Microcontroladores II

Juan Esteban Giraldo Hoyos
Ingeniero Electrónico
Magíster en Gestión de Ciencia, Tecnología e Innovación

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Proyectos de Aula

Proyectos:

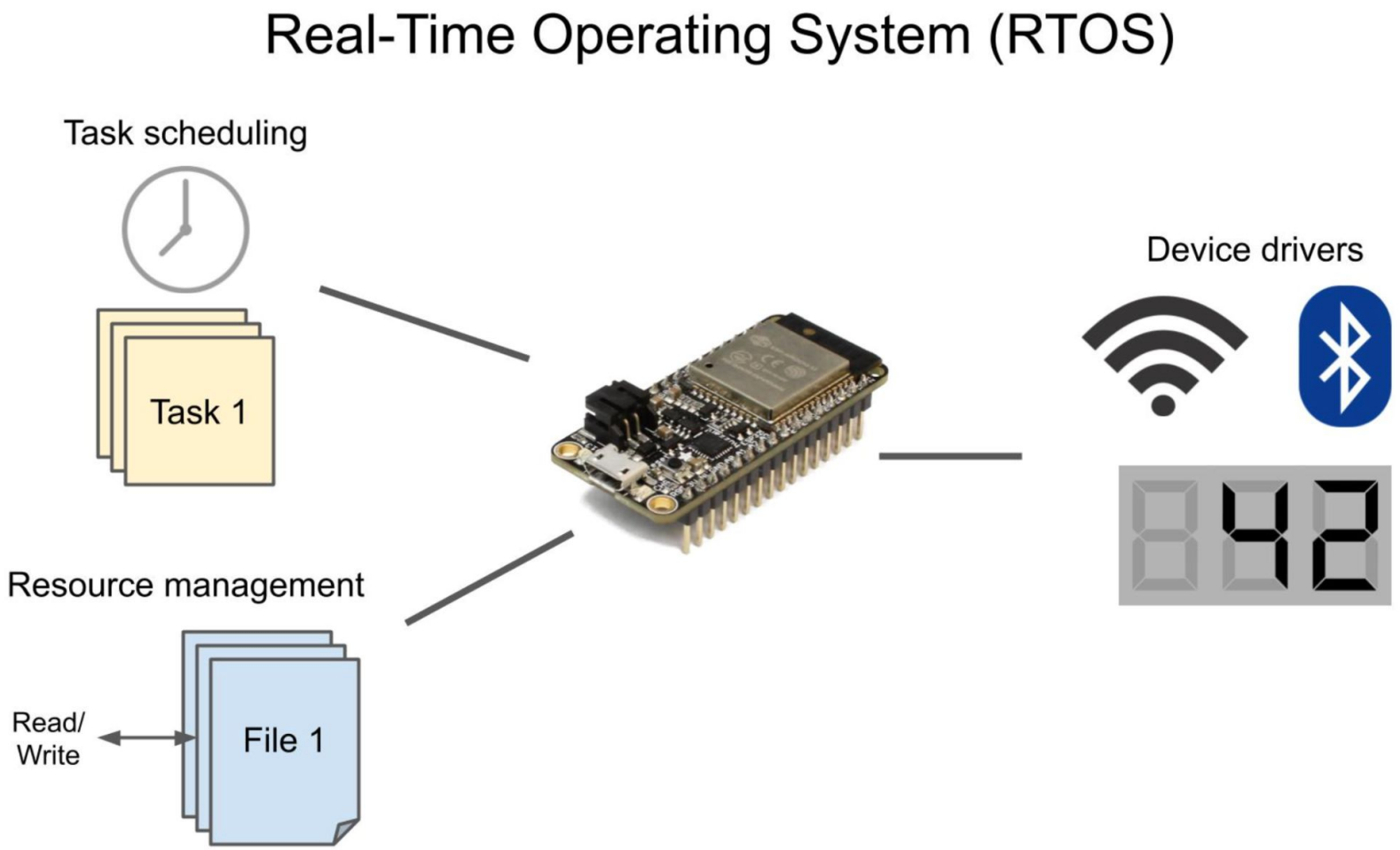
1) Sistema de alerta por detección de gas y/o incendio, visualización de temperatura y humedad en tiempo real. Y visualización del histórico gráfica de la temperatura. Debe tener una interfaz de visualización en el PC, usando el ESP32 con FreeRTOS
2) Sistema de Control de Acceso por RFID: Crea un sistema para controlar el acceso mediante tarjetas RFID. FreeRTOS puede manejar la lectura de tarjetas, la activación de mecanismos de apertura de puertas y la comunicación en un PC que muestre si el acceso se ha denegado o se ha habilitado, usando el ESP32 con FreeRTOS

Microcontroladores II - Proyectos de Aula

Proyectos:

