Microcontroladores II

Juan Esteban Giraldo Hoyos
Ingeniero Electrónico
Magíster en Gestión de Ciencia, Tecnología e Innovación

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Anteriormente trabajamos

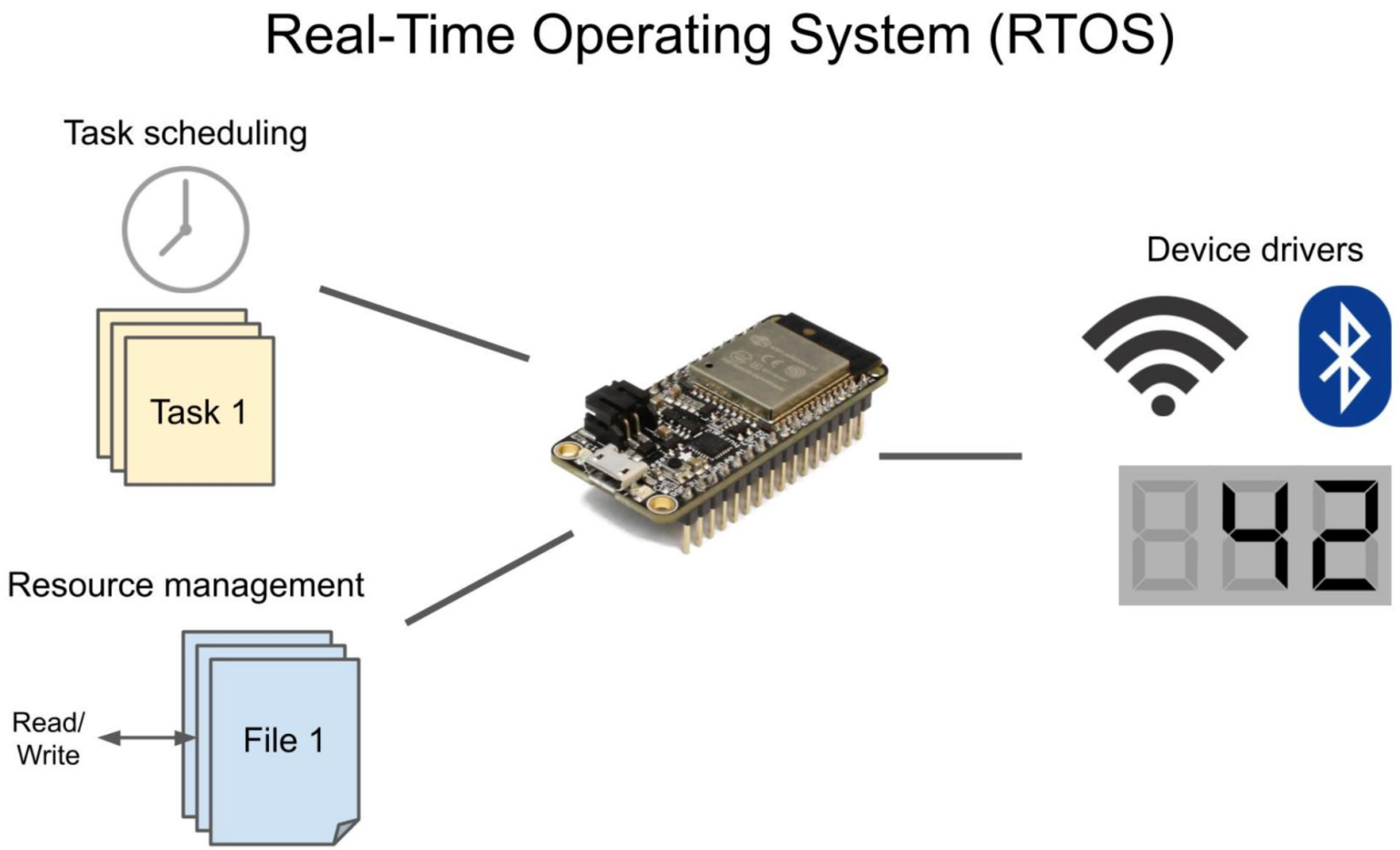
com x

Escuela de Ingenierías

IUSH Institución Universitaria SALAZAR Y HERRERA

SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Hoy trabajaremos



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS

Los sistemas operativos en tiempo real (RTOS, por sus siglas en inglés) son un tipo especial de sistema operativo diseñado para aplicaciones en las que es crucial que las operaciones se realicen dentro de tiempos específicos o bajo condiciones de tiempo restringido. Estos sistemas se utilizan comúnmente en aplicaciones de microcontroladores, donde el rendimiento, la eficiencia y la fiabilidad son esenciales.

Los RTOS son útiles porque proporcionan una estructura organizada para ejecutar múltiples tareas y manejar interrupciones y eventos en un marco de tiempo preciso. Estas características son especialmente importantes en dispositivos embebidos y sistemas industriales, donde el tiempo de respuesta puede ser crítico para la seguridad y el rendimiento.

Microcontroladores II - RTOS

- **Predecibilidad y determinismo:** En un RTOS, el tiempo de respuesta es predecible y controlado, permitiendo que las tareas críticas se completen dentro de los límites de tiempo establecidos.
- **Gestión de múltiples tareas:** Los RTOS permiten la ejecución concurrente de múltiples tareas, facilitando la implementación de sistemas complejos con distintos módulos de software trabajando simultáneamente.
- **Manejo de interrupciones y eventos:** Los RTOS están diseñados para manejar interrupciones y eventos con eficiencia, lo que es fundamental para la operación de dispositivos de control y comunicación.
- **Eficiencia:** Los RTOS suelen ser ligeros y están optimizados para funcionar en hardware de baja potencia y recursos limitados, como microcontroladores.

