



UNIVERSITAT  
POLITÀCNICA  
DE VALÈNCIA

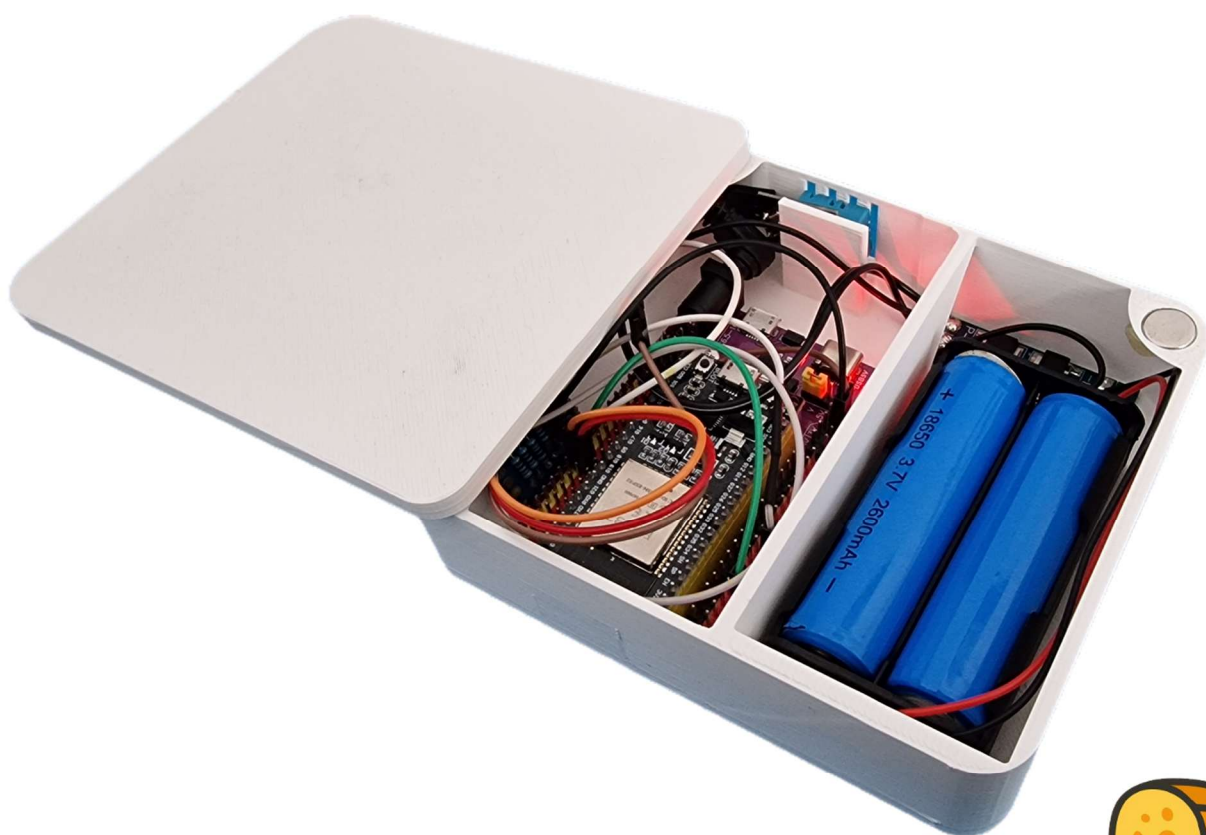
Escola Tècnica  
Superior d'Enginyeria  
Informàtica



Escuela Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# AutoCheese



## Nodo de Temperatura



**Grupo A9** | SEM | *Informática Industrial y Robótica*

*Integrantes del equipo:*

Mario Martínez Carneros,

Kai Aoiz Canillas,

Vicente Burdeus Sánchez,

Francisco Nortes Novikov



# Índice

---

## Tabla de contenidos

Índice .....	2
1. Introducción .....	4
2. Objetivos.....	4
2.1 Alcance.....	4
2.1.1 Nodos .....	5
2.1.2 Base de datos.....	5
2.1.3 SCADA.....	5
2.2 Requisitos que cumplir .....	6
3. Especificaciones.....	7
3.1 Componentes Hardware .....	7
3.1.1 Controlador .....	7
3.1.2 LEDs de estado .....	8
3.1.2 Sensor temperatura .....	8
3.1.4 Alimentación.....	8
3.1.5Carcasa .....	9
3.1.6 Interruptor sensor + divisor resistivo.....	10
3.2 Software.....	11
3.2.1 Constantes y pines .....	12
3.2.2 Pin_init.....	12
3.2.3 WiFi.....	12
3.2.4 MQTT .....	13
3.2.5 Sensor temperatura y humedad.....	14
	2

