INTRODUCTION TO RTOS PART 1 - WHAT IS RTOS?

RTOS = real-time operating system

Operating system (OS) = software that runs on a computer/microcontroller

→ accomplishes a number of important functions

↳ ① Task scheduling

↳ due to background processes executing at the same time

↳ OS figures out how to divide the time between the applications so it appears to run concurrently

↳ ② Resource management

↳ virtual resources eg. files, libraries & folders so applications can access them when needed

↳ ③ Device drivers

↳ allows system to read & write from external disks

↳ respond to keyboard & mouse i/p

General Purpose Operating System (GPOS)

↳ eg. Windows, Linux, MacOS, iOS, Android

↳ prioritises human interaction → so scheduler prioritises such tasks

↳ ∴ some timing deadlines can be pushed or delayed

↳ as long as the lag is unnoticable by humans

↳ scheduler is non-deterministic

↳ when the task executes & its duration is unknown

↳ ∴ for OS w/ strict timing deadline is catastrophic!!

Real-Time Operating System (RTOS)

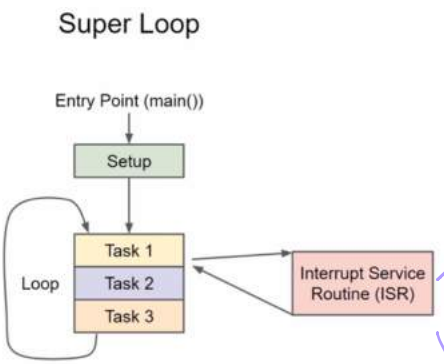
↳ so scheduler can meet timing deadlines for the task

↳ device drivers includes things intended for microcontroller

↳ eg. Wi-Fi & Bluetooth stack, LCD drivers

↳ can run multiple tasks concurrently

PROGRAM FLOWS → ① Super Loop

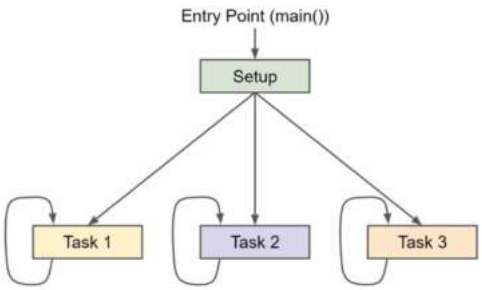


↳ can be used to meet strict timing deadlines for 1 or 2 tasks

↳ for timing deadlines < 1 ms

Advantages	Disadvantages
• for simple microcontroller projects	• cannot execute tasks concurrently
• saves CPU cycle or memory	↳ can be fixed by running multiple tasks @ same time on multicore processor
• easier to debug	• can handle one process
• can handle multiple processes	

RTOS

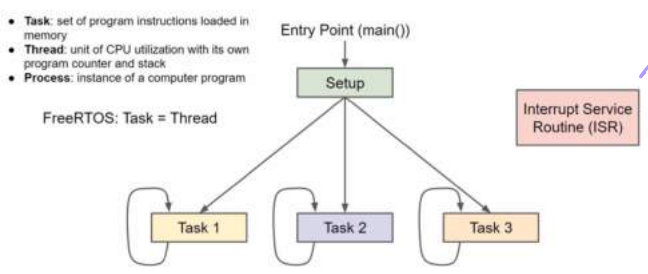


- **Task**: set of program instructions loaded in memory
- **Thread**: unit of CPU utilization with its own program counter and stack
- **Process**: instance of a computer program

→ a program usually has ≥1 threads to accomplish tasks

In FreeRTOS → task = thread

RTOS



→ can interrupt task if it has higher prioritization

→ to handle tasks concurrently

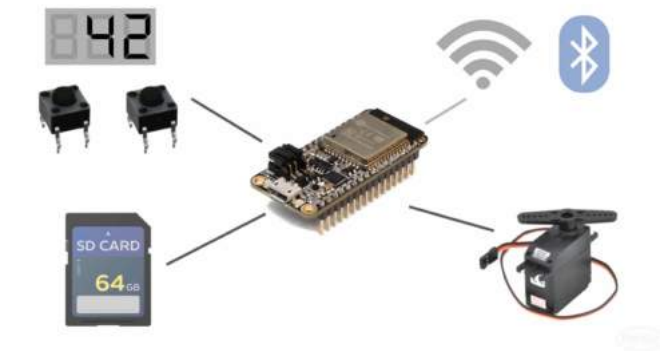
ATmega 328p
- 16 MHz
- 32 kB flash
- 2 kB RAM

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

- RTOS allows multiple tasks to run concurrently

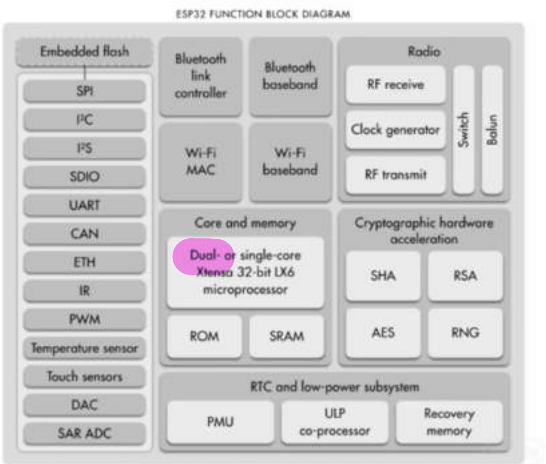


- RTOS allows division of tasks in projects

INTRODUCTION TO RTOS PART 2

ESP 32 architecture:

Symmetric Multiprocessing (SMP)



- ESP32 runs a modified version of FreeRTOS (ESP-IDF)
- Running ESP32 with 1-core:

- vTaskDelay function → tells scheduler to run other tasks until the delay time is up → then continue running the task
- tick timers → microcontroller's hardware timer → allocated to interrupt the processor at a specified interval

tick = interrupt period

1 tick period = 1 ms

- xTaskCreatePinnedToCore (→ run task in the specified core

```
void setup() {

  // Configure pin
  pinMode(led_pin, OUTPUT);

  // Task to run forever
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
              toggleLED,    // Function to be called
              "Toggle LED", // 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.

}
```

→ higher no. = higher priority
↳ for this, there's 25 slots,
0 is lowest priority &
24 is highest

Task: LED blink at different rates
↳ create tasks & set delay to something different