Prática 1

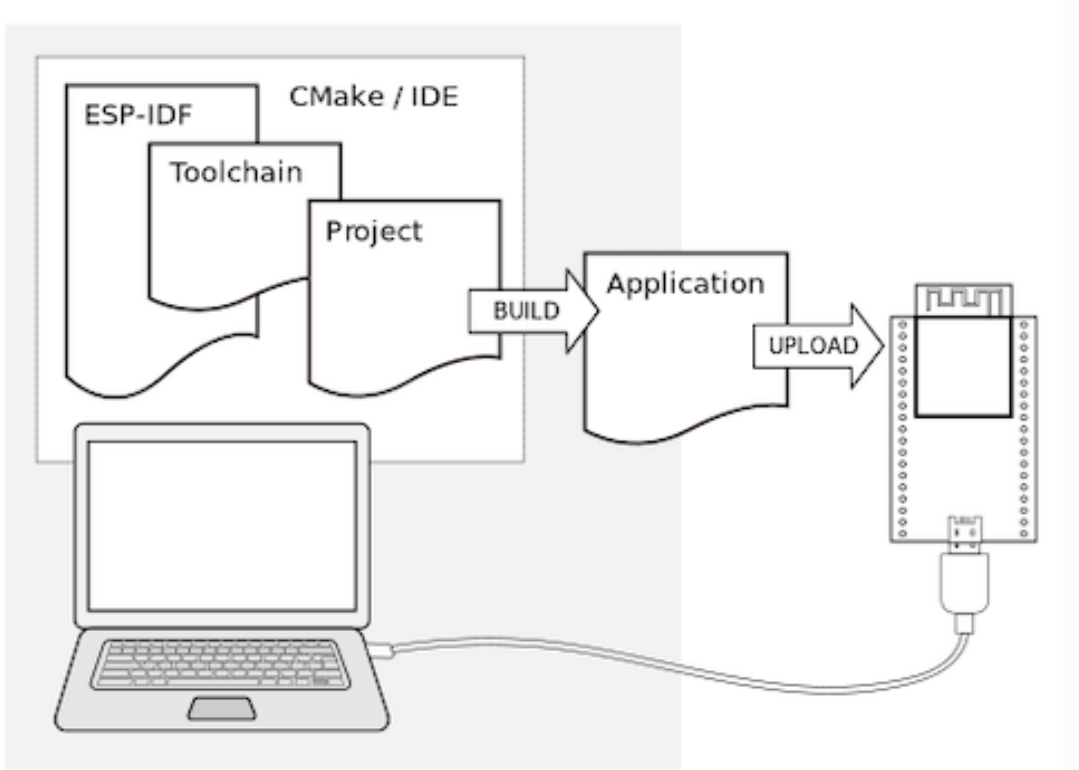
Hello World

ESP32

ESP32 is a system on a chip that integrates the following features:

- Wi-Fi (2.4 GHz band)
- Bluetooth
- Dual high performance Xtensa® 32-bit LX6 CPU cores
- Ultra Low Power co-processor
- Multiple peripherals

Software


CMake / IDE
ESP-IDF
Toolchain
Project
BUILD
Application
UPLOAD

Esp_chip_info_t

`void esp_chip_info(esp_chip_info_t *out_info)`

Fill an esp_chip_info_t structure with information about the chip.

Parameters: **out_info** – **[out]** structure to be filled

struct **esp_chip_info_t**

The structure represents information about the chip.

Public Members

esp_chip_model_t **model**

chip model, one of esp_chip_model_t

uint32_t **features**

bit mask of CHIP_FEATURE_x feature flags

uint16_t **revision**

chip revision number (in format MXX; where M - wafer major version, XX - wafer mi version)

uint8_t **cores**

number of CPU cores

enum **esp_chip_model_t**

Chip models.

