

freeRTOS con ESP32

FreeRTOS es un sistema operativo de tiempo real que permite gestionar los recursos hardware y los tiempos de ejecución de las diferentes tareas implementadas en un sistema embebido. FreeRTOS está diseñado para consumir pocos recursos de memoria y se encuentra desarrollado principalmente en lenguaje ANSI-C, conformado por un conjunto de funciones que se ejecutan en momentos determinados del programa y distribuido gratuitamente bajo licencia "Open Source", incluyendo aplicaciones comerciales.



Aunque FreeRTOS no es el único RTOS que se encuentra en el mercado, este mismo ha sido portado a 35 plataformas, permite compilar solo las funciones que se van a utilizar, acotando así su impacto en la memoria de código. Adicionalmente, ofrece funciones de temporización, comunicación entre tareas, sincronización entre tareas e interrupciones y de definición de secciones críticas, entre otros.

Finalmente, es importante mencionar que el estudio de los RTOS consiste en la planificación y programación de la carga de trabajo en un procesador, de modo que las garantías de la línea de tiempo para la carga de trabajo nunca se viole. La carga de trabajo se cuantifica en piezas discretas a las que nos referimos como una tarea. Una tarea es, entonces, un programa independiente que desarrolla operaciones específicas usando uno o varios recursos (memoria, RAM, CPU, UART, GPIO, etc.) del microcontrolador o SOC en el caso del ESP32.

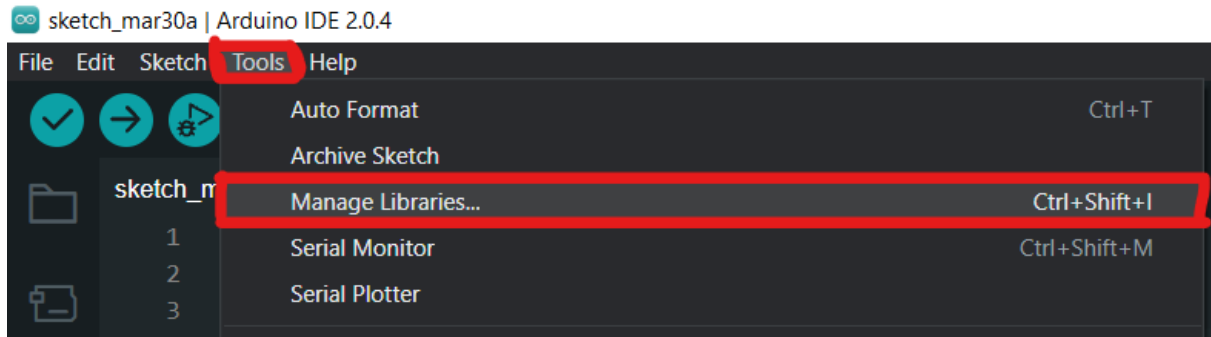
En la siguiente practica aprenderemos a instalar la libreria freeRTOS y haremos una practica que nos servira para aprender las bases de esta creando nuestra primera tarea.

Materiales necesarios:

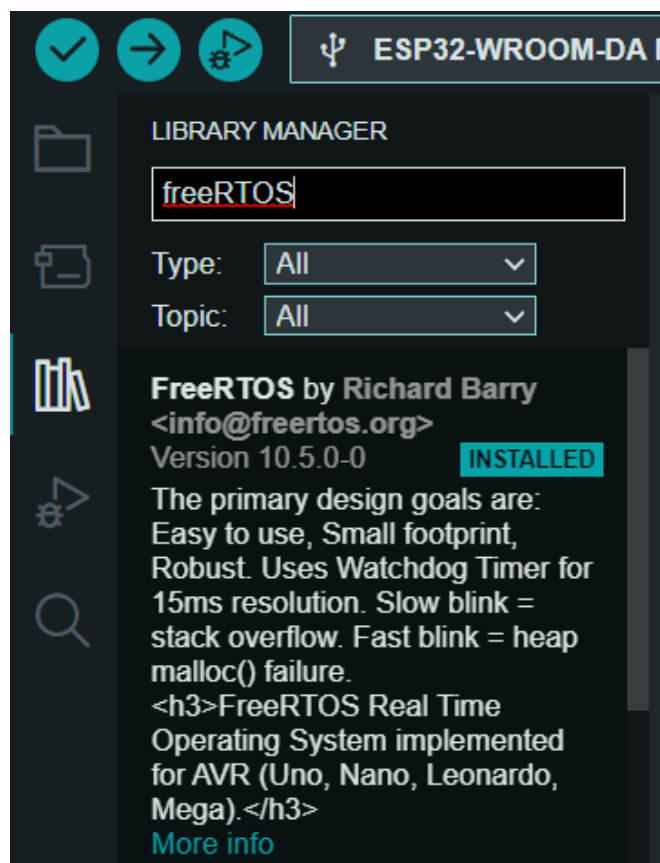
- ESP32
- Cable USB
- Computadora con el IDE de Arduino instalado