Microcontroladores II - RTOS

American Megatrends

RTOS and GPOS Differences

RTOS

Real-Time Operating System

- Deterministic: no random execution pattern
- Predictable Response Times
- Time Bound
- Preemptive Kernel

Examples:

Contiki source code, FreeRTOS™, Zephyr™ Project

Use Case:

Embedded Computing

GPOS

General-Purpose Operating System

- Dynamic memory mapping
- Random Execution Pattern
- Response Times not Guaranteed

Examples:

Microsoft® Windows® operating system, Apple® macOS® operating system, Red Hat® Enterprise Linux® operating system

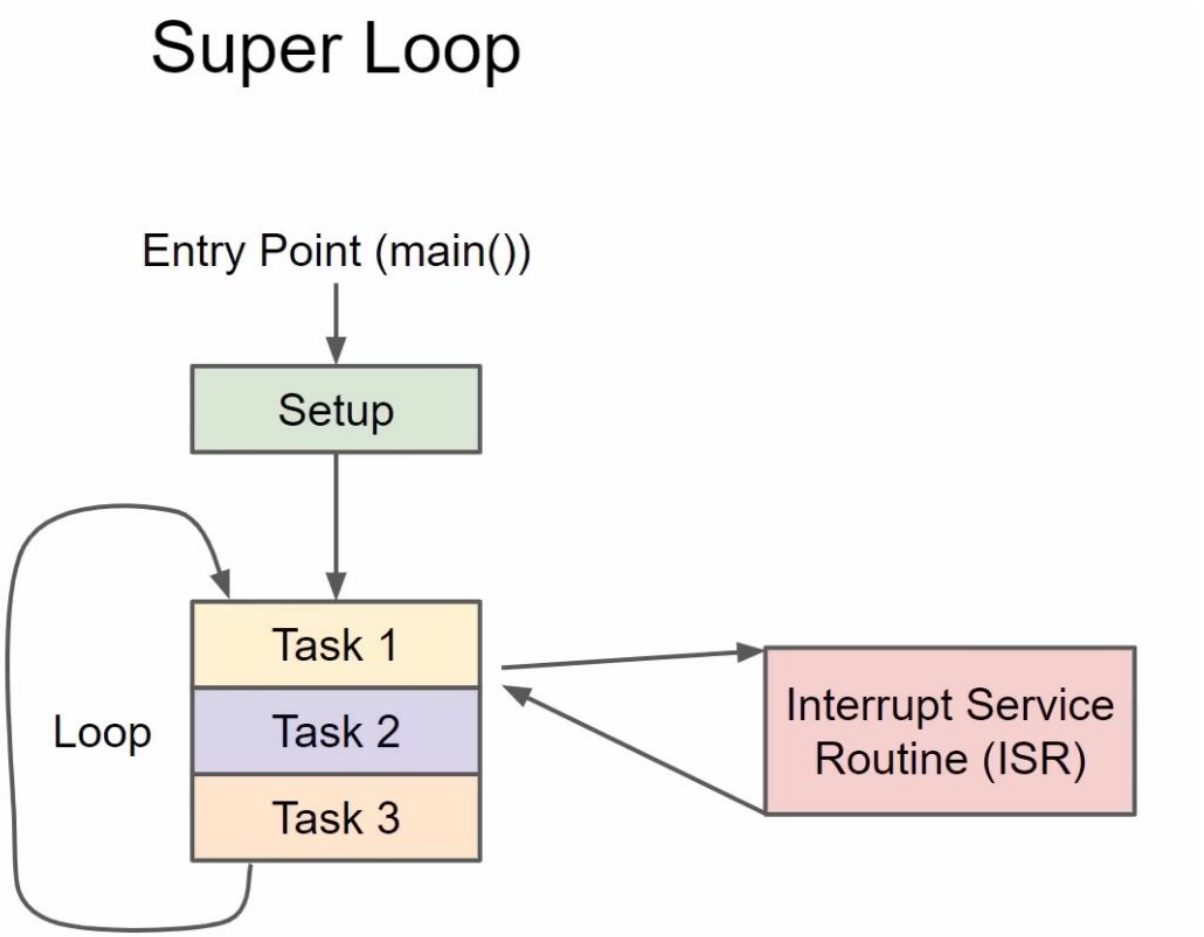
Use Case:

Desktop, Laptop, Tablet computers

Microcontroladores II - RTOS Más populares

- **FreeRTOS:** Es uno de los RTOS más populares y ampliamente utilizados para microcontroladores. Es ligero y tiene una licencia de código abierto. Ofrece funcionalidades básicas para multitarea y sincronización, y tiene una comunidad activa y un soporte sólido.
- **RT-Thread:** Es otro RTOS de código abierto, diseñado para aplicaciones embebidas y de tiempo real. Proporciona un entorno flexible para el desarrollo de aplicaciones y es compatible con una amplia gama de microcontroladores.
- **Zephyr:** Este RTOS es una plataforma modular y escalable para dispositivos embebidos, respaldado por la Linux Foundation. Ofrece compatibilidad con múltiples arquitecturas y una amplia gama de funciones para aplicaciones de tiempo real.
- **ChibiOS/RT:** Un RTOS pequeño y eficiente, diseñado para microcontroladores con recursos limitados. Proporciona funcionalidad para multitarea, sincronización y comunicación entre tareas.
- **NuttX:** Un RTOS de código abierto diseñado para ser compatible con POSIX y ANSI C, permitiendo portabilidad y flexibilidad para una amplia gama de aplicaciones embebidas.
- **QNX:** Aunque generalmente se usa en sistemas más grandes, también es utilizado en aplicaciones embebidas y de tiempo real para microcontroladores de alto rendimiento. Es conocido por su estabilidad y fiabilidad.