---

3.3.6 Medidor tensión de la batería .....	14
3.2.7 Deep Sleep .....	15
Bibliografia .....	16

# 1. Introducción

Esta rama del proyecto consiste en el diseño e implementación de una forma de medir la temperatura y humedad de las salas de curado de queso. Para esto, se busca una solución que permita una instalación rápida y simple sin necesidad de modificar las salas de curado ya existentes.

También se busca que esta solución tenga el mantenimiento mínimo posible.

Para la implementación del sistema de curado de quesos completamente automatizado es necesario mantener bajo control la humedad y temperatura de las salas de curado, pero estas salas generalmente son muy grandes y necesitan tomar las mediciones en diferentes puntos para asegurarse de un control correcto y adicionalmente tener un registro de estas.

La idea desarrollada consiste en la instalación de diferentes nodos sensores a lo largo de la sala los cuales miden la temperatura y la transmiten a la base de datos donde se guarda el registro.

Para asegurar la conectividad de los nodos se aprovecha la red MESH instalada.

Los nodos durmientes funcionan por baterías las cuales han de ser reemplazadas por los operarios de mantenimiento. Estos notificaran cuando necesiten un cambio de batería.

## 2. Objetivos

### 2.1 Alcance

Este subproyecto consiste en el desarrollo e implementación de:

1. Nodos sensores
2. Implementación de las estructuras de datos en la BBDD.
3. Implementación de un SCADA para los operarios.

Si hay algo que no esté nombrado aquí o en las siguientes partes del proyecto significa que esa parte no se va a desarrollar o no pertenece al proyecto.

El sistema MQTT y el Wifi ya existen en la instalación.

### 2.1.1 Nodos

Los nodos sensores de temperatura consisten en una pequeña carcasa de plástico (HDPE o PLA)

Esta cuenta con una rejilla la cual permite al sensor de temperatura y humedad tomar los datos de la sala, este compartimento del sensor está separado NO herméticamente del interior de la carcasa.

Tiene una tapa magnética en la parte superior la cual al retirarla permite ver el alojamiento de las baterías intercambiables. Estas son dos celdas li-ion 18650 con una tensión nominal de 3.7 V.

Por último, el lado izquierdo del nodo tiene 2 led de estado uno rojo y uno blanco estos solo están activos al encenderse por primera vez el nodo (cambio de batería) o si estos no se han iniciado correctamente se mantienen indicando el código de error.

Los nodos mandan de forma periódica los datos por MQTT a la base de datos. Tras esto los nodos se “duermen” (entran en Deep Sleep) para ahorrar energía.

### 2.1.2 Base de datos

La base de datos creada en el proyecto de automatización de quesos ya existe, por lo tanto

En la base de datos se incluyen 2 tablas:

Una contiene la información de cada nodo (nombre, donde esta, ultimo cambio de batería, etc.).

La segunda contiene los datos registrados en cada nodo en función del tiempo.

### 2.1.3 SCADA

Página en la que se puede ver el registro de la temperatura y humedad de las salas.

<https://www.futurerobotics.es/nt.php>

## 2.2 Requisitos que cumplir

Los siguientes puntos resumen los requisitos básicos del proyecto:

- El trabajo se realizará en grupos de 4 alumnos
- El hardware de referencia a emplear será un microcontrolador Espressif ESP32-S3 sobre el kit que se quiera (ESP32-S3-DEVKITC-1-N8R2 usado en clase o cualquier otro como M5STACK, LilyGo, Wagner, etc.
- La parte de desarrollo de programación realizada por el grupo será obligatoria
- realizarla en C o C++ sobre ESP-IDF y empleando FreeRTOS como RTOS ( a excepción de trabajos específicos). Se podrán incorporar/aprovechar código C/C++ de Arduino a través del componente xxx.
- Está prohibido usar MicroPython ... en este trabajo (pero es algo a tener en cuenta en el futuro).
- El equipo ha de aportar el hardware adicional, quedando después de su propiedad.
- Se dedicará esfuerzo especialmente en el aspecto que se seleccione de la tabla de items.
- Se valorará que el proyecto incorpore alguna tecnología de conectividad (WIFI/Bluetooth u otra) .
- Se valorará que el proyecto exporte algún tipo de datos a Internet y que se use algo fácil para explotarlo.
- Se valorarán las implicaciones sociales, ecológicas y sostenibles de la propuesta.

