

Primer banco de Programas ESP32

2024-I

OK

Ejemplo 1a: Blink con el ESP32 en la IDE de Arduino

```
#define LED_1 2 2
// 2 == led interno GPIO general purpose input/output

void setup() {
    pinMode(LED_1, OUTPUT);
}

void loop() {
    digitalWrite(LED_1, HIGH); // turn the LED on (HIGH is the
    voltage level)
    digitalWrite(LED_1, LOW); // turn the LED off by making the
    voltage LOW
```

}

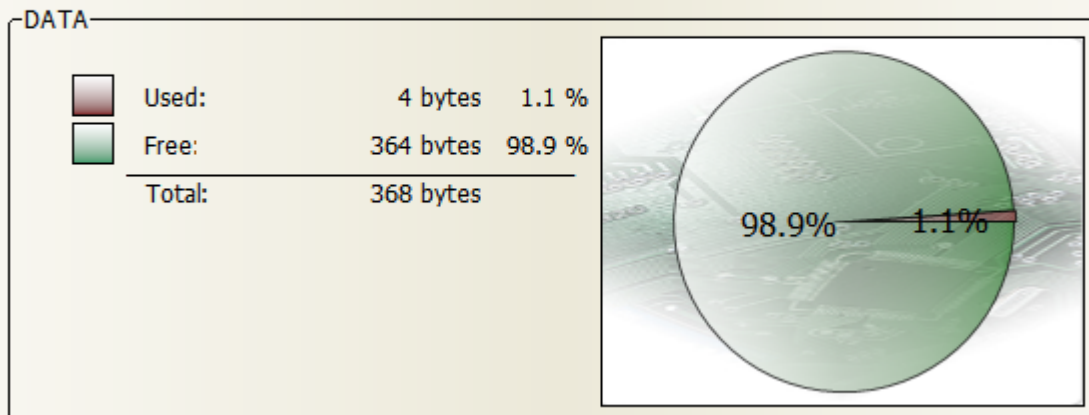
// "El Sketch usa 237421 bytes (18 %) del espacio de almacenamiento de programa. El máximo es 1'310.720 bytes == 1,3 Mb de 4 Mb".

// "Las variables Globales usan 21048 bytes (6 %) de la memoria dinámica, dejando 306632 bytes para las variables locales. El máximo es 327680 bytes."

Ejemplo 1b: Blink en MikroC PRO



```
void main()  
{  
    OSCCON = 0b11110000;           // 32MHz ===  
    OSCCON = 0b11110000;  
    TRISA = 0;  
    ANSELA = 0;  
    while (1)  
    {  
        lata = 0;  
        lata = 255;  
    }  
}
```

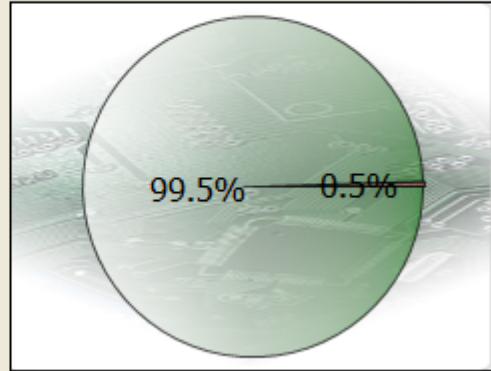
RAM Memory Usage



ROM Memory Usage

ROM Usage

	Used:	21 program words	0.5 %
	Free:	4074 program words	99.5 %
		<hr/>	
	Total:	4095 program words	



Ejemplo 2: GPIO I/O

```
#define LED_1 2
// 2 == led interno GPIO general purpose input/output

#define LED_2 17

const int pulsador = 0;
// 0 == boot botton, mala técnica.

int estado_del_pulsador;

void setup() {
    setCpuFrequencyMhz(240);
    Serial.begin(115200);
    pinMode(LED_1, OUTPUT);
    pinMode(LED_2, OUTPUT);
    //pinMode(pulsador, INPUT);
    //pinMode(pulsador, INPUT_PULLDOWN);
    pinMode(pulsador, INPUT_PULLUP);
}
```

```

void loop() {

    estado_del_pulsador = digitalRead(pulsador);

    Serial.println(" Estado del pulsador " +
String(estado_del_pulsador));

    if (estado_del_pulsador == HIGH)

        {

            digitalWrite(LED_1, HIGH);

// turn the LED on (HIGH is the voltage level)

            digitalWrite(LED_2, LOW);

// turn the LED off by making the voltage LOW

            delay(100);

// wait

        }

    else

    {

        digitalWrite(LED_1, LOW);

// turn the LED off by making the voltage LOW

        digitalWrite(LED_2, HIGH);

// turn the LED on (HIGH is the voltage level)

        delay(100);

// wait

    }

    Serial.printf(" Núcleo %d\n ", xPortGetCoreID());

    Serial.println("Corre en el núcleo --> " +
String(xPortGetCoreID()));

}

```

Ejemplo 3a: CRUNCH_CLK_ESP32

```
// 260 veces más rápido que en un pic a 32 MHz

// Se demora a 240 MHz, 807 mS

#define LED 2

#define start_Pin 0


unsigned long loops1 = 1000;

unsigned long loops2 = 1000;


//double tt, aa = 12345.8;

float t1; // Más lento

int t2;

int t3;

unsigned long qq = 0;


void setup() {

    //rtc_clk_cpu_freq_set (RTC_CPU_FREQ_80M);

    setCpuFrequencyMhz(240);

    Serial.begin (115200);

    delay (1000);

    Serial.println ("START");

    pinMode (LED, OUTPUT);

    pinMode (start_Pin, INPUT_PULLUP);
```

```

        digitalWrite (LED, LOW);
    }

void loop()
{
    long start = millis();
    digitalWrite(LED, HIGH);
    for ( long i = 0; i < loops1; i++) {

        for (long j = 1; j <
loops2; j++) {

            qq++;

            t1 = 5000.0 * i;

            t2 = 150 * 1234 * i;

            t3 = j % 554 ;

        }

    }
}

```

```

    digitalWrite(LED, LOW);
    Serial.print(millis() - start);
    Serial.println(" mS");
    Serial.println("Finish");
    Serial.println(xPortGetCoreID());
    delay(10);
}

```


Ejemplo 3b: CRUNCH PIC16F1827 a 32 MHz

```
unsigned long loops1 = 1000;           // 100*100 = 2 Seg
unsigned long loops2 = 1000;           // 1000*1000 = 3,5 minutos
```

```
//double tt, aa = 12345.8;
float t1; // Más lento
int t2;
int t3;
long i, j;
unsigned long qq = 0;
//unsigned long i, j, qq = 0;
```

```
void main() {
    OSCCON = 0xF0;
    ansela = 0;
    trisa = 0;           // pinMode (LED, OUTPUT);
    lata.f0 = 1;         // digitalWrite (LED, LOW);
    while (1)
    {

        for (i = 0; i < loops1; i++) {
```

```
        for (j = 1; j < loops2; j++)
        {
            qq++;
            t1 = 5000.0 * i;
            t2 = 150 * 1234 * i;
            t3 = j % 554 ;
        }
        lata.f1 = !lata.f1;
    }
    lata.f0 = !lata.f0;                // digitalWrite(LED, LOW)
}
}
```

Ejemplo 4: Sensor táctil

```
// Tomado de la IDE de Arduino

// ESP32 Touch Test

// Just test touch pin - Touch0 is T0 which is on GPIO 4. T1 = GPIO 0, ...

void setup()
{
    Serial.begin(115200);

    delay(1000); // give
me time to bring up serial monitor

    Serial.println("ESP32 Touch Test");
}

void loop()
{
    Serial.println(touchRead(T0)); // get
value using T0 = GPIO 4, using T1 = GPIO 0

    delay(100);
}
```

RTOS

Ejemplo 5a: Tareas en “paralelo”, un solo núcleo

Task Scheduling

```
// Tareas, video 10
```

```
//
```

```
https://www.youtube.com/watch?v=Vus7HE3wc6A&list=PL-Hb9zZP9qC48GcXj\_BsDipCPAzwcps6e&index=2
```

```
// Easy Learning, min 11´50, si se resetea, aumenta el tamaño de la tarea de 1024 a 2048.
```

```
// Tres leds que prenden con demoras de 1, 2 y 4 segundos == contador binario de 0 a 7
```

```
//#include "esp_system.h"
```

```
//#include "nvs_flash.h"
```

```
#include <stdio.h>
```

```
#include "esp_log.h"
```

```
#include "driver/gpio.h"
```

```
#include "freertos/task.h"
```

```
#include "freertos/FreeRTOS.h"
```

```
#define Stack_Size 1024
```

```
#define LedR 2
// 33, 25, 26 (ESP32 CAM, 4)

#define LedG 25

#define LedB 26


#define R_delay 500
// *2

#define G_delay 1000

#define B_delay 2000


const char *tag = "Main";

float cont = 500;


void tareaR(void *pvParameter)
{
    while(1)
    {
        printf("Ejecutando tarea que
demora 1 Segundo\n");

        digitalWrite(LedR, HIGH);
// turn the LED on (HIGH is the voltage level)

//vTaskDelay(R_delay/portTICK_PERIOD_MS);

vTaskDelay(pdMS_TO_TICKS(R_delay));

        digitalWrite(LedR, LOW);
```

```

//vTaskDelay(R_delay/portTICK_PERIOD_MS);

vTaskDelay(pdMS_TO_TICKS(R_delay));

        Serial.println();
    }

    vTaskDelete(NULL);
}

void tareaG(void *pvParameter)
{
    while(1)
    {
        printf("Ejecutando tarea que
demora 2 Segundos\n");

        digitalWrite(LedG, HIGH);
        // turn the LED on (HIGH is the voltage level)

        //vTaskDelay(G_delay/portTICK_PERIOD_MS);

        vTaskDelay(pdMS_TO_TICKS(G_delay));

        digitalWrite(LedG, LOW);

        //vTaskDelay(G_delay/portTICK_PERIOD_MS);

        vTaskDelay(pdMS_TO_TICKS(G_delay));

        Serial.println();
    }

    vTaskDelete(NULL);
}

```

```

    }

void tareaB(void *pvParameter)
{
    while(1)
    {
        printf("Ejecutando tarea que
demora 4 Segundos\n");

        digitalWrite(LedB, HIGH);
        // turn the LED on (HIGH is the voltage level)

        //vTaskDelay(B_delay/portTICK_PERIOD_MS);

        vTaskDelay(pdMS_TO_TICKS(B_delay));

        digitalWrite(LedB, LOW);

        //vTaskDelay(B_delay/portTICK_PERIOD_MS);

        vTaskDelay(pdMS_TO_TICKS(B_delay));

        Serial.println();
    }

    vTaskDelete(NULL);
}

void tarea4(void *pvParameter)
{
    while(1)
    {

```

```

        printf("Ejecutando tarea rápida
\n");

//printf("*****\n");

        Serial.print("El contador de mS
va en: ");

        Serial.println(cont);

        cont = cont + 500;

printf("*****\n");

        Serial.println();

        vTaskDelay(pdMS_TO_TICKS(500));

//vTaskDelay(500/portTICK_PERIOD_MS);

    }

    vTaskDelete(NULL);

}

void setup()
{
    //nvs_flash_init();

    Serial.begin(115200);

    pinMode(LedR, OUTPUT);

    pinMode(LedG, OUTPUT);

    pinMode(LedB, OUTPUT);

```



```
        xTaskCreate(&tareaR, "tarea1", Stack_Size, NULL, 1,
NULL);

        xTaskCreate(&tareaG, "tarea2", Stack_Size, NULL, 1,
NULL);

        xTaskCreate(&tareaB, "tarea3", Stack_Size, NULL, 1,
NULL);

        xTaskCreate(&tarea4, "tarea4", Stack_Size, NULL, 1,
NULL);    // prioridad 2
```

```
        //vTaskStartScheduler();
// Se usa para lanzar tareas en RTOS, aquí no se necesita

    }
```

```
void loop()

{

    vTaskDelete(NULL);
// Recomendación para arduino !!!!!!! se puede demorar el doble.

}
```

Ejemplo 5b: Planeación de tareas, Task Scheduling //

```
// app_cpu (núcleo 1) y pro_cpu (núcleo 0)
```

```
// Digikey 3
```

```
// https://www.youtube.com/watch?v=95yUbClyf3E&list=LL&index=1&t=7s
```

```
/**
```

```
 * FreeRTOS Task Demo
```

```
 *
```

```
 * Toggles LED and prints "Hello, World" to console in separate tasks.
```

```
 *
```

```
 * Date: December 3, 2020
```

```
 * Author: Shawn Hymel
```

```
 * License: 0BSD
```

```
 */
```

```
// NOTA: Configurar el monitor serial a: 300 bps
```

```
#if CONFIG_FREERTOS_UNICORE
```

```
// Use only core 1 for demo purposes
```

```
    static const BaseType_t app_cpu = 0;
```

```
#else
```

```
    static const BaseType_t app_cpu = 1;
```

```
#endif
```

```
const char msg[] = "Esto es una demostración RTOS";
```

```
// Some string to print
```

```

static TaskHandle_t task_1 = NULL;
// Task handles A

static TaskHandle_t task_2 = NULL;

//*****
***

// Tasks

// Task: print to Serial Terminal with lower priority
void startTask1(void *parameter)
{
    int msg_len = strlen(msg);
// Count number of characters in string

    while (1)
    {
        Serial.println();

        for (int i = 0; i < msg_len; i++)
        {
            Serial.print(msg[i]);

// Print string to Terminal

        }

        Serial.println();

        vTaskDelay(1000 /
portTICK_PERIOD_MS);    // Task in lock state by 1 S

    }
}

// Task: print to Serial Terminal with higher priority (2)

```

```

void startTask2(void *parameter)
{
    while (1)
    {
        Serial.print('*');

        vTaskDelay(100 / portTICK_PERIOD_MS);
    }
}

//*****
***

// Main (runs as its own task with priority 1 on core 1)

void setup()
{
    Serial.begin(300);
    // Configure Serial (go slow so we can watch the preemption)

    vTaskDelay(1000 / portTICK_PERIOD_MS);

    Serial.println();

    Serial.println("---FreeRTOS Task Demo---");

    // Print self priority

    Serial.print("Setup and loop task running on core ");
    Serial.print(xPortGetCoreID());
    Serial.print(" with priority ");
    Serial.println(uxTaskPriorityGet(NULL));
}

```

```

// Task to run forever

xTaskCreatePinnedToCore(startTask1,
                        "Task 1",
                        1024,
                        NULL,
                        1,
                        &task_1,
                        app_cpu);

// Task to run once with higher priority

xTaskCreatePinnedToCore(startTask2,
                        "Task 2",
                        1024,
                        NULL,
                        2,
                        &task_2,
                        app_cpu);

}

void loop()
{

// Suspend the higher priority task for some intervals

for (int i = 0; i < 3; i++)
{
    vTaskSuspend(task_2);

    vTaskDelay(2000 / portTICK_PERIOD_MS);
}
}

```

```
        vTaskResume(task_2);

        vTaskDelay(2000 / portTICK_PERIOD_MS);

    }

    // Delete the lower priority task
    if (task_1 != NULL)
    {
        vTaskDelete(task_1);

        task_1 = NULL;
    }
}
```

Ejemplo 6: Cambiando de núcleos (0 o 1)

app_cpu (núcleo 1) y pro_cpu (núcleo 0)

```
// Andreas Spiess # 168
```

```
#define LED1 2
```

```
unsigned long loops1 = 1000;
```

```
unsigned long loops2 = 1000;
```

```
float t1;
```

```
int t2;
```

```
int t3;
```

```
unsigned long qq = 0;
```

```
TaskHandle_t Task1;
```

```
void artificialload ()
```

```
{
```

```
    for ( long i = 0; i < loops1; i++) {
```

```
        for (long j =
```

```
1; j < loops2; j++) {
```

```
    qq++;
```

```
    t1 = 5000.0 * i;
```

```
t2 = 150 * 1234 * i;
```

```
t3 = j % 554 ;
```

```
}
```

```
}
```

```
}
```

```
void tarea1( void * parameter )
```

```
{
```

```
    unsigned long i, j;
```

```
    for (;;) {
```

```
        long start = millis();
```

```
        artificialLoad();
```

```
        Serial.print("Finish
```

```
Task 1 which runs on Core ");
```

```
        Serial.print(
```

```
xPortGetCoreID());
```

```
        //Serial.printf("Core
```

```
%d\n ", xPortGetCoreID());
```

```
        Serial.print(" Time ");
```

```
        Serial.println(millis()
```

```
- start);
```

```
        delay(10);
```

```
    }
```

```
}
```

```
void setup()
```



```

{
    Serial.begin(115200);

    pinMode(LED1, OUTPUT);

    xTaskCreatePinnedToCore(tarea1,"moliendo
números",1000,NULL,1,&Task1,1);

                                //( Task function, name of task, size of
task, parameter of the task, priority of the task, Task handle to keep track
of created task, Core)

}

void loop()
{
    //delay(1000); //
Con el delay comentado, se demora el doble con el núcleo 1 (1646 vs 823).

    //vTaskDelete(NULL); //
Corrección al anterior problema

}