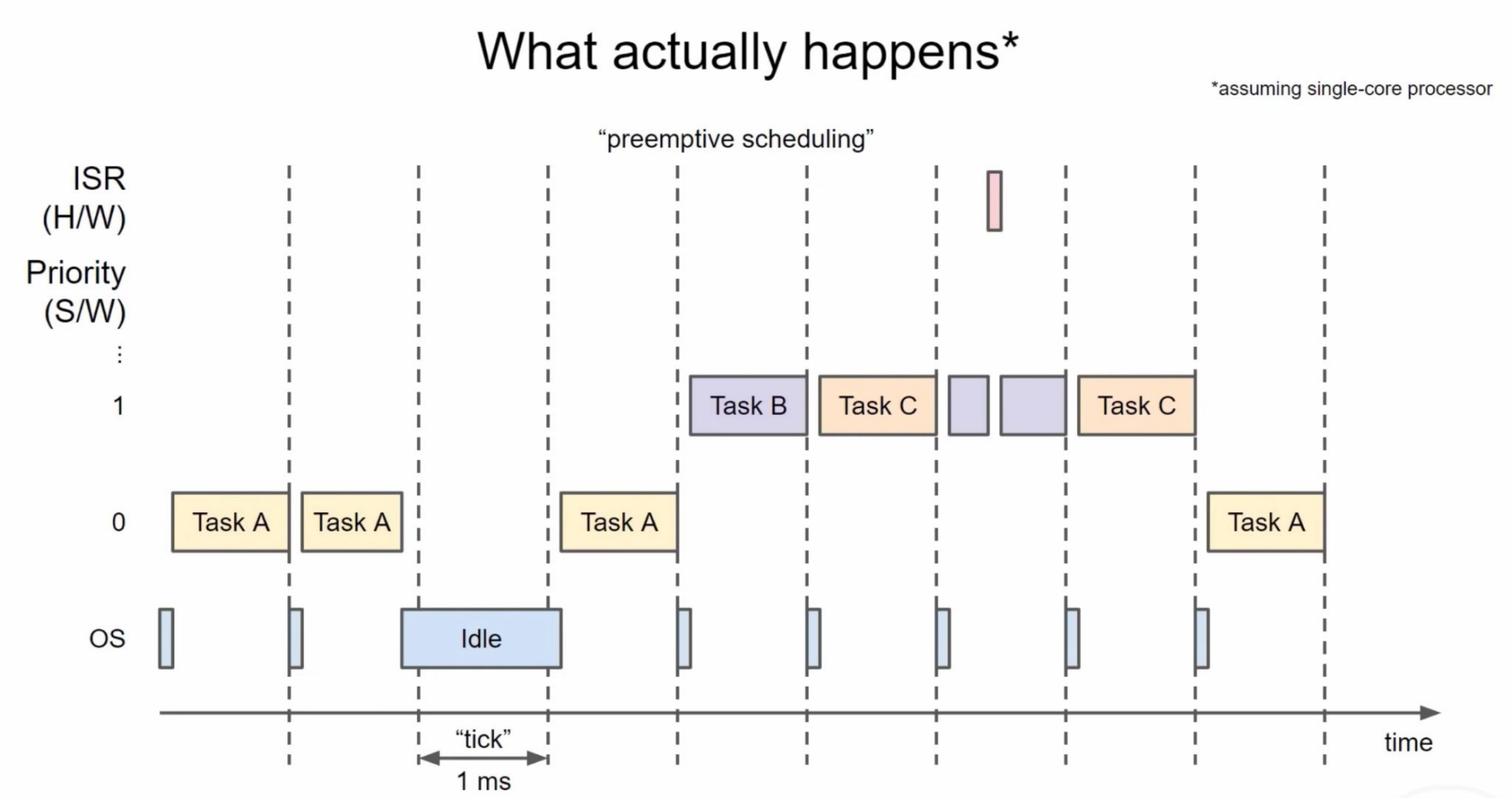
3) Reloj despertador usando un display LCD, que se configure por medio de comunicación bluetooth desde el celular y que reproduzca un sonido cuando se llegue a la hora configurada, usando el ESP32 con FreeRTOS.
4) Robot Controlado por Bluetooth: Construye un pequeño robot con motores controlados mediante un módulo Bluetooth. Utiliza FreeRTOS para gestionar tareas como el control de movimiento, la comunicación Bluetooth y la detección de obstáculos.

Microcontroladores II - Hoy trabajaremos



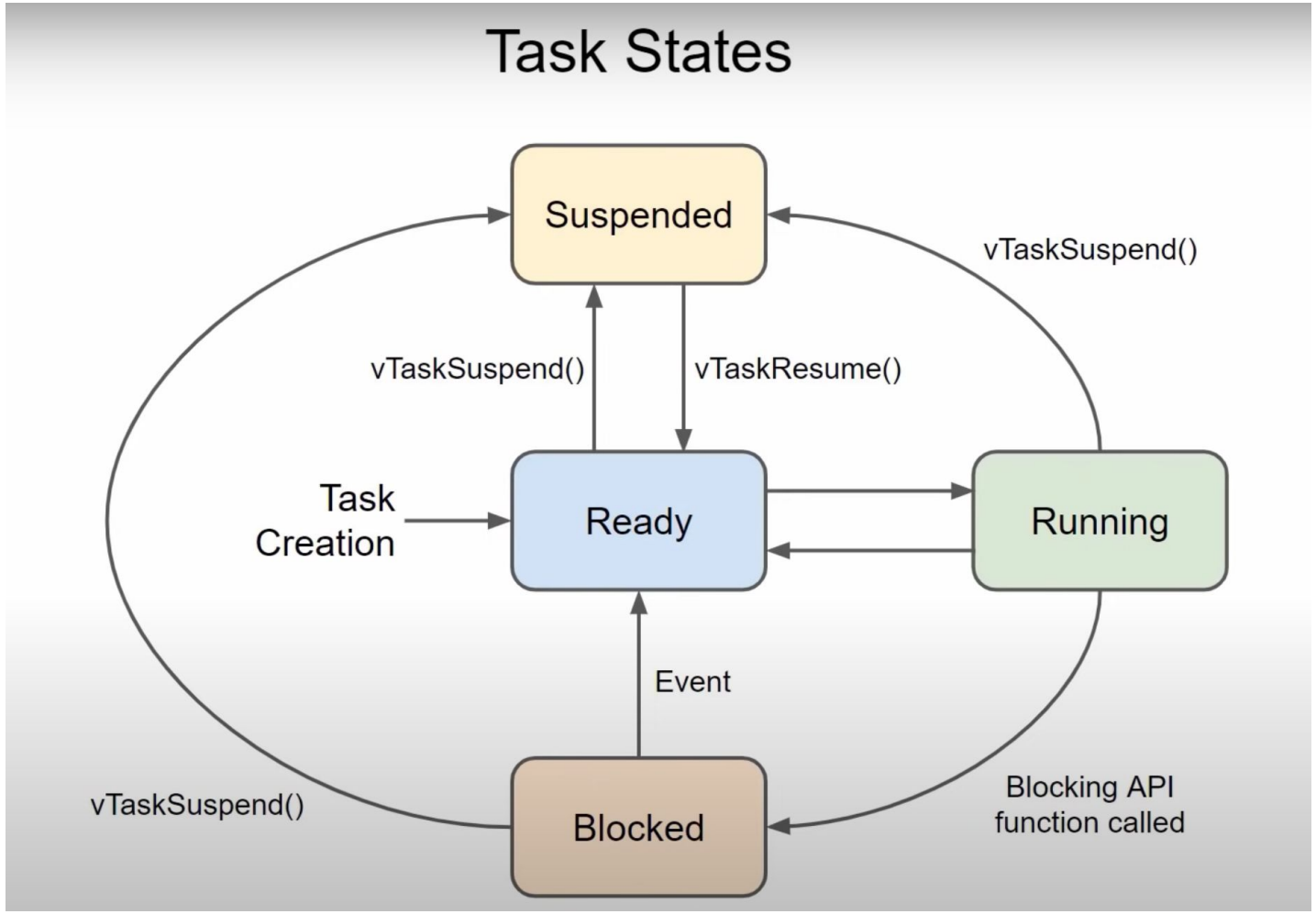
Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Hoy trabajaremos



