
3031 Tisch Way 110 Plaza West  
San Jose, CA 95128 USA