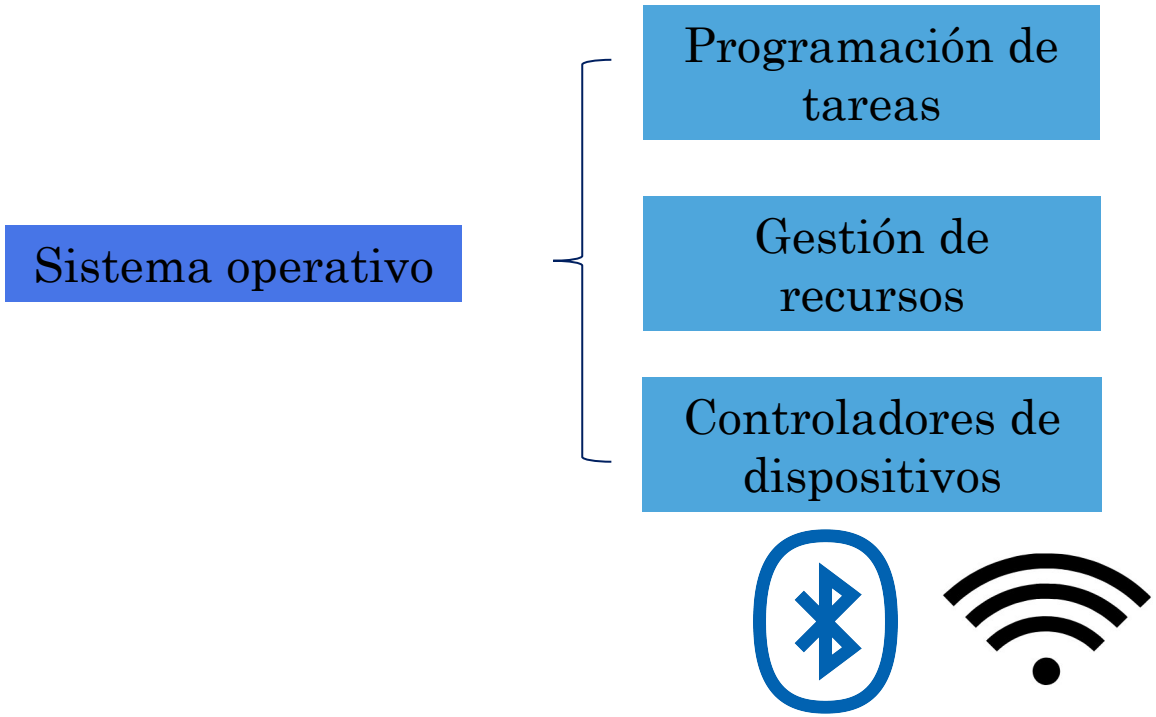
Sistema operativo

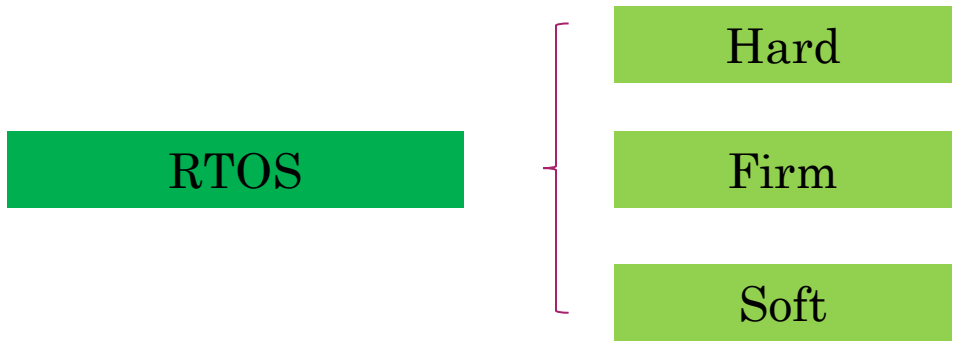


Real-time operating system

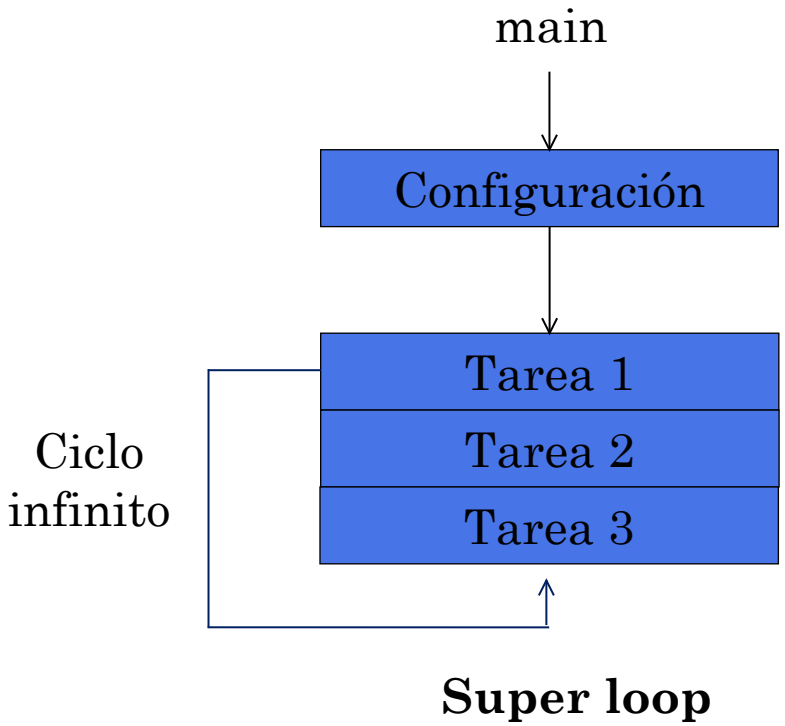
RTOS

- Diseñado para manejar **tareas críticas** en términos de tiempo con precisión y eficiencia.
- Se centra en respuestas en tiempo real y tiempos precisos.
- Prioriza las tareas críticas para garantizar que se ejecuten dentro de plazos estrictos.
- Comportamiento **determinista**.
- Manejo eficiente de interrupciones.

