UTFPR

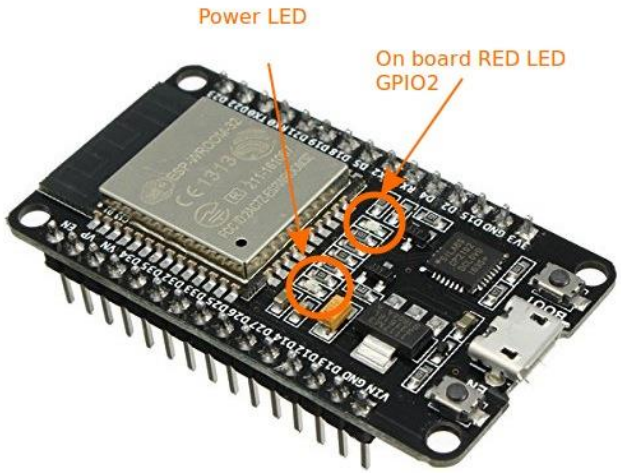
- FreeRTOS

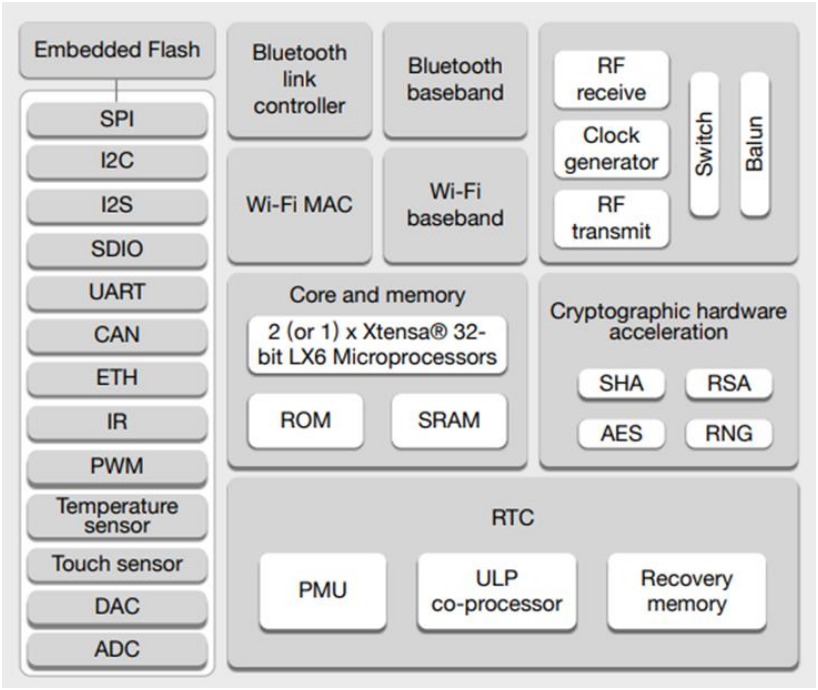
Microcontroladores: (ELF74)
Prof: DaLuz

FreeRTOS (Mini Curso)

- FreeRTOS

Hardware:



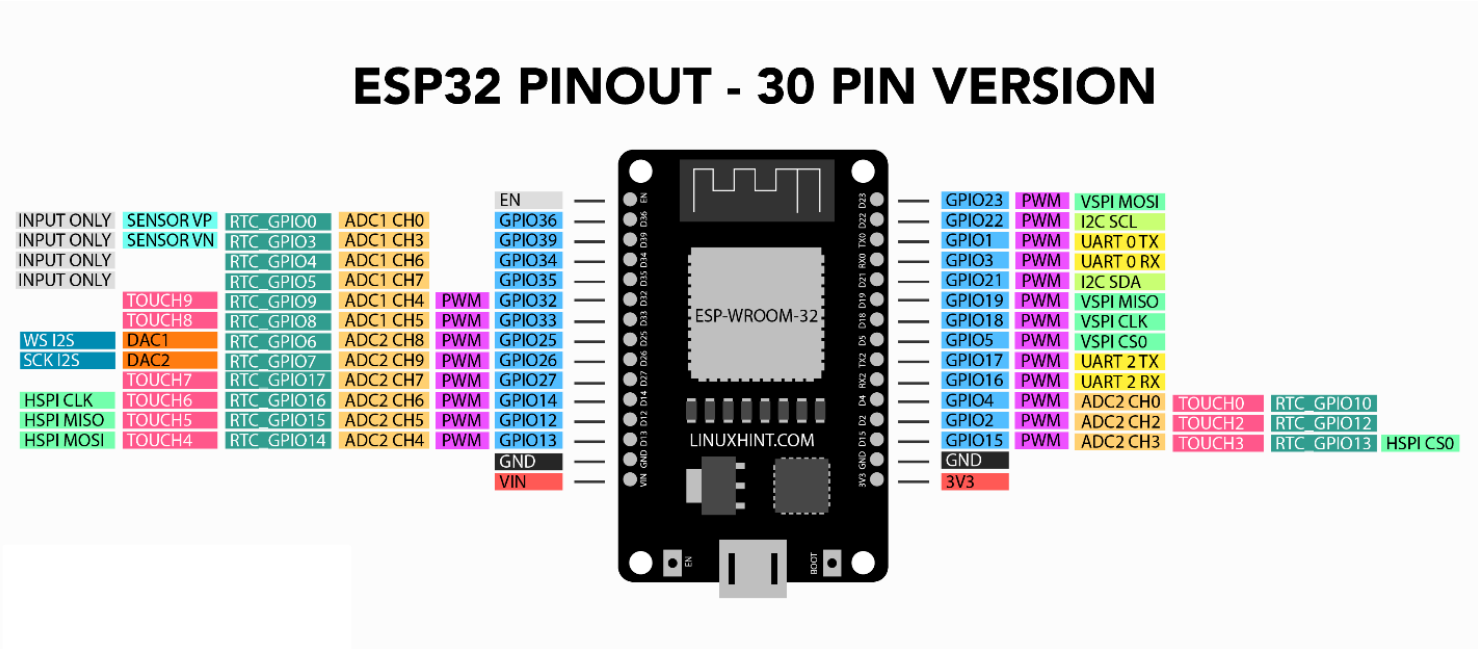


UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (Kit ESP32-Doit 30Pin):