## 3. Especificaciones

### 3.1 Componentes Hardware

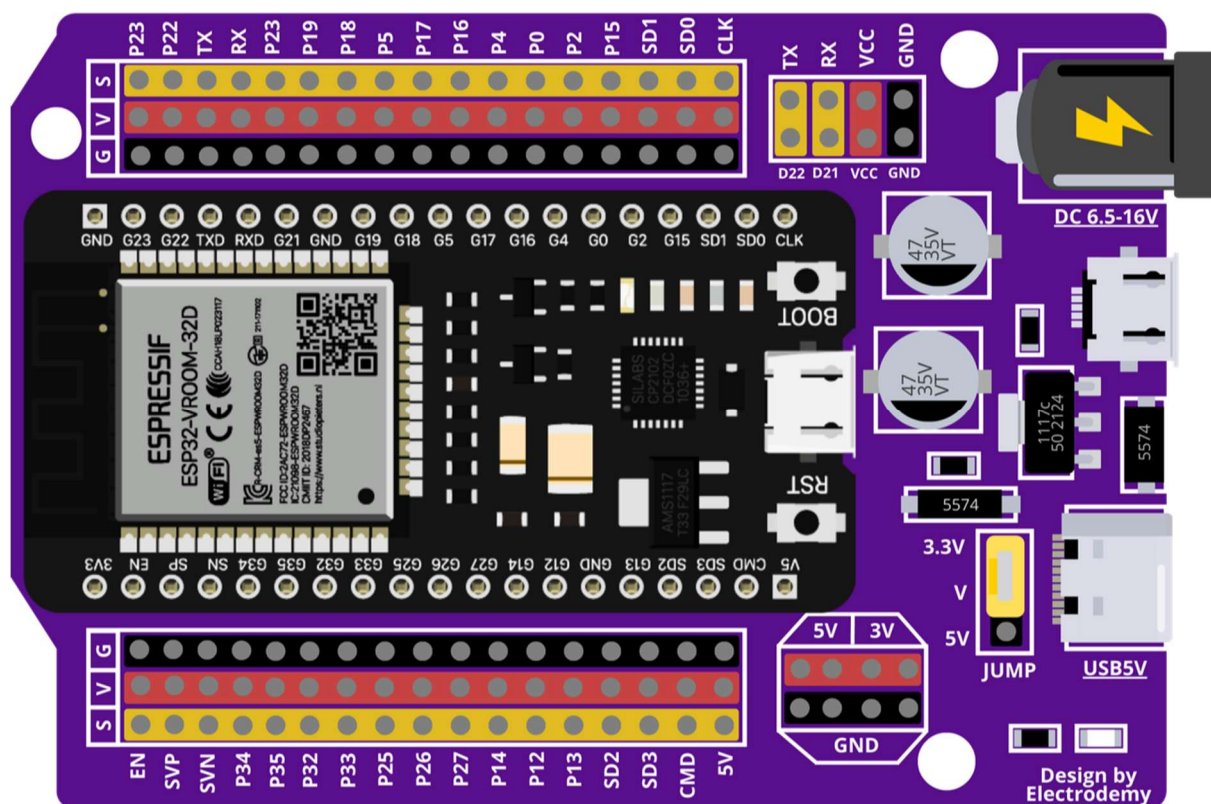
#### 3.1.1 Controlador

Para la implementación del nodo sensor se utiliza el microcontrolador esp32.

Para el desarrollo de prototipo/proyecto se usa la esp32-wroom dev-board: esta cuenta con:

- Puente uart: permite cargar fácilmente el programa en al ESP.
- Botones reset y boot.
- Protección contra sobretensión y sobre corriente
- Regulador de voltaje

Para el montaje de esta placa en el prototipo contamos con una placa de expansión la cual nos permite clavar la dev-board. Adicionalmente sobre esta placa también se montan los leds ya que la placa de expansión tiene pines adicionales que facilita la conectividad.