Values:

enumerator **CHIP_ESP32**

ESP32.

enumerator **CHIP_ESP32S2**

ESP32-S2.

enumerator **CHIP_ESP32S3**

ESP32-S3.

enumerator **CHIP_ESP32C3**

ESP32-C3.

enumerator **CHIP_ESP32H2**

ESP32-H2.

enumerator **CHIP_ESP32C2**

ESP32-C2.

```
typedef enum {
    CHIP_ESP32  = 1, //!< ESP32
    CHIP_ESP32S2 = 2, //!< ESP32-S2
    CHIP_ESP32S3 = 9, //!< ESP32-S3
    CHIP_ESP32C3 = 5, //!< ESP32-C3
    CHIP_ESP32H2 = 6, //!< ESP32-H2
    CHIP_ESP32C2 = 12, //!< ESP32-C2
} esp_chip_model_t;
```

Macros

`CHIP_FEATURE_EMB_FLASH`

Chip has embedded flash memory.

`CHIP_FEATURE_WIFI_BGN`

Chip has 2.4GHz WiFi.

`CHIP_FEATURE_BLE`

Chip has Bluetooth LE.

`CHIP_FEATURE_BT`

Chip has Bluetooth Classic.

`CHIP_FEATURE_IEEE802154`

Chip has IEEE 802.15.4.

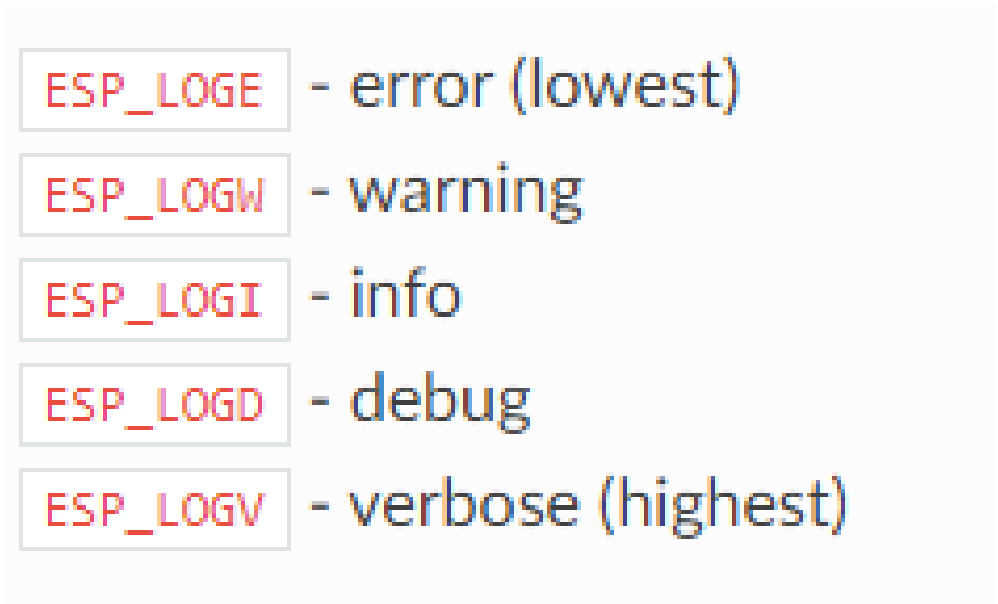
`CHIP_FEATURE_EMB_PSRAM`

Chip has embedded psram.

https://docs.espressif.com/projects/esp-idf/en/v5.1/esp32/api-reference/system/misc_system_api.html