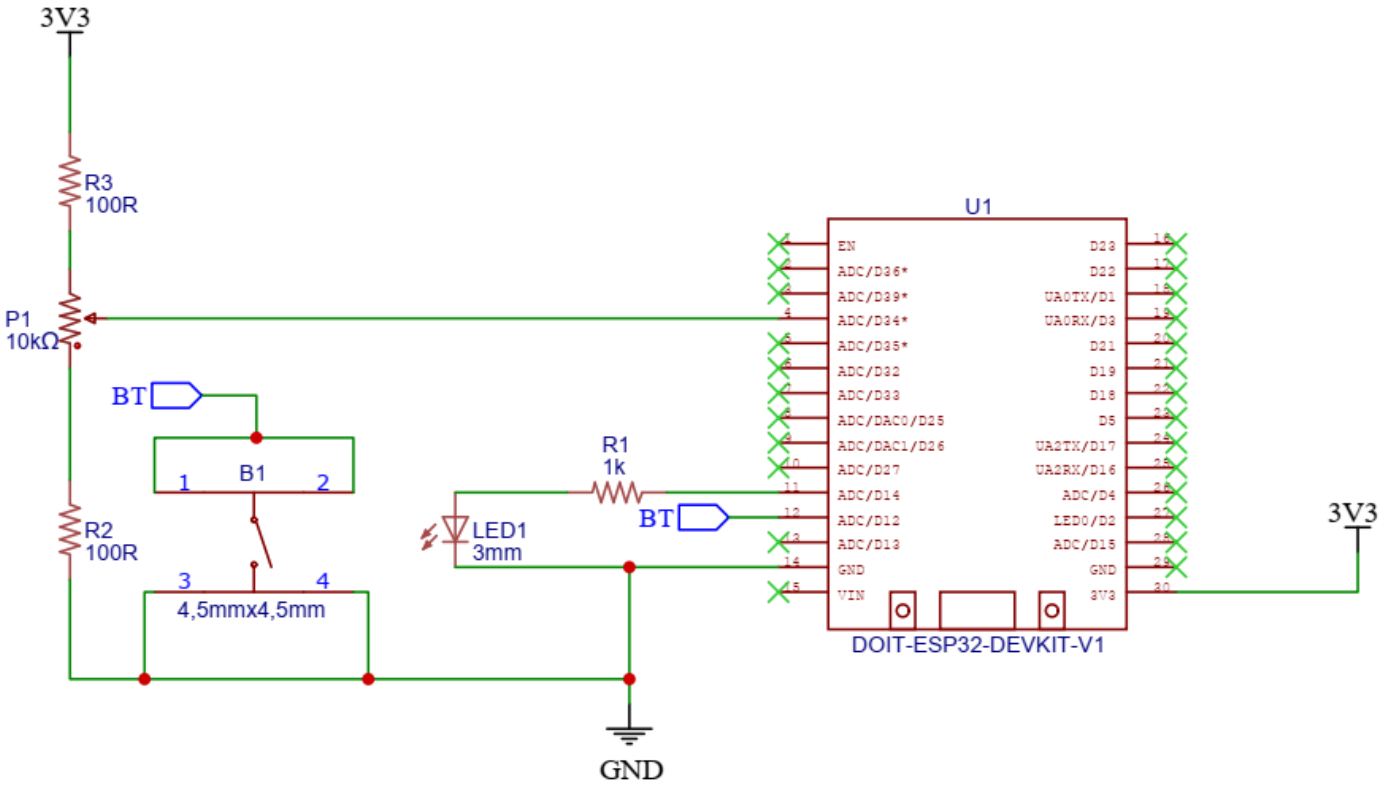


UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

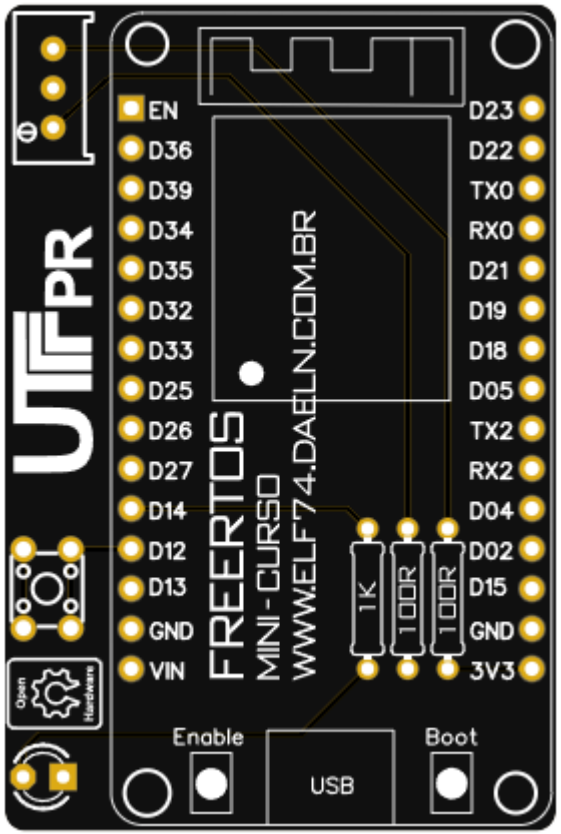
Hardware (Esquemático):



- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (PCB):



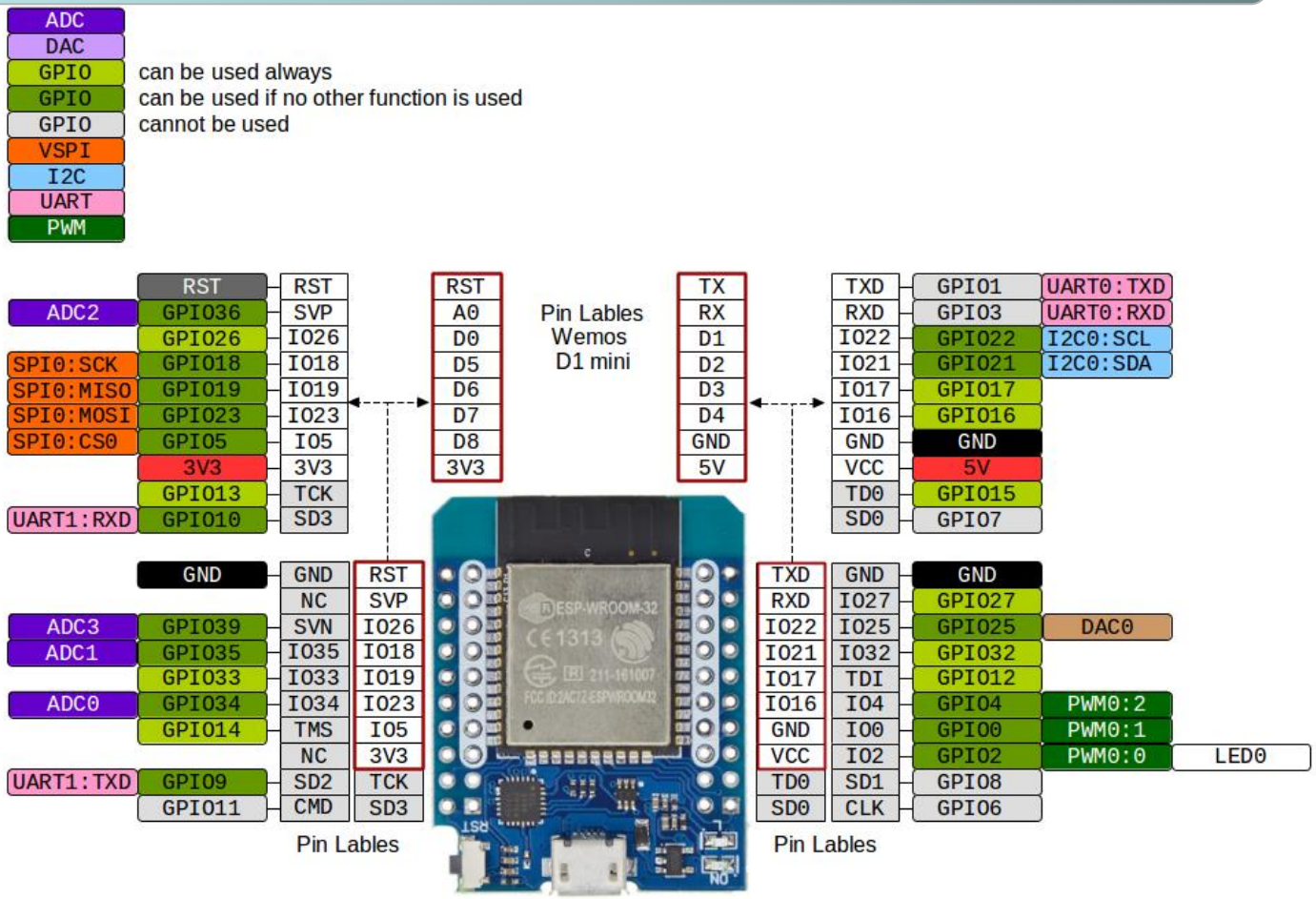
https://oshwlab.com/garcez/freertos_minipcb