### 3.1.2 LEDs de estado

Dos LEDs uno rojo y uno blanco montados en una pieza de plástico engastada entre la dev-board y la placa de expansión estos están conectados a los pines de la placa de expansión y a GND mediante unas resistencias de  $220\Omega$  evitando la sobre corrientes.

### 3.1.2 Sensor temperatura

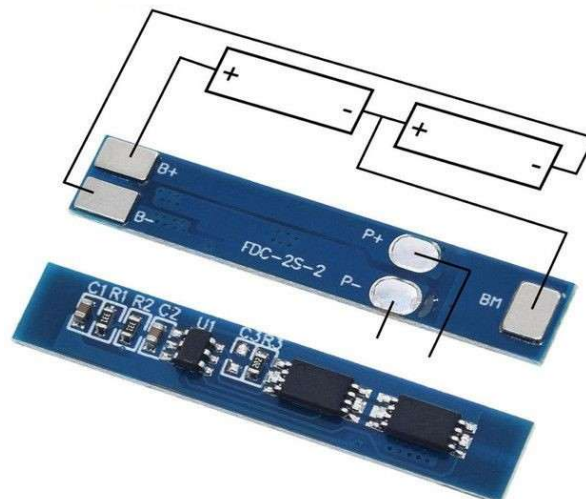
La medición de temperatura y humedad se realiza mediante el DHT 11 el módulo de este sensor viene con una resistencia de  $5k\Omega$  entre el Vcc y el pin de señal el cual es un puerto serie.

Conexiones	
DHT 11	Board
Vcc	Vcc
Serial	Pin_serial_sensor
GND	T1(source)

### 3.1.4 Alimentación

Para la alimentación del nodo se han elegido dos celdas 18650 li-ion 3.7V.

Están montadas en serie proporcionando una tensión de 8.4 (6V descargadas) esta tensión cubre las necesidades del módulo. Para controlar estas dos celdas se ha instalado un BMS (Battery Manager System).



### 3.1.5Carcasa

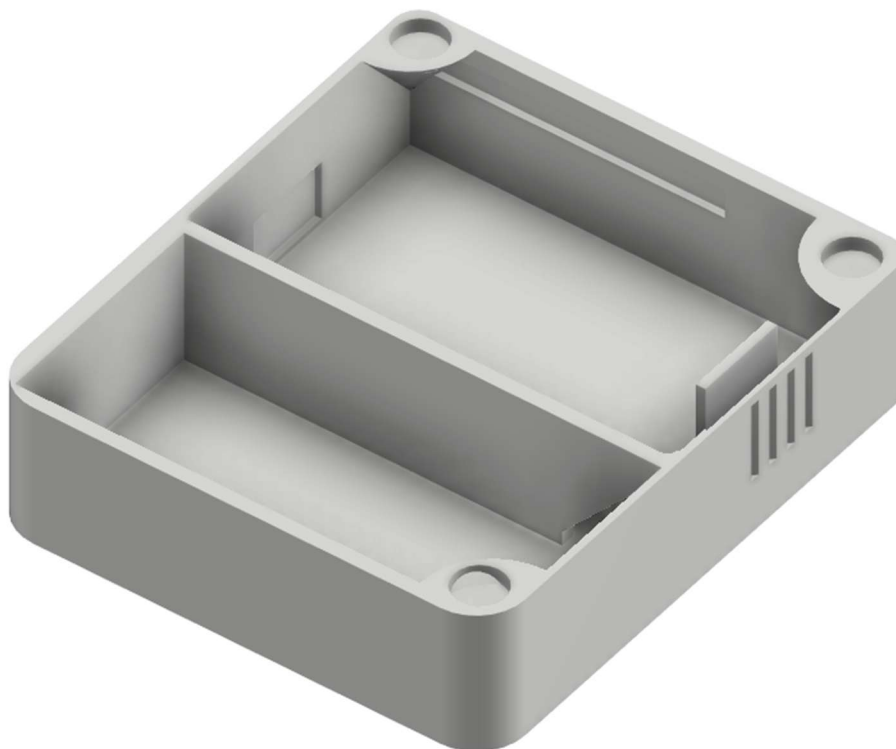
Carcasa desarrollada a medida para el prototipo (esto hace que sea más grande de lo necesario).