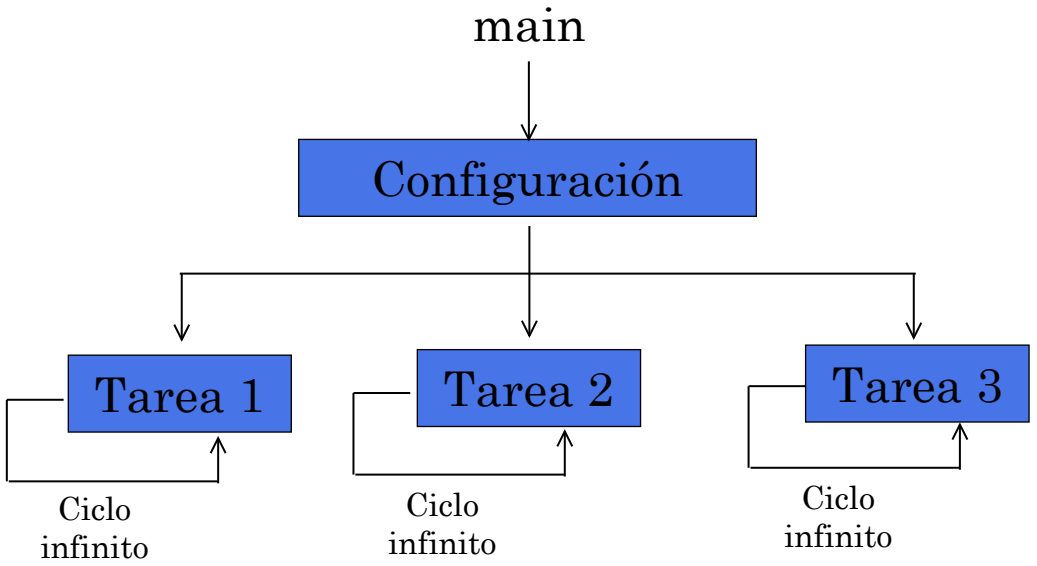
Real-time operating system



Bare-metal y RTOS en microcontroladores

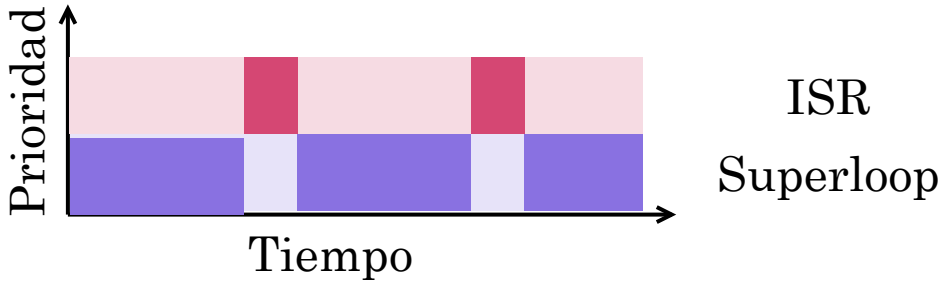


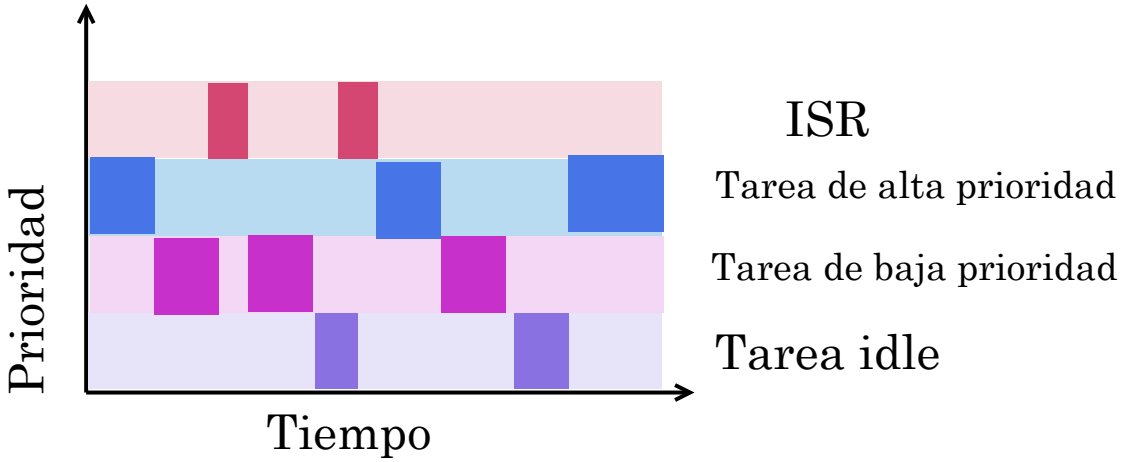
Super loop



RTOS