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

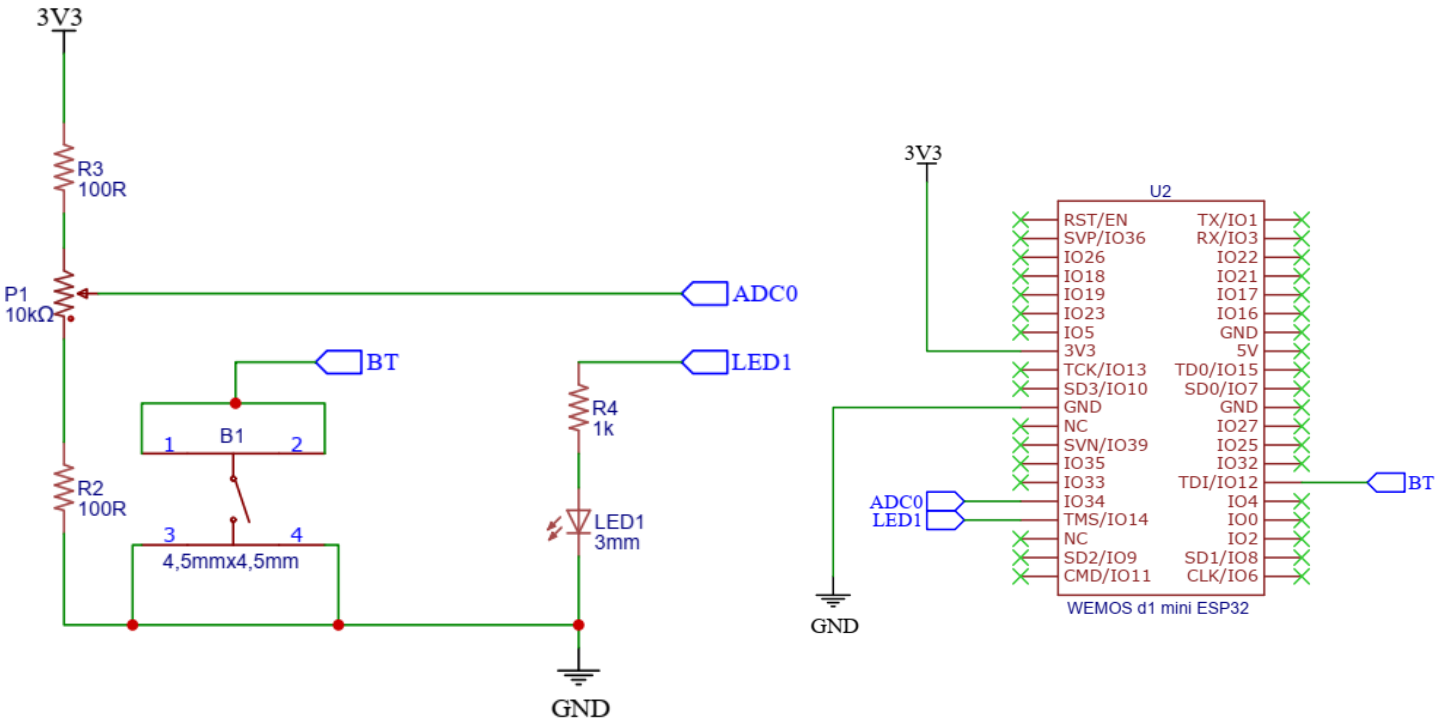
Hardware (Kit ESP2 Wemos D1 Mini):



- FreeRTOS

FreeRTOS (Mini Curso)

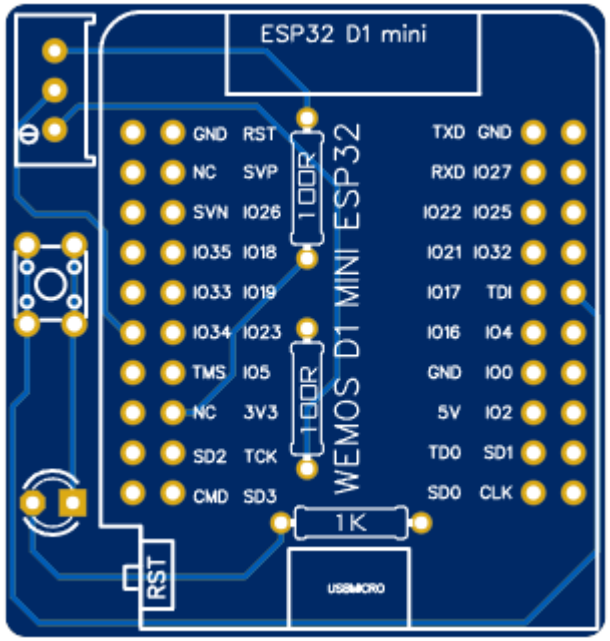
Hardware (Esquemático):

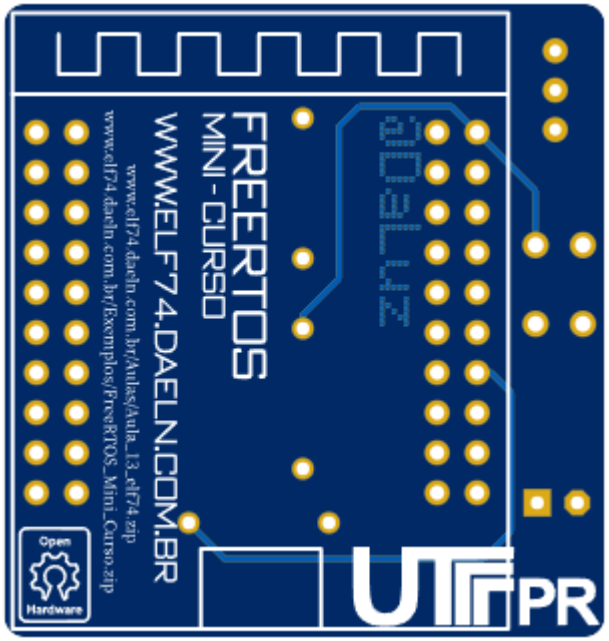


- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (PCB):