Esta está hecha de Polietileno de alta densidad (está impresa en 3D PLA) blanco, cuenta con puntos de anclaje con imanes para colocar la tapa.

Se divide en 2 cámaras:

En la cámara inferior está colocado le porta celdas lo cual facilita so desmontaje y sustitución, este va soldado a BMS B+, B- y BM.

En la cámara superior está la placa de expansión, en la pared izquierda hay una pequeña ventana donde el grosor de la carcasa es muy inferior, permitiendo a los LEDs ser visibles desde fuera, y en el lado derecho hay una “reja”, que permite que el sensor de temperatura y humedad (montado dentro) tome las medidas.

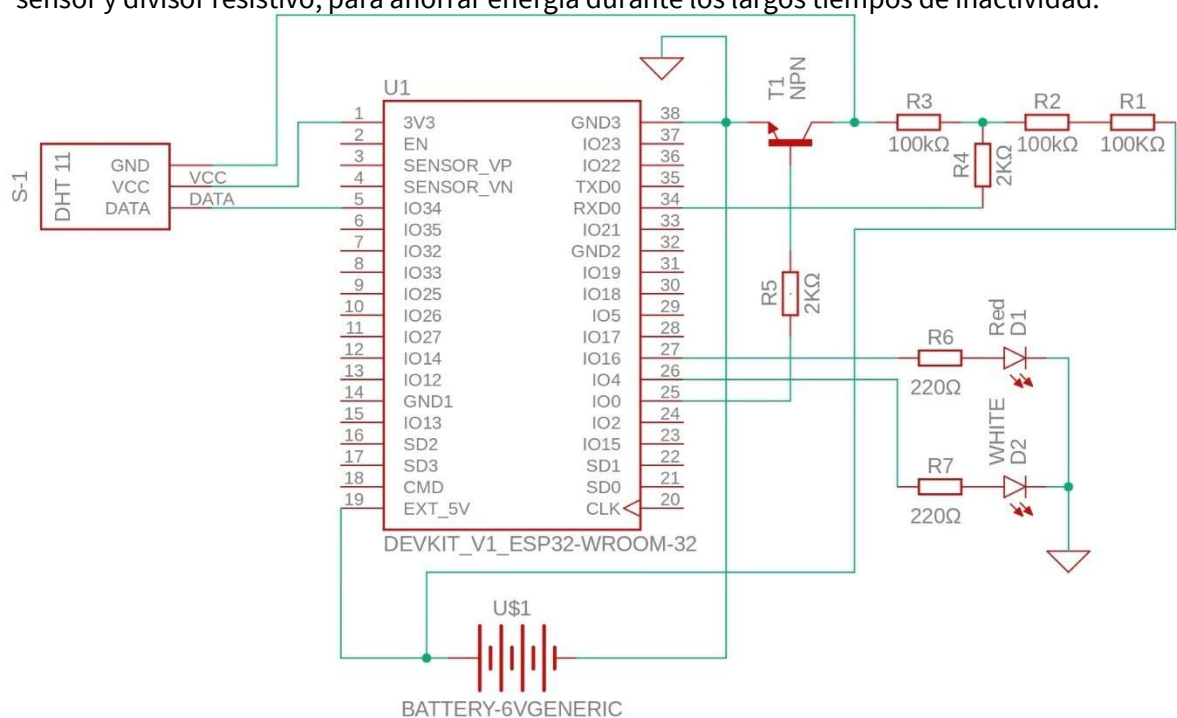


## 3.1.6 Interruptor sensor + divisor resistivo

El circuito cuenta con un divisor resistivo de  $200\text{k}\Omega + 100\text{k}\Omega$  para poder tomar medidas de la carga de la batería, el punto intermedio está conectado al Pin\_bateria mediante una resistencia de  $2\text{k}\Omega$

Conexiones	
T1 P2N2222A	Componente
Colector	DHT11(GND)  Divisor (GND)
Base	Pin_enable_divisorR  (resistencia de $2\text{k}\Omega$ )
Emisor	GND

Este divisor resistivo y el sensor de temperatura y humedad tiene su GND unido al colector de T1. T1 es un transistor NPN (modelo P2N2222A), el cual permite la conexión y desconexión del sensor y divisor resistivo, para ahorrar energía durante los largos tiempos de inactividad.