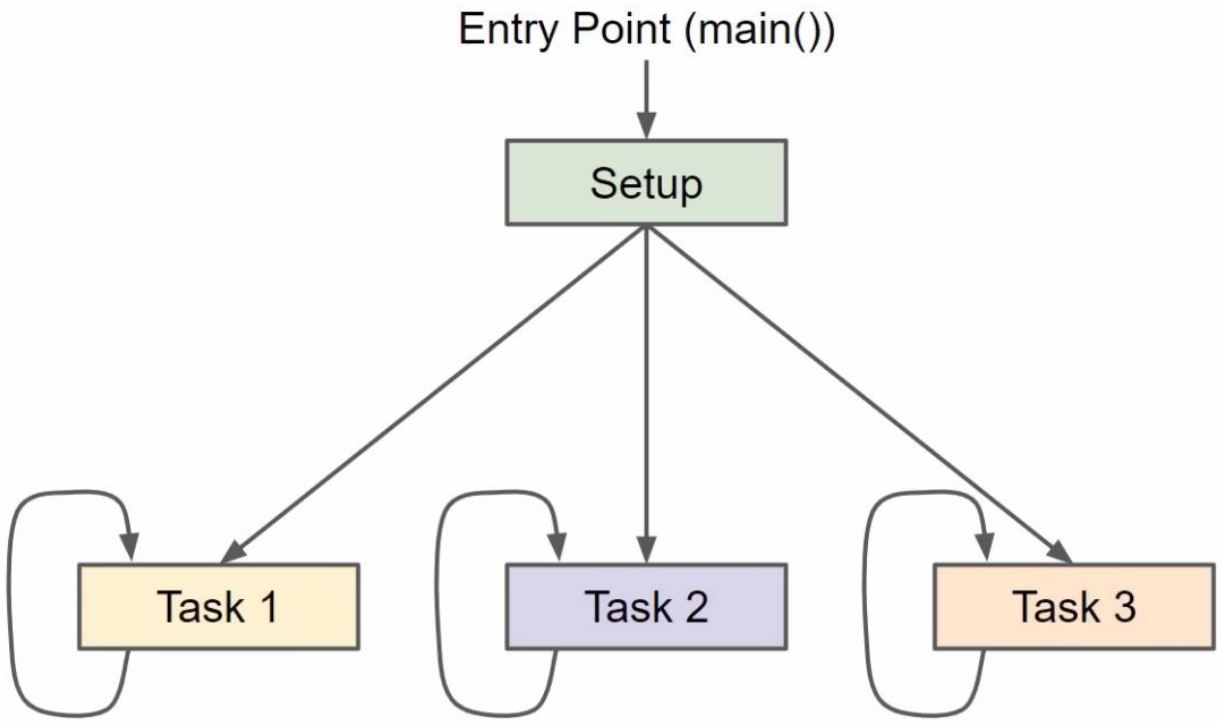
Microcontroladores II - RTOS



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS

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

Microcontroladores II - RTOS

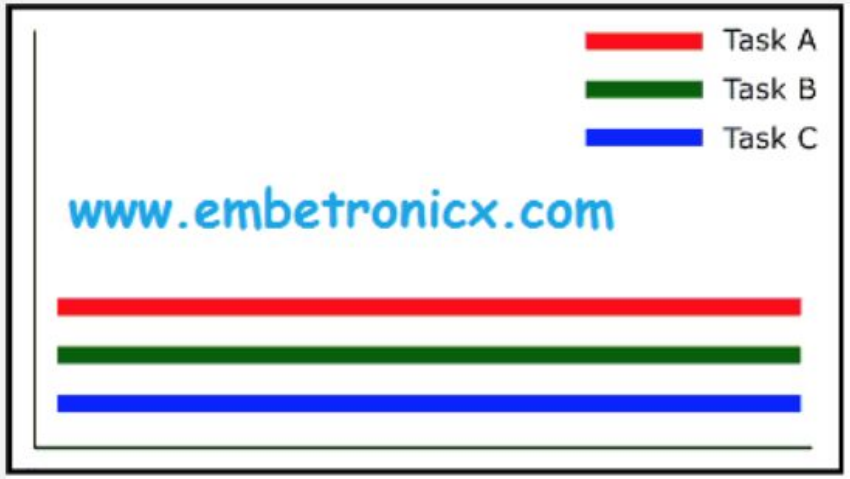


Fig 1: Task A, Task B and Task C running concurrently

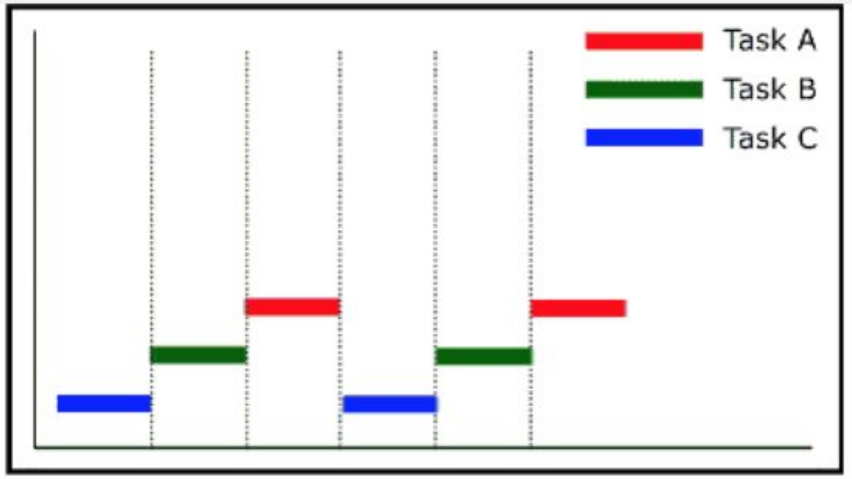
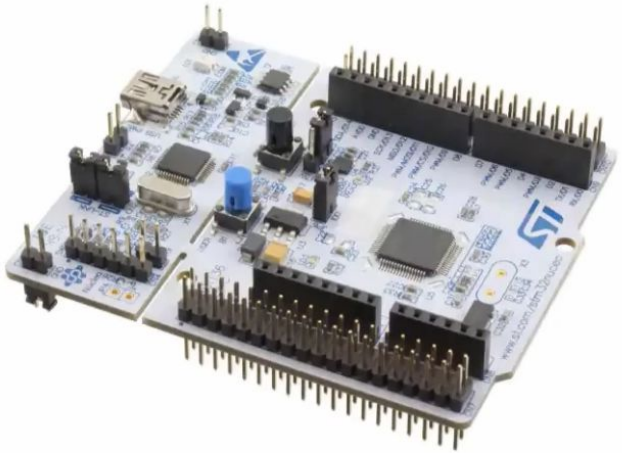