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Hoy trabajaremos



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS PRACTIQUEMOS

Realizar el siguiente ejemplo que permite configurar por medio de la comunicación serial el tiempo de encendido y apagado de un led usando 2 tareas

FREERTOS_DEMO_2

```
// 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
#define led_pin 23

// Globals
static int led_delay = 500;   // ms

//*****************************************************************************
// Tasks

// Task: Blink LED at rate set by global variable
void toggleLED(void *parameter) {
  while (1) {
    digitalWrite(led_pin, HIGH);
    vTaskDelay(led_delay / portTICK_PERIOD_MS);
    digitalWrite(led_pin, LOW);
    vTaskDelay(led_delay / portTICK_PERIOD_MS);
  }
}
```

FREERTOS_DEMO_2

```
// Task: Read from serial terminal
void readSerial(void *parameters) {
  String serialData = "";
  // Loop forever
  while (1) {
    // Read characters from serial
    if (Serial.available() > 0) {
      //Recibe caracteres hasta el de retorno
      serialData = Serial.readStringUntil('\n');
      //Actualiza la variable para el tiempo del blink
      led_delay = serialData.toInt();
      //Reinicia string para obtener dato del serial
      serialData = "";
      Serial.print("Updated LED delay to: ");
      Serial.println(led_delay);
    }
  }
}
```

Microcontroladores II - RTOS PRACTIQUEMOS

Realizar el siguiente ejemplo que permite configurar por medio de la comunicación serial el tiempo de encendido y apagado de un led usando 2 tareas

```
void setup() {
  pinMode(led_pin, OUTPUT); // Configure pin
  Serial.begin(115200); // Configure serial and wait a second
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  Serial.println("Multi-task LED Demo");
  Serial.println("Enter a number in milliseconds to change the LED delay.");

  // Start blink task
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
            toggleLED,      // Function to be called
            "Toggle LED",   // Name of task
            1024,           // Stack size (bytes in ESP32, words in FreeRTOS)
            NULL,           // Parameter to pass
            1,              // Task priority
            NULL,           // Task handle
            app_cpu);       // Run on one core for demo purposes (ESP32 only)

  // Start serial read task
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
            readSerial,     // Function to be called
            "Read Serial",  // Name of task
            1024,           // Stack size (bytes in ESP32, words in FreeRTOS)
            NULL,           // Parameter to pass
            1,              // Task priority (must be same to prevent lockup)
            NULL,           // Task handle
            app_cpu);       // Run on one core for demo purposes (ESP32 only)

  vTaskDelete(NULL); // Delete "setup and loop" task
}
```