```

Ejemplo 7: Prueba velocidad dos núcleos (asíncrona)

```
// app_cpu (núcleo 1) y pro_cpu (núcleo 0)
```

```
// Andreas Spiess # 168
```

```
unsigned long loops1 = 1000;
```

```
unsigned long loops2 = 1000;
```

```
float t1;
```

```
int t2;
```

```
int t3;
```

```
unsigned long qq = 0;
```

```
TaskHandle_t Task1, Task2;
```

```
void artificialLoad ()
```

```
{
```

```
    for ( long i = 0; i < loops1; i++) {
```

```
        for (long j = 1; j < loops2; j++) {
```

```
            qq++;
```

```
            t1 = 5000.0 * i;
```

```
            t2 = 150 * 1234 * i;
```

```
            t3 = j % 554 ;
```

```
        }
```

```
    }
```

```
}
```

```

void tarea1( void * parameter )
{
    unsigned long i, j;
    for (;;) {
        long start = millis();
        artificialLoad();
        Serial.print("Finish Tarea 1 which runs on Core ");
        Serial.print( xPortGetCoreID());
        Serial.print(" Time ");
        Serial.println(millis() - start);
        delay(10);
    }
}

```

```

void tarea2( void * parameter )
{
    for (;;) {
        long start = millis();
        artificialLoad();
        Serial.print("Finish Tarea 2 which runs on Core ");
        Serial.print( xPortGetCoreID());
        Serial.print(" Time ");
        Serial.println(millis() - start);
        delay(10);
    }
}

```

```

void setup() {
    Serial.begin(115200);

    xTaskCreatePinnedToCore(tarea1,"moliendo1",1000,NULL,1,&Task1,0);
        //( Task function, name of task, size of task, parameter of the task,
priority of the task, Task handle to keep track of created task, Core)

    delay(500);

    xTaskCreatePinnedToCore(tarea2,"moliendo2",1000,NULL,1,&Task2,1);
}

void loop() {
    //delay(1000);
    vTaskDelete(NULL);
    //Serial.printf("Core %d\n ", xPortGetCoreID());
}

```

Ejemplo 8a: Colas

```
// Easy learning, video 12
```

```
//
```

```
https://www.youtube.com/watch?v=abwyjmfZ0mQ&list=PL-Hb9zZP9qC48GcXj\_BsDipCPAzwcps6e&index=4
```

```
// https://www.youtube.com/watch?v=abwyjmfZ0mQ
```

```
// Tarea escribe 8 datos cada 400 mS y la tarea lee los toma cada 2000 mS,  
se bloquea
```

```
// Tarea escribe 8 datos cada 400 mS y descansa 7000 mS. La tarea lee los  
toma cada 2000 mS, no se bloquea
```

```
#include <stdio.h>
```

```
#include "esp_log.h"
```

```
#include "driver/gpio.h"
```

```
#include "freertos/task.h"
```

```
#include "freertos/queue.h"
```

```
// A0, incluir.
```

```
#include "freertos/FreeRTOS.h"
```

```
#define ledR 2
```

```
#define ledG = 25
```

```
//#define ledB = 26
```

```

#define R_delay 400

#define G_delay 2000
// 2000, lee más rapido

#define STACK_SIZE 1024*2
// 1024*1 ERROR!!!!!!!!!!!!!!!!!!!!!!

#define TAM_COLA 20
/*20 mensajes*/ //era 10 se llena la cola????

#define TAM_MSG 7
/*Cada Mensaje 7 caracteres*/

const char *tag = "Main";

QueueHandle_t cola_Mensaje = 0;
// A otra forma, junio 6

//xQueueHandle cola_Mensaje;
// A, GlobalQueue

void escribe1(void *pvParameter)
{
    char cadena[7];

    // Variable local

    while(1)
    {
        // on led

        for (int i =0; i<8; i++)
        {

```

```

vTaskDelay(pdMS_TO_TICKS(R_delay/2));

cola\n", i);

// on led
//ESP_LOGW(tag,
"enviando %i a la cola", i);

//if
(!xQueueSendToBack(cola_Mensaje, &i, 2000/portTICK_RATE_MS))

if
(!xQueueSend(cola_Mensaje, &i, pdMS_TO_TICKS(1))) // C
{
printf("\t\t\t
Fallo al enviar %i a la cola\n", i);

//ESP_LOGE(tag,
"Fallo al enviar %i a la cola", i);

}

//vTaskDelay(2000 /
// Retardo para poder ver el
mensaje

vTaskDelay(pdMS_TO_TICKS(R_delay/2));

// off led
}

vTaskDelay(pdMS_TO_TICKS(7000)); //
Si se comenta, la cola se llena 7 seg.

}

}

```

```

void lee1(void *pvParameter)
{
    int i, valor_recibido= 0;
    //char Rx[7];

    while(1)
    {

        //if(!xQueueReceive(cola_Mensaje,&Rx,10000/portTICK_RATE_MS)==pdTRUE)

        if(!xQueueReceive(cola_Mensaje,&valor_recibido, pdMS_TO_TICKS(100)))
        // D
        {
            printf("Fallo al recibir de la
cola, desocupada\n");
        }
        else
        {

            vTaskDelay(pdMS_TO_TICKS(G_delay/2));

            // on led
            printf("Valor recibido %i de la
cola\n", valor_recibido);

            vTaskDelay(pdMS_TO_TICKS(G_delay/2));

            // off led
        }

        vTaskDelay(pdMS_TO_TICKS(1));
    }
    // Recomendación para que no se active el WDT

```



```

    }

}

void setup()
{
    Serial.begin(115200);

    //cola_Mensaje= xQueueCreate(TAM_COLA, TAM_MSG);
    // B, Crea la cola

    cola_Mensaje= xQueueCreate(TAM_COLA, sizeof(uint32_t));
    // B, Crea la cola de 10 elementos, enteros

    xTaskCreate(&lee1, "lee1", STACK_SIZE, NULL, 1,
NULL); // La de mayor prioridad, la cola nunca se
llena. ¿sobra?

    xTaskCreate(&escribe1, "escribe1", STACK_SIZE, NULL, 1,
NULL); //

}

void loop()
{
    vTaskDelete(NULL);
    // Recomendación para arduino !!!!!!! se puede demorar el doble.

}

```

Ejemplo 8b: Colas y boot sw

```
// ESP32_SIMPLE_QUEUE vs global
```

```
// https://www.youtube.com/watch?v=ywbq1qR-fY0
```

```
TaskHandle_t Task0;
```

```
TaskHandle_t Task1;
```

```
QueueHandle_t queue;
```

```
#define LED0 2
```

```
#define LED1 15
```

```
#define SWITCH 0
```

```
void loop0(void * parameter) {
```

```
    for (;;) {
```

```
        // Add a random number to the
```

```
queue
```

```
        // auto rndNumber = esp_random();
```

```
        // Use Arduino implementation for
```

```
a number between limits
```

```
int rndNumber = random(1, 9);
```

```
        // Add to the queue - wait forever
```

```
until space is available
```

```

                                Serial.println("\t\t\t\t\tManager
- Adding " + String(rndNumber) + " to queue");

                                xQueueSend(queue, &rndNumber,
portMAX_DELAY);                                // FIFO, wait for ever

                                // Artificial wait here

                                digitalWrite(LED0, HIGH);
                                delay(200);
                                digitalWrite(LED0, LOW);
                                delay(200);

                                // Slow motion delay here
                                //delay(5000);

                                // Force a tea-break here if

switch is pressed

                                bool TeaBreak = false;
                                while (digitalRead(SWITCH) ==

LOW){

                                delay(100);

                                if (!TeaBreak) {

                                Serial.println("\t\t\t\t\tMgr 0 - Tea Break");

                                TeaBreak = true;

                                }

```

```
}
```

```
}
```

```
}
```

```
void loop1(void * parameter) {
```

```
    for (;;) {
```

```
        // Get the number of flashes
```

```
        required
```

```
        int flashTotal;
```

```
        xQueueReceive(queue, &flashTotal,
```

```
        portMAX_DELAY);
```

```
        Serial.println("Worker - reading "
```

```
        + String(flashTotal));
```

```
        // Flash that number
```

```
        for (int cnt = 0; cnt <
```

```
        flashTotal; cnt++) {
```

```
            digitalWrite(LED1, HIGH);
```

```
            delay(150);
```

```
            digitalWrite(LED1, LOW);
```

```
            delay(150);
```

```
}
```

```

        // Slow motion delay here

        //delay(1000);

    }

}

void setup()
{
    Serial.begin(115200);

    Serial.println("Setup started.");

    pinMode(LED0, OUTPUT);

    pinMode(LED1, OUTPUT);

    pinMode(SWITCH, INPUT_PULLUP);

    queue = xQueueCreate(5, sizeof(int)); //
    Create the queue with 5 slots of 2 bytes

    xTaskCreatePinnedToCore(
        loop0, //
        "Task0", // Name
        1000, // Stack
        NULL, // Task
        input parameter */
    of the task */
    Function to implement the task */
    size in words */
}

```

```

                                0,                                /*
Priority of the task */

                                &Task0,                        /* Task
handle. */

                                1                                /* Core
where the task should run */

                                );

```

```

                                xTaskCreatePinnedToCore(
                                loop1,                            /*
Function to implement the task */

                                "Task1",                        /* Name
of the task */

                                1000,                            /* Stack
size in words */

                                NULL,                            /* Task
input parameter */

                                0,                                /*
Priority of the task */

                                &Task1,                        /* Task
handle. */

                                1                                /* Core
where the task should run */

                                );

```

```

                                Serial.println("Setup completed.");
                                }

```

```

void loop()
{

```

```

        vTaskDelete (NULL); // ===
    delay (); min 23`50
}

```

Ejemplo 9a: Race condition, sin Mutex

// <https://www.youtube.com/watch?v=nNFhv5nyddw&list=LL&index=2&t=300s>

```
// ADTechKnow
```

```
#if CONFIG_FREERTOS_UNICORE
```

```
#define ARDUINO_RUNNING_CORE 0
```

```
#else
```

```
#define ARDUINO_RUNNING_CORE 1
```

```
#endif
```

```
// Race conditiong
```

```
//SemaphoreHandle_t mutex;
```

```
int glbvar = 0;
```

```
void task1(void *pvParameters)
```



```

{
    while (1)
    {
        //Serial.println("Task 1:
Mutex Acquired");
        // Access shared resource

        glbvar++;

        Serial.print("Task 1 - ");
        Serial.println(glbvar);

        vTaskDelay(1/
portTICK_PERIOD_MS);
        // se puede cambiar a 2
    }
}

```

```

void task2(void *pvParameters)
{
    while (1)

    {
        //Serial.println("Task 2:
Mutex Acquired");
        // Access shared resource

        glbvar++;

        Serial.print("Task 2 - ");
        Serial.println(glbvar);

        vTaskDelay(1 /
portTICK_PERIOD_MS);

    }
}

```

```
}
```

```
void setup()
{
    Serial.begin(115200);

    //mutex = xSemaphoreCreateMutex();

    xTaskCreatePinnedToCore(task1, "Task 1", 2048, NULL, 1,
NULL, ARDUINO_RUNNING_CORE);

    xTaskCreatePinnedToCore(task2, "Task 2", 2048, NULL, 1,
NULL, ARDUINO_RUNNING_CORE);
}
```

```
void loop()
{

// Do nothing

}
```

```
/*
```

```
SemaphoreHandle_t mutex;
```

```
int glbvar = 0;
```

```
// Race conditiong solucionada con mutex
```

```
void task1(void *pvParameters)
```

```

        {
            while (1)
            {
                if
(xSemaphoreTake(mutex, portMAX_DELAY) == pdTRUE)
                {

Serial.println("Task 1: Mutex Acquired");           // Access shared resource
                                                    glbvar++;

Serial.print("Task 1 - ");

Serial.println(glbvar);

                                                    vTaskDelay(1 /
portTICK_PERIOD_MS);

xSemaphoreGive(mutex);

Serial.println("Task 1: Mutex Released");

Serial.println("");

Serial.println("");

                                                    }
            }
        }

```

```

void task2(void *pvParameters)

```

```

    {
        while (1)
        {
            if (xSemaphoreTake(mutex,
portMAX_DELAY) == pdTRUE)

            {
                Serial.println("Task 2:
Mutex Acquired");          // Access shared resource

                glbvar++;

                Serial.print("Task 2 -

");

                Serial.println(glbvar);

                vTaskDelay(1 /

portTICK_PERIOD_MS);

                xSemaphoreGive(mutex);

                Serial.println("Task 2:

Mutex Released");

                Serial.println("");

                Serial.println("");

            }

        }
    }
}

```

```

void setup()
{
    Serial.begin(115200);

    mutex = xSemaphoreCreateMutex();
}

```

```
        xTaskCreatePinnedToCore(task1, "Task 1", 2048, NULL, 1, NULL,
,ARDUINO_RUNNING_CORE);
```

```
        xTaskCreatePinnedToCore(task2, "Task 2", 2048, NULL, 1, NULL,
ARDUINO_RUNNING_CORE);
```

```
    }
```

```
void loop()
```

```
{
```

```
// Do nothing
```

```
}
```

```
*/
```

Ejemplo 9b: Mutex

// Easy learning, video 13 MUTEX

//

https://www.youtube.com/watch?v=xWk1hshwGqk&list=PL-Hb9zZP9qC48GcXj_BsDipCPAzwcps6e&index=5

```
#include <stdio.h>
```

```
#include "esp_log.h"
```

```
#include "driver/gpio.h"
```

```
#include "freertos/task.h"
```

```
//#include "freertos/queue.h"
```

```
#include "freertos/semphr.h"
```

```
// A0, incluir.
```

```
#include "freertos/FreeRTOS.h"
```

```
#define ledR 2
```

```
// 33
```

```
#define ledG 25
```

```
#define ledB 26
```

```
#define R_delay 1000
```

```
#define G_delay 2000
```

```
//
```

```
#define STACK_SIZE 1024*2
```

```
// 1024*1 ERROR!!!!!!!!!!!!!!!!!!!!!!
```

```
const char *tag = "Main";
```

```
//xSemaphoreHandle_t cola_Mensaje = 0;
```

```
// A error
```

```
xSemaphoreHandle cola_Mensaje = 0;
```

```
// A, GlobalQueue
```

```
void recurso_compartido(int led)
```

```
{
```

```
    for (size_t i =0; i<8; i++)
```

```
    {
```

```
        vTaskDelay(pdMS_TO_TICKS(400));
```

```
        printf("on led\n");
```

```
        digitalWrite(led, 1);
```

```
// on led
```

```
        vTaskDelay(pdMS_TO_TICKS(400));
```

```
        printf("off led\n");
```

```
        digitalWrite(led, 0);
```

```
// off led
```

```
    }
```

```
}
```

```
void escribe1(void *pvParameter)
```

```
{
```

```
    while(1)
```

```
    {
```

```

if
(xSemaphoreTake(cola_Mensaje, pdMS_TO_TICKS(100))) // C. Usa
el recurso dos veces seguidas

{

printf("La tarea
ESCRIBE tomo el recurso compartido\n");

//ESP_LOGE(tag,
"Fallo al enviar %i a la cola", i); // Error

//ESP_LOGI(tag,
"Fallo al enviar %i a la cola", i); // Information

//ESP_LOGW(tag,
"Fallo al enviar %i a la cola", i); // Warning

recurso_compartido(ledR);

xSemaphoreGive(cola_Mensaje); // D

}

//vTaskDelay(2000 /
portTICK_RATE_MS); // Retardo para poder ver
el mensaje

vTaskDelay(pdMS_TO_TICKS(R_delay));

}

}

```

```

void lee1(void *pvParameter)

{

```



```

        while(1)
        {
            if (xSemaphoreTake(cola_Mensaje,
pdMS_TO_TICKS(100)))
                // C
            {
                printf("La tarea LEE1 tomo el
recurso compartido\n");

                //ESP_LOGE(tag, "Fallo al
enviar %i a la cola", i);

                recurso_compartido(ledG);
                xSemaphoreGive(cola_Mensaje);

                // D
            }

            //vTaskDelay(2000 /
            portTICK_RATE_MS);
            poder ver el mensaje
            // Reatardo para

            vTaskDelay(pdMS_TO_TICKS(G_delay));
        }
    }
}

```

```

void setup()
{
    Serial.begin(115200);
    pinMode(ledR, OUTPUT);
    pinMode(ledG, OUTPUT);
    pinMode(ledB, OUTPUT);
}

```

```

        cola_Mensaje= xQueueCreate(TAM_COLA, TAM_MSG);
// B, Crea la cola

        cola_Mensaje= xSemaphoreCreateMutex();
// B, Crea el mutex


        xTaskCreate(&lee1,    "lee1",    STACK_SIZE, NULL, 1,
NULL);
                        // Igual prioridad

        xTaskCreate(&escribe1, "escribe1", STACK_SIZE, NULL, 1,
NULL);
                        //

    }


void loop()
{
    vTaskDelete(NULL);
// Recomendación para arduino !!!!!!! se puede demorar el doble.

}

```

Ejemplo 9c: Dos núcleos, asíncronos, sin MUTEX

*

* This blinks two LEDs independently and not synchronized. Both have other blink frequencies.