Fig 2: Task A, Task B and Task C running interleaved

Microcontroladores II - RTOS



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

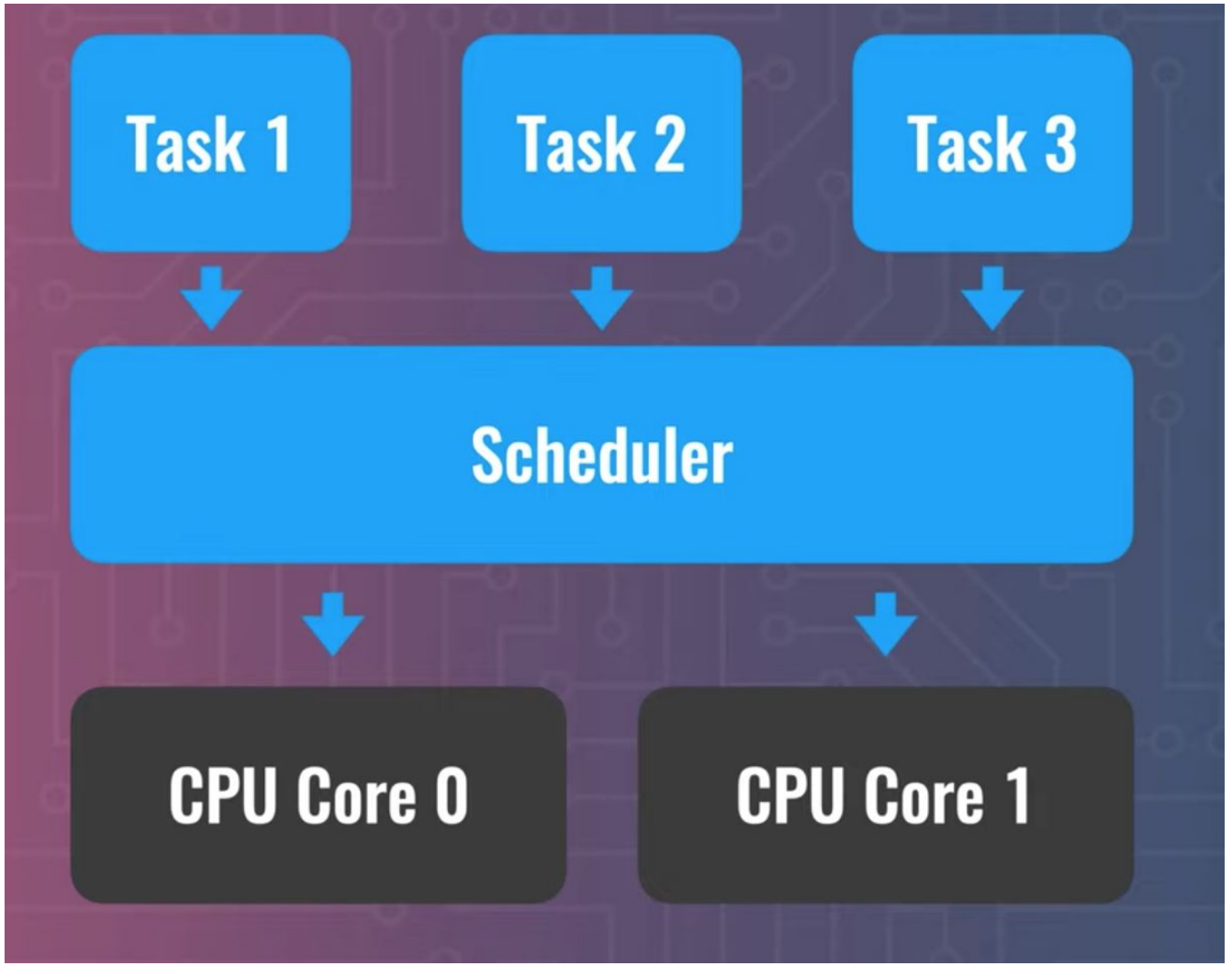
STM32L476RG
- 80 MHz
- 1 MB flash
- 128 kB RAM