Bare-metal y RTOS en microcontroladores

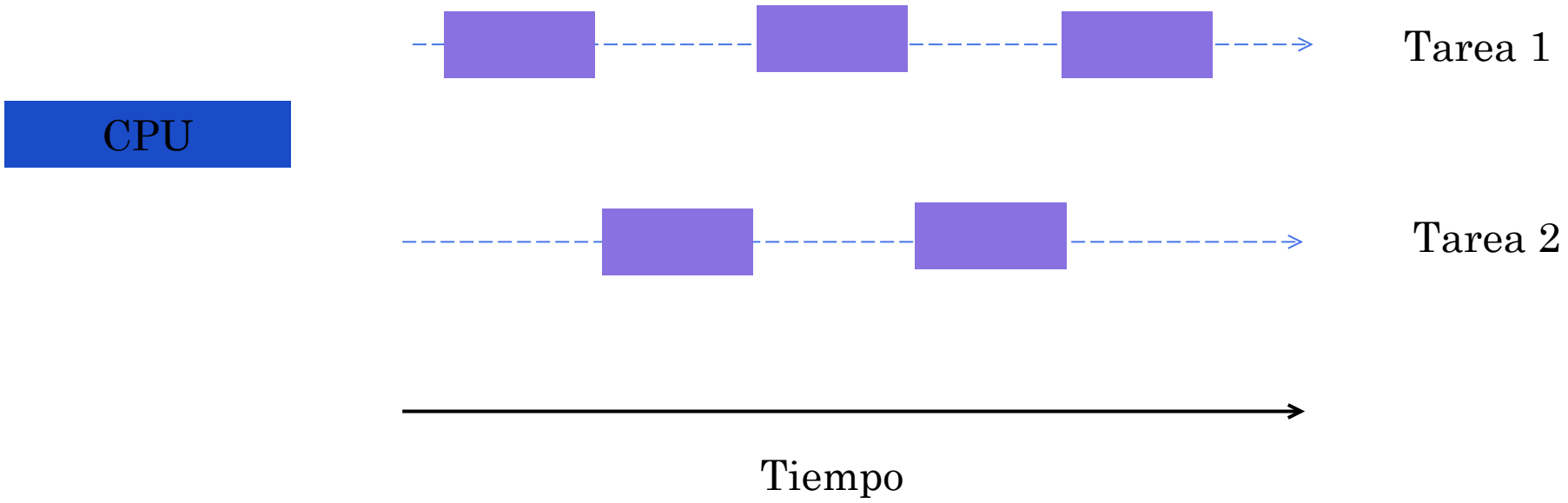




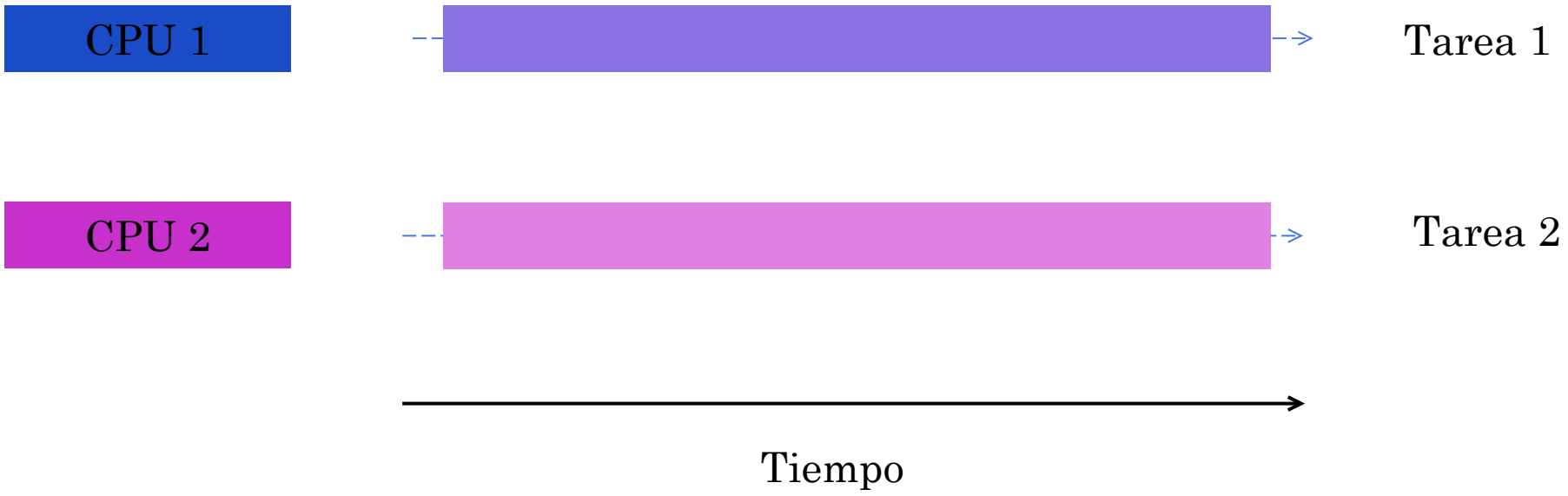
FreeRTOS

Procesador principal:
Tensilica Xtensa 32-bit LX6 microprocessor de uno o dos cores dependiendo la serie.