Logging library

- `static const char* TAG = "MyModule";`
- `ESP_LOGW(TAG, "Baud rate error %.1f%%. Requested: %d baud, actual: %d baud", error * 100, baud_req, baud_real);`



https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/log.html

Logging library

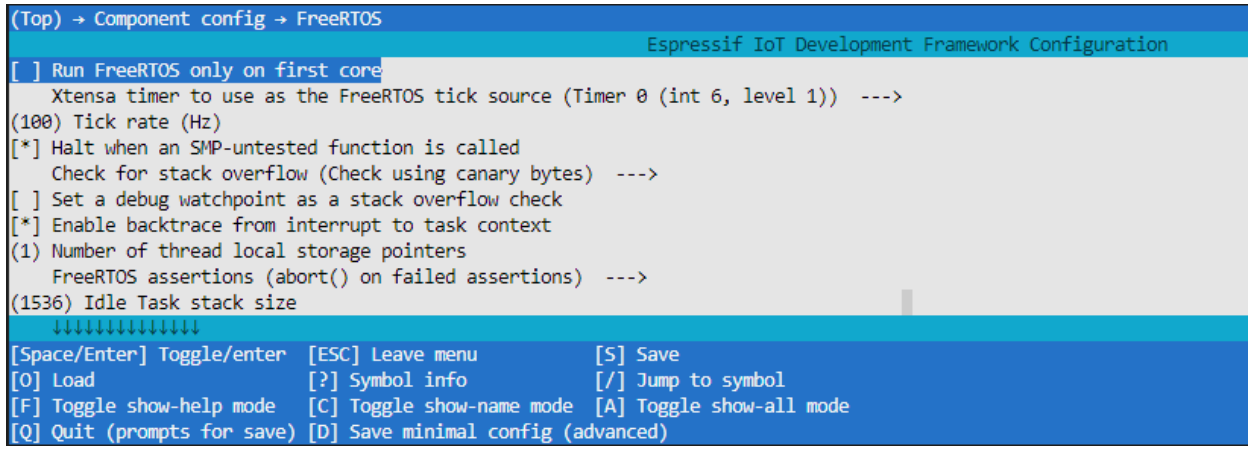
```
esp_log_level_set("*", ESP_LOG_ERROR);        // set all components to ERROR level
esp_log_level_set("wifi", ESP_LOG_WARN);      // enable WARN logs from WiFi stack
esp_log_level_set("dhcpc", ESP_LOG_INFO);     // enable INFO logs from DHCP client
```

void vTaskDelay(*const* TickType_t xTicksToDelay)

```
void vTaskFunction( void * pvParameters )
{
// Block for 500ms.
const TickType_t xDelay = 500 / portTICK_PERIOD_MS;

  for( ;; )
  {
      // Simply toggle the LED every 500ms, blocking between each toggle.
      vToggleLED();
      vTaskDelay( xDelay );
  }
}
```