ESP-WROOM-32
- 240 MHz (dual core)
- 4 MB flash
- 520 kB RAM

Super Loop → RTOS

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS

Keep WiFi connected

Measure electricity use

Update LCD

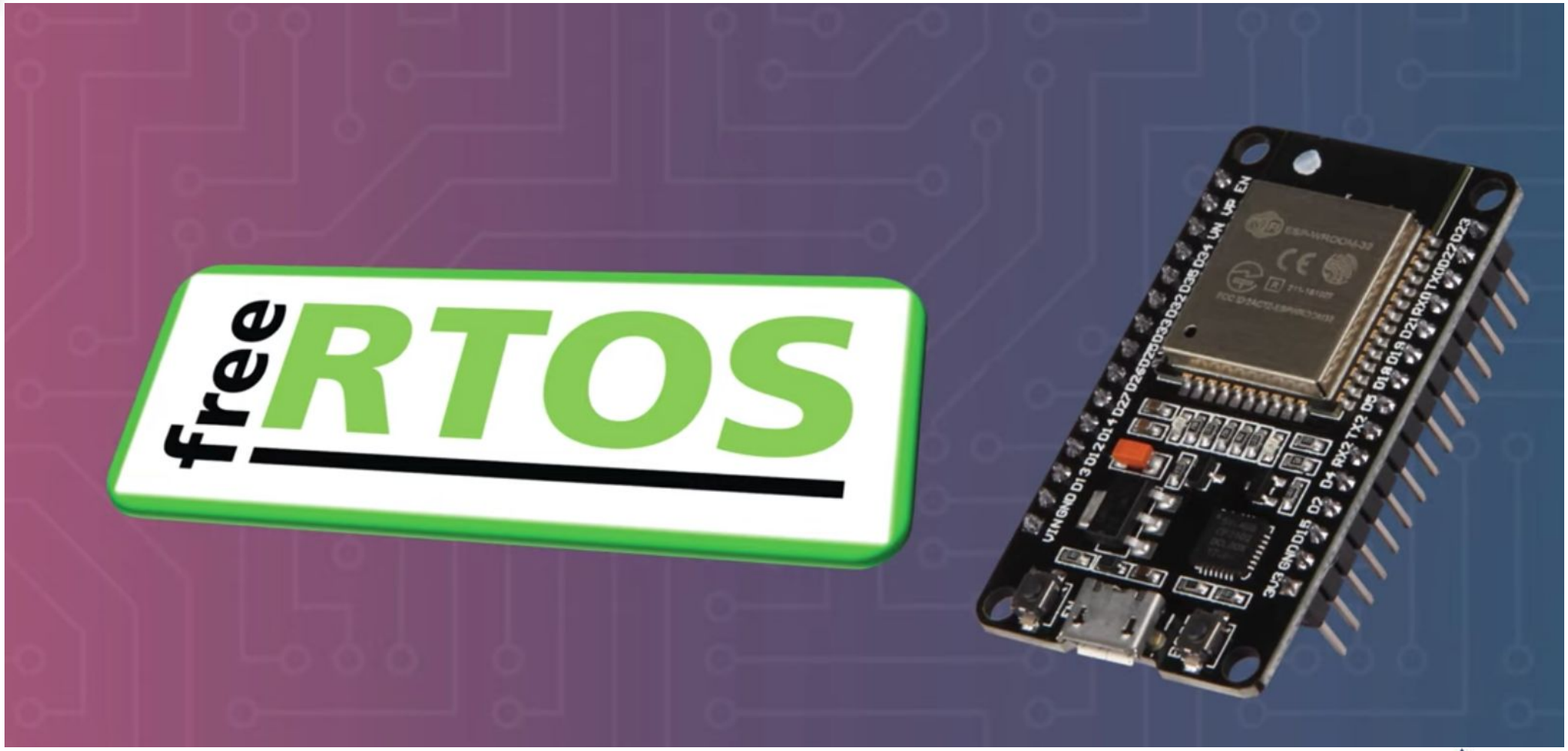
Upload data to AWS

Escuela de Ingenierías

IUSH Institución Universitaria SALAZAR Y HERRERA

SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

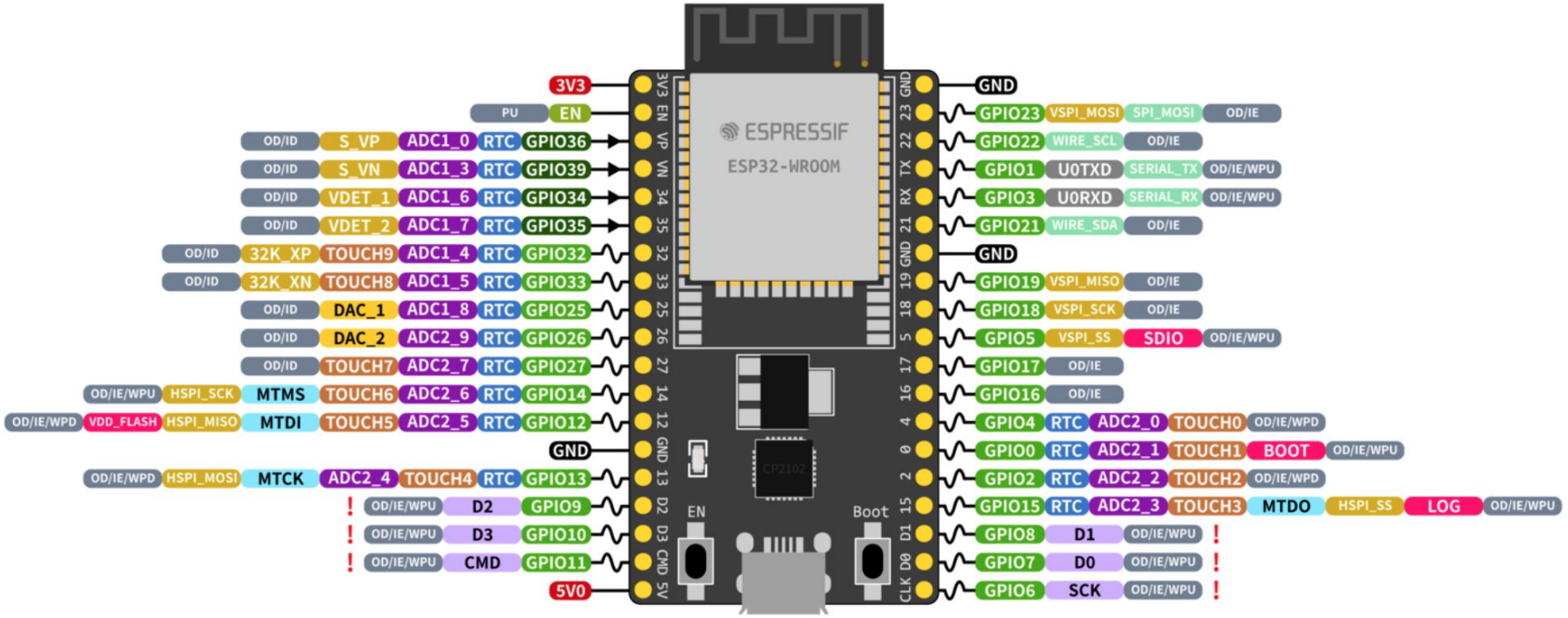


Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

ESP32-DevKitC

ESPRESSIF



ESP32 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
BLuetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
DAC_X Digital-to-Analog Converter
DEBUG JTAG for Debugging
FLASH External Flash Memory (SPI)
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
ARDUINO Arduino Related Functions
STRAP Strapping Pin Functions

RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)
! Pin Shared with the Flash Memory *Can't be used as regular GPIO*

GPIO STATE
WPU: *Weak Pull-up (Internal)*
WPD: *Weak Pull-down (Internal)*
PU: *Pull-up (External)*
IE: *Input Enable (After Reset)*
ID: *Input Disabled (After Reset)*
OE: *Output Enable (After Reset)*
OD: *Output Disabled (After Reset)*

IUSH Institución Universitaria SALAZAR Y HERRERA
ABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