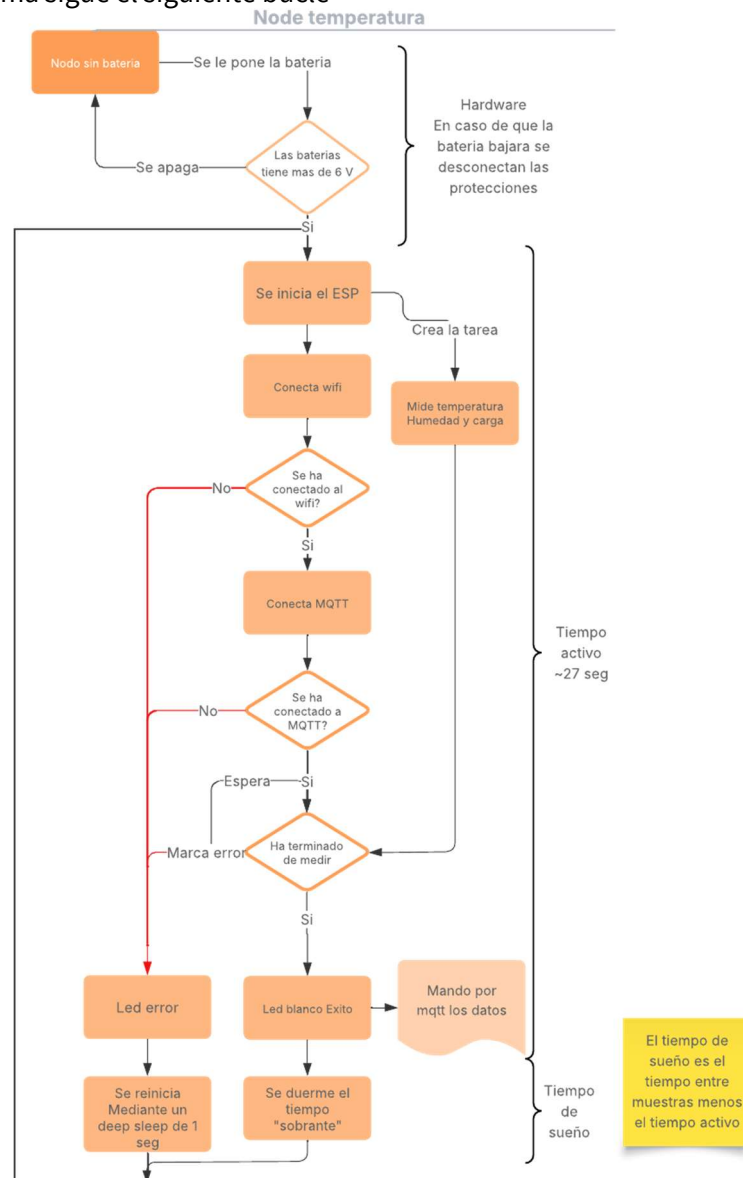


10

## 3.2 Software

Para el control del nodo de temperatura el programa implementado utiliza FreeRTOS un micro kernel el cual permite la utilización de concurrencias y facilita la creación y gestión de tareas.

El programa sigue el siguiente bucle



A continuación, se hace un pequeño resumen de cada una de las funcionalidades implementadas, Para una información más detallada consulte el código.

### 3.2.1 Constantes y pines

Se declaran como constantes los pines utilizados para evitar confusión:

```
#define Pin_Led_rojo 27           // Pin salida para led Rojo
#define Pin_Led_blanco 26        // Pin salida para led Blanco
#define Pin_senial_sensor 5      // Pin entrada para señal del sensor
#define Pin_bateria 34           // Pin entrada para señal de la batería
#define Pin_enable_divisorR 25   // Pin salida para activar el divisor de tensión de la batería
```

### 3.2.2 Pin\_init

Esta función se encarga de configurar los registros de los pines para configurar:

- Los pines de entrada digital Pin\_senial\_sensor sin resistencia de Pull-up/down y sin interrupciones.
- Los pines de salida Pin\_Led\_blanco Pin\_Led\_rojo y Pin\_enable\_divisorR.
- El pin\_batería se configura habilitando con el canal adc\_6.

### 3.2.3 WiFi

Utilizando la librería de ESP\_WiFi.h se ha implementada un cliente de internet el cual utilizan la estructura de datos propia wifi\_config\_t esta configura el cliente wifi e implementa una gestión de eventos estos que se encargan de gestionar cuando el wifi se desconecta por error o se ha conectado y tiene que pedir la IP, adicionalmente el evento de conexión activa una variable que indica al main que el wifi está conectado correctamente.