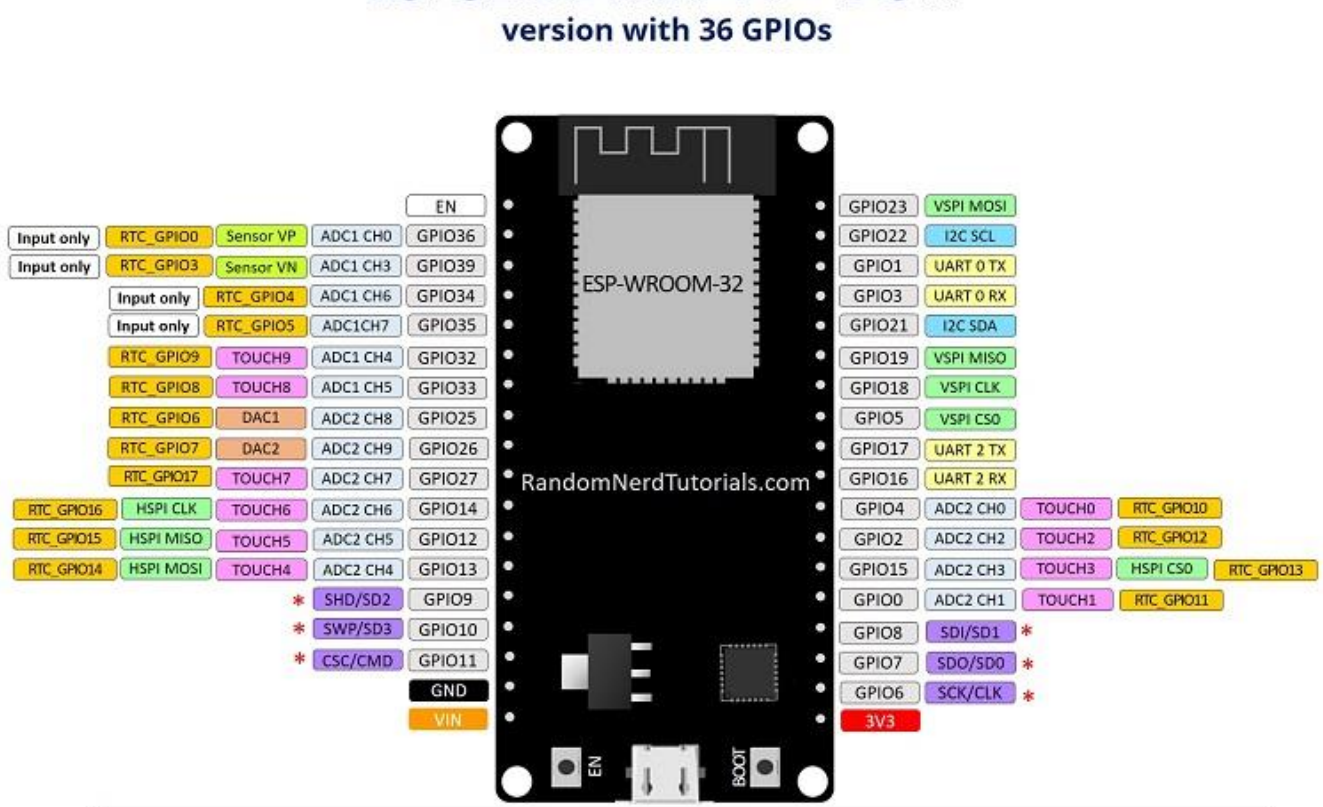
https://oshwlab.com/garcez/freertos_minipcb_wemos

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (Kit SP32-Doit 36Pin):



UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (Esquemático):

Hardware (PCB):

Em Produção ...

UTFPR

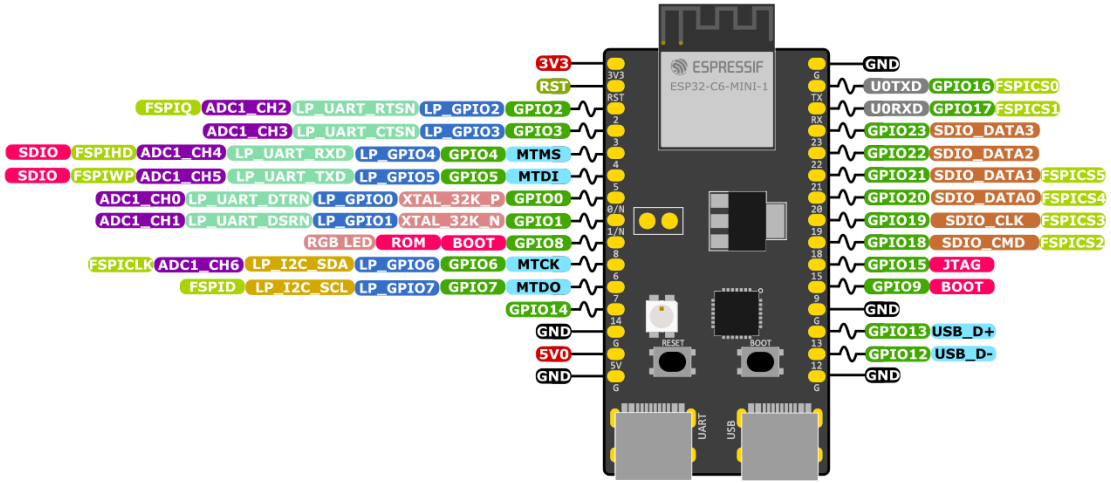
- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (Kit ESP32-C6 RISC-V):

ESP32-C6-DevKitM-1

ESPRESSIF



ESP32-C6 Specs
32-bit RISC-V single-core @160 MHz
Wi-Fi IEEE 802.11 ax 2.4 GHz + Bluetooth LE 5 + IEEE 802.15.4 (Zigbee and Thread)
512 KB SRAM (21 KB for cache)
320 KB ROM
30 or 22 GPIOs, 3x SPI, 2x UART, 1x I2C, RMT
LED PWM 6ch, 1x 12-bit ADC with 7ch, TWAI®
USB Serial/JTAG, ETM, MCPWM, SDIO Slave

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Hardware (Esquemático):

Hardware (PCB):

Em Produção ...

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Software:

DownLoad:
https://code.visualstudio.com/download

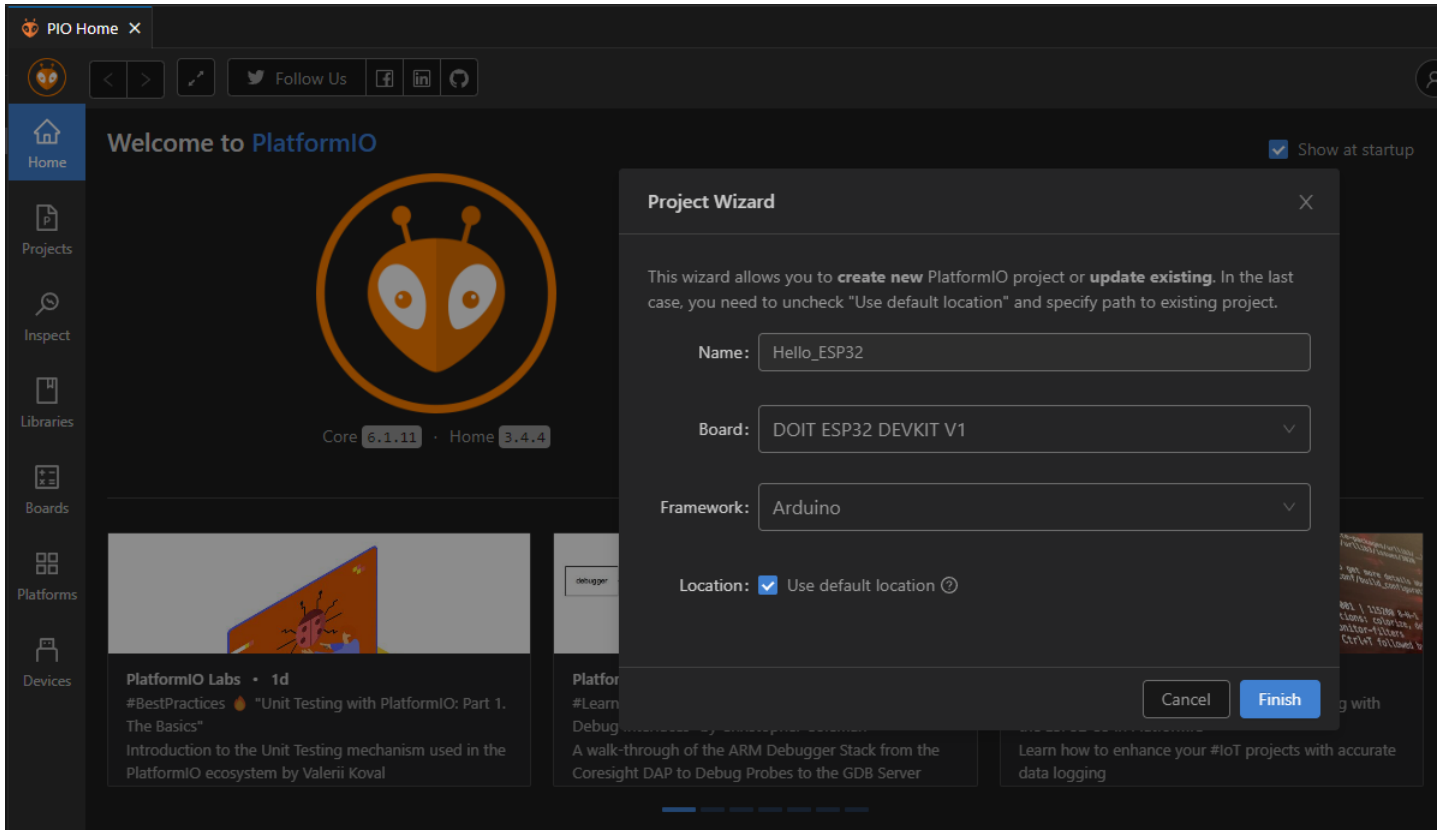
DownLoad: **(Obs: Usar Extensão do VSCode)**
https://platformio.org/

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Software:

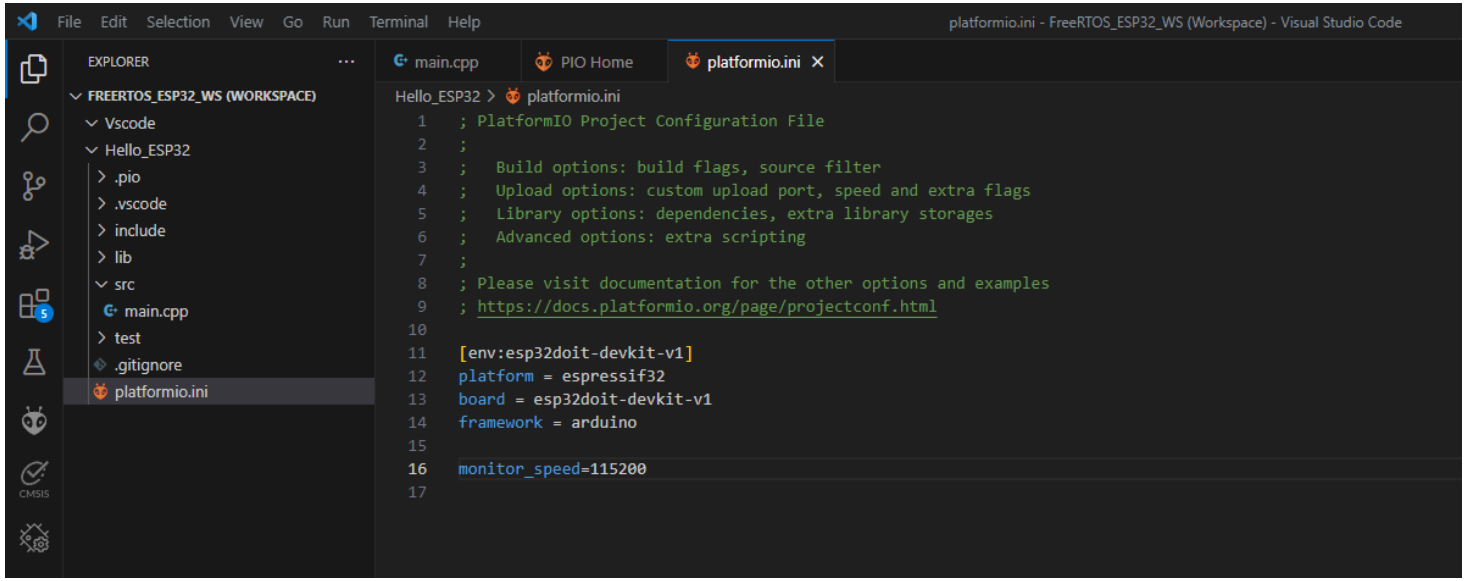


UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Software:



UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Software:

main.cpp | PIO Home | platformio.ini

Hello_ESP32 > src > main.cpp > ...

```
/*#########################################################################
# Exemplo Hello World - ESP32 DOIT
# Pisca o LED da placa em intervalos de 1 segundo (500ms ON / 500ms OFF)
# Exemplo para entender o fluxo de criação de um projeto no Platformio com VSCode
# Por: Prof. Dr. Paulo Denis Garcez da Luz
# http://www.elf74.daeln.com.br
#########################################################################*/

//Inclusão de biblioteca Arduino
#include <Arduino.h>

/*mapeamento de pinos*/
#define LEDBUILTIN 2

//Função setup
void setup()
{
  pinMode(LEDBUILTIN,OUTPUT);               //configura pino do LED como saída
  Serial.begin(115200);                     //configura comunicação serial com baudarate de 115200 bps
}

//Função Loop
void loop()
{
  digitalWrite(LEDBUILTIN,HIGH);            //liga LED
  delay(500);                               //espera 0,5 s
  digitalWrite(LEDBUILTIN,LOW);             //desliga LED
  delay(500);                               //espera 0,5 s
  Serial.println("Hello World ESP32-DOIT"); // imprime mensagem na serial
}
```

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Convenções:

- Variáveis ***unsigned*** começam com "**u**".
- Variáveis do tipo ***char*** (8 bits) começam com "**c**".
- Variáveis do tipo ***short*** (16 bits) começam com "**s**".
- Variáveis do tipo ***long*** (32 bits) começam com "**l**".
- Ponteiros começam com "**p**".
- Funções privadas em um arquivo começam com "**prv**".
- As funções da API são prefixadas com seu tipo de retorno, conforme a convenção definida para variáveis, com a adição do prefixo "**v**" de "**void**".
- Os nomes das funções da API começam com o nome do arquivo no qual estão definidos. Por exemplo "**vTaskDelete"** é definido em "**tasks.c**" e possui um tipo de retorno "**void**".