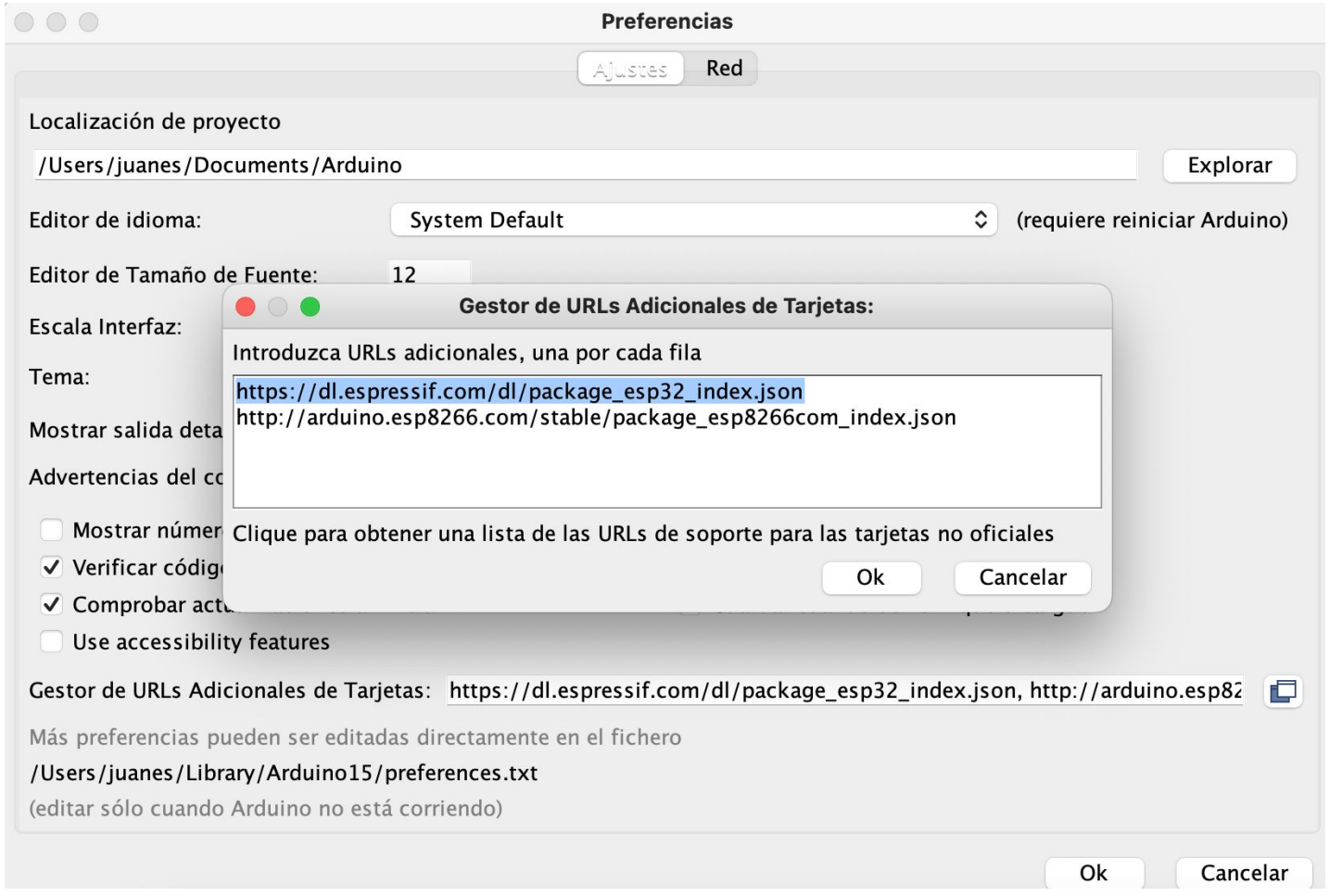
Especificación	Detalle
Fabricante	Espressif Systems
CPU	Dual-core Xtensa LX6
Frecuencia de CPU	Hasta 240 MHz
Memoria RAM	520 KB SRAM
Memoria Flash	4 MB (configurable hasta 16 MB o más, según el modelo y uso de memoria externa)
Conectividad inalámbrica	Wi-Fi 802.11 b/g/n, Bluetooth 4.2 BR/EDR y BLE
GPIO	34 pines de propósito general (configurables como entrada o salida)
ADC	12-bit, hasta 18 canales
DAC	2 canales de 8-bit
Interfaz de comunicación	UART, I2C, SPI, I2S, CAN, IR, RMT
Interfaz de hardware	JTAG, SD/SDIO/MMC, Ethernet MAC
PWM	Hasta 16 canales con frecuencia ajustable
Temporizadores y Watchdog	4 temporizadores de hardware y Watchdog Timer (WDT) configurable
Sensores internos	Sensor de temperatura interna
Periféricos adicionales	RTC (Real-Time Clock), Touch Sensor (hasta 10 canales)
Voltaje de operación	2.2 V - 3.6 V
Consumo de energía	Varía según el modo de operación (modo de baja potencia, normal, y de alta potencia)
Temperatura de operación	-40°C a 125°C
Factor de forma	Varía según el modelo, pero comúnmente se encuentra en módulos compactos y tableros de desarrollo
Entorno de desarrollo	Soporte para Ard[illegible] IDE, Espressif IDF, y otras herramientas de desarrollo embebido

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

Configurando el entorno de desarrollo para usar el ESP32:

1) Agregar en **preferences** el nuevo microcontrolador ESP32 **http://dl.espressif.com/dl/package_esp32_index.json**

