# PRACTICA 3 : SISTEMAS OPERATIVOS EN TIEMPO REAL

## B: Ejercicio 2

En esta práctica, debemos realizar un programa que utilice dos tareas una enciende un led y otra lo apaga dichas tareas deben estar sincronizadas. Para conseguirlo, vamos a utilizar semaforo mutex para cooridnar el encendido y apagado del led.

### Código

```
#include <Arduino.h>

void ledON (void * pvParameters);
void ledOFF(void * pvParameters);
int LED =2;

SemaphoreHandle_t semafor;

void setup(){

Serial.begin(9600);
pinMode(LED, OUTPUT);
semafor= xSemaphoreCreateMutex();

xTaskCreate(
    ledON,
    "LED ON",
    10000,
    NULL,
    1,
    NULL);

xTaskCreate(
    ledOFF,
    "LED OFF",
    10000,
    NULL,
    1,
    NULL);

}

void loop(){}

void ledON (void * pvParameters){
    for(;;){
        xSemaphoreTake(semafor, portMAX_DELAY);
        digitalWrite(LED, HIGH);
        delay(3000);
        xSemaphoreGive(semafor);
```

```
    }
}

void ledOFF (void * pvParameters){
    for(;;){
        xSemaphoreTake(semafor, portMAX_DELAY);
        digitalWrite(LED, LOW);
        delay(1500);
        xSemaphoreGive(semafor);
    }
}
```

## Funcionamiento del programa

En primer lugar, vamos a declarar dos funciones una llamada "ledON" y otra llamada "ledOFF", y otras variables que nos harán falta.

```
#include <Arduino.h>

void ledON (void * pvParameters);
void ledOFF(void * pvParameters);
int LED =2;

SemaphoreHandle_t semafor;
```

A continuación definimos nuestra función Setup que asigna a la variable "semafor" creada anteriormente un Mutex y nos devuelve un controlador mediante el cual se puede hacer referencia al Mutex creado.

```
void setup(){

Serial.begin(9600);
pinMode(LED, OUTPUT);
semafor= xSemaphoreCreateMutex();
```

Ahora declaramos dos tareas, una para cada LED que vamos a necesitar. Para ello debemos pasarle una función (ledON y ledOFF), el nombre de la tarea, el tamaño de la stack en bytes, la prioridad de la tarea y finalmente el identificador de tarea.

```
xTaskCreate(
    ledON,
    "LED ON",
    10000,
    NULL,
    1,
    NULL);
```

```
xTaskCreate(
    ledOFF,
    "LED OFF",
    10000,
    NULL,
    1,
    NULL);

}
```

Finalmente definimos las funciones ledON y ledOFF de forma que queden sincornizadas y se produzca el efecto "Semaforo".

```
void ledON (void * pvParameters){
    for(;;){
        xSemaphoreTake(semafor, portMAX_DELAY);
        digitalWrite(LED, HIGH);
        delay(3000);
        xSemaphoreGive(semafor);
    }
}

void ledOFF (void * pvParameters){
    for(;;){
        xSemaphoreTake(semafor, portMAX_DELAY);
        digitalWrite(LED, LOW);
        delay(1500);
        xSemaphoreGive(semafor);
    }
```