Desarrollo

1. Abrimos nuestro IDE arduino y nos dirigimos a la pestaña “Herramientas” (Tools) y al abrirla seleccionamos “Gestionar Bibliotecas” (Manage Libraries).



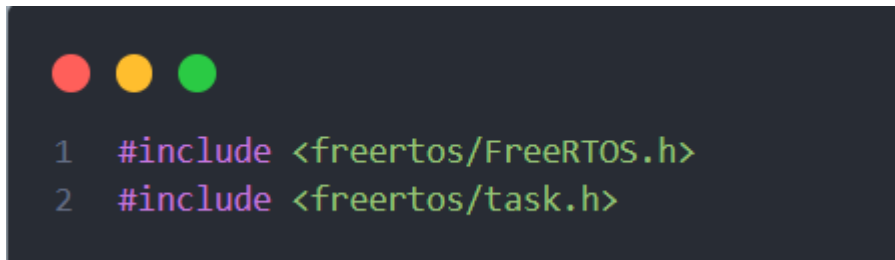
2. Se abra una pestaña en la cual buscaremos la libreria “freeRTOS” y la instalaremos.



3. Cuando este inslada procederamos a hacer la primer practica para probar la instalacion de la libreria.

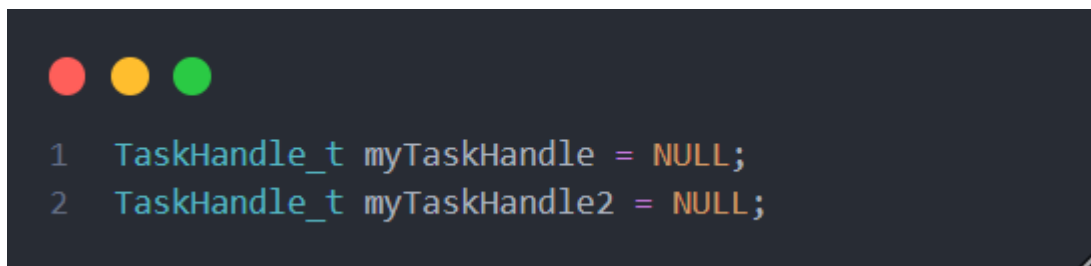
Desarrollo de la primer Practica

1. Para iniciar Incluimos las librerias en la parte superior del programa.



```
1  #include <freertos/FreeRTOS.h>
2  #include <freertos/task.h>
```


2. Se declaran dos variables de tipo `TaskHandle_t`, que se utilizarán para almacenar los identificadores de las tareas creadas. El valor inicial de ambas variables es `NULL`, lo que significa que en ese momento no están asociadas a ninguna tarea en particular. Estas variables se utilizan para almacenar el manejador devuelto por la función que crea la tarea, lo que permite realizar operaciones sobre esa tarea en el futuro, como suspenderla, reanudarla o eliminarla.




```
1  TaskHandle_t myTaskHandle = NULL;
2  TaskHandle_t myTaskHandle2 = NULL;
```

3. Crearemos 2 funciones los nombres de las funciones son "Demo_Task" y "Demo_Task2" toma un puntero a void como argumento, que puede ser utilizado para pasar datos a la tarea.