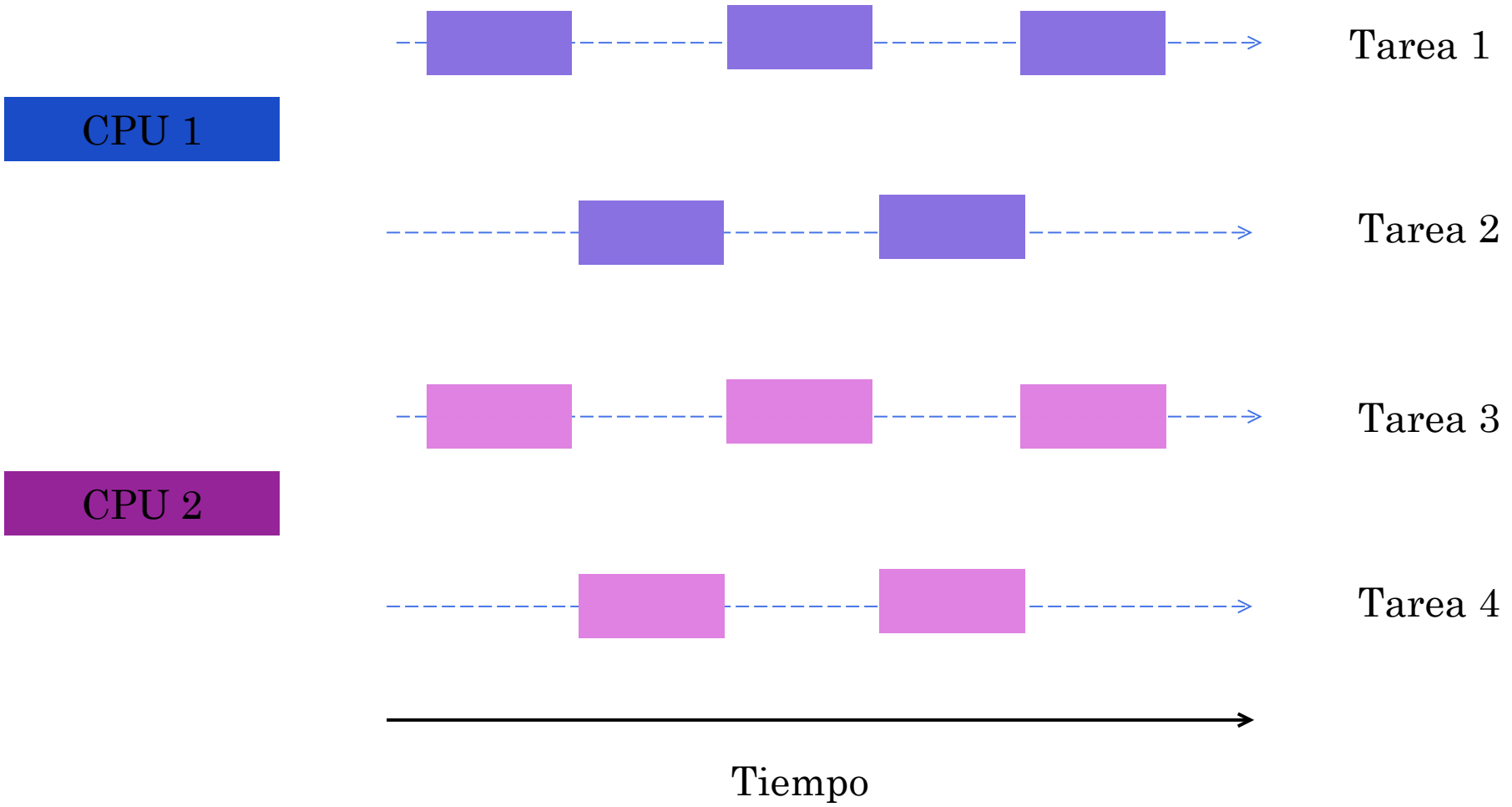
Ejecución concurrente



Ejecución paralela



Ejecución paralela y concurrente



Crear tareas

```c
#include <stdio.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include "esp_timer.h"
```

```c
TaskHandle_t handleTarea1 = NULL;
TaskHandle_t handleTarea2 = NULL;
```

```c
void app_main(void)
{
  printf("configMAX_PRIORITIES: %d\n",configMAX_PRIORITIES);
  xTaskCreate(tarea1, "tarea1", 4096, NULL, 10, &handleTarea1);
  xTaskCreate(tarea2, "tarea2", 4096, NULL, 10, &handleTarea2);
}
```

https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos_idf.html?highlight=xtaskcreate#_CPPv411xTaskCreate14TaskFunction_tPCKcK22configSTACK_DEPTH_TYPEPCv11UBaseType_tPC12TaskHandle_t

```
xTaskCreate(tarea1, "tarea1", 4096, NULL, 10, &handleTarea1);
```

```
BaseType_t xTaskCreate(
    TaskFunction_t pvTaskCode,   /* Puntero a la función que se ejecutará como la tarea */
    const char * const pcName,   /* Nombre opcional que puedes asignar a la tarea para
                                    propósitos de depuración */
    const uint32_t usStackDepth, /* Tamaño de la pila en bytes */
    void *pvParameters,          /* Puntero a datos que deseas pasar a la tarea */
    UBaseType_t uxPriority,      /* Prioridad de la tarea */
    TaskHandle_t *pxCreatedTask  /* Puntero para devolver el handle de la tarea */
);
```