Ex. Tick Rate = 100 Hz.
portTICK_PERIOD_MS = 10 mS



Idf.py menuconfig

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html

Task Management

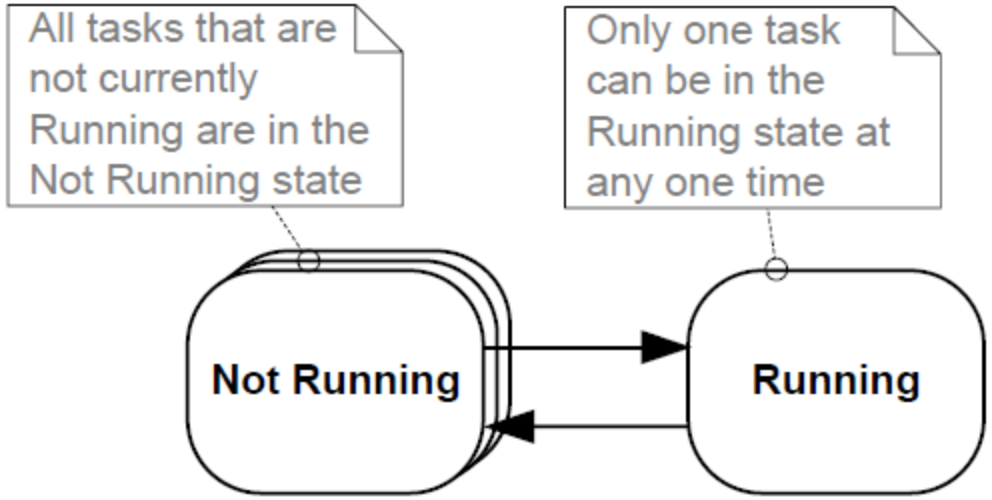


Figure 9. Top level task states and transitions

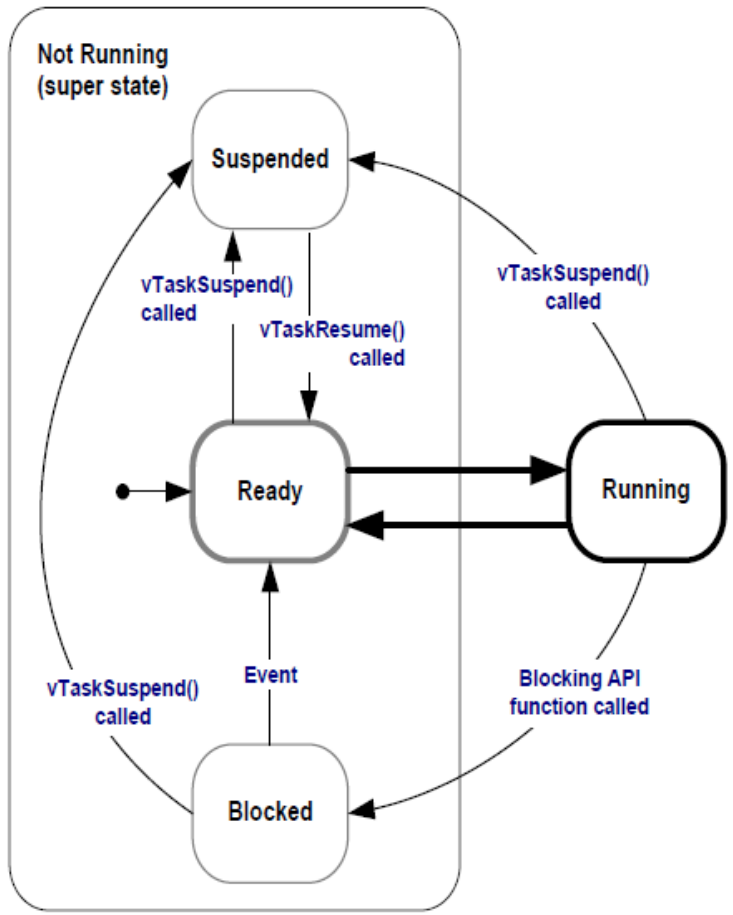
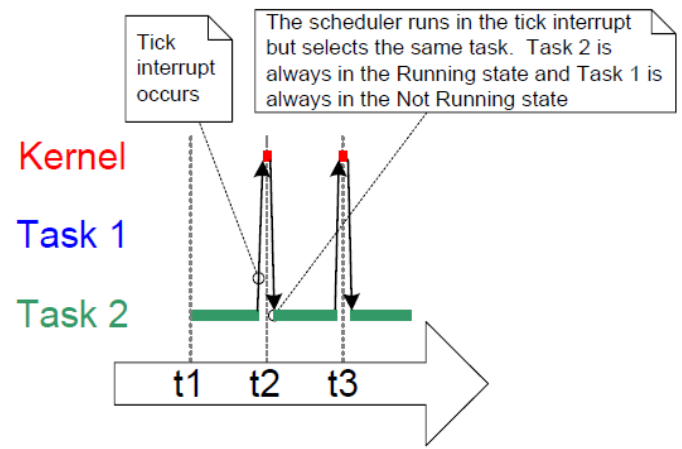


Figure 15. Full task state machine



Referência

- **Miscellaneous System APIs**

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/misc_system_api.html

- https://freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf

- https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf