PRÁCTICA 4 : SISTEMAS OPERATIVOS EN TIEMPO REAL

CÓDIGO

```
#include <Arduino.h>

void anotherTask( void * parameter );

void setup()
{
  Serial.begin(112500);
  /* we create a new task here */
  xTaskCreate(
    anotherTask, /* Task function. */
    "another Task", /* name of task. */
    10000, /* Stack size of task */
    NULL, /* parameter of the task */
    1, /* priority of the task */
    NULL); /* Task handle to keep track of created task */
}

/* the forever loop() function is invoked by Arduino ESP32 loopTask */
void loop()
{
  Serial.println("this is ESP32 Task");
  delay(1000);
}

/* this function will be invoked when additionalTask was created */
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 );
}
```

SALIDA POR EL PUERTO SERIE

Por el puerto serie va saliendo en bucle:

this is ESP32 Task
this is another Task

En la siguiente imagen se muestra una captura del terminal:

FUNCIONAMIENTO

Dentro del *setup*, primero inicializamos una comunicación en serie a una velocidad de 115200 bauds. A continuación informamos al planificador sobre nuestra tarea con `xTaskCreate` , donde tenemos que definir la función de la *Task*, el nombre, el tamaño del *stack*, el parámetro y su prioridad. El código es el siguiente:

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

Después dentro del `void loop()` creamos un bucle infinito donde se muestra por el terminal "this is ESP32 Task" cada segundo,

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

Para finalizar llamamos a la función `void anothertask( void * parameter)` y dentro de esta también hacemos un bucle infinito que va mostrando en el terminal "this is another Task" cada segundo, y después con `vTaskDelate( NULL )` eliminamos la tasca cuando finalice, pero esto nunca sucederá porque es un bucle infinito.

```
void anotherTask( void * parameter )
{
  for(;;)
  {
    Serial.println("this is another Task");
    delay(1000);
  }
  vTaskDelete( NULL );
}
```