Dentro de la función, hay un bucle while que se ejecutará continuamente mientras la tarea esté activa. En cada iteración del bucle, se imprimirá un mensaje de texto a través del objeto Serial. La función `vTaskDelay` se utiliza para pausar la tarea durante un período de tiempo específico, en este caso 1000 milisegundos (1 segundo), antes de continuar con la próxima iteración del bucle. La función `pdMS_TO_TICKS` se utiliza para convertir el valor de tiempo en milisegundos a un valor de tiempo en ticks que es utilizado por el sistema operativo en tiempo real para programar tareas.



```
1 void Demo_Task(void *arg)
2 {
3     while (1) {
4         Serial.println("Demo_Task printing..");
5         vTaskDelay(pdMS_TO_TICKS(1000));
6     }
7 }
```



```
1 void Demo_Task2(void *arg)
2 {
3     while (1) {
4         Serial.println("Demo_Task2 printing..");
5         vTaskDelay(pdMS_TO_TICKS(1000));
6     }
7 }
8
```

4. El código se encuentra dentro de la función "setup", que es parte del ciclo de vida de un programa que utiliza la plataforma Arduino. En este fragmento de código, se están creando dos tareas (también conocidas como procesos o hilos) utilizando la función "xTaskCreate" y "xTaskCreatePinnedToCore", que son funciones proporcionadas por FreeRTOS para crear tareas. Estas tareas se ejecutarán simultáneamente en diferentes núcleos del microcontrolador.

El primer parámetro en ambas funciones es el nombre de la función que se ejecutará como tarea (en este caso, "Demo_Task" y "Demo_Task2"). El segundo parámetro es el nombre que se le dará a la tarea. El tercer parámetro es el tamaño de la pila que se asignará a la tarea (4096 bytes en este caso). El cuarto parámetro es un puntero a cualquier dato que se desee pasar a la tarea (en este caso, NULL indica que no se pasará ningún dato). El

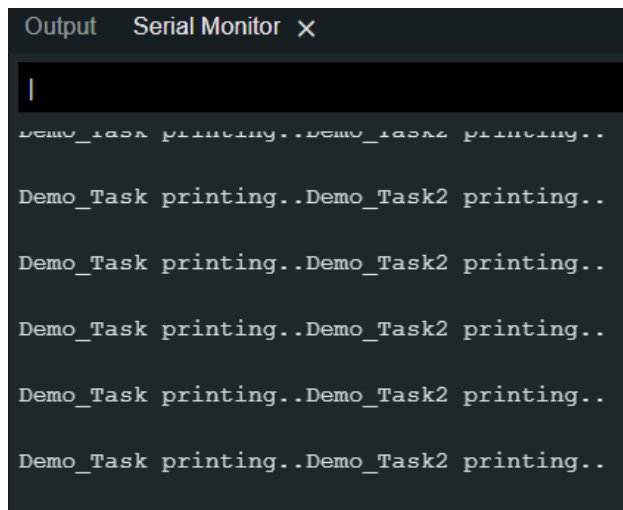
quinto parámetro es la prioridad de la tarea. En este caso, ambas tareas tienen una prioridad de 10. Finalmente, el sexto parámetro es un puntero a una variable que contendrá un identificador para la tarea que se está creando.

En resumen, este código está creando dos tareas utilizando FreeRTOS, que se ejecutarán simultáneamente en diferentes núcleos del microcontrolador. La función "setup" se utiliza comúnmente en los programas de Arduino para realizar configuraciones iniciales antes de que el programa principal comience a ejecutarse.

```
1 void setup()
2 {
3     Serial.begin(115200);
4     xTaskCreate(Demo_Task, "Demo_Task", 4096, NULL, 10, &myTaskHandle);
5     xTaskCreatePinnedToCore(Demo_Task2, "Demo_Task2", 4096, NULL, 10, &myTaskHandle2, 1);
6 }
```

Nota: El void loop lo dejamos vacío.

5. Como último paso subimos el programa a nuestra esp32 y si seguiste los pasos correctamente aparecerá en el monitor serie lo siguiente.



```
Output  Serial Monitor x
|
Demo_Task printing..Demo_Task2 printing..
Demo_Task printing..Demo_Task2 printing..
Demo_Task printing..Demo_Task2 printing..
Demo_Task printing..Demo_Task2 printing..
Demo_Task printing..Demo_Task2 printing..
Demo_Task printing..Demo_Task2 printing..
```

Y ¡FELICIDADES creaste tu primer código con FreeRTOS!

Puedes encontrar el código completo y los pdf en la siguiente página de GitHub:

https://github.com/GabrielCorUs/freeRTOS_ESP32/tree/main/freeRTOS_ESP32