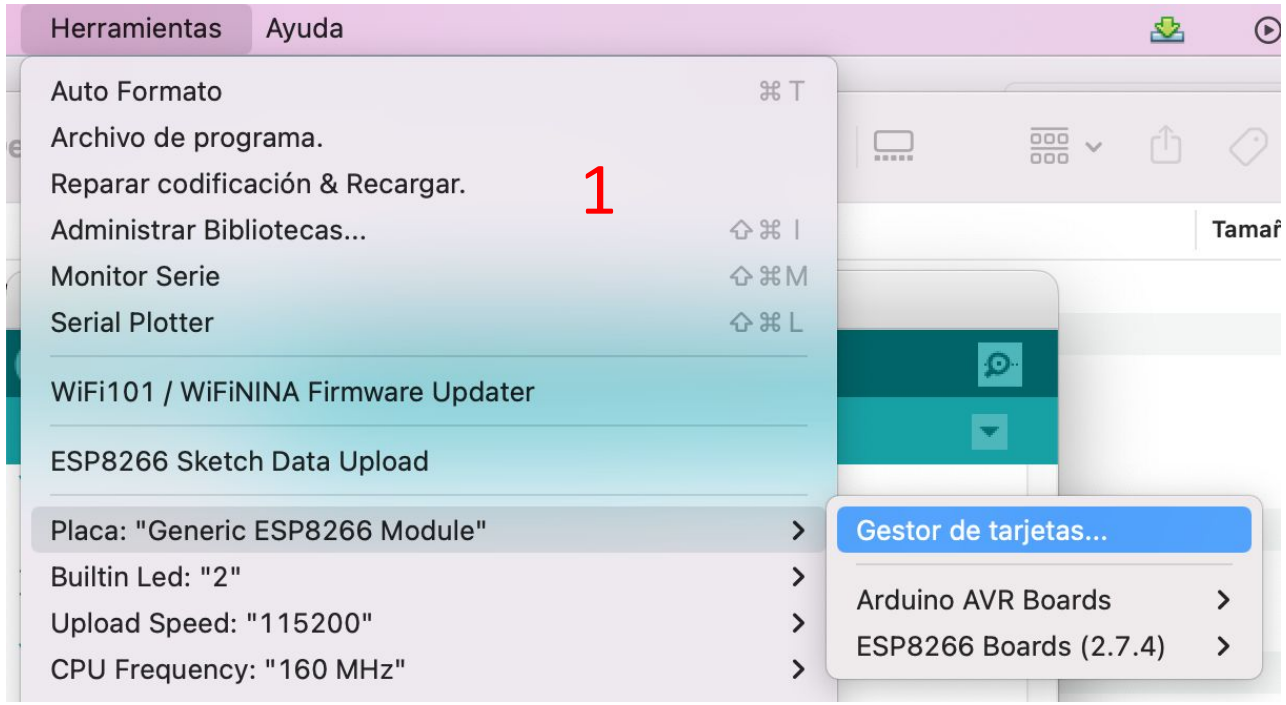


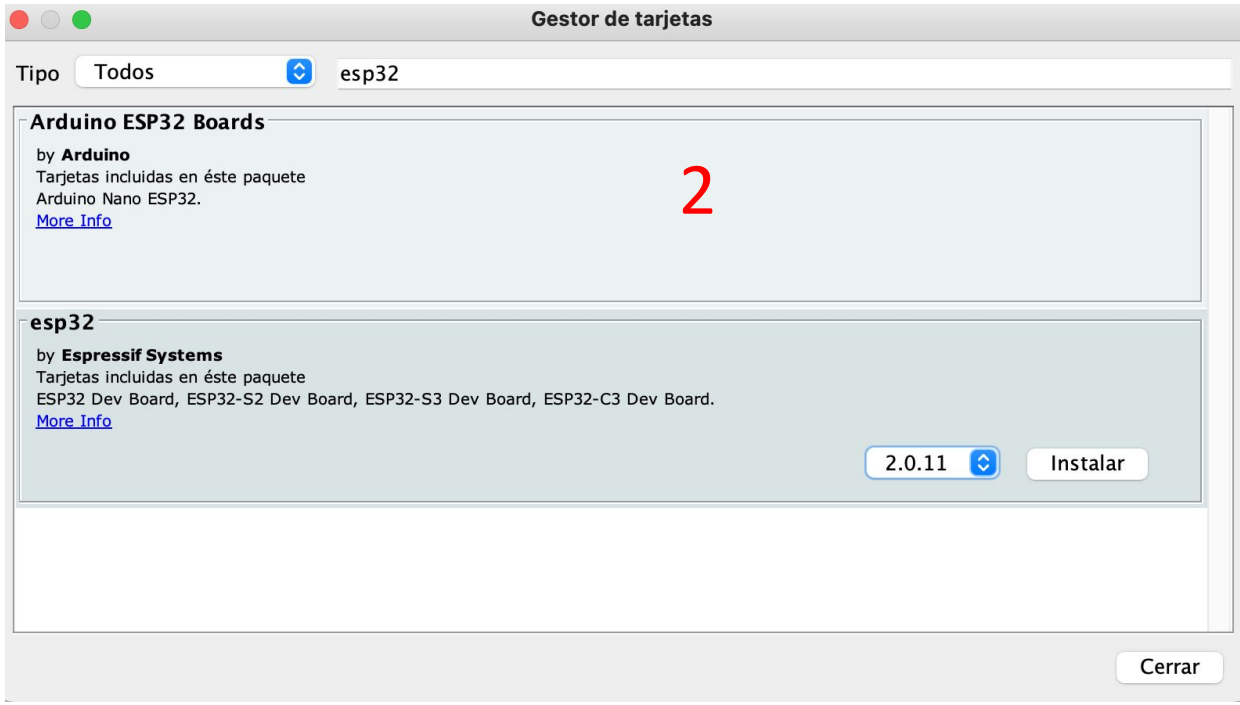
Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

Configurando el entorno de desarrollo para usar el ESP32:

2) Instalamos la tarjeta para poder desarrollar con el microcontrolador ESP32 desde el Arduino IDE: Herramientas (Tools) > Gestor de tarjetas