* The blink sketches run in two tasks and on two cores.

*/

```
#define LED1 2
```

```
#define LED2 14
```

```
TaskHandle_t Task1, Task2;
```

```
int counter = 0;
```

```
void blink(byte pin, int duration) {
```

```
    digitalWrite(pin, HIGH);
```

```
    delay(duration);
```

```
    digitalWrite(pin, LOW);
```

```
    delay(duration);
```

```
}
```

```
void codeForTask1( void * parameter )
```

```
{
```

```
    for (;;) {
```

```

        blink(LED1, 1000);

        delay(50);

        Serial.println("Task 1:
");
    }

}

void codeForTask2( void * parameter )
{
    for (;;) {

        blink(LED2, 1200);

        delay(50);

        Serial.println("
Task 2: ");
    }
}

void setup() {
    Serial.begin(115200);

    pinMode(LED1, OUTPUT);

    pinMode(LED2, OUTPUT);

    xTaskCreatePinnedToCore(codeForTask1,"led1Task",1000,NULL,1,&Task1,0);

    delay(500); // needed to start-up task1

    xTaskCreatePinnedToCore(codeForTask2,"led2Task",1000,NULL,1,&Task2,1);
}

```

```
void loop() {  
    delay(1000);  
}
```

Ejemplo 9d: Dos núcleos síncronos con MUTEX

serie/paralelo

```
/*  
    This sketch uses semaphores to synchronize two LEDs. The blink sketches  
    run in two tasks and on two cores.  
  
    As shown in the video you can either blink them sequentially or in  
    parallel  
*/  
  
#define LED1 2  
#define LED2 14  
  
TaskHandle_t Task1, Task2;  
SemaphoreHandle_t baton;  
  
int counter = 0;  
  
void blink(byte pin, int duration) {  
    digitalWrite(pin, HIGH);  
    delay(duration);  
    digitalWrite(pin, LOW);  
    delay(duration);  
}
```

```

    }

void codeForTask1( void * parameter )
{
    for (;;) {

        xSemaphoreTake(baton,
portMAX_DELAY);

        //xSemaphoreGive(baton);

        // En paralelo si se descomenta

        Serial.print("millis Task
1: ");

        Serial.println(millis());
        blink(LED1,1000);
        counter++;

        xSemaphoreGive(baton);

        // En serie si se descomenta

        delay(10);

        // Evita que la misma tarea arranque nuevamente....

        Serial.print("Counter in
Task 1: ");

        Serial.println(counter);

        //counter++;

    }
}

void codeForTask2( void * parameter )
{
    for (;;) {

```

```

portMAX_DELAY);

millis Task 2: ");

Counter in Task 2: ");

xSemaphoreTake(baton,

Serial.print("

Serial.println(millis());

blink(LED2,1200);

counter++;

xSemaphoreGive(baton);

delay(10);

Serial.print("

Serial.println(counter);

//counter++;

}

}

```

```

void setup() {

    Serial.begin(115200);

    pinMode(LED1, OUTPUT);

    pinMode(LED2, OUTPUT);

    baton = xSemaphoreCreateMutex();

    // A viewer suggested to use : &codeForTask1, because
his ESP crashed

    xTaskCreatePinnedToCore(codeForTask1,"led1Task",1000,NULL,1,&Task1,0);

```



```
                                                                    //delay(500);  
  
// needed to start-up task1  
  
xTaskCreatePinnedToCore(codeForTask2,"led2Task",1000,NULL,1,&Task2,1);  
    }  
  
void loop() {  
    delay(10);  
}
```

Ejemplo 9e: Uso del monitor serial con tareas de diferente prioridad con MUTEX

```
// https://www.youtube.com/watch?v=pWz-yddfBy8
```

```
// Vladimir Trujillo
```

```
#include <stdio.h>
```

```
#include "esp_log.h"
```

```
//#include "driver/gpio.h"
```

```
#include "freertos/task.h"
```

```
//#include "freertos/queue.h"
```

```
#include "freertos/semphr.h"
```

```
// A0, incluir.
```

```
#include "freertos/FreeRTOS.h"
```

```
// A0, incluir.
```

```
#define ledR 2
```

```
#define pulsador 0
```

```
#define STACK_SIZE 1024
```

```
// 1024
```

```

//SemaphoreHandle_t Global_Key = 0;           m
// A error

xSemaphoreHandle semaforo = 0;
// A, GlobalQueue === Testigo

//void leeSW(void *pvParameters);

//void leeADC(void *pvParameters);

void leeSW(void *pvParameter)
// Se empieza a ejecutar, no tiene restricciones. Da la llave a la tarea
suspendida

{
    //unsigned int pulsador = 0;
    //pinMode(pulsador, INPUT_PULLUP);

    while(1)
    {
        int estado_SW =
digitalRead(pulsador);

        if
(xSemaphoreTake(semaforo, (TickType_t) 5 == pdTRUE))           // C, SE
PUEDE COMENTAR #####.

// Si el testigo es uno, se ejecuta y decrementa el testigo.

{
    Serial.print("-- El
estado del pulsador es igual a : ");

    Serial.println(estado_SW);

```

```

xSemaphoreGive(semaforo); // Pone
el puerto serie disponible, incrementa el semaforo a 1

    }

    vTaskDelay(1);

// Necesario para estabilidad

//vTaskDelay(pdMS_TO_TICKS(R_delay));

    }

}

void leeADC(void *pvParameter)
{
    while(1)
    {
        int lectura_ADC = analogRead(A0);

        if (xSemaphoreTake(semaforo,
pdMS_TO_TICKS(5))) // C, SE PUEDE COMENTAR
#####

        //if (xSemaphoreTake(semaforo,
(TickType_t) 5 == pdTRUE))

        //if (xSemaphoreTake(semaforo,
        // C, Si el testigo es uno,
        se ejecuta y decrementa el testigo. portMAX_DELAY = 0xffffffff UL == 2^32
        == 4,29 e9 = (12 horas??), 49,7 días

        {

            Serial.print("La lectura del
canal analogico cero es igual a : ");

            Serial.println(lectura_ADC);

            xSemaphoreGive(semaforo);

```

```

    }

    vTaskDelay(2);
    // 10 o más... Si es 1 mS, solo se ejecuta esta tarea, más prioridad.

    //vTaskDelay(2000 /
    portTICK_RATE_MS); // Retardo para
    poder ver el mensaje

    //vTaskDelay(pdMS_TO_TICKS(G_delay));

    //vTaskDelay(pdMS_TO_TICKS(10));
}

}

```

```

void setup()
{
    Serial.begin(115200);
    //Serial.begin(9600);
    pinMode(ledR, OUTPUT);
    pinMode(pulsador, INPUT_PULLUP);
    if (semaforo == NULL)
    {
        //semaforo = xSemaphoreCreateBinary();
        // B. Crea el semáforo binario. Arranca en cero

        semaforo = xSemaphoreCreateMutex();
        // B. Crea el semáforo MUTEX.

        if (semaforo != NULL)
            xSemaphoreGive(semaforo);
        // Pone el puerto serie disponible
    }
}

```

```
}
```

```
        //xTaskCreate(leeSW, "leesw", STACK_SIZE, NULL, 1, NULL);  
// Baja prioridad  
        //xTaskCreate(leeADC, "leeadc", STACK_SIZE, NULL, 1, NULL);  
// Alta prioridad  
        xTaskCreate(&leeSW, "leesw", STACK_SIZE, NULL, 1, NULL);  
// Baja prioridad  
        xTaskCreate(&leeADC, "leeadc", STACK_SIZE, NULL, 2, NULL);  
// Alta prioridad  
    }
```

```
void loop()  
{  
    vTaskDelete(NULL);  
// Recomendación para arduino !!!!!!! se puede demorar el doble.  
}
```

Semáforos

Ejemplo 10a: Semáforos binarios

```
// Easy learning, video 14 Semáforos binarios  
  
//  
https://www.youtube.com/watch?v=8dGv1kRkLPs&list=PL-Hb9zZP9qC48GcXj\_BsDipCPA  
zwcps6e&index=6
```

```
#include <stdio.h>  
  
#include "esp_log.h"  
  
#include "driver/gpio.h"  
  
#include "freertos/task.h"  
  
// #include "freertos/queue.h"  
  
#include "freertos/semphr.h"  
// A0, incluir.  
  
#include "freertos/FreeRTOS.h"
```

```
#define ledR 2  
  
#define ledG 25  
  
#define ledB 26  
  
#define R_delay 10000  
// Con 1000 se repite la rutina, hay interrupciones
```

```

#define G_delay 2000
//

#define STACK_SIZE 1024*2
// 1024*1 ERROR!!!!!!!!!!!!!!!!!!!!!!

const char *tag = "Main";

//SemaphoreHandle_t Global_Key = 0;           m
// A error

xSemaphoreHandle Global_Key = 0;
// A, GlobalQueue === Testigo

/*
void recurso_compartido(int led)
{
    for (size_t i =0; i<8; i++)
    {
        vTaskDelay(pdMS_TO_TICKS(400));
        printf("on led\n");
        // on led
        vTaskDelay(pdMS_TO_TICKS(400));
        printf("off led\n");
        // off led
    }
}

*/

```



```

void escribe1(void *pvParameter)
// Se empieza a ejecutar, no tiene restricciones. Da la llave a la tarea
suspendida

{

    while(1)
    {
        for (int i =0; i<8; i++)
        {

vTaskDelay(pdMS_TO_TICKS(400));

printf("Enviando %i
semaforo_Binario\n", i);

digitalWrite(ledR, 1);

// on led

vTaskDelay(pdMS_TO_TICKS(400));

digitalWrite(ledR, 0);

// off led

// ESP_LOGW(tag,
"enviando %i a la cola", i);

//v TaskDelay(2000 /
// Retardo para poder ver el
mensaje

}

printf("La tarea ESCRIBE
entrega la llave\n");

xSemaphoreGive(Global_Key);

// B, incrementa el semaforo a 1

vTaskDelay(pdMS_TO_TICKS(R_delay));

}

```

```
}
```

```
void lee1(void *pvParameter)
{
    while(1)
    {
        //if (xSemaphoreTake(Global_Key,
pdMS_TO_TICKS(100)))
        // C

        if (xSemaphoreTake(Global_Key,
portMAX_DELAY))
        // C, Si el testigo es uno,
se ejecuta y decrementa el testigo. portMAX_DELAY = 0xffffffff UL == 2^32
=== 4,29 e9 = (12 horas??), 49,7 dias

        {
            printf("La tarea LEE1 se
desperto y toma la llave\n");

            for (int i =0; i<8; i++)
            {

vTaskDelay(pdMS_TO_TICKS(200));

            printf("LEE1
trabajando\n");

            // on led

vTaskDelay(pdMS_TO_TICKS(200));

            // off led
```

```

// ESP_LOGW(tag,
"enviando %i a la cola", i);

//v TaskDelay(2000 /
portTICK_RATE_MS); // Retardo para poder ver el
mensaje

}

printf("La tarea LEE1 se va a
dormir\n");

}

//vTaskDelay(2000 /
portTICK_RATE_MS); // Reatardo para
poder ver el mensaje

//vTaskDelay(pdMS_TO_TICKS(G_delay));

vTaskDelay(pdMS_TO_TICKS(10));

}

}

void setup()
{
    Serial.begin(115200);

    pinMode(ledR, OUTPUT);

    Global_Key = xSemaphoreCreateBinary();
    // B. Crea el semáforo binario. Arranca en cero

    xTaskCreate(&lee1, "lee1", STACK_SIZE, NULL, 1,
NULL); //

```

```
        xTaskCreate(&escribe1, "escribe1", STACK_SIZE, NULL, 1,  
NULL);  
        //  
    }
```

```
void loop()  
{  
    vTaskDelete(NULL);  
    // Recomendación para arduino !!!!!!! se puede demorar el doble.  
}
```

Ejemplo 10b: Semáforos binarios, tarea con argumento.

```
// xSemaphoreTake(bin_sem, portMAX_DELAY); // Do  
nothing until binary semaphore has been returned, xSemaphoreGive(bin_sem) le  
suma uno
```

```
// Digikey 7
```

```
// https://www.youtube.com/watch?v=5JcMtbA9QEE&t=7s
```

```
// ok, 26/09/2023
```

```
// Sending: 1200
```

```
// Received: 1200
```

```
// Done!
```

```
/**
```

```
* FreeRTOS Binary Semaphore Demo
```

```
*
```

```
* Pass a parameter to a task using a binary semaphore.
```

```
*
```

```
* Date: January 23, 2021
```

```
* Author: Shawn Hymel
```

```
* License: 0BSD
```

```

*/

// You'll likely need this on vanilla FreeRTOS

//#include <semphr.h>

// Use only core 1 for demo purposes

#if CONFIG_FREERTOS_UNICORE

    static const BaseType_t app_cpu = 0;

#else

    static const BaseType_t app_cpu = 1;

#endif

// Pins (change this if your Arduino board does not have LED_BUILTIN
defined)

//static const int led_pin = LED_BUILTIN;

static const int led_pin = 2;

// Globals

static SemaphoreHandle_t bin_sem;

//*****
***

// Tasks

// Blink LED based on rate passed by parameter

void blinkLED(void *parameters)

{

```

```

// Copy the parameter into a local
variable

int num = *(int *)parameters;

//xSemaphoreGive(bin_sem);
// Release the binary semaphore so that the creating function can finish, le
suma uno

Serial.print("Received: ");

// Print the parameter

Serial.println(num);

pinMode(led_pin, OUTPUT);

while (1)

// Blink forever and ever

{

    digitalWrite(led_pin, HIGH);

    vTaskDelay(num /

portTICK_PERIOD_MS);

    digitalWrite(led_pin, LOW);

    vTaskDelay(num /

portTICK_PERIOD_MS);

    xSemaphoreGive(bin_sem);

// Release the binary semaphore so that the creating function can finish, le
suma uno

}

}

//*****
***

```

```

// Main (runs as its own task with priority 1 on core 1)

void setup()
{
    long int delay_arg;

    Serial.begin(115200);

    vTaskDelay(1000 / portTICK_PERIOD_MS);

    Serial.println();

    Serial.println("---FreeRTOS Mutex Solution---");

    Serial.println("Enter a number for delay (milliseconds)");


    while (Serial.available() <= 0);
// Wait for input from Serial

    delay_arg = Serial.parseInt();
// Read integer value

    Serial.print("Sending: ");

    Serial.println(delay_arg);


    bin_sem = xSemaphoreCreateBinary();
// Create binary semaphore before starting tasks, se inicializa en cero, no
se necesita tomarlo take


    xTaskCreatePinnedToCore(blinkLED,
// Start task 1

                            "Blink LED",

                            1024,

                            (void *)&delay_arg,

```



```

        1,
        NULL,
        app_cpu);

        xSemaphoreTake(bin_sem, portMAX_DELAY);
// Do nothing until binary semaphore has been returned,
xSemaphoreGive(bin_sem) le suma uno

        Serial.println("Done!");
// Show that we accomplished our task of passing the stack-based argument,
no vuelve a leer el puerto serial.

    }

void loop()
{
    vTaskDelay(1000 / portTICK_PERIOD_MS);
// Do nothing but allow yielding to lower-priority tasks

}

```

Ejemplo 10c: Semáforos CONTADORES, tareas con argumento.

Tomado de la IDE de Arduino

Counting_Semaphore_IDE.ino

```
/* Basic Multi Threading Arduino Example
```

```
   This example code is in the Public Domain (or CC0 licensed, at your option.)
```

```
   Unless required by applicable law or agreed to in writing, this software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```
*/
```

```
// Please read file README.md in the folder containing this example.
```

```
#include <Arduino.h>
```

```
SemaphoreHandle_t package_delivered_semaphore;
```

```
void delivery_truck_task(void *pvParameters)
```

```
{
```

```
    int truck_number = (int) pvParameters;
```

```
    while(1)
```

```
    {
```

```
        // Wait for a package to be delivered
```

```
        // ...
```

```

        // Notify the warehouse that a package has been
delivered

        xSemaphoreGive(package_delivered_semaphore);

        Serial.printf("Package delivered by truck: %d\n",
truck_number);

        //wait for some time

        vTaskDelay(1000 / portTICK_PERIOD_MS);

    }

}

void warehouse_worker_task(void *pvParameters)
{
    int worker_number = (int) pvParameters;

    while(1)
    {
        // Wait for a package to be delivered

        xSemaphoreTake(package_delivered_semaphore,
portMAX_DELAY);

        Serial.printf("Package received by worker: %d\n",
worker_number);

        // Receive the package

        // ...

    }

}

void setup()
{
    Serial.begin(115200);

```

```

        while(!Serial){ delay(100); }

// Create the counting semaphore

package_delivered_semaphore = xSemaphoreCreateCounting(10, 0);

// Máxima cuenta y valor inicial

// Create multiple delivery truck tasks

for (int i = 0; i < 5; i++)
{
    xTaskCreate(delivery_truck_task, "Delivery Truck", 2048,
(void *)i, tskIDLE_PRIORITY, NULL);
}

// Create multiple warehouse worker tasks

for (int i = 0; i < 3; i++) {
    xTaskCreate(warehouse_worker_task, "Warehouse Worker", 2048,
(void *)i, tskIDLE_PRIORITY, NULL);
}

}

void loop()
{
    // Empty loop
}

```

Ejemplo 10d: Semáforos binarios, con TMR0, ver ejemplo 13b.