Tenga en cuenta que el parámetro **pvParameters** debe existir durante la vida útil de la tarea, por lo que se declara estático o global. Si fuera solo una variable de pila automática, es posible que ya no exista, o al menos que se haya dañado, para cuando la nueva tarea intente acceder a ella.

uxPriority

- **Prioridad mínima**:

La prioridad más baja es 0 y se define en la macro **tskIDLE_PRIORITY**. Usualmente la tarea idle tiene está prioridad.

- **Prioridad máxima**: (configMAX_PRIORITIES - 1):

En FreeRTOS se puede configurar la prioridad máxima por medio de la macro **configMAX_PRIORITIES**. La prioridad máxima es (**configMAX_PRIORITIES - 1**).

```c
void tarea1(void *arg)
{
  uint8_t cnt = 0;
  while (1) {
    printf("Tarea 1 corriendo. Cnt: %d. Tiempo: %lld\n", cnt++, esp_timer_get_time());
    vTaskDelay(500 / portTICK_PERIOD_MS);
  }
}

void tarea2(void *arg)
{
  uint8_t cnt = 0;
  while (1) {
    printf("Tarea 2 corriendo. Cnt: %d. Tiempo: %lld\n", cnt++, esp_timer_get_time());
    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
```

```
configMAX_PRIORITIES: 25
Tarea 1 corriendo. Cnt: 0. Tiempo: 307194
Tarea 2 corriendo. Cnt: 0. Tiempo: 308233
Tarea 1 corriendo. Cnt: 1. Tiempo: 805601
Tarea 2 corriendo. Cnt: 1. Tiempo: 1305602
Tarea 1 corriendo. Cnt: 2. Tiempo: 1305684
Tarea 1 corriendo. Cnt: 3. Tiempo: 1805601
Tarea 2 corriendo. Cnt: 2. Tiempo: 2305602
Tarea 1 corriendo. Cnt: 4. Tiempo: 2305684
Tarea 1 corriendo. Cnt: 5. Tiempo: 2805601
Tarea 2 corriendo. Cnt: 3. Tiempo: 3305602
Tarea 1 corriendo. Cnt: 6. Tiempo: 3305684
Tarea 1 corriendo. Cnt: 7. Tiempo: 3805601
Tarea 2 corriendo. Cnt: 4. Tiempo: 4305602
```