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Convenções:

- Indentação:
- **TAB** é usado para identação. Um **TAB** é igual a 4 espações.

- Comentários:
- Os comentários nunca passam da coluna 80, a menos que sigam e descrevam um parâmetro.
- Comentários de barra dupla no estilo C++ (//) não são usados.

- *Layout*:
- O *layout* do código-fonte do **FreeRTOS** foi projetado para ser o mais fácil de visualizar e ler quanto possível.

```
/* Booleans use the microcontroller architecture's most efficient type, which
is defined in the kernel's portable layer as BaseType_t. */
BaseType_t xSchedulerRunning = pdFALSE;

/* Function names are always written on a single line, including the return
type.  As always, there is no space before the opening parenthesis.  There
is a space after an opening parenthesis.  There is a space before a closing
parenthesis.  There is a space after each comma.  Parameters are given
verbose, descriptive names (unlike this example!).  The opening and closing
curly brackets appear on their own lines, lined up underneath each other. */
void vAnExampleFunction( uint32_t ulParameter1, uint16_t usParameter2 )
{
/* Variable declarations are not indented, and use stdint.h defined types. */
uint8_t ucByte;

    /* Code is indented.  Curly brackets are always on their own lines
    and lined up underneath each other. */
    for( ucByte = 0U; ucByte < fileBUFFER_LENGTH; ucByte++ )
    {
        /* Indent again. */
    }
}

/* For, while, do and if constructs follow a similar pattern.  There is no
space before the opening parenthesis.  There is a space after an opening
parenthesis.  There is a space before a closing parenthesis.  There is a
space after each semicolon (if there are any).  There are spaces before and
after each operator.  No reliance is placed on operator precedence -
parenthesis are always used to make precedence explicit.  Magic numbers,
other than zero, are always replaced with a constant or #defined constant.
The opening and closing curly brackets appear on their own lines. */
for( ucByte = 0U; ucByte < fileBUFFER_LENGTH; ucByte++ )
{
```

https://www.freertos.org/FreeRTOS-Coding-Standard-and-Style-Guide.html#NamingConventions

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

FreeRTOS API:

Task Creation	xTaskCreate xTaskCreateStatic vTaskDelete xTaskGetStaticBuffers
Task Control	vTaskDelay vTaskDelayUntil xTaskDelayUntil uxTaskPriorityGet uxTaskPriorityGetFromISR uxTaskBasePriorityGet uxTaskBasePriorityGetFromISR vTaskPrioritySet vTaskSuspend vTaskResume xTaskResumeFromISR xTaskAbortDelay

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

FreeRTOS API:

Task Utilities	
	uxTaskGetSystemState vTaskGetInfo xTaskGetCurrentTaskHandle xTaskGetIdleTaskHandle uxTaskGetStackHighWaterMark eTaskGetState pcTaskGetName xTaskGetHandle xTaskGetTickCount xTaskGetTickCountFromISR xTaskGetSchedulerState uxTaskGetNumberOfTasks vTaskList vTaskListTasks vTaskStartTrace ulTaskEndTrace vTaskGetRunTimeStats vTaskGetRunTimeStatistics vTaskGetIdleRunTimeCounter ulTaskGetRunTimeCounter ulTaskGetRunTimePercent ulTaskGetIdleRunTimeCounter ulTaskGetIdleRunTimePercent vTaskSetApplicationTaskTag xTaskGetApplicationTaskTag xTaskCallApplicationTaskHook pvTaskGetThreadLocalStoragePointer vTaskSetThreadLocalStoragePointer vTaskSetTimeOutState xTaskCheckForTimeOut

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

FreeRTOS API:

Kernel Control	taskYIELD taskENTER_CRITICAL taskEXIT_CRITICAL taskENTER_CRITICAL_FROM_ISR taskEXIT_CRITICAL_FROM_ISR taskDISABLE_INTERRUPTS taskENABLE_INTERRUPTS vTaskStartScheduler vTaskEndScheduler vTaskSuspendAll xTaskResumeAll vTaskStepTick

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

FreeRTOS API:

Queue Management	xQueueCreate xQueueCreateStatic vQueueDelete xQueueSend xQueueSendFromISR xQueueSendToBack xQueueSendToBackFromISR xQueueSendToFront xQueueSendToFrontFromISR xQueueReceive xQueueReceiveFromISR uxQueueMessagesWaiting uxQueueMessagesWaitingFromISR uxQueueSpacesAvailable xQueueReset xQueuePeek xQueuePeekFromISR vQueueAddToRegistry pcQueueGetName vQueueUnregisterQueue xQueueIsQueueEmptyFromISR xQueueIsQueueFullFromISR xQueueOverwrite xQueueOverwriteFromISR xQueueGetStaticBuffers

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

FreeRTOS API:

Semaphores	xSemaphoreCreateBinary xSemaphoreCreateBinaryStatic vSemaphoreCreateBinary [use xSemaphoreCreateBinary() for new designs] xSemaphoreCreateCounting xSemaphoreCreateCountingStatic xSemaphoreCreateMutex xSemaphoreCreateMutexStatic xSemaphoreCreateRecursiveMutex xSemaphoreCreateRecursiveMutexStatic vSemaphoreDelete xSemaphoreGetMutexHolder xSemaphoreTake xSemaphoreTakeFromISR xSemaphoreTakeRecursive xSemaphoreGive xSemaphoreGiveRecursive xSemaphoreGiveFromISR uxSemaphoreGetCount xSemaphoreGetStaticBuffer

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

FreeRTOS API:

Software Timers	xTimerCreate xTimerCreateStatic xTimerIsTimerActive pvTimerGetTimerID pcTimerGetName vTimerSetReloadMode xTimerStart xTimerStop xTimerChangePeriod xTimerDelete xTimerReset xTimerStartFromISR xTimerStopFromISR xTimerChangePeriodFromISR xTimerResetFromISR pvTimerGetTimerID vTimerSetTimerID xTimerGetTimerDaemonTaskHandle xTimerPendFunctionCall xTimerPendFunctionCallFromISR pcTimerGetName xTimerGetPeriod xTimerGetExpiryTime xTimerGetReloadMode

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

Tarefas:

Protótipo de uma Tarefa:

```
void vTaskCode( void * pvParameters );
```

Corpo de uma Tarefa:

```
/* Task to be created. */
void vTaskCode( void * pvParameters )
{
    for( ;; )
    {
        /* Task code goes here. */
    }
}
```

task. h

```
BaseType_t xTaskCreate(TaskFunction_t pvTaskCode,
                       const char * const pcName,
                       configSTACK_DEPTH_TYPE usStackDepth,
                       void *pvParameters,
                       UBaseType_t uxPriority,
                       TaskHandle_t *pxCreatedTask);
```

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EXEMPLOS:

DownLoad:
http://www.elf74.daeln.com.br/Exemplos/FreeRTOS_Mini_Curso.zip

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX01:

Task1:	Led Blink de 200ms
Task2:	Imprime contador de tempo no monitor serial a casa 1s.