Interrupciones

Ejemplo 11a: Interrupción por GPIO

```
#include <Arduino.h>

//
// De Arduino

struct Button {

    const uint8_t PIN;

    uint32_t numberKeyPresses;

    bool pressed;

};

Button button1 = {0, 0, false};
// GPIO 0

Button button2 = {15, 0, false};
// GPIO 15

void IRAM_ATTR isr(void* arg) {
    // Elaborada

    Button* s = static_cast<Button*>(arg);

    s->numberKeyPresses += 1;

    s->pressed = true;

}
```

```

/*

void IRAM_ATTR isr()    {
// Convencional

        button1.numberKeyPresses += 1;

        button1.pressed = true;

    }

*/


void IRAM_ATTR isr_INT() {
// Convencional

        button2.numberKeyPresses += 1;

        button2.pressed = true;

    }


void setup() {

    Serial.begin(115200);

    pinMode(button1.PIN, INPUT_PULLUP);

    pinMode(button2.PIN, INPUT_PULLUP);


    attachInterruptArg(button1.PIN, isr, &button1, FALLING);
// Elaborada (1)

    //attachInterrupt(button1.PIN, isr, FALLING);


    attachInterrupt(button2.PIN, isr_INT, FALLING);
// Convencional

    }

```

```

void loop() {

    if (button1.pressed) {

        Serial.printf("Button 1 has been
pressed %u times\n", button1.numberKeyPresses);

        button1.pressed = false;

    }

    if (button2.pressed) {

        Serial.printf("Button 2 has been
pressed %u times\n", button2.numberKeyPresses);

        button2.pressed = false;

    }

    static uint32_t lastMillis = 0;

    if (millis() - lastMillis > 10000) {

        lastMillis =
millis();

detachInterrupt(button1.PIN);    // El button1, deja de servir como
interrupción después de 10 segundos.

    }

}

```


Ejemplo 11b: Interrupción por GPIO, con semáforo binario.

```
// https://www.youtube.com/watch?v=XFTvT82xsAo
```

```
// Biblioman
```

```
// OK, 25/04/2024
```

```
/* 3. Ejemplo: Interrupciones y semáforo binario para sincronizar tareas.
```

```
*/
```

```
#include <stdio.h>
```

```
#include "driver/gpio.h"
```

```
#include "freertos/FreeRTOS.h"
```

```
#include "freertos/task.h"
```

```
#include "esp_system.h"
```

```
#include "freertos/semphr.h"
```

```
// A
```

```
#define ESP_INTR_FLAG_DEFAULT 0
```

```
#define PULSADOR 0
```

```
SemaphoreHandle_t xSemaphore = NULL;
```

```
// B, Se crea el manejador para el semáforo como variable global
```

```

void IRAM_ATTR isr_INT() {

    xSemaphoreGiveFromISR(xSemaphore, NULL);
    // C, Da el semáforo para que quede libre para la tarea pulsador. Le suma 1

}

```

```

void task_pulsador(void* arg)
{

    while(1)
    {

        if(xSemaphoreTake(xSemaphore,portMAX_DELAY) == pdTRUE)    // D, Si es
cero, espero por la notificación de la ISR. Si es uno, le resta uno y
ejecuta la tarea.

        {

            printf("Pulsador
presionado!\n");

        }

    }

}

```

```
void setup()
{
    Serial.begin(115200);

    pinMode(PULSADOR, INPUT_PULLUP);
    attachInterrupt(PULSADOR, isr_INT, FALLING);

    xSemaphore = xSemaphoreCreateBinary();
    // E, se crea el semáforo binario. Arranca en cero.

    xTaskCreate(task_pulsador, "task_pulsador", 2048, NULL, 5,
    NULL); // F, crea la tarea task_pulsador
}

void loop()
{
    //delay(1000);

    vTaskDelete(NULL);
}
```

Ejemplo 12: Interrupción por TOUCH

/ *

This is an example of how to use Touch Interrupts

The bigger the threshold, the more sensible is the touch

 $\ast/$

```
// Arduino, 21/septiembre/2021
```

```
int threshold = 55;
```

```
volatile bool touch1detected = false;
```

```
volatile bool touch2detected = false;
```

```
void IRAM_ATTR gotTouch1(){
```

```
touch1detected = true;
```

}

```
void IRAM_ATTR gotTouch2(){
```

```
touch2detected = true;
```

}

```
void setup() {
```

```
Serial.begin(115200);
```

```
delay(1000);
```

```
me time to bring up serial monitor
```

```
// give
```

```
Serial.println("ESP32 Touch Interrupt Test");
```

```
        touchAttachInterrupt(T0, gotTouch1, threshold);        // GPIO
4
        touchAttachInterrupt(T3, gotTouch2, threshold);        // GPIO
15
    }

void loop(){
    if(touch1detected){
        touch1detected = false;
        Serial.println("Touch 1 detected");
        delay(200);
    }
    if(touch2detected){
        touch2detected = false;
        Serial.println("Touch 2 detected");
        delay(200);
    }
}
```

Ejemplo 13a: Interrupción por TIMER

```
// 2017 pcbreflux
```

```
# define LED_0 2
```

```
// Build in
```

```
// ESP32 CAM GPIO 4
```

```
# define LED_1 17
```

```
# define pulsador 0
```

```
// Boot del esp32
```

```
bool b;
```

```
volatile uint8_t ledstat = 0;
```

```
long ticks = 1000000;
```

```
hw_timer_t *timer1 = NULL;
```

```
// Apuntador, Primer paso
```

```
portMUX_TYPE hab_desh_INT =
```

```
portMUX_INITIALIZER_UNLOCKED;////////////////////////////////////NEW!!!!!!!
|||||
```

```

void IRAM_ATTR funcion_interrupcion()
{
    // Se necesita el atributo Interrupt RAM IRAM, no debería tener código
    bloqueante!!!!!!!!!!!!!!!!!!!!!!!!!!!! Segundo paso

    portENTER_CRITICAL_ISR(&hab_desh_INT);

    timerAlarmWrite (timer1, ticks, true);
    // Para cargar una cuenta diferente. ticks e3 uS, autoreload == true
    C

    /*

        Serial.println(String("onTimer() ") +
String(millis())); // no debería tener código
bloqueante!!!!!!!!!!!!!!!!!!!!!!!!!!!!

        Serial.println(String("led status ") +
String(ledstat)); // no debería tener código
bloqueante!!!!!!!!!!!!!!!!!!!!!!!!!!!!

        Serial.println(String("\t\t\t\t pulsador ") +
String(b)); // no debería tener código bloqueante!!!!!!!!!!!!!!!!!!!!!!!!!!!!

        Serial.println(String("ticks ") + String(ticks));

    */

    ledstat = 1 - ledstat;

    digitalWrite(LED_0, ledstat);

    // turn the LED on or off

    digitalWrite(LED_1, ledstat);

    // turn the LED on or off

    portEXIT_CRITICAL_ISR(&hab_desh_INT);

}

void setup(){

    Serial.begin(115200);

```

```

    pinMode(LED_0, OUTPUT);

    pinMode(LED_1, OUTPUT);

    pinMode(pulsador, INPUT_PULLUP);


    Serial.println("start timer ");


    timer1 = timerBegin(0, 80, true);
// timer canal 0, PS === 80 MHz === 12,5 ns; 12,5 ns*80 === 1 us, countUp
A

    timerAttachInterrupt(timer1, &funcion_interrupcion, true);
// edge (not level) triggered
B

    timerAlarmWrite(timer1, ticks, true);
// 1000000 * 1 us = 1 s, autoreload true
C

    timerAlarmEnable(timer1);
// enable
D

}

void loop() {

    b = digitalRead(pulsador);

    if (b == LOW){

        ticks = 100000;

    }

    else {

        ticks = 1000000;

    }

}

```



```
        //timerWrite (timer1, 0);

        //vTaskDelay(portMAX_DELAY);
// wait as much as possible ...Si se coloca código, se debe quitar!

    }
```

Ejemplo 13b: Interrupción por GPIO con semáforo

TMR_Semaforo.ino

```
/*
```

```
Repeat timer example
```

```
This example shows how to use the hardware timer in ESP32. The timer calls  
onTimer
```

```
function every second. The timer can be stopped with button attached to PIN  
0
```

```
(IO0).
```

```
This example code is in the public domain.
```

```
*/
```

```
#define LED_0 2
```

```
#define BTN_STOP_ALARM 0
```

```
// Stop button is attached to PIN 0 (IO0)
```

```
volatile uint8_t ledstat = 0;
```

```
hw_timer_t * timer = NULL;

volatile SemaphoreHandle_t timerSemaphore;

portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

```
volatile uint32_t lastIsrAt = 0;

volatile uint32_t isrCounter = 0;
```

```
void ARDUINO_ISR_ATTR onTimer()

{

    portENTER_CRITICAL_ISR(&timerMux);
    // Increment the counter and set the time of ISR

    isrCounter++;

    lastIsrAt = millis();

    portEXIT_CRITICAL_ISR(&timerMux);

    xSemaphoreGiveFromISR(timerSemaphore, NULL);
    // Give a semaphore that we can check in the loop

    ledstat = 1 - ledstat;
    // It is safe to use digitalWrite, Write here if you want to toggle an output

    digitalWrite(LED_0, ledstat);
    // turn the LED on or off
```

```
}
```

```
void setup()
```

```
{
```

```
    Serial.begin(115200);
```

```
    pinMode(LED_0, OUTPUT);
```

```
    pinMode(BTN_STOP_ALARM, INPUT);
```

```
// Set BTN_STOP_ALARM to input mode
```

```
    //pinMode(BTN_STOP_ALARM, INPUT_PULLUP);
```

```
    timerSemaphore = xSemaphoreCreateBinary();
```

```
// Create semaphore to inform us when the timer has fired
```

```
    timer = timerBegin(0, 80, true);
```

```
// Use 1st timer of 4 (counted from zero), Set 80 divider for prescaler,  
countup
```

```
    timerAttachInterrupt(timer, &onTimer, true);
```

```
// Attach onTimer function to our timer, edge triggered.
```

```
    timerAlarmWrite(timer, 1000000, true);
```

```
// Repeat the alarm (third parameter), Set alarm to call onTimer function  
every second (value in microseconds), autoreload.
```

```
    timerAlarmEnable(timer);
```

```
// Start an alarm
```

```
}
```

```
void loop()
```

```
{
```

```

        if (xSemaphoreTake(timerSemaphore, 0) == pdTRUE)
// If Timer has fired

    {

        uint32_t isrCount = 0, isrTime = 0;


        portENTER_CRITICAL(&timerMux);
// Read the interrupt count and time

        isrCount = isrCounter;

        isrTime = lastIsrAt;

        portEXIT_CRITICAL(&timerMux);


        Serial.print("onTimer no. ");
        Serial.print(isrCount);
        Serial.print(" at ");
        Serial.print(isrTime);
        Serial.println(" ms");
    }

    if (digitalRead(BTN_STOP_ALARM) == LOW)
    {
        if (timer)
// If timer is still running

        {

            timerEnd(timer);
// Stop and free timer

            timer = NULL;
        }
    }

```


Ejemplo 13c: Interrupción por TIMER con semáforo

TMR_Semaforo.ino

```
/*
```

```
Repeat timer example
```

```
This example shows how to use the hardware timer in ESP32. The timer calls  
onTimer
```

```
function every second. The timer can be stopped with button attached to PIN  
0
```

```
(IO0).
```

```
This example code is in the public domain.
```

```
*/
```

```
#define LED_0 2
```

```
#define BTN_STOP_ALARM 0
```

```
// Stop button is attached to PIN 0 (IO0)
```

```
volatile uint8_t ledstat = 0;
```

```
hw_timer_t * timer = NULL;

volatile SemaphoreHandle_t timerSemaphore;

portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

```
volatile uint32_t lastIsrAt = 0;

volatile uint32_t isrCounter = 0;
```

```
void ARDUINO_ISR_ATTR onTimer()

{

    portENTER_CRITICAL_ISR(&timerMux);
    // Increment the counter and set the time of ISR

    isrCounter++;

    lastIsrAt = millis();

    portEXIT_CRITICAL_ISR(&timerMux);

    xSemaphoreGiveFromISR(timerSemaphore, NULL);
    // Give a semaphore that we can check in the loop

    ledstat = 1 - ledstat;
    // It is safe to use digitalWrite, Write here if you want to toggle an output

    digitalWrite(LED_0, ledstat);
    // turn the LED on or off
```



```
}
```

```
void setup()
```

```
{
```

```
    Serial.begin(115200);
```

```
    pinMode(LED_0, OUTPUT);
```

```
    pinMode(BTN_STOP_ALARM, INPUT);
```

```
// Set BTN_STOP_ALARM to input mode
```

```
    //pinMode(BTN_STOP_ALARM, INPUT_PULLUP);
```

```
    timerSemaphore = xSemaphoreCreateBinary();
```

```
// Create semaphore to inform us when the timer has fired
```

```
    timer = timerBegin(0, 80, true);
```

```
// Use 1st timer of 4 (counted from zero), Set 80 divider for prescaler,  
countup
```

```
    timerAttachInterrupt(timer, &onTimer, true);
```

```
// Attach onTimer function to our timer, edge triggered.
```

```
    timerAlarmWrite(timer, 1000000, true);
```

```
// Repeat the alarm (third parameter), Set alarm to call onTimer function  
every second (value in microseconds), autoreload.
```

```
    timerAlarmEnable(timer);
```

```
// Start an alarm
```

```
}
```

```
void loop()
```

```
{
```

```

        if (xSemaphoreTake(timerSemaphore, 0) == pdTRUE)
// If Timer has fired

    {

        uint32_t isrCount = 0, isrTime = 0;


        portENTER_CRITICAL(&timerMux);
// Read the interrupt count and time

        isrCount = isrCounter;

        isrTime = lastIsrAt;

        portEXIT_CRITICAL(&timerMux);


        Serial.print("onTimer no. ");

        Serial.print(isrCount);

        Serial.print(" at ");

        Serial.print(isrTime);

        Serial.println(" ms");

    }

    if (digitalRead(BTN_STOP_ALARM) == LOW)

    {

        if (timer)
// If timer is still running

        {

            timerEnd(timer);
// Stop and free timer

            timer = NULL;

        }

```


Ejemplo 14: PWM

```
// Tomado de arduino, like analogWrite

#define LEDC_CHANNEL_0    0
// use first channel of 16 channels (started from zero)

#define LEDC_TIMER_13_BIT 13
// use 13 bit precision for LEDC timer

#define LEDC_BASE_FREQ    5000
// use 5000 Hz as a LEDC base frequency


#define LED_PIN1          2
// Build in

#define LED_PIN2          15

#define LED_PIN3          26


int brightness = 0;
// how bright the LED is

int fadeAmount = 5;
// how many points to fade the LED by

//int valueMax = 255;
// Rápido. valuemax = Resolución =  $2^8 - 1 = 255$  == Rápido
```

```

int valueMax = 8191;
// Lento,  valuemax = Resolución = 2 ^ 13 - 1 = 8191 == lento

// value has to be between 0 and valueMax


void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax){

uint32_t duty = (8191 / valueMax) * min(value, valueMax); // calculate duty,
8191 from 2 ^ 13 - 1

ledcWrite(channel, duty);                                // write duty to
LEDC                                                    C

                                                    }

void setup() {

    Serial.begin(115200);

    pinMode(LED_PIN1, OUTPUT);

    pinMode(LED_PIN2, OUTPUT);

    pinMode(LED_PIN3, OUTPUT);


    ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ,
LEDC_TIMER_13_BIT);    // Setup timer and attach timer to a led pin
A

    ledcAttachPin(LED_PIN1, LEDC_CHANNEL_0);

//
B

    ledcAttachPin(LED_PIN2, LEDC_CHANNEL_0);

```

```

        ledcAttachPin(LED_PIN3, LEDC_CHANNEL_0);
    }

void loop() {
    ledcAnalogWrite(LEDC_CHANNEL_0, brightness, valueMax);
    // set the brightness on LEDC channel 0

    brightness = brightness + fadeAmount;
    // change the brightness for next time through the loop

    Serial.print(String("brightness = "));
    Serial.println(brightness);

    if (brightness <= 0 || brightness >= valueMax){
        // reverse the direction of the fading at the ends of the fade:

        fadeAmount =
        -fadeAmount;
    }

    delay(30);
    // wait for 30 milliseconds to see the dimming effect

}