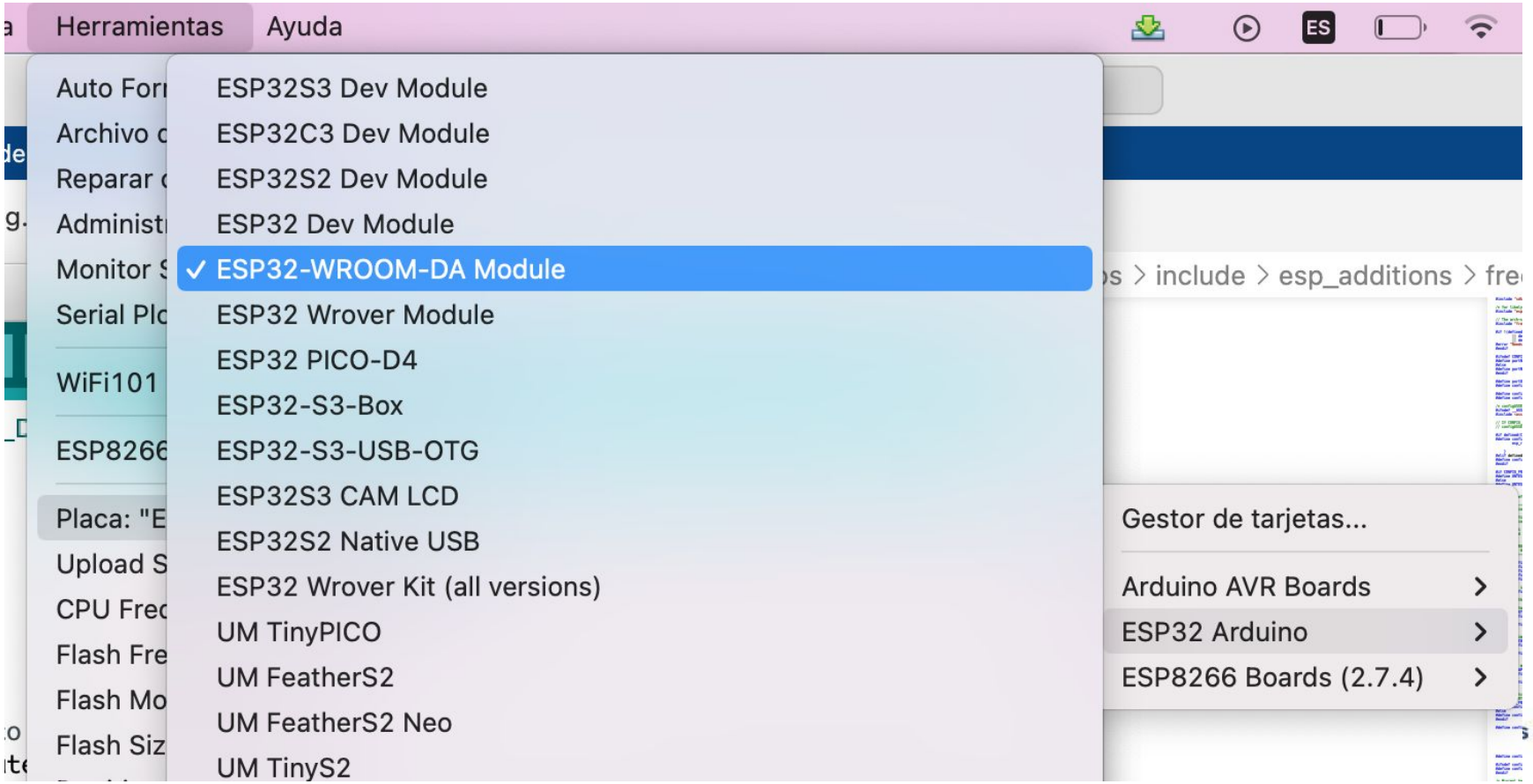




Microcontroladores II - FREERTOS Y ESP32

Configurando el entorno de desarrollo para usar el ESP32:

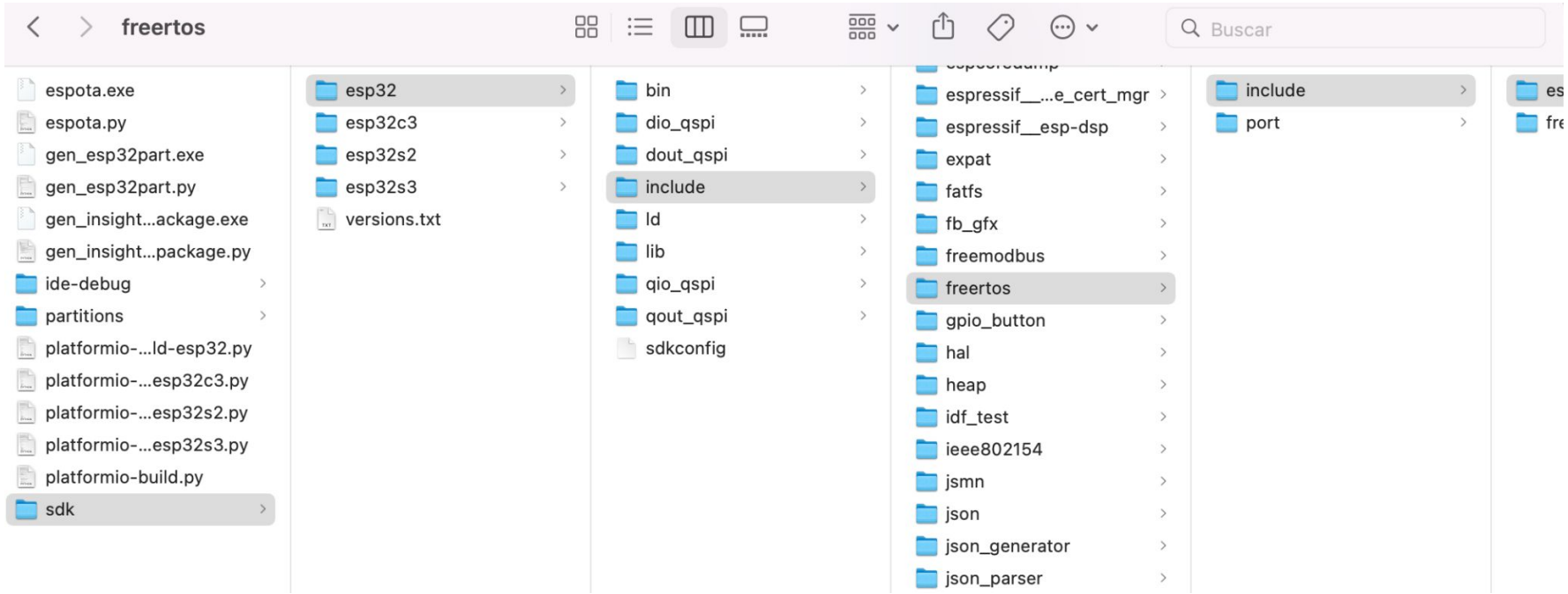
3) Al instalar la tarjeta ESP32 debemos seleccionar la versión con la que vamos a trabajar. Herramientas > Placa > ESP32 > Seleccionar del listado



IUSH Institución Universitaria SALAZAR Y HERRERA
SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

Con la instalación se agrega al Arduino IDE de forma automática también todo el soporte para utilizar FREERTOS en la tarjeta de desarrollo con el microcontrolador ESP32



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

Ahora vamos a realizar nuestro primer programa usando FREERTOS:
(ENCENDER Y APAGAR UN LED)

FREERTOS_DEMO §

```
// 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
#define rate_1 500 // ms


// Pins
#define led_pin 23

// 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);
  }
}
```

```
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)

}

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

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

Ahora vamos a agregar una segunda tarea (task) que encienda el mismo led en un intervalo de tiempo diferente

```
    // LED rates
    #define rate_1 500 // ms
  > #define rate_2 323  // 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);
  }
}
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - FREERTOS Y ESP32

Ahora vamos a agregar una segunda tarea (task) que encienda el mismo led en un intervalo de tiempo diferente

```
// 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)
```

FREERTOS Y ESP32 PRACTICA 10%

Agregar a la implementación anterior un segundo led y manejar un encendido y apagado para el led 1 de cada 2 segundos. Y para el led 2 un encendido y apagado de cada 3 segundos.

Adicionar un sensor de temperatura y realizar una tarea nueva que consulte el estado del sensor cada 30 seg y que muestre su valor por serial

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Bibliografía

- https://www.freertos.org/Documentation/Mastering-the-FreeRTOS-Real-Time-Kernel.v1.0.pdf
- https://www.digikey.com/en/maker/projects/what-is-a-realtime-operating-system-rtos/28d8087f53844decafa5000d89608016
- https://www.youtube.com/watch?v=kP-pP6FEu8I&list=PLzvRQMJ9HDiQ3OIuBWCEW6yE0S0LUWhGU&index=20
- https://drive.google.com/file/d/1Lf2deyf0xiE3iye8TglEaVfmIN05to8a/view