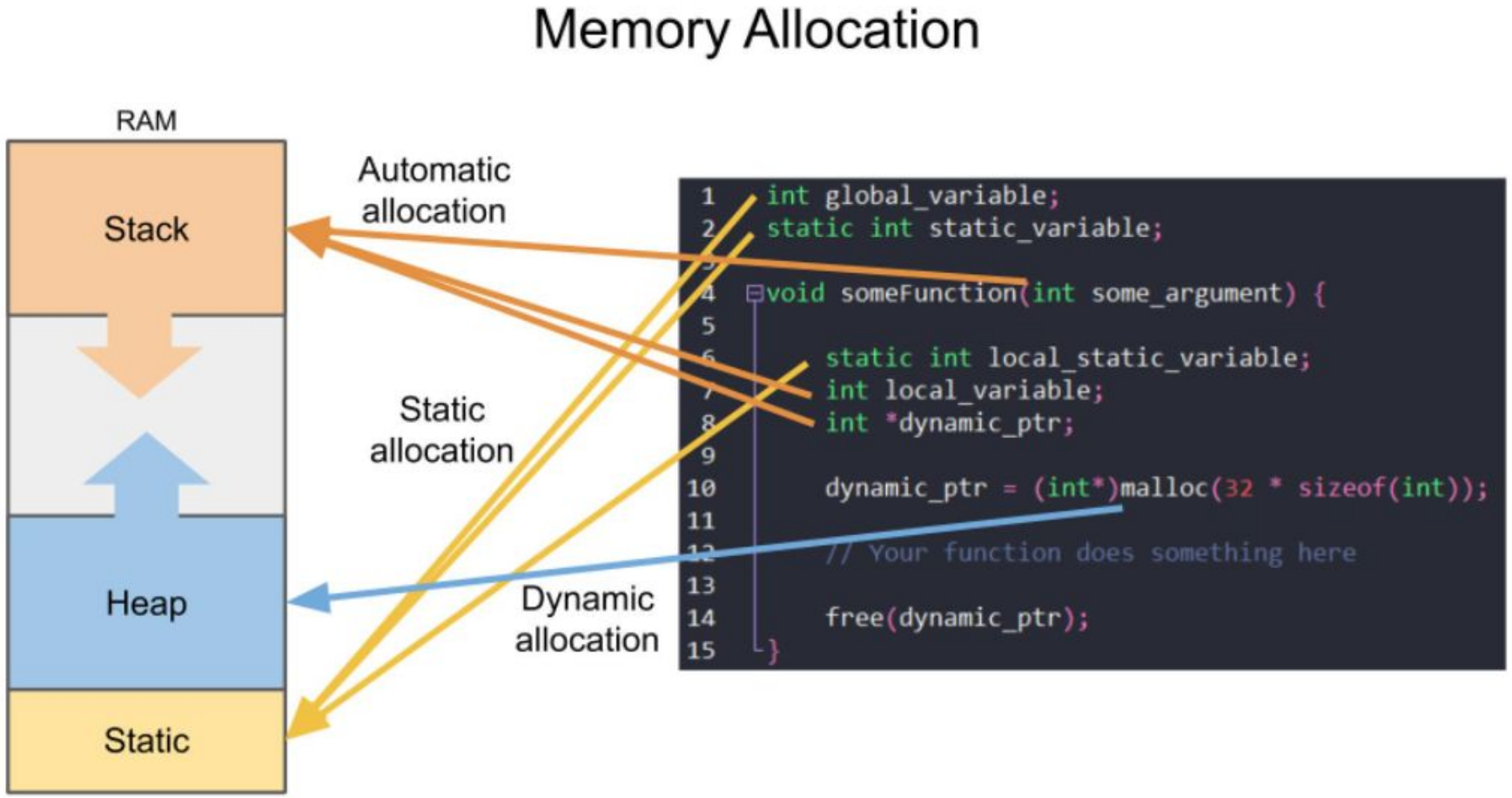
```
void loop() {
  // Execution should never get here
}
```

Escuela de Ingenierías

IUSH Institución Universitaria SALAZAR Y HERRERA

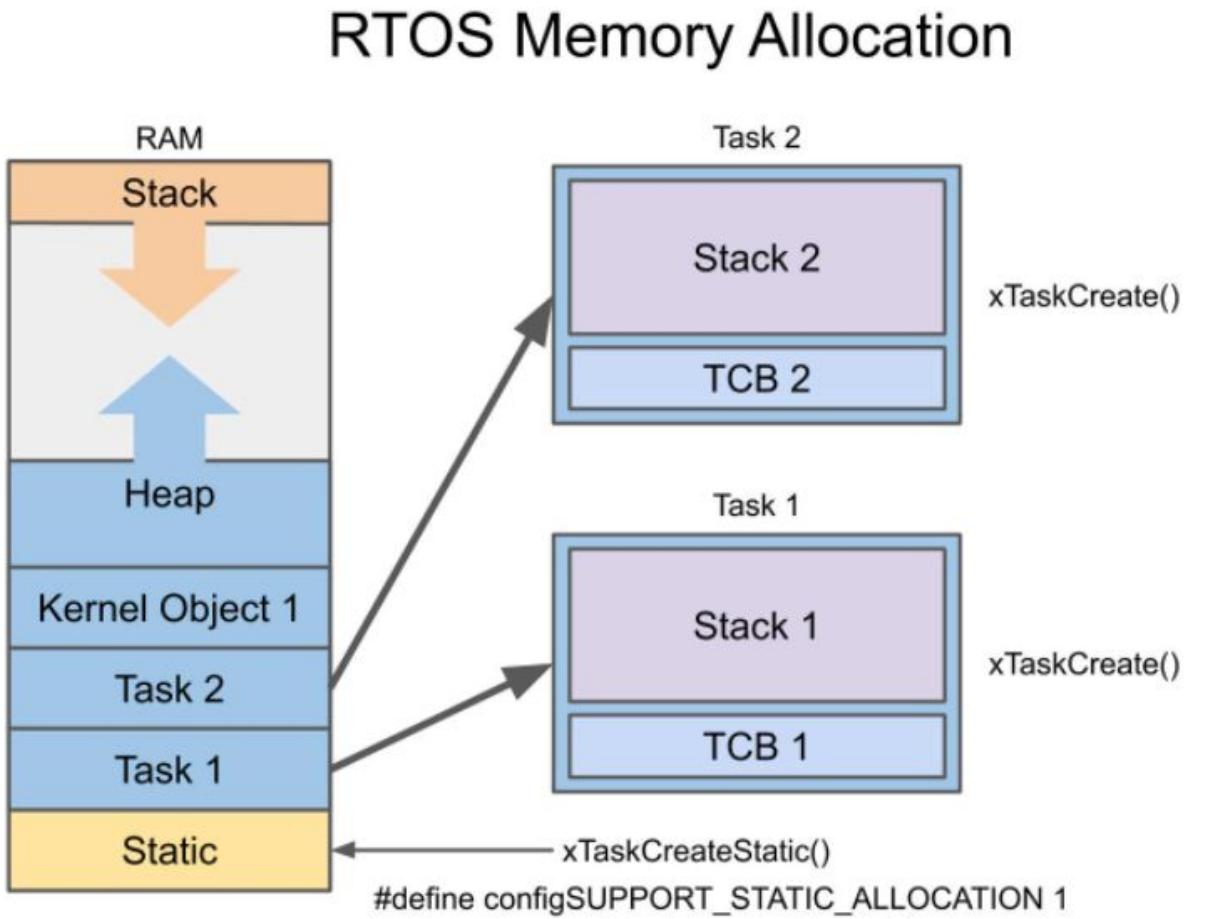
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS



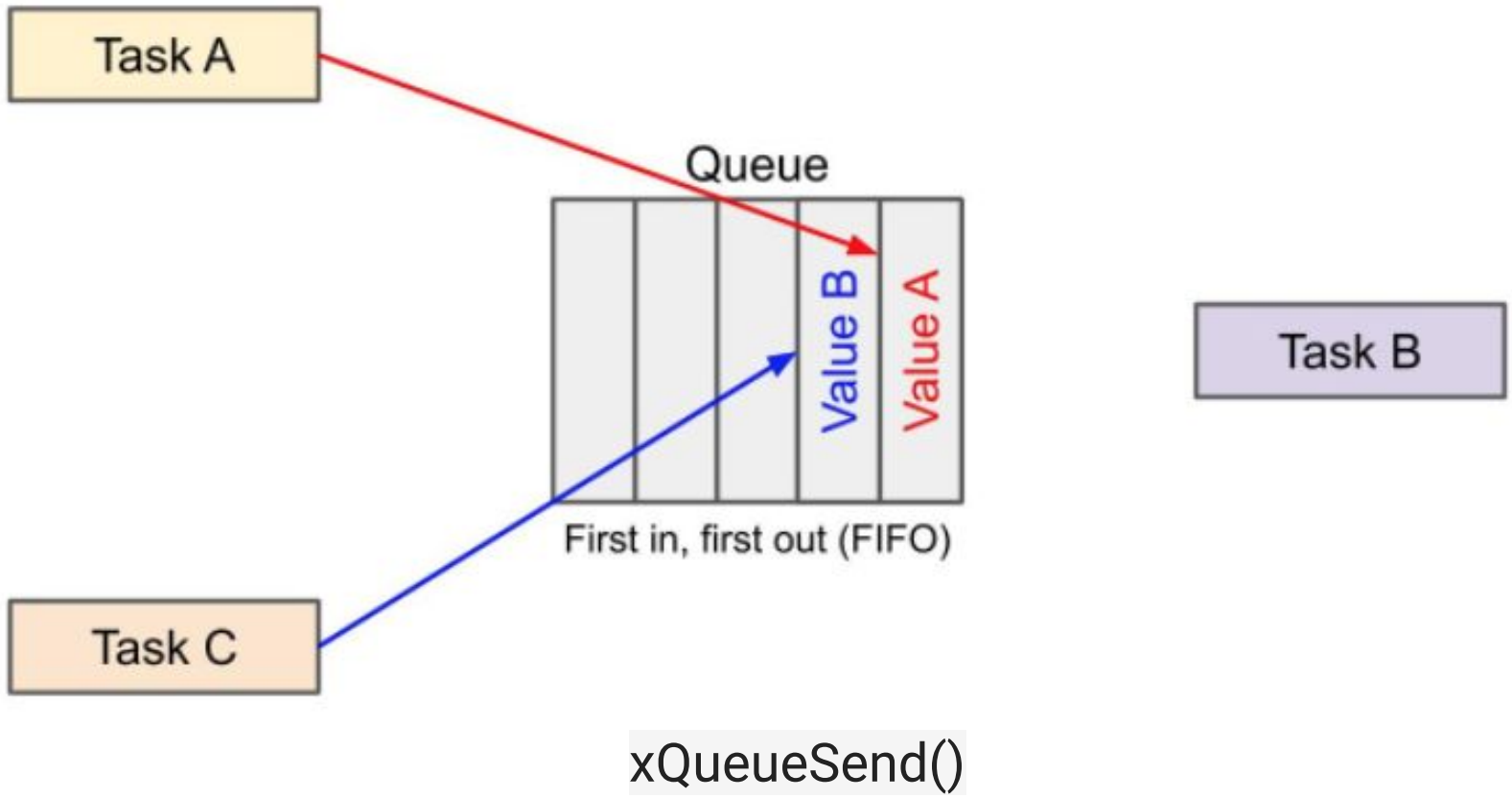
Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS



Microcontroladores II - RTOS

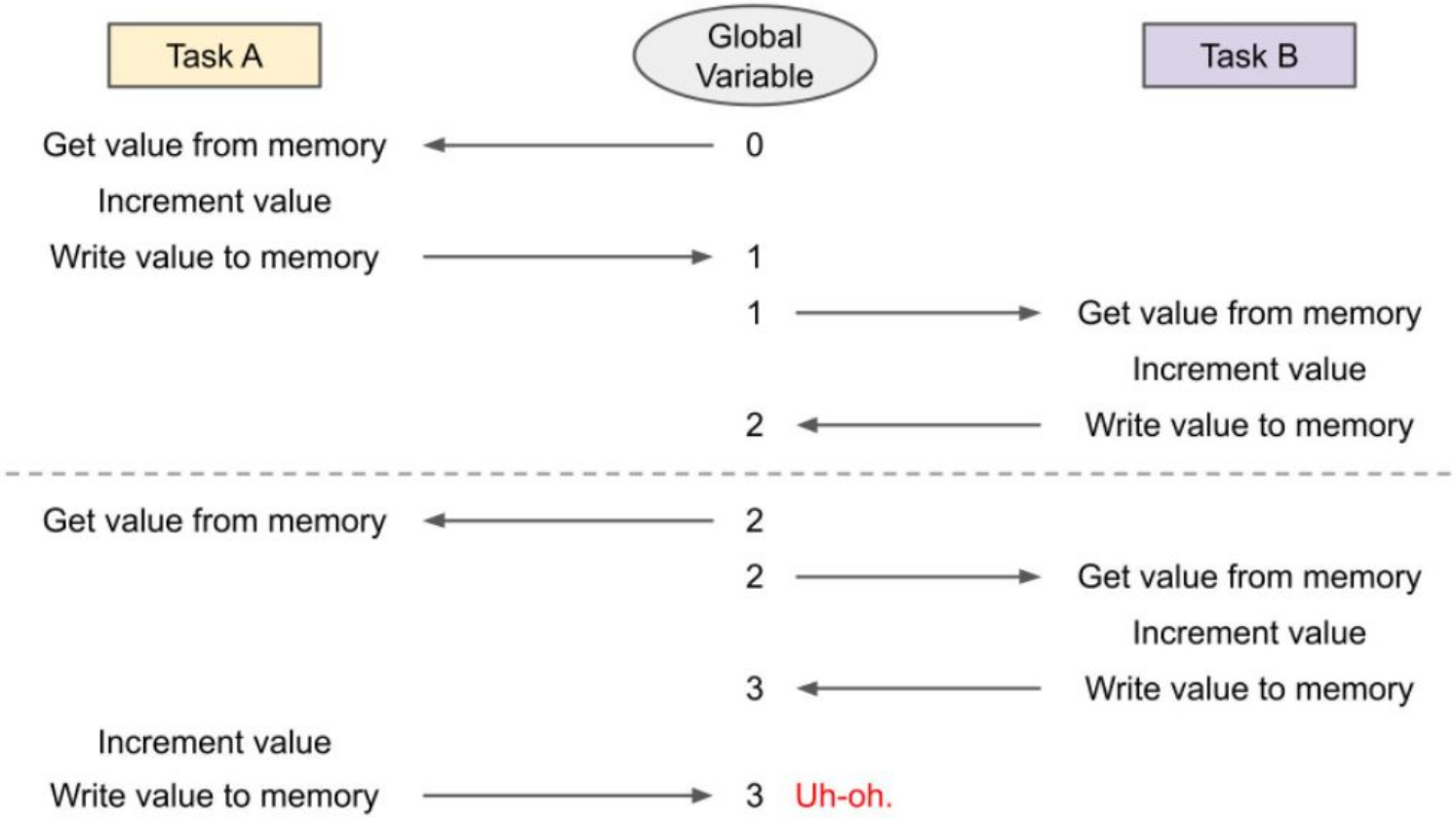
Una cola (Queue) en un sistema operativo en tiempo real (RTOS) es un objeto del núcleo que es capaz de pasar información entre tareas sin incurrir en sobrescrituras de otras tareas ni entrar en una condición de carrera. Una cola es un sistema de primero en entrar, primero en salir (FIFO) donde los elementos se eliminan de la cola una vez leídos.



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS

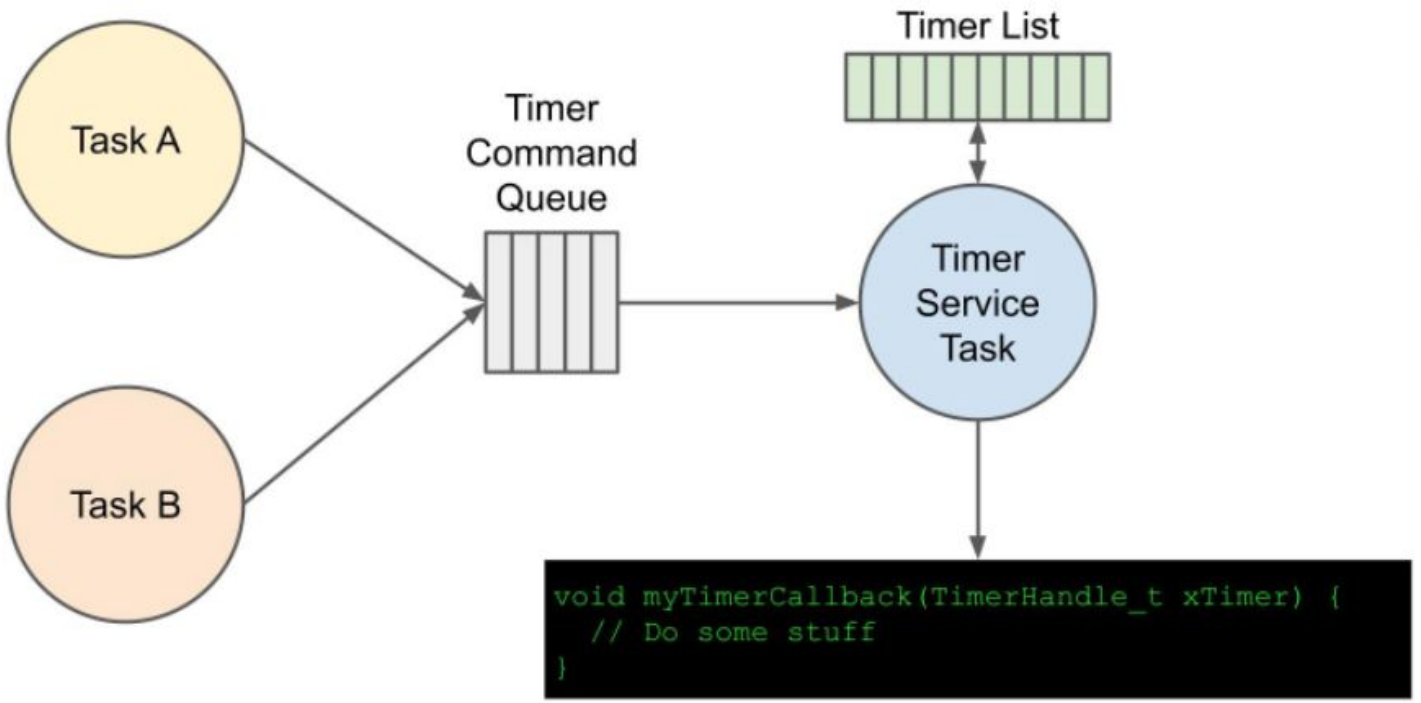
Un Mutex (abreviatura de MUTual EXclusion) es una bandera o bloqueo que se utiliza para permitir que solo un hilo acceda a una sección de código a la vez. Bloquea (o bloquea) el acceso de todos los demás subprocesos al código o recurso. Esto garantiza que todo lo que se ejecute en esa sección crítica sea seguro y que otros subprocesos no corrompan la información.