```

Ejemplo 15a: WDT; Perro guardián

```
// 23/septiembre/2021
```

// 18/enero/2023

```
// De arduino: ejemplos/ESP32/TIMERS/WDT
```

```
#include "esp_system.h"
```

```
// No se necesita
```

```
const int button = 0;
```

```
// gpio to use to trigger delay
```

```
const int wdtTimeout = 3000;
```

```
// time in ms to trigger the watchdog
```

```
hw_timer_t *timer = NULL;
```

```
// timer == nombre de la función (*timer)
```

A

```
void IRAM_ATTR resetModule() {
```

```
// función resetModule === nombre de la función de interrupción
```

C

```
ets_printf("\t\t\t\t\tToco morder,  
reboot\n");
```

```
esp_restart();
```

//

C*1

```
//esp_restart_noos();
```

// <https://www.youtube.com/watch?v=7kLy2iwIvy8>

```
}
```

```
void setup() {  
  
    Serial.begin(115200);  
  
    Serial.println();  
  
    Serial.println("running setup");  
  
    pinMode(button, INPUT_PULLUP);  
    // init control pin  
  
  
    timer = timerBegin(0, 80, true);  
    // timer === nombre de la función, canal 0, PS = 80 (1 MHZ), countup  
    B*4  
  
    timerAttachInterrupt(timer, &resetModule, true);  
    // timer === nombre de la función, attach callback a la funcion resetModule  
  
    timerAlarmWrite(timer, wdtTimeout * 1000, false);  
    // timer === nombre de la función, set time in us (true === autoreload)  
  
    timerAlarmEnable(timer);  
    // timer === nombre de la función, enable interrupt  
  
}
```

```
void loop()  
  
    {  
  
        Serial.println("running main loop == alimentando el WDT");  
  
        timerWrite(timer, 0);  
        // reset timer (feed watchdog)  
        D  
  
        long loopTime = millis();  
  
        //while button is pressed, delay up to 3 seconds to trigger  
        the timer
```



```
        while (!digitalRead(button)) {  
            Serial.println("button pressed  
=== perrito con ganas de morder"); // Si se oprime mucho tiempo, se  
reseta!!!!!!  
  
            delay(500);  
        }  
  
        delay(1000);  
        // simulate work  
  
        loopTime = millis() - loopTime;  
        Serial.print("loop time is = ");  
  
        Serial.println(loopTime);  
        // should be under 3000  
    }  
}
```

Ejemplo 15b: WDT, FreeRTOS

<https://www.youtube.com/watch?v=C2xF306qkbg>

//

<https://github.com/nkolban/esp32-snippets/blob/master/tasks/watchdogs/main.c>

pp

```
#include "freertos/FreeRTOS.h"
```

```
#include "esp_wifi.h"
```

```
#include "esp_system.h"
```

```
#include "esp_event.h"
```

```
#include "esp_event_loop.h"
```

```
#include "nvs_flash.h"
```

```
#include "driver/gpio.h"
```

```
#include <esp_task_wdt.h>
```

```
/*
```

```
esp_err_t event_handler(void *ctx, system_event_t *event)
```

```
{
```

```
    return ESP_OK;
```

```
}
```

```
extern "C"
```

```
{
```

```
    void app_main(void);
```

```
}
```

```

*/

void highPriorityTask(void *myData)
{
    printf("High priority task started and now looping for 10 seconds.
Our priority is %d.\n", uxTaskPriorityGet(nullptr));

    TickType_t startTicks = xTaskGetTickCount();

    while (xTaskGetTickCount() - startTicks < (10 * 1000 /
portTICK_PERIOD_MS))
    {
        // Do nothing but loop
    }

    printf("High priority task ended\n");
    vTaskDelete(nullptr);
}

void hardLoopTask(void *myData)
{
    printf("Hard loop task started ...\n");
    while (1)
    {
        // do nothing but burn CPU
    }
}

void hardLoopTaskNoInterrupts(void *myData)
{
    printf("Hard loop task disabling interrupts started ...\n");

```

```

    taskDISABLE_INTERRUPTS();

    while (1)
    {
        // do nothing but burn CPU
    }
}

void myTask(void *myData)
{
    printf("# Running in myTask\n");
    printf("# Registering our new task with the task watchdog.\n");
    esp_task_wdt_add(nullptr);

    printf("# Looping 5 times with a delay of 1 second and not feeding
the watchdog.\n");
    for (int i = 0; i < 5; i++)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        printf("Tick\n");
    }

    printf("# Looping 5 times with a delay of 1 second and positively
feeding the watchdog.\n");
    esp_task_wdt_reset();
    for (int i = 0; i < 5; i++)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);

```

```

        printf("Tick\n");

        esp_task_wdt_reset();
    }

    printf("# Removing our watchdog registration so we can do something
expensive.\n");

    esp_task_wdt_delete(nullptr);

    printf("# Looping 5 times with a delay of 1 second and not feeding
the watchdog.\n");

    for (int i = 0; i < 5; i++)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);

        printf("Tick\n");
    }

    printf("# Re-registering our task with the task watchdog.\n");

    esp_task_wdt_add(nullptr);

    printf("# Looping 5 times with a delay of 1 second and not feeding
the watchdog.\n");

    for (int i = 0; i < 5; i++)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);

        printf("Tick\n");
    }

    printf("# Our current task priority is %d.\n",
uxTaskPriorityGet(nullptr));

```

```

printf("# Spwaning a higher priority task\n");

xTaskCreate(highPriorityTask, // Task code
            "Priority task", // Name of task
            16 * 1024,       // Stack size
            nullptr,         // Task data
            5,               // Priority
            nullptr          // task handle
        );

printf("# Looping 5 times with a delay of 1 second and positively
feeding the watchdog.\n");

esp_task_wdt_reset();

for (int i = 0; i < 5; i++)
{
    vTaskDelay(1000 / portTICK_PERIOD_MS);
    printf("Tick\n");
    esp_task_wdt_reset();
}

printf("Spawning a hard-loop function!\n");

xTaskCreate(hardLoopTaskNoInterrupts, // Task code
            "Hard Loop", // Name of task
            16 * 1024,    // Stack size
            nullptr,      // Task data
            5,            // Priority
            nullptr       // task handle
        );

```

```
    printf("# Looping 5 times with a delay of 1 second and positively  
feeding the watchdog.\n");
```

```
    esp_task_wdt_reset();
```

```
    for (int i = 0; i < 5; i++)
```

```
    {
```

```
        vTaskDelay(1000 / portTICK_PERIOD_MS);
```

```
        printf("Tick\n");
```

```
        esp_task_wdt_reset();
```

```
    }
```

```
    printf("# Removing our watchdog registration before we end the  
task.\n");
```

```
    esp_task_wdt_delete(nullptr);
```

```
    printf("# Ending myTask\n");
```

```
    vTaskDelete(nullptr);
```

```
} // myTask
```

```
void setup()
```

```
{
```

```
    xTaskHandle handle;
```

```
    printf("App starting\n");
```

```
    printf("Initializing the task watchdog subsystem with an  
interval of 2 seconds.\n");
```

```
    esp_task_wdt_init(2, false);
```

```
    printf("Creatign a new task.\n");
```

```
    // Now let us create a new task.
```

```
    xTaskCreate(myTask,    // Task code  
               "My Task", // Name of task  
               16 * 1024, // Stack size  
               nullptr,   // Task data  
               0,          // Priority  
               &handle     // task handle  
               );
```

```
    //printf("App Ended!\n");
```

```
}
```

```
void loop()
```

```
{
```

```
    // put your main code here, to run repeatedly:
```

```
}
```


Ejemplo 16: WAKE UP by Touch

```
/*
```

```
Deep Sleep with Touch Wake Up
```

```
=====
```

```
This code displays how to use deep sleep with  
a touch as a wake up source and how to store data in  
RTC memory to use it over reboots
```

```
This code is under Public Domain License.
```

```
Author:
```

```
Pranav Cherukupalli <cherukupallip@gmail.com>
```

```
*/
```

```
#define Threshold 50 // Por debajo de este  
umbral, se activa la interrupción TOUCH, Greater the value, more the  
sensitivity
```

```
int contador = 0;
```

```
RTC_DATA_ATTR int bootCount = 0, cc = 0; // En al memoria del RTC  
(8 Kb de SRAM en RTC), no se pierde la información durante deepsleep, si con  
reset
```

```
touch_pad_t touchPin;
```

```

void print_wakeup_reason(){                                     // Method to print the
reason by which ESP32 has been awoken from sleep

    esp_sleep_wakeup_cause_t wakeup_reason;

    wakeup_reason = esp_sleep_get_wakeup_cause();

// D

    switch(wakeup_reason)
    {
        case
ESP_SLEEP_WAKEUP_EXT0      : Serial.println("\t\t\t Wakeup caused by
external signal using RTC_IO"); break;

        case
ESP_SLEEP_WAKEUP_EXT1      : Serial.println("\t\t\t Wakeup caused by
external signal using RTC_CNTL"); break;

        case
ESP_SLEEP_WAKEUP_TIMER     : Serial.println("\t\t\t Wakeup caused by
timer"); break;

        case
ESP_SLEEP_WAKEUP_TOUCHPAD  : Serial.println("\t\t\t Wakeup caused by
touchpad"); break;

        case
ESP_SLEEP_WAKEUP_ULP       : Serial.println("\t\t\t Wakeup caused by ULP
program"); break;

        case
ESP_SLEEP_WAKEUP_UNDEFINED : Serial.println("\t\t\t Wakeup caused is
undefined"); break;

        default :
Serial.println("\t\t\t Wakeup was not caused by deep sleep"); break;

    }

```

```
}
```

```
void print_wakeup_touchpad() { // Method to print the  
touchpad by which ESP32 has been awoken from sleep
```

```
touch_pad_t pin;
```

```
touchPin =  
esp_sleep_get_touchpad_wakeup_status();
```

```
switch(touchPin)
```

```
{
```

```
case 0 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 4"); break;
```

```
case 1 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 0"); break;
```

```
case 2 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 2"); break;
```

```
case 3 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 15 !!!!"); break;
```

```
case 4 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 13"); break;
```

```
case 5 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 12"); break;
```

```
case 6 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 14"); break;
```

```
case 7 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 27"); break;
```

```
case 8 :
```

```
Serial.println("\t\t\t Touch detected on GPIO 33"); break;
```

```

                                case 9 :
Serial.println("\t\t\t Touch detected on GPIO 32 !!!!"); break;

                                default :
Serial.println("\t\t\t Wakeup not by touchpad"); break;

                                }

        }

void callback0(){ // IRAM_ATTR !!!
    // placeholder callback function, usar volatile!!!!
}

void callback3(){ // IRAM_ATTR !!!
    // placeholder callback function, usar volatile!!!!
}

void callback8(){ // IRAM_ATTR !!!
    //placeholder callback function
}

void setup(){
    Serial.begin(115200);
    delay(1000);
    ++bootCount;
    // Increment boot number and print it every reboot
    contador++;
    Serial.println("\t\t\t Se ha dormido un número de veces
igual a: " + String(bootCount));

```

```
        Serial.println("\t\t\t La variable CONTADOR se pierde al  
WAKEUP: " + String(contador));
```

```
        //Serial.println("\t\t\t Esta variable se pierde con cada  
"levantada : " + String(contador));
```

```
        print_wakeup_reason();  
// Print the wakeup reason for ESP32 and touchpad too
```

```
        print_wakeup_touchpad();
```

```
        touchAttachInterrupt(T0, callback0, Threshold);  
// Setup interrupt on Touch Pad T0 (GPIO4)  
A
```

```
        touchAttachInterrupt(T3, callback3, Threshold);  
// Setup interrupt on Touch Pad T3 (GPIO15)
```

```
        touchAttachInterrupt(T8, callback8, Threshold);  
// Setup interrupt on Touch Pad T8 (GPIO33).....está  
invertido con el 32...
```

```
        esp_sleep_enable_touchpad_wakeup();  
// Configure Touchpad as wakeup source  
B
```

```
        Serial.println("\t\t\t Going to sleep now");  
        Serial.flush();  
        esp_deep_sleep_start();  
//  
C
```

```
        Serial.println("This will never be printed");
```

```

    }

void loop(){
    // This will never be reached
}

```

Ejemplo 17: WAKE UP by Timer

```

/*
Simple Deep Sleep with Timer Wake Up
=====

ESP32 offers a deep sleep mode for effective power saving as power is an
important factor for IoT

applications. In this mode CPUs, most of the RAM, and all the digital
peripherals which are clocked

from APB_CLK are powered off. The only parts of the chip which can still be
powered on are:

RTC controller, RTC peripherals, and RTC memories This code displays the
most basic deep sleep with

a timer to wake it up and how to store data in RTC memory to use it over
reboots

This code is under Public Domain License.

Author:

Pranav Cherukupalli <cherukupallip@gmail.com>

*/

```

```

#define uS_TO_S_FACTOR 1000000
/* Conversion factor for micro seconds to seconds */

#define TIME_TO_SLEEP 5
/* Time ESP32 will go to sleep (in seconds) */

#define LED 2

RTC_DATA_ATTR int bootCount = 0;
// En al memoria del RTC (8 Kb de SRAM en RTC), no se pierde la información
durante deepsleep si con reset

void print_wakeup_reason(){
// Method to print the reason by which ESP32 has been awoken from sleep

    esp_sleep_wakeup_cause_t wakeup_reason;

    wakeup_reason = esp_sleep_get_wakeup_cause();

// D

    switch(wakeup_reason)
    {
        case 1 :
Serial.println("\t\t\t Wakeup caused by external signal using RTC_IO");
break;

        case 2 :
Serial.println("\t\t\t Wakeup caused by external signal using RTC_CNTL");
break;

        case 3 :
Serial.println("\t\t\t Wakeup caused by touchpad"); break;

        case 4 :
Serial.println("\t\t\t Wakeup caused by timer");
break;////////// ERA TOUCH

        case 5 :
Serial.println("\t\t\t Wakeup caused by ULP program"); break;

```

```

                                default :
Serial.println("\t\t\t Wakeup was not caused by deep sleep");

                                break;

                                }

                                }

void setup(){

    Serial.begin(115200);

    delay(100);
// Take some time to open up the Serial Monitor

    pinMode (LED, OUTPUT);

    digitalWrite (LED, HIGH);

    delay (200);

    digitalWrite (LED, LOW);

    ++bootCount;
// Increment boot number and print it every reboot

    Serial.println("\t\t\t Boot number: " + String(bootCount));

    print_wakeup_reason();
// Print the wakeup reason for ESP32

    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
// First we configure the wake up source. We set our ESP32 to wake up every
5 seconds          // B

    Serial.println("\t\t\t Setup ESP32 to sleep for every " +
String(TIME_TO_SLEEP) + " Seconds");

    /*

    Next we decide what all peripherals to shut down/keep on

```


By default, ESP32 will automatically power down the peripherals

not needed by the wakeup source, but if you want to be a poweruser

this is for you. Read in detail at the API docs

http://esp-idf.readthedocs.io/en/latest/api-reference/system/deep_sleep.html

Left the line commented as an example of how to configure peripherals.

The line below turns off all RTC peripherals in deep sleep.

```
*/  
  
//esp_deep_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH,  
ESP_PD_OPTION_OFF);  
  
//Serial.println("Configured all RTC Peripherals to be powered  
down in sleep");
```

```
/*
```

Now that we have setup a wake cause and if needed setup the peripherals state in deep sleep, we can now start going to deep sleep.

In the case that no wake up sources were provided but deep sleep was started, it will sleep forever unless hardware reset occurs.

```
*/
```

```
Serial.println("\t\t\t Going to sleep now");  
Serial.flush();  
esp_deep_sleep_start();
```

```
// C
```

```
Serial.println("\t\t\t This will never be printed");
```

```
}
```

```
void loop(){
```

```
    //This is not going to be called
```

```
}
```

Ejemplo 18: WAKE UP EXT0, un GPIO

/ *

Deep Sleep with External Wake Up

=====

This code displays how to use deep sleep with an external trigger as a wake up source and how to store data in RTC memory to use it over reboots

This code is under Public Domain License.

Hardware Connections

=====

Push Button to GPIO 33 pulled down with a 10K Ohm resistor

NOTE :

=====

Only RTC IO can be used as a source for external wake source. They are pins: 0,2,4,12-15,25-27,32-39.