También se implementa una función de desconexión del wifi

### 3.2.3.1 *wifi\_config\_t*

Esto cuneta con dos datos char\* para indicar el ssid y la pasword de la red

```
typedef struct{  
    char *ssid; // Nombre de la red Wi-Fi  
    char *password; // Contraseña de la red Wi-Fi  
} Wifi_config_t;
```

### 3.2.4 MQTT

Se define una estructura propia MQTT\_config\_t, que almacena los parámetros necesarios para conectarse al broker MQTT. La función inicializa el cliente mqtt mediante la librería mqtt\_client.h adicionalmente se implementa la gestión de eventos keepalive y la actualización de la variable global mqtt\_ready .

También se implementa una función de publicar mensaje y desconectar MQTT

### 3.2.4.1 *MQTT\_config\_t*

```
typedef struct  
{ char *uri; // Puerto del broker MQTT  
  char *client_id; // ID del cliente MQTT  
  char *username; // Nombre de usuario para el broker MQTT  
  char *password; // Contraseña para el broker MQTT  
  char *topic; // Tema al que se suscribe el cliente MQTT  
} mqtt_config_t;
```

### 3.2.5 Sensor temperatura y humedad

La medición de la temperatura y la humedad se realiza por medio del sensor de temperatura DHT 11, este sensor necesita una alimentación de 3.3Vcc y cuenta con un pin serie por el cual es se comunica con el controlador, este pin está conectado a 3.3Vcc mediante una resistencia de 5kΩ necesaria para el funcionamiento correcto del sensor.

El sistema para recibir información de este sensor consta de hacer una “serie” de señales con el pin para que el sensor comience a emitir información por eso para esta función se utiliza la librería dht11.h de **Michele Biondi** esta librería permite tomar las medidas de manera rápida y eficiente.

La librería esta adjuntada como un archivo adicional en el proyecto.

Debido a que el sensor de temperatura y humedad tiene un consumo pasivo este solo está activo cuando el Pin\_enable\_divisorR está activo por lo tanto si intentamos tomar las medidas sin iniciar el sensor esta función da error.

### 3.3.6 Medidor tensión de la batería

Para poder hacer un seguimiento de la batería restante en cada nodo

La salida del divisor resistivo está conectada a el Pin\_bateria el cual también en el ADc\_chanel6 para poder hacer la conversión de tensión a porcentaje de batería:

El divisor resistivo es 200KΩ + 100KΩ por lo tanto la tensión medida teórica son

Batería completamente cargada:  $8,4V * \frac{1}{3} \simeq 2,8V$

Batería completamente agotada:  $6,0V * \frac{1}{3} \simeq 2,0V$

El divisor resistivo solo da medidas correctas si el Pin\_enable\_divisorR está activo ya que si se dejara siempre actico tendrías un consumo mayor del nodo de temperatura.

### 3.2.7 Deep Sleep

Esta función permite al ESP mediante el uso de FreeRTOS la capacidad de entrar en modo de bajo consumo, su funcionamiento se base en dos partes, la primera como argumento toma el tiempo que va a “dormir” el ESP esto configura el RTC(Real Time Clock) para de esta manera “despertarse” en el tiempo definido la segunda es la función que activa el Deep Sleep .

El Deep Sleep deshabilita o apaga todo excepto el estado de algunos pines y los RTC por lo tanto cuando el ESP se “despierta” podemos pensar en este como un reinicio completo del ESP en el cual hemos perdido todas las variables y RAM debido a esto debemos de reconfigurar los pines en cada ciclo.



## Bibliografia

- Espressif Systems. (s.f.). *ESP-IDF Programming Guide*.  
<https://docs.espressif.com/projects/esp-idf/en/latest/>
- Barr, M. (2001). *Embedded C Coding Standard*. Barr Group.  
<https://barrgroup.com/embedded-c-coding-standard>
- Adafruit Industries. (s.f.). *DHT sensor library*. GitHub.  
<https://github.com/adafruit/DHT-sensor-library>
- Barry, R. (s.f.). *FreeRTOS Kernel*. FreeRTOS.org.  
<https://www.freertos.org/>