Johny Silva Mendes

PRACTICA 4: SISTEMAS OPERATIVOS EN TIEMPO REAL

Objetivo: observar el funcionamiento de un sistema operativo en tiempo Real

Pregunta 1: ¿qué sucede si está utilizando una pantalla de tinta electrónica que tarda unos segundos en actualizarse?

En este caso veríamos a cada cierto tiempo una temperatura acutalizada, pero no a tiempo real, sinó una temperatura con bastante retraso. Por lo tanto, sería imprescindible utilizar tareas para modificar el valor de la variable una vez realizada la tarea de mostrar por pantalla. Seguiría sin ser a tiempo real pero se aproximaría más que en el anterior caso, también, de esta forma, se podría medir la temperatura casi a tiempo real y tener más precisón en caso de que fusese necesario para alguna otra tarea.

Ejercicio Practico 1

Código proporcionado en la práctica:

void setup()

```
void setup() {
Serial.begin(112500);
xTaskCreate(anotherTask,"another Task", 10000, NULL, 1, NULL);
}
```

Se inicializa el puerto serie. Se crea las tareas a utilizar mediante la funcion **xTaskCreate()** en este caso se crea una llamada anotherTask, la cual se llamará a través de la función anotherTask. El tamaño reservado para esta tarea es 10000. No se le pasa ningún valor, se puede declarar poniendo el primer NULL. La prioridad de la tarea es 1. Las prioridades van de 0 a 24, siendo la 24 la mayor prioridad. Las prioridades nos sirven cuando dos o más tareas compiten por el uso de la CPU, en ese caso se ejecuta primero la de mayor prioridad. En caso de que dos tareas tengan el mismo grado de prioridad, el tiempo de CPU se repartirá.

void loop()

```
void loop()
{
Serial.println("this is ESP32 Task");
delay(1000);
}
```

Esta funcióon es una tarea llamada por el mismo código de forma automática. Hay formas de anularla como se verá más adelante, pero en este caso la usamos solamente para mostrar por el puerto serie una frase.

DECLARACIÓN DE LA TAREA

```
void anotherTask( void * parameter )
{
/* loop forever */
for(;;)
{
Serial.println("this is another Task");
delay(1000);
}
/* delete a task when finish,
this will never happen because this is infinity loop */
vTaskDelete( NULL );
}
```

Se declara la tarea de la misma forma que una función y se pasa como parámetro un puntero que en este caso se llama *parameter*.

Como podemos observar, este código crea una tarea llamada *anotherTask* la cual se va a ejecutar en paralelo con el *loop* del main. En el puerto serie se muestra los mensajes "this is ESP32 Task" y "this is another Task" cada una cada 1000ms, idependientemente una de la otra.