Funções utilizadas:	**API Reference:**
void vTaskDelay(const TickType_t xTicksToDelay);	Link URL
Macro: pdMS_TO_TICKS(xTimeInMs);	#define pdMS_TO_TICKS(xTimeInMs) ((TickType_t) (((TickType_t) (xTimeInMs) * (TickType_t) configTICK_RATE_HZ) / (TickType_t) 1000))

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX02:

Task1:	Led Blink de 200ms
Task2:	Imprime contador de tempo no monitor serial a casa 1s. Quando o contador chegar em 11 deletamos a Task1. E quando o contador chegar em 17 deletamos a Task2.

Funções utilizadas:	**API Reference:**
void vTaskDelete(TaskHandle_t xTask);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX03:

Task1:	Led Blink de 200ms
Task2:	Imprime contador de tempo no monitor serial a casa 1s. Quando o contador chegar em 11 suspendemos a Task1. E quando o contador chegar em 17 reiniciamos a Task1.

Funções utilizadas:	**API Reference:**
void vTaskSuspend(TaskHandle_t xTaskToSuspend);	Link URL
void vTaskResume(TaskHandle_t xTaskToResume);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX04:

Task1:	Led Blink de 200ms com passagem de parâmetro (pino) de inicialização da task1.
Task2:	Imprime contador de tempo no monitor serial a casa 1s. O valor inicial de contagem é passado como parâmetro na inicialização da task2.

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

ESP32-IDF -> FrameWork -> Arduino

C:\Users\xxxxxxx\.platformio\packages\framework-arduinoespressif32\cores\main.cpp

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_task_wdt.h"
#include "Arduino.h"

TaskHandle_t loopTaskHandle = NULL;

bool loopTaskWDTEnabled;

void loopTask(void *pvParameters)
{
  setup();
  for (;;)
      {
        if (loopTaskWDTEnabled) esp_task_wdt_reset();
        loop();
        if (serialEventRun) serialEventRun();
      }
}