Author:

Pranav Cherukupalli <cherukupallip@gmail.com>

*/

```
// Necesita una resistencia de pullUp o pullDown (GPIO_x)
```

```

#include <driver/rtc_io.h>

RTC_DATA_ATTR int bootCount = 0;
// En al memoria del RTC (8 Kb de SRAM en RTC), no se pierde la información
durante deepsleep si con reset

void print_wakeup_reason() {
// Method to print the reason by which ESP32 has been awoken from sleep

    esp_sleep_wakeup_cause_t wakeup_reason;

    wakeup_reason = esp_sleep_get_wakeup_cause();

// D

    Serial.println("");
    Serial.println("");
    Serial.println("");
    Serial.println("EXT0 Test");

    switch (wakeup_reason)
    {
        case 1 : Serial.println("Wakeup caused by
external signal using RTC_IO"); break;

        case 2 : Serial.println("Wakeup caused by
external signal using RTC_CNTL"); break;

        case 3 : Serial.println("Wakeup caused by
touchpad"); break;

```

```

                                case 4 : Serial.println("Wakeup caused by
timer"); break;
                                // Era touch

                                case 5 : Serial.println("Wakeup caused by ULP
program"); break;

                                default : Serial.println("Wakeup was not
caused by deep sleep"); break;
                                }
                                }

```

```

void setup() {

    Serial.begin(115200);

    delay(1000);
    // Take some time to open up the Serial Monitor

    ++bootCount;
    // Increment boot number and print it every reboot

    Serial.println("\t\t\t Boot number: " + String(bootCount));

    print_wakeup_reason();
    // Print the wakeup reason for ESP32

```

```

/*

    First we configure the wake up source

    We set our ESP32 to wake up for an external trigger.

    There are two types for ESP32, ext0 and ext1 .

    ext0 uses RTC_IO to wake up thus requires RTC peripherals
    to be on while ext1 uses RTC Controller so doesn't need
    peripherals to be powered on.

```

```

        Note that using internal pullups/pulldowns also requires
        RTC peripherals to be turned on.

    */

    //rtc_gpio_pullup_en (GPIO_NUM_4);
    // En vez de las dos siguientes

    //pinMode (4, INPUT_PULLDOWN);

    //
    // A

    pinMode (0, INPUT_PULLUP);

    //rtc_gpio_hold_en (GPIO_NUM_4);
    // Para mantener el estado durante el deep sleep...sobra

    //esp_sleep_enable_ext0_wakeup(GPIO_NUM_4, 1);
    // Con H se "despierta". Puede necesitar resistencia de pull Up o pull Down.
    1 = High, 0 = Low                                     // B

    //esp_sleep_enable_ext0_wakeup(GPIO_NUM_4, 0);

    esp_sleep_enable_ext0_wakeup(GPIO_NUM_0, 0);

    Serial.println("Going to sleep now");

    Serial.flush();

    esp_deep_sleep_start();

    // C

    Serial.println("This will never be printed");

}

void loop() {

    //This is not going to be called

}

```

Ejemplo 19: WAKE UP EXT1, múltiples GPIO

```
/*  
  
    Deep Sleep with External Wake Up  
  
    =====  
  
    This code displays how to use deep sleep with  
    an external trigger as a wake up source and how  
    to store data in RTC memory to use it over reboots  
  
    This code is under Public Domain License.  
  
    Hardware Connections  
  
    =====  
  
    Push Button to GPIO 33 pulled down with a 10K Ohm  
    resistor  
  
    NOTE:  
  
    =====  
  
    Only RTC IO can be used as a source for external wake  
    source. They are pins: 0,2,4,12-15,25-27,32-39.  
  
    Author:  
  
    Pranav Cherukupalli <cherukupallip@gmail.com>  
  
*/
```

```
#include <driver/rtc_io.h>
```

```
//#define BUTTON_PIN_BITMASK 0x0100000000  
// GPIO_32 (40 bits)
```

```

// #define BUTTON_PIN_BITMASK 0x0200000000
// GPIO_33 (40 bits)

#define BUTTON_PIN_BITMASK 0x0300000000
// GPIO_32, GPIO_33 (40 bits)

// #define BUTTON_PIN_BITMASK 0xFF00000000
// all pins

RTC_DATA_ATTR int bootCount = 0;
// En la memoria del RTC (8 Kb de SRAM en RTC), no se pierde la información
durante deepsleep si con reset

uint64_t a;

void print_wakeup_reason() {
// Method to print the reason by which ESP32 has been awoken from sleep

    esp_sleep_wakeup_cause_t wakeup_reason;

    wakeup_reason = esp_sleep_get_wakeup_cause();

    Serial.println("");
    Serial.println("");
    Serial.println("\t\t\t EXT1 Test");

    switch (wakeup_reason)
    {
        case 1 :
Serial.println("\t\t\t Wakeup caused by external signal using RTC_IO");
break;

        case 3 : {

// ???

```



```
Serial.println("\t\t\t Wakeup caused by external signal using RTC_CNTL_EXT1");
```

```
Serial.println((uint32_t)esp_sleep_get_ext1_wakeup_status(),HEX);
```

```
Serial.println();
```

```
Serial.println();
```

```
                                a =  
esp_sleep_get_ext1_wakeup_status(), HEX;  
  
                                print64(a);
```

```
Serial.println();
```

```
                                break;  
                                }
```

```
                                case 2 :  
Serial.println("\t\t\t Wakeup caused by touchpad"); break;
```

```
                                case 4 :  
Serial.println("\t\t\t Wakeup caused by timer"); break;
```

```
                                case 5 :  
Serial.println("\t\t\t Wakeup caused by ULP program"); break;
```

```
                                default :  
Serial.println("\t\t\t Wakeup was not caused by deep sleep"); break;  
                                }
```

```
}
```

```
void print64(uint64_t number) {
```

```
    for (int i = 0; i < 40; i++) {
```

```

                                                                    bool bitt =
number & 0xFF0000000;

Serial.println(bitt);

                                                                    number =
number << 1;

                                                                    }

                                                                    }

```

```

void setup() {
    Serial.begin(115200);
    delay(1000);
    // Take some time to open up the Serial Monitor

    ++bootCount;
    // Increment boot number and print it every reboot

    Serial.println("\t\t\t Boot number: " + String(bootCount));

    print_wakeup_reason();
    // Print the wakeup reason for ESP32

```

```

/*
    First we configure the wake up source
    We set our ESP32 to wake up for an external trigger.
    There are two types for ESP32, ext0 and ext1 .
    ext0 uses RTC_IO to wake up thus requires RTC peripherals
    to be on while ext1 uses RTC Controller so doesn't need

```

peripherals to be powered on.

Note that using internal pullups/pulldowns also requires

RTC peripherals to be turned on.

```
*/
```

```
//If you were to use ext1, you would use it like
```

```
pinMode(GPIO_NUM_32, INPUT_PULLDOWN);
```

```
rtc_gpio_hold_en(GPIO_NUM_32);
```

```
pinMode(GPIO_NUM_33, INPUT_PULLDOWN);
```

```
rtc_gpio_hold_en(GPIO_NUM_33);
```

```
    esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,  
ESP_EXT1_WAKEUP_ANY_HIGH);           // Se activa el GPIO 32  
en H
```

```
//Serial.println(esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,  
ESP_EXT1_WAKEUP_ANY_HIGH));
```

```
    Serial.println("Going to sleep now");
```

```
// Go to sleep now
```

```
    Serial.flush();
```

```
    esp_deep_sleep_start();
```

```
    Serial.println("This will never be printed");
```

```
}
```

```
void loop() {
```

```
    //This is not going to be called
```


Ejemplo 20: WAKE UP by ULP

```
// Ejemplo de arduino
// esp 32 wakeup atomic14
// https://www.youtube.com/watch?v=KQS\_xDDWfLw&t=1s
//
/*
    This example smothly blinks GPIO_2 using different frequencies changed
    after Deep Sleep Time

    The PWM and control of blink frequency is done by ULP exclusively

    This is an example about how to program the ULP using Arduino

    It also demonstrates use of RTM MEMORY to persist data and states
*/

#include <Arduino.h>
#include "esp32/ulp.h"
#include "driver/rtc_io.h"

// RTC Memory used for ULP internal variable and Sketch interfacing
#define RTC_dutyMeter 0
#define RTC_dir      4
#define RTC_fadeDelay 12

// *fadeCycleDelay is used to pass values to ULP and change its behaviour
uint32_t *fadeCycleDelay = &RTC_SLOW_MEM[RTC_fadeDelay];
```

```

#define ULP_START_OFFSET 32

// For ESP32 Arduino, it is usually at offset 512, defined in sdkconfig
RTC_DATA_ATTR uint32_t ULP_Started = 0; // 0 or 1

//Time-to-Sleep

//#define uS_TO_S_FACTOR 1000000ULL /* Conversion factor for micro seconds
to seconds */

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to
seconds */

#define TIME_TO_SLEEP 5 /* Time ESP32 will go to sleep (in
microseconds); multiplied by above conversion to achieve seconds*/

void ulp_setup()
{
    if (ULP_Started)
    {
        return;
    }

    *fadeCycleDelay = 5;
    // 5..200 works fine for a full Fade In + Out cycle

    ULP_Started = 1;

    // GPIO2 initialization (set to output and initial value is 0)

    const gpio_num_t MeterPWMPin = GPIO_NUM_2;

    rtc_gpio_init(MeterPWMPin);

```

```

        rtc_gpio_set_direction(MeterPWMPin,
RTC_GPIO_MODE_OUTPUT_ONLY);

        rtc_gpio_set_level(MeterPWMPin, 0);

        // if LED is connected to GPIO2 (specify by
+RTC_GPIO_OUT_DATA_S : ESP32 is 14, S2/S3 is 10)

        const uint32_t MeterPWMBit =
rtc_io_number_get(MeterPWMPin) + RTC_GPIO_OUT_DATA_S;

enum labels
{
    INIFINITE_LOOP,
    RUN_PWM,
    NEXT_PWM_CYCLE,
    PWM_ON,
    PWM_OFF,
    END_PWM_CYCLE,
    POSITIVE_DIR,
    DEC_DUTY,
    INC_DUTY,
};

// Define ULP program
const ulp_insn_t ulp_prog[] =
{
    // Initial Value setup
    I_MOVI(R0, 0),           // R0 = 0

```

```

                                I_ST(R0, R0, RTC_dutyMeter), //
RTC_SLOW_MEM[RTC_dutyMeter] = 0

                                I_MOVI(R1, 1),           // R1 = 1
                                I_ST(R1, R0, RTC_dir),     //
RTC_SLOW_MEM[RTC_dir] = 1

                                M_LABEL(INIFINITE_LOOP),   // while(1) {

                                // run certain PWM Duty for about
                                (RTC_fadeDelay x 100) microseconds

                                I_MOVI(R3, 0),             // R3 = 0
                                I_LD(R3, R3, RTC_fadeDelay), // R3 =
RTC_SLOW_MEM[RTC_fadeDelay]

                                M_LABEL(RUN_PWM),          // do { //
repeat RTC_fadeDelay times:

                                // execute about 10KHz PWM on GPIO2 using as
                                duty cycle = RTC_SLOW_MEM[RTC_dutyMeter]

                                I_MOVI(R0, 0),             // R0 = 0
                                I_LD(R0, R0, RTC_dutyMeter), // R0 =
RTC_SLOW_MEM[RTC_dutyMeter]

                                M_BL(NEXT_PWM_CYCLE, 1),   // if (R0
> 0) turn on LED

                                I_WR_REG(RTC_GPIO_OUT_W1TS_REG, MeterPWMBit,
MeterPWMBit, 1), // W1TS set bit to clear GPIO - GPIO2 on

                                M_LABEL(PWM_ON),           // while
(R0 > 0) // repeat RTC_dutyMeter times:

                                M_BL(NEXT_PWM_CYCLE, 1),   // {
                                //I_DELAY(8),              // // 8
is about 1 microsecond based on 8MHz

```



```

R0 - 1                                I_SUBI(R0, R0, 1),           //      R0 =

M_BX(PWM_ON),                        //      }

M_LABEL(NEXT_PWM_CYCLE),             //      //

toggle GPIO_2

I_MOVI(R0, 0),                       //      R0 = 0

I_LD(R0, R0, RTC_dutyMeter),         //      R0 =

RTC_SLOW_MEM[RTC_dutyMeter]

I_MOVI(R1, 100),                     //      R1 =

100

I_SUBR(R0, R1, R0),                  //      R0 =

100 - dutyMeter

M_BL(END_PWM_CYCLE, 1),              //      if (R0

> 0) turn off LED

I_WR_REG(RTC_GPIO_OUT_W1TC_REG, MeterPWMBit,
MeterPWMBit, 1), // W1TC set bit to clear GPIO - GPIO2 off

M_LABEL(PWM_OFF),                    //      while

(R0 > 0) // repeat (100 - RTC_dutyMeter) times:

M_BL(END_PWM_CYCLE, 1),              //      {

//I_DELAY(8),                        //      // 8

is about 1us: ULP fetch+execution time

I_SUBI(R0, R0, 1),                   //      R0 =

R0 - 1

M_BX(PWM_OFF),                       //      }

M_LABEL(END_PWM_CYCLE),              //

I_SUBI(R3, R3, 1),                   //      R3 = R3

- 1 // RTC_fadeDelay

I_MOVR(R0, R3),                      //      R0 = R3

// only R0 can be used to compare and branch

```

```

                                M_BGE(RUN_PWM, 1),           // } while
(R3 > 0) // ESP32 repeatinf RTC_fadeDelay times

                                // increase/decrease DutyMeter to apply Fade
In/Out loop

                                I_MOVI(R1, 0),               // R1 = 0
                                I_LD(R1, R1, RTC_dutyMeter), // R1 =
RTC_SLOW_MEM[RTC_dutyMeter]

                                I_MOVI(R0, 0),               // R0 = 0
                                I_LD(R0, R0, RTC_dir),        // R0 =
RTC_SLOW_MEM[RTC_dir]

                                M_BGE(POSITIVE_DIR, 1),      //
if(dir == 0) { // decrease duty by 2
                                // Dir is 0, means decrease Duty by 2
                                I_MOVR(R0, R1),              // R0 =
Duty

                                M_BGE(DEC_DUTY, 1),           // if
(duty == 0) { // change direction and increase duty
                                I_MOVI(R3, 0),                // R3 =
0
                                I_MOVI(R2, 1),                 // R2 =
1
                                I_ST(R2, R3, RTC_dir),         //
RTC_SLOW_MEM[RTC_dir] = 1 // increasing direction
                                M_BX(INC_DUTY),                //
goto "increase Duty"

                                M_LABEL(DEC_DUTY),             //
} "decrease Duty":

                                I_SUBI(R0, R0, 2),             // Duty -=
2

```

```

        I_MOVI(R2, 0),           //      R2 = 0

        I_ST(R0, R2, RTC_dutyMeter), //
RTC_SLOW_MEM[RTC_dutyMeter] += 2

        M_BX(INIFINITE_LOOP),    // }

// dir == 1 // increase duty by 2

        M_LABEL(POSITIVE_DIR),    //      else {
// Dir is 1, means increase Duty by 2

        I_MOVR(R0, R1),           //      R0 =
Duty

        M_BL(INC_DUTY, 100),      //      if
(duty == 100) { // change direction and decrease duty

        I_MOVI(R2, 0),           //      R2 =
0

        I_ST(R2, R2, RTC_dir),    //
RTC_SLOW_MEM[RTC_dir] = 0 // decreasing direction

        M_BX(DEC_DUTY),           //      goto
"decrease Duty"

        M_LABEL(INC_DUTY),        //      }
"increase Duty":

        I_ADDI(R0, R0, 2),        //      Duty +=
2

        I_MOVI(R2, 0),           //      R2 = 0

        I_ST(R0, R2, RTC_dutyMeter), //
RTC_SLOW_MEM[RTC_dutyMeter] -= 2

//      } // if
(dir == 0)

        M_BX(INIFINITE_LOOP),    //      } //
while(1)

};

```

```

        // Run ULP program

        size_t size = sizeof(ulp_prog) / sizeof(ulp_insn_t);

        ulp_process_macros_and_load(ULP_START_OFFSET, ulp_prog,
&size);

        esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH,
ESP_PD_OPTION_ON);

        ulp_run(ULP_START_OFFSET);
    }

void setup()
{
    Serial.begin(115200);

    while (!Serial) {}
// wait for Serial to start

    ulp_setup();
// it really only runs on the first ESP32 boot

    Serial.printf("\nStarted smooth blink with delay %d\n",
*fadeCycleDelay);

    // *fadeCycleDelay resides in RTC_SLOW_MEM and persists along
deep sleep waking up

    // it is used as a delay time parameter for smooth blinking, in
the ULP processing code

    if (*fadeCycleDelay < 195)
    {
        *fadeCycleDelay += 10;
    }

```

```

        else
        {
            *fadeCycleDelay = 5;
            // 5..200 works fine for a full Fade In + Out cycle

        }

        Serial.println("Entering in Deep Sleep");

        esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR /*/
4*/);           // time set with variable above

        esp_deep_sleep_start();

        // From this point on, no code is executed in DEEP SLEEP mode
    }

void loop()
{

    // It never reaches this code because it enters in Deep Sleep mode at the
    end of setup()

}