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS

Software Timers in FreeRTOS



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS

Realizar el siguiente ejemplo que permite configurar una interrupción por software de timer con su rutina de atención de la interrupción. El ejemplo recibe por serial algún dato, cuando eso pasa enciende un led y activa el timer para que al pasar 5 seg se apague el led en la rutina de atención de la interrupción del timer

FREERTOS_DEMO_4

```
// 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

// Settings
static const TickType_t dim_delay = 5000 / portTICK_PERIOD_MS;

// Pins (change this if your Arduino board does not have LED_BUILTIN defined)
#define led_pin 23

// Globals
static TimerHandle_t one_shot_timer = NULL;

//*****************************************************************************
// Callbacks

// Turn off LED when timer expires
void autoDimmerCallback(TimerHandle_t xTimer) {
  digitalWrite(led_pin, LOW);
}
```

```
//*****************************************************************************
// Tasks

// Echo things back to serial port, turn on LED when while entering input
void doCLI(void *parameters) {
  char c;

  while (1) {
    // See if there are things in the input serial buffer
    if (Serial.available() > 0) {

      // If so, echo everything back to the serial port
      c = Serial.read();
      Serial.print(c);

      // Turn on the LED
      digitalWrite(led_pin, HIGH);

      // Start timer (if timer is already running, this will act as
      // xTimerReset() instead)
      xTimerStart(one_shot_timer, portMAX_DELAY);
    }
  }
}
```

Microcontroladores II - RTOS

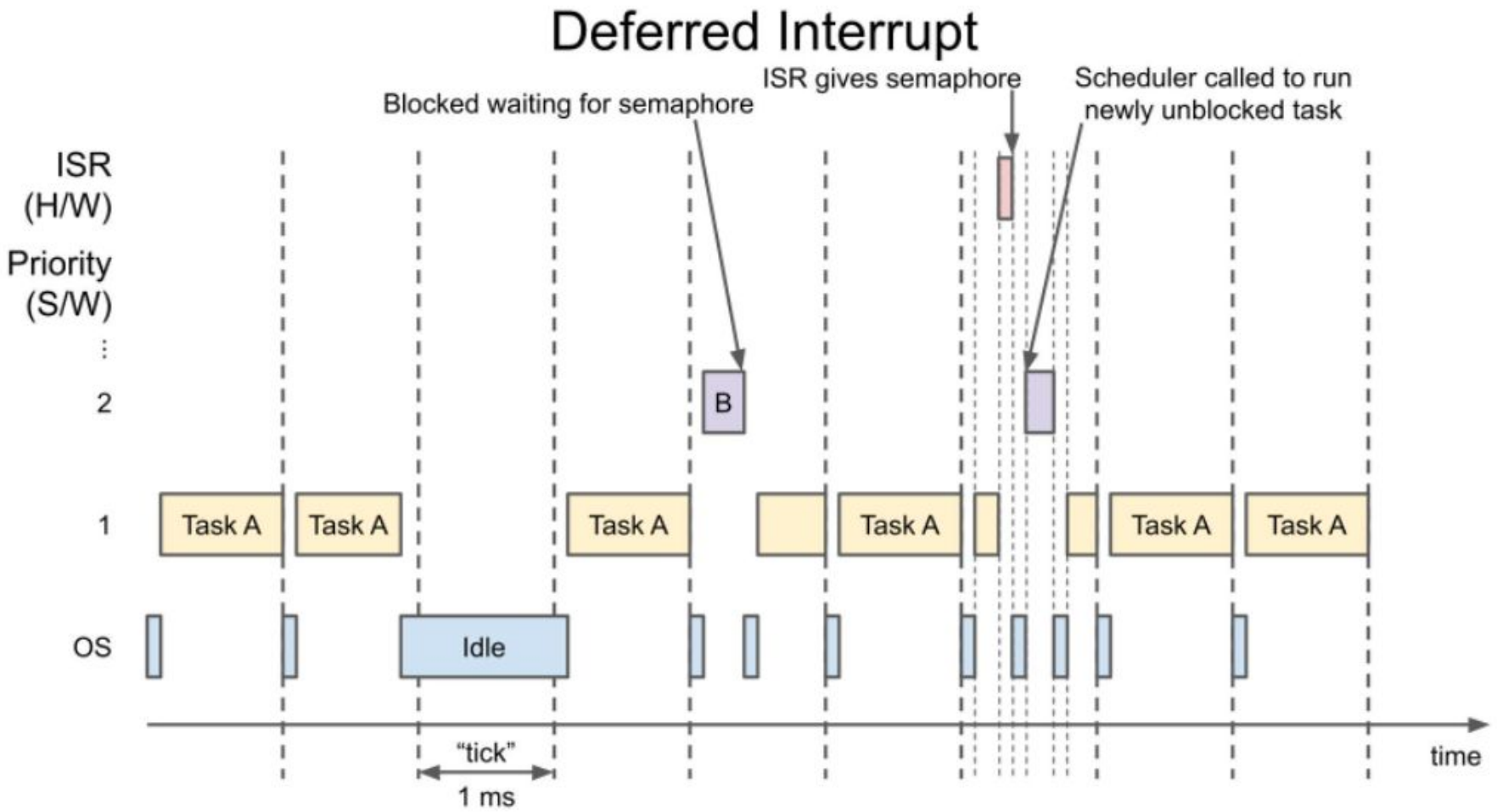
Realizar el siguiente ejemplo que permite configurar una interrupción por software de timer con su rutina de atención de la interrupción. El ejemplo recibe por serial algún dato, cuando eso pasa enciende un led y activa el timer para que al pasar 5 seg se apague el led en la rutina de atención de la interrupción del timer

```
//*****************************************************************************
// Main (runs as its own task with priority 1 on core 1)
void setup() {
  pinMode(led_pin, OUTPUT); // Configure LED pin
  Serial.begin(115200);  // Configure Serial
  vTaskDelay(1000 / portTICK_PERIOD_MS); // Wait a moment to start (so we don't miss Serial output)
  Serial.println();
  Serial.println("---FreeRTOS Timer Solution---");

  // Create a one-shot timer
  one_shot_timer = xTimerCreate(
                     "One-shot timer",     // Name of timer
                     dim_delay,            // Period of timer (in ticks)
                     pdFALSE,              // Auto-reload
                     (void *)0,            // Timer ID
                     autoDimmerCallback);  // Callback function

  // Start command line interface (CLI) task
  xTaskCreatePinnedToCore(doCLI,
                          "Do CLI",
                          1024,
                          NULL,
                          1,
                          NULL,
                          app_cpu);

  // Delete "setup and loop" task
  vTaskDelete(NULL);
}
```

```
void loop() {
  // Execution should never get here
}
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS Hardware interrupts



Escuela de Ingenierías
SER PERSONA DESDE EL SABER Y EL HACER
IUSH Institución Universitaria SALAZAR Y HERRERA

Microcontroladores II - RTOS Hardware interrupts

FREERTOS_DEMO_5