extern "C" void app_main()
{
  loopTaskWDTEnabled = false;
  initArduino();
  xTaskCreateUniversal(loopTask, "loopTask", getArduinoLoopTaskStackSize(), NULL, 1,
                       &loopTaskHandle, ARDUINO_RUNNING_CORE);
}
```

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX05:

Task1:	Led Blink de 200ms com passagem de parâmetro (pino led_built_in) de inicialização da task1. Utilizando o core principal.
Task2:	Imprime o contador de tempo no monitor serial a casa 1s. O valor inicial de contagem é passado como parâmetro na inicialização da task2. Utilizando o core secundário.
Task3:	Led Blink de 500ms com passagem de parâmetro (pino led_pcb_ext1) de inicialização da task1. Utilizando o core principal.

Funções utilizadas:	**API Reference:**
void xTaskCreatePinnedToCore (...);	Função fornecida pelo fabricante, não é da API do FreeRTOS

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX06:

Task1:	Led Blink de 200ms com passagem de parâmetro (pino led_built_in) de inicialização da task1. Utilizando o core principal.
Task2:	Imprime o contador de tempo no monitor serial a casa 1s e o tamanho de sua Stack. O valor inicial de contagem é passado como parâmetro na inicialização da task2. Utilizando o core secundário.
Task3:	Led Blink de 500ms com passagem de parâmetro (pino led_pcb_ext1) de inicialização da task1. Utilizando o core principal.

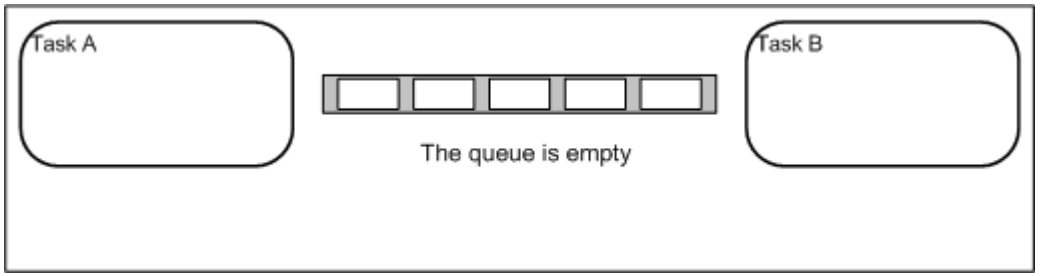
Funções utilizadas:	**API Reference:**
UBaseType_t uxTaskGetStackHighWaterMark (TaskHandle_t xTask);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX07:



Task1:	Coloca inteiros na Fila a cada 500ms até a o contador chegar em 10. Quando o contador chega em 10 a tarefa entra em espera de 5s.
Task2:	Retira os inteiros da fila e mostra no monitor serial. Quando a tarefa1 está em espera exibe um TimeOut.

Funções utilizadas:	**API Reference:**
QueueHandle_t xQueueCreate(UBaseType_t uxQueueLength, UBaseType_t uxItemSize); BaseType_t xQueueReceive(QueueHandle_t xQueue, void *pvBuffer, TickType_t xTicksToWait);	Link URL Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX08:

Task0: (loop)	Executa o Led Blinky a 333ms.
Task1:	Retira da fila quantas vezes o botão foi pressionado e envia pela serial o valor.
ISR:	Interrupção que pega o pressionamento do botão, sem tratamento do repique. Quando em execução coloca o número de vezes que o botão foi pressionado na fila.

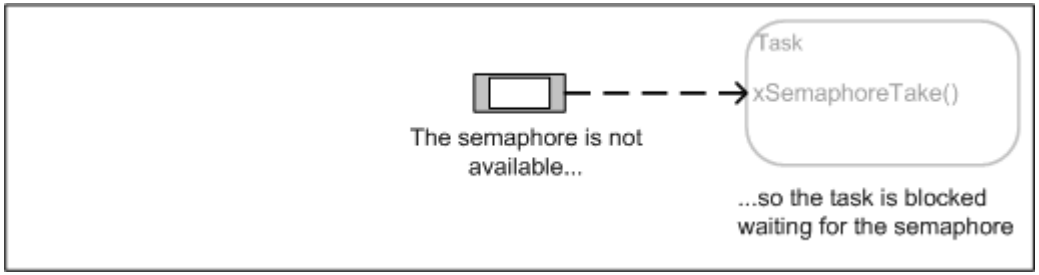
Funções utilizadas:	**API Reference:**
BaseType_t xQueueSendFromISR (QueueHandle_t xQueue, const void *pvItemToQueue, BaseType_t *pxHigherPriorityTaskWoken);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX09:



Task0: (loop)	Inverte o estado do Led a cada 2,5s e libera o semáforo binário neste mesmo intervalo.
Task1:	Quando adquire o semáforo binário, executa uma leitura analógica do potenciômetro e envia esse dado pela serial, voltando a aguardar a liberação do semáforo binário.

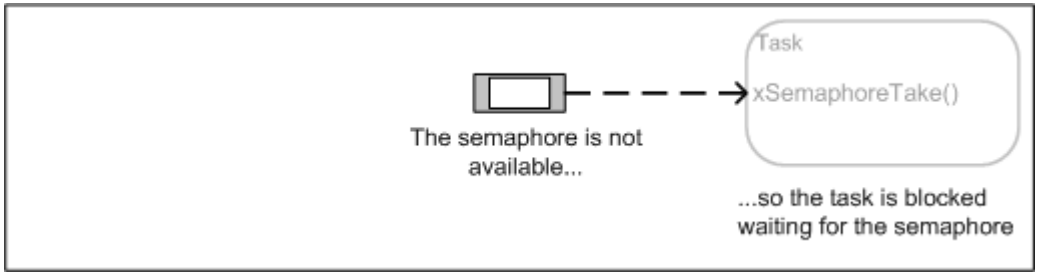
Funções utilizadas:	**API Reference:**
SemaphoreHandle_t xSemaphoreCreateBinary(void);	Link URL
xSemaphoreGive(SemaphoreHandle_t xSemaphore);	Link URL
xSemaphoreTake(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX10:



Task0: (loop)	Pisca o Led (100ms=ON e 900ms=OFF)
Task1:	Aguarda o semáforo binário ser liberado e informa o número de vezes que o botão foi pressionado via ISR do botão.
ISR:	Interrupção que pega o pressionamento do botão, sem tratamento do repique. Quando em execução libera o semáforo e força a troca de contexto.

Funções utilizadas:	**API Reference:**
xSemaphoreGiveFromISR(SemaphoreHandle_t xSemaphore, signed BaseType_t *pxHigherPriorityTaskWoken)	Link URL
xTaskNotifyAndQueryFromISR / xTaskNotifyAndQueryIndexedFromISR	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX11:

Task0: (loop)	Pisca o Led (200ms=ON e 800ms=OFF)
Task1:	Aguarda o semáforo contador ter eventos e informa o número de vezes que o botão foi pressionado via ISR do botão, tendo um delay(500ms) de espera entre os tratamentos.
ISR:	Interrupção que pega o pressionamento do botão, sem tratamento do repique. Quando em execução libera o semáforo contador e força a troca de contexto.

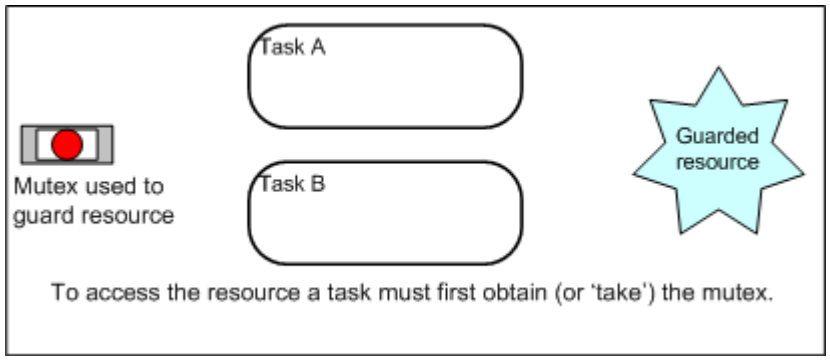
Funções utilizadas:	**API Reference:**
SemaphoreHandle_t xSemaphoreCreateCounting(UBaseType_t uxMaxCount, UBaseType_t uxInitialCount);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX12:



Task0: (loop)	Pisca o Led (300ms=ON e 700ms=OFF)
Task1 (prioridade 1):	Aguarda pelo Mutex e informa o número da Task pela serial através de uma função. Congela por 2s a Task e libera para troca de contexto (100ms).
Task2 (prioridade 4): *Obs: recompilar com a linha do vTaskDelay(pdMS_TO_TICKS(100)); comentada e verificar o que acontece ...	Aguarda pelo Mutex e informa o número da Task pela serial através de uma função. Congela por 500ms a Task e libera para troca de contexto (100ms).

Funções utilizadas:	**API Reference:**
SemaphoreHandle_t xSemaphoreCreateMutex(void)	Link_URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX13:

Task0: (loop)	Idle ...
Task1:	Faz *pooling* no botão com tratamento do repique, caso o botão seja pressionado ativa o timer2, desativa o timer1 e liga o Led da pcb.
vFTimer1:	Função de *callback* do S.T.1: Led_*built-in blinky* de 750ms. (*auto-reload)*
vFTimer2:	Função de *callback* do S.T.2: reativa o S.T.1 e desliga o Led da pcb. (*one-shot*)

Funções utilizadas:	**API Reference:**
TimerHandle_t xTimerCreate (const char * const pcTimerName, const TickType_t xTimerPeriod, const UBaseType_t uxAutoReload, void * const pvTimerID, TimerCallbackFunction_t pxCallbackFunction);	Link URL
BaseType_t xTimerStart(TimerHandle_t xTimer, TickType_t xBlockTime);	Link URL
BaseType_t xTimerStop(TimerHandle_t xTimer, TickType_t xBlockTime)	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX14:

Task0: (loop)	Pisca o Led (300ms=ON e 700ms=OFF)
Task1:	Aguarda sinalização da FLAG do grupo de eventos no bit 0, enviando pela serial o evento, após o S.T.1 ativar a FLAG.
Task2:	Aguarda sinalização da FLAG do grupo de eventos no bit 1, enviando pela serial o evento, após o S.T.1 ativar a FLAG.
vFTimer1:	Ativa as FLAGS ... Com 3s, 6s e 9s.

Funções utilizadas:	**API Reference:**
EventGroupHandle_t xEventGroupCreate(void);	Link URL
EventBits_t xEventGroupWaitBits (const EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToWaitFor, const BaseType_t xClearOnExit, const BaseType_t xWaitForAllBits, TickType_t xTicksToWait);	Link URL
EventBits_t xEventGroupSetBits(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToSet);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX15:

Task0: (loop)	Led Blinky (250ms=ON e 250ms=OFF)
Task1:	Aguarda uma notificação da ISR para mudar o estado do Led e enviar pela serial a notificação de tratamento.
ISR:	ISR do pino do PushButton (não faz tratamento do repique), usa Task Notification como sinalizador de eventos do botão para a Task1

Funções utilizadas:	**API Reference:**
void vTaskNotifyGiveFromISR(TaskHandle_t xTaskToNotify, BaseType_t *pxHigherPriorityTaskWoken); void vTaskNotifyGiveIndexedFromISR(TaskHandle_t xTaskHandle, UBaseType_t uxIndexToNotify, BaseType_t *pxHigherPriorityTaskWoken);	Link URL

UTFPR

- FreeRTOS

FreeRTOS (Mini Curso)

EX16:

Task0: (loop)	Idle ...
Task1:	Led Blinky de 1s com passagem de struct como parâmetro durante a criação da Task1
Task2:	Led Blinky de 200ms com passagem de struct como parâmetro durante a criação da Task2

UTFPR

- FreeRTOS

Referências:

Continuação dos Labs e do Projeto Final ...

* *Refs* ↔ freertos.org, *microntrollerslab.com, jblopen.com, microsoft.com, udemy.com, embarcados.com.br.*