```

Ejemplo 21a: ADC

```
// https://www.youtube.com/watch?v=mzlq65fr3uI
```

```
// G6EJD
```

```
// VIN === 3.00 V
```

```
#include <Arduino.h>
```

```
#include <WiFi.h>
```

```
#include "esp_adc_cal.h"
```

```
#define ADC_GPIO_PIN 36
```

```
#define DEFAULT_VREF 1100
```

```
// calibration values for the adc
```

```
esp_adc_cal_characteristics_t *adc_chars;
```

```
float VERFERENCE_CAL;
```

```
float read_Voltage (byte adcpin)
```

```
{
```

```
    float cal = 1.0;
```

```
    float vref = 1100;
```

```
// 1100 mV
```

```
    esp_adc_cal_characteristics_t adc_chars;
```

```

        esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_12, DEFAULT_VREF, &adc_chars);          // Obtiene las
caracteristicas del ADC

        vref = adc_chars.vref;
// Obtiene el VREF

        VERFERENCE_CAL = vref;

        Serial.println();

        Serial.println(" VREF = " + String(VERFERENCE_CAL,3));

        Serial.println();

        return(analogRead(adcpin)/4095.0)*3.3*(1100/vref)*cal;

    }

```

```

void setup()
{
    Serial.begin(115200);

    Serial.println("Started up");

}

```

```

void loop()
{

    //int sample = adc1_get_raw(ADC1_CHANNEL_7);
// read a sample from the adc using GPIO35

    float volt_sin_aju = (analogRead(ADC_GPIO_PIN)/4095.0*3.3);

    float volt_con_aju = read_Voltage (ADC_GPIO_PIN);

```

```
Serial.println(" VIN = 3.00 V ");

Serial.println(" Voltaje sin corregir = " + String
(volt_sin_aju,3) + " V " + String (volt_sin_aju/3.0*100-100) + " % error");

Serial.println(" Voltaje con correcci = " + String
(volt_con_aju,3) + " V " + String (volt_con_aju/3.0*100-100) + " % error");

Serial.println();

delay(5000);

}
```


Ejemplo 21b: ADC

```
// ESP32 Audio Input Using I2S and Internal ADC  
// https://www.youtube.com/watch?v=pPh3\_ciEmzs&t=203s
```

```
#include <Arduino.h>  
#include <WiFi.h>  
#include "esp_adc_cal.h"
```

```
#define DEFAULT_VREF 1100  
// calibration values for the adc
```

```
esp_adc_cal_characteristics_t adc_chars;  
//esp_adc_cal_characteristics_t *adc_chars === error
```

```
void setup()  
{  
    Serial.begin(115200);  
    Serial.println("Started up");  
}
```

```

        adc1_config_width(ADC_WIDTH_BIT_12);
// Range 0-4096

        adc1_config_channel_atten(ADC1_CHANNEL_7, ADC_ATTEN_DB_11);
// full voltage range GPIO 35


        //adc2_config_width(ADC_WIDTH_BIT_12);
// Range 0-4096

        //adc2_config_channel_atten(ADC2_CHANNEL_0, ADC_ATTEN_DB_11);
// full voltage range GPIO 4


/*

        if (esp_adc_cal_check_efuse(ESP_ADC_CAL_VAL_EFUSE_VREF) ==
ESP_OK)
            // check to see what calibration is available

{

Serial.println("Using voltage ref stored in eFuse");

}

        if (esp_adc_cal_check_efuse(ESP_ADC_CAL_VAL_EFUSE_TP) ==
ESP_OK)

{

Serial.println("Using two point values from eFuse");

}

        if (esp_adc_cal_check_efuse(ESP_ADC_CAL_VAL_DEFAULT_VREF) ==
ESP_OK)

```

```

{

Serial.println("Using default VREF");

}

        adc_chars = (esp_adc_cal_characteristics_t *)calloc(1,
sizeof(esp_adc_cal_characteristics_t));           // Characterize ADC

*/

        esp_adc_cal_characteristics_t adc_chars;

// new

        esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_12, DEFAULT_VREF, &adc_chars);           // Obtiene las
características del ADC sin & original

        //esp_adc_cal_characterize(ADC_UNIT_2, ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_12, DEFAULT_VREF, &adc_chars);           // Obtiene las
características del ADC

    }

void loop()

{

    int sample = adc1_get_raw(ADC1_CHANNEL_7);
// read a sample from the adc using GPIO35

    //int sample1 = analogRead(35);

```

```
        //int sample2 = adc2_get_raw(ADC2_CHANNEL_0);  
// GPIO 4 ADC2  
  
        //int sample3 = analogRead(4);  
  
        int milliVolts = esp_adc_cal_raw_to_voltage(sample,  
&adc_chars); // get the  
calibrated value, sin & error  
  
        Serial.printf("Sample=%d, mV=%d\n", sample, milliVolts);  
  
        delay(500);  
    }
```

Ejemplo 22a: DAC

```
// Unir GPIO 4 con GPIO 25/26

// 33 mS == 30 Hz y 360 muestras CON DAC Y ADC

// 66 mS == 15 Hz y 720 muestras con DAC y ADC

// 13 mS === 77 Hz 720 muestras; 18 uS por muestra (55,55 KHz). Solo DAC


#include <soc/sens_reg.h>

#include "soc/rtc.h"
// A

#include "esp_adc_cal.h"

#include "driver/adc.h"


#define DAC_CHANNEL_1 25
// DAC1 = 25; DAC2 = 26

#define DAC_CHANNEL_2 26

#define SAMPLES 360.0


uint32_t a, b, i, ls = 5000, li = 0;

float theta, dtheta = 2*PI/SAMPLES, vector[720], vector1[720];


void setup() {

    Serial.begin(115200);

    //analogSetCycles(8);
    // Default is 8 and seems to do well. Range is 1 - 255

    //void analogSetClockDiv(uint8_t clockDiv);
```

```

        //analogSetClockDiv(18);
// Default is 1. Range is 1 - 255. No afecta

        for (i=0 ; i<SAMPLES ; i++)
// SAMPLES  ángulos.

        {

            vector[i] = 128 + 100*sin(dtheta*i)+ 50*sin(dtheta*3*i)
+ 10*sin(dtheta*5*i);                                // 77 HZ    (720
muestras)

            //vector[i] = 128 + 38*sin (dtheta*i) + 38*sin
(dtheta*i*3) + 38*sin (dtheta*i*5) + 30*sin (dtheta*i*20);        // 150
HZ    (360 muestras)

        }

        pinMode (2, OUTPUT);

    }

void loop() {

        //digitalWrite(2, HIGH);

        for (i=0 ; i<SAMPLES ; i++)

// SAMPLES  ángulos.

        {

            digitalWrite (DAC_CHANNEL_1,
// 13 mS === 77

            vector [i]);

            Hz 720 muestras; 18 uS por muestra (55,55 KHz)

            digitalWrite (DAC_CHANNEL_2,
// DAC 8 bits,

            i/3.0);

            valor máximo === 255

            //digitalWrite(2, HIGH);

            a = analogRead(4);

            // 4 === ADC2_CH0; 25 === ADC2_CH8

```

```

//digitalWrite(2, LOW);

vector1 [i] = a;

Serial.println(String(ls)+String(",")+String(li)+String(",")+String(a));

//delayMicroseconds(10);

}

//digitalWrite(2, LOW);

/*
while (1)
{
    for (i=0 ; i<SAMPLES ; i++)
        {

Serial.println(String(ls)+String(",")+String(li)+String(",")+String(vector1
[i]));

delayMicroseconds(1000);

}

} */
}

```

Ejemplo 22b: ADC y DAC

// <https://www.youtube.com/watch?v=2YIwTfJtCoM&t=851s>

// Rubén Hidalgo Carrillo

// Unir GPIO25 con VP (GPIO36) y aplicar $128 \Rightarrow 3,3/2 = 4096/2 = 2160$

// ADC1_CHANNEL_0 === GPIO 36

```
#include <driver/adc.h>
```

```
#define DAC_CHANNEL_1 25
```

```
// DAC1 = 25; DAC2 = 26
```

```
#define DAC_CHANNEL_2 26
```

```
#define max 2250
```

```
// 2150 +/- 100
```

```
#define min 2050
```

```
int lectura;
```

```
int lecturamax;
```

```
int lecturamin;
```

```
int vueltas = 0;
```

```
void setup()
```

```
{
```



```

        Serial.begin(111500);

        dacWrite (DAC_CHANNEL_1, 128);
// 13 mS === 77 Hz 720 muestras; 18 uS por muestra (55,55 KHz)

        dacWrite (DAC_CHANNEL_2, 80);
// DAC 8 bits, valor máximo === 255

        lecturamin = adc1_get_raw(ADC1_CHANNEL_0);

        lecturamax = lecturamin;
    }

void loop()
{
    lectura = adc1_get_raw(ADC1_CHANNEL_0);

    if (lectura < lecturamin) lecturamin = lectura;
    if (lectura > lecturamax) lecturamax = lectura;

    char buff[1024];

    sprintf (buff, "max:%d min:%d adc:%d lmin:%d lmax:%d", max,
min, lectura, lecturamin, lecturamax);

    Serial.println(buff);

    delay(10);

    if (++vueltas > 1000)
    {
        vueltas = 0;

        lecturamin = lectura;

        lecturamax = lectura;
    }
}

```

Ejemplo 22c: DAC y ADC con histograma

// <https://www.youtube.com/watch?v=2YIwTfJtCoM&t=851s>

// Rubén Hidalgo Carrillo

// Unir GPIO25 con VP (GPIO36) y aplicar 128 == $3,3/2 = 4096/2 =$

```
#include <driver/adc.h>
```

```
#define DAC_CHANNEL_1 25
```

```
// DAC1 = 25; DAC2 = 26
```

```
#define DAC_CHANNEL_2 26
```

```
#define directo 0
```

```
#define multisampling 1
```

```
#define escalon 2
```

```
#define grafica 0
```

```
#define histograma 1
```

```
#define FILTRO 500
```

```
int j = 0;
```

```

//int modo = directo;

//int modo = multisampling;

int modo = escalon;


int visualizar = grafica;

//int visualizar = histograma;


int lectura;

int rawlectura;

double filtro;

int vueltas = 0;

int veces[4096];

int milisant = 0;


void setup()
{
    Serial.begin(115200);

    pinMode(ADC1_CHANNEL_0, INPUT_PULLUP);

    for (int i=0; i<4096; i++) veces[i]= 110;

    dacWrite (DAC_CHANNEL_1, 128);
    // 13 mS === 77 Hz 720 muestras; 18 uS por muestra (55,55 KHz)

    dacWrite (DAC_CHANNEL_2, 80);
    // DAC 8 bits, valor máximo === 255

}


void loop()
{

```

```

if (modo == directo)
{
    lectura = adc1_get_raw(ADC1_CHANNEL_0);
    //lectura = analogRead(4);
    rawlectura = lectura;
    //filtro = lectura;
}

else if (modo == multisampling)
{
    int suma = 0;
    for (int i=0; i<64; i++) suma +=
adc1_get_raw(ADC1_CHANNEL_0);
    //for (int i=0; i<64; i++) suma += analogRead(4);
    lectura = suma/64;
    rawlectura = adc1_get_raw(ADC1_CHANNEL_0);
}

else if (modo == escalon)
{
    lectura = adc1_get_raw(ADC1_CHANNEL_0);
    //lectura = analogRead(4);
    rawlectura = lectura;
    filtro += ((double)lectura - filtro)/ (double)FILTRO;
    lectura = (int) filtro;
}

//veces[lectura] = veces[lectura] + 1;
veces[lectura]++;
//veces[1600] = veces[1600] + 1;

```

```

if (visualizar == histograma)
{
    if (millis() > (milisant + 1000))
    {
        for (int i=(2160-25);i<(2160+25);i++)
        {
            Serial.println(veces[i]);
        }
        for (int i=0; i<4096; i++) veces[i]=0;
        vueltas = 0;
        milisant = millis();
    }
}
else if (visualizar == grafica)
{
    if (millis() > milisant + 100)
    {
        Serial.print("raw:");
        Serial.print(rawlectura);
        Serial.print(" filter:");
        Serial.println(filtro);
        milisant = millis();
    }
}
}

```

WI-FI

Ejemplo 23a: WI-FI SCAN MEJORADO

Number of networks found: 23

Network name: DANIELA_6G

Signal strength: -29

MAC address: 2C:96:82:81:BA:E8

Encryption type: WPA2_PSK

Network name: DIRECT-F5-HP DeskJet 5820 series

Signal strength: -62

MAC address: 40:B0:34:66:B9:F6

Encryption type: WPA2_PSK

Network name: MOVISTAR_8BB9

Signal strength: -63

MAC address: 74:93:DA:A0:8B:BB

Encryption type: WPA2_PSK

Network name: FAMILIA_MORENO

Signal strength: -75

MAC address: 46:48:B9:12:FE:D4

Encryption type: WPA2_PSK

Network name: Ana

Signal strength: -77

MAC address: F4:69:42:A0:3E:8F

Encryption type: WPA2_PSK

// <https://www.youtube.com/watch?v=0AlATlN95Y0>

/*

Rui Santos

Complete project details at

<https://RandomNerdTutorials.com/solved-reconnect-esp32-to-wifi/>

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files.

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

*/

// Funciono, 12/06/2024

#include <WiFi.h>

#include <clave.h>

//const char* ssid = "xxxx";

//const char* password = "xxxx";

unsigned long interval = 30000;

unsigned long previousMillis = 0;

String translateEncryptionType(wifi_auth_mode_t encryptionType)

{

switch (encryptionType)

{

case (WIFI_AUTH_OPEN):

```

        return "Open";
    case (WIFI_AUTH_WEP):
        return "WEP";
    case (WIFI_AUTH_WPA_PSK):
        return "WPA_PSK";
    case (WIFI_AUTH_WPA2_PSK):
        return "WPA2_PSK";
    case (WIFI_AUTH_WPA_WPA2_PSK):
        return "WPA_WPA2_PSK";
    case (WIFI_AUTH_WPA2_ENTERPRISE):
        return "WPA2_ENTERPRISE";
    }
}

```

```

void scanNetworks() {
    int numberOfNetworks = WiFi.scanNetworks();

    Serial.print("Number of networks found: ");
    Serial.println(numberOfNetworks);

    for (int i = 0; i < numberOfNetworks; i++)
    {
        Serial.print("Network name: ");
        Serial.println(WiFi.SSID(i));

        Serial.print("Signal strength: ");
    }
}

```



```

        Serial.println(WiFi.RSSI(i));

        Serial.print("MAC address: ");
        Serial.println(WiFi.BSSIDstr(i));

        Serial.print("Encryption type: ");
        String encryptionTypeDescription =
translateEncryptionType(WiFi.encryptionType(i));
        Serial.println(encryptionTypeDescription);
        Serial.println("-----");
    }
}

```

```

void connectToNetwork()
{
    unsigned long currentMillis = millis();
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(1000);
        Serial.println("Establishing connection to WiFi..");
    }

    Serial.println("Connected to network");
}

```

```

/*

```

```
        if ((WiFi.status() != WL_CONNECTED) && (currentMillis - previousMillis
>=interval)) {                                     // if WiFi is
down, try reconnecting every CHECK_WIFI_TIME seconds
```

```
Serial.print(millis());
```

```
Serial.println("Reconnecting to WiFi...");
```

```
WiFi.disconnect();
```

```
WiFi.reconnect();
```

```
previousMillis = currentMillis;
```

```
}
```

```
*/
```

```
}
```

```
void setup()
```

```
{
```

```
    Serial.begin(115200);
```

```
    scanNetworks();
```

```
    connectToNetwork();
```

```
    Serial.println(WiFi.macAddress());
```

```
    Serial.println(WiFi.localIP());
```

```
    WiFi.disconnect(true);
```

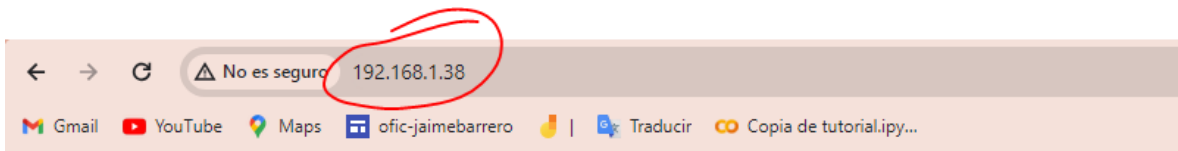
```
    Serial.println(WiFi.localIP());
```

```
}
```

```
void loop() {}
```

Ejemplo 23b: WI-FI SERVER KT

```
Try Connecting to  
DANIELA_6G  
.  
WiFi connected successfully  
Got IP: 192.168.1.38  
HTTP server started
```



My First Web Server with ESP32 - Station Mode 😊

```
// Funcionó, 12/06/2024
```

```
// Kelly Tatiana
```

```
#include <WiFi.h>
```

```
#include <clave.h>
```

```
#include <WebServer.h>
```

```
//const char* ssid = "xxxx";
```

```

//const char* password = "xxxx";

WebServer server(80); //
Object of WebServer(HTTP port, 80 is default)

void setup(){

    Serial.begin(115200);

    Serial.println("Try Connecting to ");

    Serial.println(ssid);

    WiFi.begin(ssid, password); //
Connect to your wi-fi modem

    while (WiFi.status() != WL_CONNECTED) //
Check wi-fi is connected to wi-fi network
    {
        delay(1000);

        Serial.print(".");

    }

    Serial.println("");

    Serial.println("WiFi connected successfully");

    Serial.print("Got IP: ");

    Serial.println(WiFi.localIP()); //
Show ESP32 IP on serial

    server.on("/", handle_root);

    server.begin();

    Serial.println("HTTP server started");

    delay(100);

```

```
}
```

```
void loop()
```

```
{
```

```
    server.handleClient();
```

```
}
```

```
// HTML & CSS contents which display on web server
```

```
String HTML = "<!DOCTYPE html>\
```

```
<html>\
```

```
<body>\
```

```
<h1>My First Web Server with ESP32 - Station Mode &#128522;</h1>\
```

```
</body>\
```

```
</html>";
```

```
void handle_root()
```

```
//
```

```
Handle root url (/)
```

```
{
```

```
    server.send(200, "text/html", HTML);
```

```
}
```

Ejemplo 23c: WI-FI SERVER JGB

```
Connecting to DANIELA_6G
..
WiFi connected.
IP address:
192.168.1.38
```



Click [here](#) to turn the LED on pin 2 on.

Click [here](#) to turn the LED on pin 2 off.

/*

WiFi Web Server LED Blink

A simple web server that lets you blink an LED via the web.

This sketch will print the IP address of your WiFi Shield (once connected) to the Serial monitor. From there, you can open that address in a web browser

to turn on and off the LED on pin 5.

If the IP address of your shield is yourAddress:

`http://yourAddress/H` turns the LED on

`http://yourAddress/L` turns it off

This example is written for a network using WPA encryption. For WEP or WPA, change the `Wifi.begin()` call accordingly.

Circuit:

- * WiFi shield attached

- * LED attached to pin 5

created for arduino 25 Nov 2012

by Tom Igoe

ported for sparkfun esp32

31.01.2017 by Jan Hendrik Berlín

`*/`

`// Funciono/Agosto/10/2020.....IP = 192.168.1.28`

`// 02/feb/2023`

`// Ver diagrama de flujo vínculo:`

```
# define LED_BUILTIN_1 2
```

```
// GPIO16,...
```

```
#include <WiFi.h>
```



```
#include "credencial.h"
```

```
WiFiServer server(80);
```

```
// A,
```

Puerto usado por navegadores

```
WEB
```

```
void setup()
```

```
{
```

```
    Serial.begin(115200);
```

```
    pinMode(LED_BUILTIN_1, OUTPUT);
```

```
// set the LED pin mode
```

```
    delay(10);
```

```
    Serial.println();
```

```
    Serial.println();
```

```
    Serial.print("Connecting to ");
```

```
    Serial.println(ssid);
```

```
    WiFi.begin(ssid, password);
```

```
// STATION MODE ¿Para cambiar seguridad?
```

```
    while (WiFi.status() != WL_CONNECTED) {
```

```
        delay(500);
```

```
        Serial.print(".");
```

```
    }
```

```
    Serial.println("");
```

```
    Serial.println("WiFi connected.");
```

```
    Serial.println("IP address: ");
```

```
    Serial.println(WiFi.localIP());
```

```

        server.begin();
// B,                                Inicializa el servidor
    }

int value = 0;

void loop(){
    WiFiClient client = server.available();
// C,                                Listen for incoming clients

    if (client) {
// if you get a client,
P0

        Serial.println("New Client.");
// print a message out the serial port

        String currentLine = "";
// make a String to hold incoming data from the client
P1

        while (client.connected()) {
// loop while the client's connected
P2

            if
            (client.available()) {                // if there's bytes to read
from the client,                                P3

char c = client.read();                        // read a byte, then

Serial.write(c);                                // print it out the serial
monitor

if (c == '\n') {                                // if the byte is a newline
character

```

```
// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

    // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

    // and a content-type so the client knows what's coming, then a blank line:

    client.println("HTTP/1.1 200 OK");

    client.println("Content-type:text/html");

    client.println();

    // the content of the HTTP response follows the header:

    client.print("Click <a href=\"/H\">here</a> to turn the LED on pin 2  
on.<br>");

    client.print("Click <a href=\"/L\">here</a> to turn the LED on pin 2  
off.<br>");

    //Lo que se ve ***Click here to turn the LED on pin 2 on.***

    //Lo que se ve ***Click here to turn the LED on pin 2 off.***
```

```
// The HTTP response ends with another blank line:

client.println();

// break out of the while loop:

break;

}

else { // if you got a newline, then clear
currentLine:

currentLine = "";

}

}

else if (c != '\r') { // if you got anything else but a carriage return
character,

currentLine += c; // add it to the end of the currentLine

}

// Check to see if the client request was "GET /H" or "GET /L":

if (currentLine.endsWith("GET /H")) {
```

```
digitalWrite(LED_BUILTIN_1, HIGH);           // GET /H turns the LED on

}

if (currentLine.endsWith("GET /L")) {

digitalWrite(LED_BUILTIN_1, LOW);           // GET /L turns the LED off

}

}

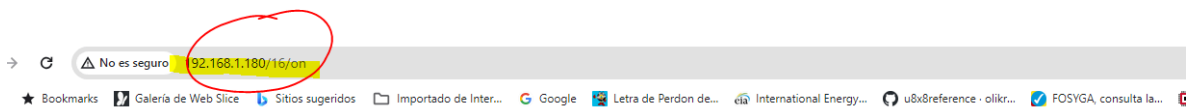
}

        client.stop();
// D, Close the connection:
        Serial.println("Client Disconnected.");
    }

}
```

Ejemplo 23d: WI-FI SERVER MEJORADO RS

```
New Client.  
GET /16/on HTTP/1.1  
Host: 192.168.1.180  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Referer: http://192.168.1.180/  
Accept-Encoding: gzip, deflate  
Accept-Language: es-ES,es;q=0.9  
  
GPIO 16 on  
Client disconnected.  
  
New Client.
```



ESP32 Web Server

GPIO 16 - State on

OFF

GPIO 17 - State off

ON

/*****

Rui Santos

Complete project details at <http://randomnerdtutorials.com>

*****/

// Funciono Agosto/10/2020.....IP =
192.168.1.28

```

//#include <ESP8266WebServer.h>

#include <WiFi.h>

#include "CRD.h"


WiFiServer server(80);


String header; //
Variable to store the HTTP request, lo que manda el browser a nuestro
servidor


String output16State = "off"; //
Auxiliar variables to store the current output state


String output17State = "off"; //
Auxiliar variables to store the current output state


const int output16 = 2; // Assign
output variables to GPIO pins


const int output17 = 17; // Assign
output variables to GPIO pins


void setup() {

    Serial.begin(115200);

    pinMode(output16, OUTPUT); //
Initialize the output variables as outputs

    pinMode(output17, OUTPUT);

    digitalWrite(output16, LOW);

```

```

digitalWrite(output17, LOW);

Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

IPAddress ip(192,168,1,180); // IP
fija, no la da el DHCP

IPAddress gateway(192,168,1,1);
IPAddress subnet (255,255,255,0);
WiFi.config(ip,gateway,subnet);

Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: "); // Print
local IP address and start web server

Serial.println(WiFi.localIP());

server.begin(); //
Inicializa el ESP32 como servidor

}

void loop(){

```



```

        WiFiClient client = server.available();           // Listen
for incoming clients

        if (client) {                                     // If a
new client connects,

                Serial.println("New Client.");           // print a
message out in the serial port

                String currentLine = "";                 // make a
String to hold incoming data from the client

                while (client.connected()) {             // loop
while the client's connected

                        if
(client.available()) {                                   // if there's
bytes to read from the client,

char c = client.read();                                  // read a byte, then

Serial.write(c);                                         // print it out the serial monitor

header += c;

if (c == '\n') {                                         // if the byte is a newline
character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

```

```

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");//????????????????????

client.println();

// turns the GPIOs on and off DE ACUERDO A LO PRESIONADO

if (header.indexOf("GET /16/on") >= 0) {

Serial.println("GPIO 16 on");

output16State = "on";

digitalWrite(output16, HIGH);

}

else if (header.indexOf("GET /16/off") >= 0) {

Serial.println("GPIO 16 off");

output16State = "off";

digitalWrite(output16, LOW);

}

else if (header.indexOf("GET /17/on") >= 0) {

```

```
Serial.println("GPIO 17 on");

output17State = "on";

digitalWrite(output17, HIGH);

}

else if (header.indexOf("GET /17/off") >= 0) {

Serial.println("GPIO 17 off");

output17State = "off";

digitalWrite(output17, LOW);

}

// Display the HTML web page CREA LA PÁGINA WEB con:
client.println("*****");

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:;\">");

// CSS to style the on/off buttons
```

```
// Feel free to change the background-color and font-size attributes to fit
your preferences
```

```
client.println("<style>html { font-family: Helvetica; display:inline-block;
margin: 0px auto; text-align: center;}");
```

```
// REPITE 1er BOTÓN
```

```
client.println(".button { background-color: #4CAF50; border:none; color:
white; padding: 16px 40px;}");
```

```
client.println("text-decoration: none; font-size: 30px; margin:2px; cursor:
pointer;}");
```

```
// REPITE 2do BOTÓN
```

```
client.println(".button2 {background-color:#555555;}</style></head>");
```

```
// Web Page Heading.....1er ENCABEZADO DE LA PÁGINA WEB
```

```
client.println("<body><h1>ESP32 Web Server</h1>");
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
// Display current state, and ON/OFF buttons for GPIO 16
```

```
client.println("<p>GPIO 16 - State " + output16State + "</p>");
```

```
// If the output16State is off, it displays the ON button
```

```

if (output16State=="off") {

    client.println("<p><a href=\"/16/on\"><button
class=\"button\">ON</button></a></p>");

}

else

{

    client.println("<p><a href=\"/16/off\"><button class=\"button
button2\">OFF</button></a></p>");

}

// Display current state, and ON/OFF buttons for GPIO 17

client.println("<p>GPIO 17 - State " + output17State + "</p>");

// If the output17State is off, it displays the ON button

if (output17State=="off") {

    client.println("<p><a href=\"/17/on\"><button
class=\"button\">ON</button></a></p>");

}

else

{

    client.println("<p><a href=\"/17/off\"><button class=\"button
button2\">OFF</button></a></p>");

```

```
}

client.println("</body></html>");

// The HTTP response ends with another blank line

client.println(); // La
página termina con una línea en blanco

// Break out of the while loop

break;

}

else { // if you got a newline, then clear
currentLine

currentLine = "";

}

}

else if (c != '\r') { // if you got anything else but a carriage return
character,

currentLine += c; // add it to the end of the currentLine

}
```

```
}
```

```
}
```

```
// Clear the header variable
```

```
header = "";
```

```
// Close the connection
```

```
client.stop();
```

```
Serial.println("Client disconnected.");
```

```
Serial.println("");
```

```
}
```

```
}
```

Ejemplo 23e: WI-FI SERVER SOFT ACCESS POINT

```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1448
load:0x40078000,len:14844
ho 0 tail 12 room 4
load:0x40080400,len:4
load:0x40080404,len:3356
entry 0x4008059c
Setting AP (Access Point) AP IP address: 192.168.4.1
```

Primer banco de Programas ESP32

Ejemplo 23e: WI-FI SERVER SOFT ACCESS POINT

```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1448
load:0x40078000,len:14844
ho 0 tail 12 room 4
load:0x40080400,len:4
load:0x40080404,len:3356
entry 0x4008059c
Setting AP (Access Point) AP IP address: 192.168.4.1
```

Red Conectado

- ESP32-Access-Point
- DANIELA_5G
- DIRECT-F5-HP Deskjet 5820 series
- FAMILIA_MORENO
- MOVISTAR_88B9
- MOVISTAR_PLUS_88B9
- MOVISTAR_CIRRA_R100

Configuración de red e Internet

Cambia los ajustes de configuración, como hacer que una conexión sea de uso medido.

Wi-Fi Modo avión Zona con cobertura

10:12 a.m. 13/06/2024


```
/*  
Rui Santos  
Complete project details at https://randomnerdtutorials.com  
*/  
  
// Load Wi-Fi library  
#include <WiFi.h>  
  
// Replace with your network credentials  
const char* ssid      = "ESP32-Access-Point";  
const char* password = "123456789";  
  
// Set web server port number to 80  
WiFiServer server(80);  
  
// Variable to store the HTTP request  
String header;  
  
// Auxiliar variables to store the current output state  
String output16State = "off";  
String output17State = "off";  
  
// Assign output variables to GPIO pins  
const int output16 = 2;  
const int output17 = 17;
```

```

void setup() {

    Serial.begin(115200);

    pinMode(output16, OUTPUT);

    pinMode(output17, OUTPUT);

    digitalWrite(output16, LOW);

    digitalWrite(output17, LOW);


    Serial.print("Setting AP (Access Point)...");

    WiFi.softAP(ssid, password);           // Remove the
password parameter, if you want the AP (Access Point) to be open. SOFT AP

                                           // Metodos:
    .softAP(const char* ssid, const char* password, int channel, int
ssid_hidden, int max_connection)

        IPAddress IP = WiFi.softAPIP();

    Serial.print("AP IP address: ");

    Serial.println(IP);

    server.begin();

}

void loop(){

    WiFiClient client = server.available(); // Listen for
incoming clients


    if (client) {                           // If a new client
connects,

        Serial.println("New Client.");      // print a
message out in the serial port

        String currentLine = "";           // make a
String to hold incoming data from the client

```

```

        while (client.connected()) {           // loop
while the client's connected

                                if
(client.available()) {           // if there's bytes to read from the
client,

char c = client.read();           // read a byte, then

Serial.write(c);                 // print it out the serial monitor

header += c;

if (c == '\n') {                 // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

```

```
// turns the GPIOs on and off

if (header.indexOf("GET /16/on") >= 0) {

    Serial.println("GPIO 16 on");

    output16State = "on";

    digitalWrite(output16, HIGH);

}

else if (header.indexOf("GET /16/off") >= 0) {

    Serial.println("GPIO 16 off");

    output16State = "off";

    digitalWrite(output16, LOW);

}

else if (header.indexOf("GET /17/on") >= 0) {

    Serial.println("GPIO 17 on");

    output17State = "on";

    digitalWrite(output17, HIGH);

}
```

```
else if (header.indexOf("GET /17/off") >= 0) {

    Serial.println("GPIO 17 off");

    output17State = "off";

    digitalWrite(output17, LOW);

}

// Display the HTML web page

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">>");

// CSS to style the on/off buttons

// Feel free to change the background-color and font-size attributes to fit
your preferences

client.println("<style>html { font-family: Helvetica; display: inline-block;
margin: 0px auto; text-align: center;});");

client.println(".button { background-color: #4CAF50; border: none; color:
white; padding: 16px 40px;});");
```

```
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;});
```

```
client.println(".button2 {background-color: #555555;}</style></head>");
```

```
// Web Page Heading
```

```
client.println("<body><h1>ESP32 Web Server</h1>");
```

```
// Display current state, and ON/OFF buttons for GPIO 16
```

```
client.println("<p>GPIO 16 - State " + output16State + "</p>");
```

```
// If the output26State is off, it displays the ON button
```

```
if (output16State=="off") {
```

```
client.println("<p><a href=\"/16/on\"><button  
class=\"button\">ON</button></a></p>");
```

```
}
```

```
else {
```

```
client.println("<p><a href=\"/16/off\"><button class=\"button  
button2\">OFF</button></a></p>");
```

```
}
```

```
// Display current state, and ON/OFF buttons for GPIO 17

client.println("<p>GPIO 17 - State " + output17State + "</p>");

// If the output17State is off, it displays the ON button

if (output17State=="off") {

    client.println("<p><a href=\""/17/on\"><button
class=\"button\">ON</button></a></p>");

}

else {

    client.println("<p><a href=\""/17/off\"><button class=\"button
button2\">OFF</button></a></p>");

}

client.println("</body></html>");

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

break;
```

```

}

else                                     { // if you got a newline, then clear
currentLine

currentLine = "";

}

}

else if (c != '\r') { // if you got anything else but a carriage return
character,

currentLine += c;      // add it to the end of the currentLine

}

}

}

// Clear the header variable
header = "";

// Close the connection
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
}

}

```


Ejemplo 24: BLUETOOTH

```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

#define led 2

void setup()
{
    SerialBT.begin("ESP--32 BL 2024");

    //bool begin(String localName = String(), bool isMaster =
false, bool disableBLE = false);

    Serial.begin(115200);
    pinMode(led, OUTPUT);
    Serial.print("ESP32 SDK: ");
    delay(1000);
    if (SerialBT.available())
```

```

    {
        Serial.print("BL Ok");
    }
    else
    {
        Serial.print("BL NO Ok");
    }
}

void loop()
{
    static uint8_t lastPinState = 1;
    uint8_t pinState = digitalRead(0);
    if (SerialBT.available())
    {
        char c = SerialBT.read();
        if (c == '1')
        {
            digitalWrite(led, HIGH);
        }
        else if (c == '0')
        {
            digitalWrite(led, LOW );
        }
    }
}

```

Ejemplo 25:

a