```
// 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

// Definimos el pin del botón
#define buttonPin 22
#define ledPin 23

// Variable para manejar la cola
QueueHandle_t xQueue;

// Prototipo de la tarea y la función de interrupción
void IRAM_ATTR isrHandler() {
  // Leemos el estado del pin
  int pinState = digitalRead(buttonPin);

  // Enviamos el estado a la cola
  BaseType_t xHigherPriorityTaskWoken = pdFALSE;
  xQueueSendFromISR(xQueue, &pinState, &xHigherPriorityTaskWoken);

  // Esto verifica si se debe dar paso a otra tarea de mayor prioridad
  if (xHigherPriorityTaskWoken) {
    portYIELD_FROM_ISR();
  }
}
```

```
void taskHandler(void *parameter) {
  int receivedValue;

  while (true) {
    // Esperamos un elemento en la cola
    if (xQueueReceive(xQueue, &receivedValue, portMAX_DELAY)) {
      // Acción que se realiza cuando se recibe un elemento
      Serial.print("Interrupción detectada, nuevo estado del pin: ");
      Serial.println(receivedValue);
      digitalWrite(ledPin, receivedValue);
    }
  }
}
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - RTOS Hardware interrupts

```
void setup() {
  // Inicializamos la comunicación serial para depurar
  Serial.begin(115200);

  // Configuramos el pin del botón como entrada con resistencia pull-up
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);

  // Creamos una cola para comunicación entre la interrupción y la tarea
  xQueue = xQueueCreate(10, sizeof(int));

  if (xQueue == NULL) {
    Serial.println("Error: No se pudo crear la cola");
    return;
  }

  // Creamos la tarea que espera eventos de la cola
  xTaskCreatePinnedToCore(taskHandler,
                          "taskHandler",
                          1024,
                          NULL,
                          1,
                          NULL,
                          app_cpu);

  // Configuramos la interrupción en el pin del botón
  attachInterrupt(digitalPinToInterrupt(buttonPin), isrHandler, CHANGE);
}
```

```
void loop() {
  // El loop se mantiene vacío ya que todo se maneja en FreeRTOS
}
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

FREERTOS Y ESP32 PRACTICA 10%

Implementar un programa que tenga:

- 1 tarea que recibe un número en milisegundos por la comunicación serial y cambia el tiempo de encendido y apagado de un led.
- 1 tarea que encienda y apague un led en el tiempo definido por el dato recibido por serial.
- 1 tarea que encienda o apague un led adicional dependiendo del estado recibido por una variable que es controlada por el estado de un botón
- 1 Interrupción de hardware de tipo FALLING que llama una rutina de atención de interrupciones cuando el estado de un botón pasa de HIGH a LOW y envía dicho estado a una variable usando una QUEUE para que sea recibida por otra tarea
- 1 Interrupción de software de Timer que cada 15 seg muestre por consola la temperatura sensada por un sensor de temperatura de su preferencia

Bibliografía

- https://www.freertos.org/Documentation/Mastering-the-FreeRTOS-Real-Time-Kernel.v1.0.pdf
- https://www.digikey.com/en/maker/projects/what-is-a-realtime-operating-system-rtos/28d8087f53844decafa5000d89608016
- https://www.youtube.com/watch?v=kP-pP6FEu8I&list=PLzvRQMJ9HDiQ3OIuBWCEW6yE0S0LUWhGU&index=20
- https://drive.google.com/file/d/1Lf2deyf0xiE3iye8TglEaVfmIN05to8a/view
- https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER