29-1-2022

**Proyecto IoT**

**GRUPO 1 Informática Industrial**

Cristina Jurado Herrero

Julián Martín Caro

# ÍNDICE

[ÍNDICE 1](#_Toc93594290)

[INTRODUCCIÓN Y OBJETIVOS 2](#_Toc93594291)

[DISEÑO HW Y ESQUEMAS DE CONEXIONADO 3](#_Toc93594292)

[Pines usados para cada sensor/dispositivo 3](#_Toc93594293)

[DISEÑO SW 4](#_Toc93594294)

[Diagrama de bloques/flujo del programa, Descripción de funcionamiento 4](#_Toc93594295)

[Técnicas que se han utilizado para hacer robusto el sistema. Por ejemplo, cómo se maneja el 6](#_Toc93594296)

[Estudio del tiempo de ejecución de distintas fases de funcionamiento y el respectivo 7](#_Toc93594297)

[consumo de energía. Posibles mejoras u optimizaciones introducidas y análisis de su 7](#_Toc93594298)

[impacto. 7](#_Toc93594299)

[Librerías utilizadas 8](#_Toc93594300)

[RESULTADOS Y CONCLUSIONES 9](#_Toc93594301)

[MANUAL DE USUARIO 10](#_Toc93594302)

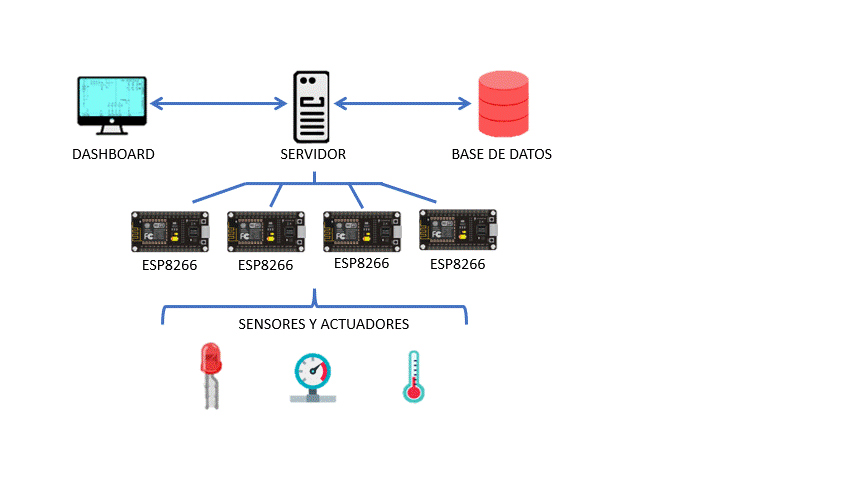
[LISTA Y DESCRIPCIÓN DE LOS FICHEROS ENTREGADOS 12](#_Toc93594303)

# INTRODUCCIÓN Y OBJETIVOS

Para la realización de este proyecto se ha partido de los requisitos mínimos proporcionados por el guion, y se ha añadido la funcionalidad de control por voz mediante una aplicación de móvil y Alexa.

El objetivo es crear un sistema SCADA accesible y comprensible para monitorizar y actualizar la información obtenida de sensores y actuadores, tales como leds y sensores de temperatura y humedad. Se ha implementado también que sea posible modificar a través del dashboard la frecuencia de muestreo de los datos, y que se almacenen en una base de datos de la que se puedan extraer y realizar operaciones.

Por último, se ha incluido que Telegram y Alexa, mediante comandos de voz, puedan ser preguntada por el estado de los sensores y que actúe sobre los actuadores.



# DISEÑO HW Y ESQUEMAS DE CONEXIONADO

### Pines usados para cada sensor/dispositivo

# DISEÑO SW

### Diagrama de bloques/flujo del programa, Descripción de funcionamiento

<https://www.diagrameditor.com/>

En Node-RED, hemos dividido el diagrama en 3 partes:

**TOPICS**

Se trata del flujo que implementa las recepciones y envíos de topics mediante MQTT en formato JSON desde el programa de Arduino. En este grupo se encuentran tanto señales de control, de actuación y de sensado.

**MONGODB**

En este flujo se implementa el almacenamiento en la base de datos de nuestro grupo, las señales medidas. Por otra parte admite peticiones de muestra de los datos obtenidos en un periodo de tiempo.

**TELEGRAM**

Mediante este flujo, se puede pedir a un Bot de Telegram las medidas actuales de los sensores, que notifique en el caso de que superen un máximo o mínimo, y también

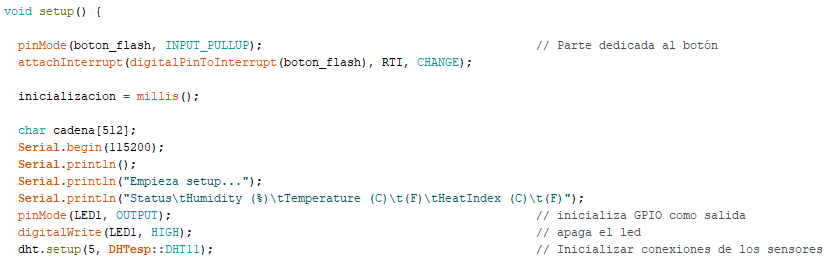
A continuación se describe el funcionamiento de el programa implementado en Arduino.

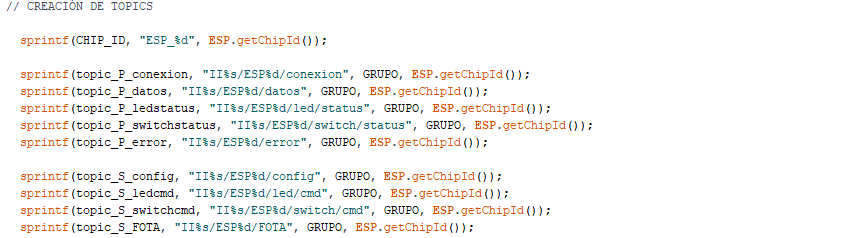
Al comienzo de dicho programa se encuentra la inicialización de una serie de variables globales. Algunas de ellas tienen valores inicializados, normalmente se trata de las que estan atadas a variables de control que deberían tener un valor por defecto si el programa de NodeRed no dice nada al respecto.

La única parte que el usuario debería tener que tocar de este programa se encuentra aquí, ya que aunque el ID del chip de la placa es obtenido automáticamente, los datos de wifi y del servidor MQTT están escritos en el propio programa, así como el número de grupo, conque para cambiar de grupo, de red Wifi, o de servidor MQTT habría que alterar sus variables en el código.

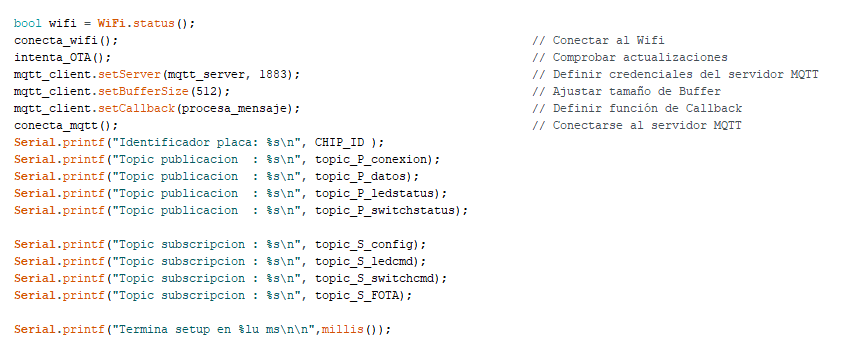
Hay varias funciones implementadas en el programa, pero para seguir un orden algo lógico se va a describir cada una a medida que son utilizadas en la operación normal.

Empezando entonces con el setup, se tiene que los sensores de las placas son inicializados, al igual que los LEDs que se van a controlar.

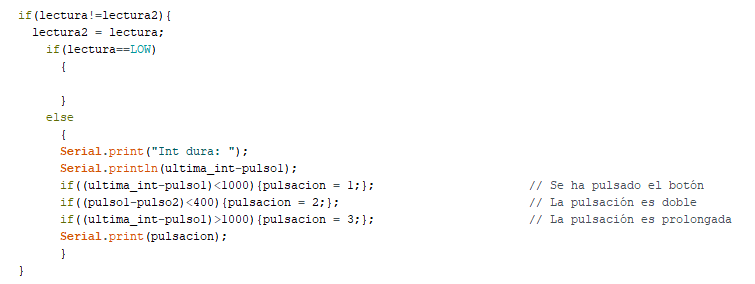
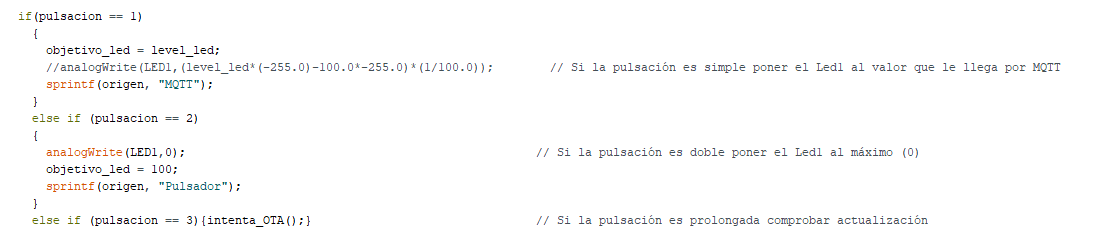
A continuación se construyen los strings que se van a utilizar para declarar los topics de transmisión de datos, siendo estos flexibles tanto al número de grupo como al ID de chip, sin necesidad de reescribir esta parte de la función si cambian alguno de estos dos factores.



Tras terminar de construir los strings, se utilizan las funciones usadas en otras tareas, **contecta\_wifi(), intenta\_OTA()** y **conecta\_mqtt()** para conectar la placa a la red Wifi, comprobar si hay actualizaciones al código disponibles, y conectarse al servidor MQTT, respectivamente. Por último, se envía un mensaje por el topic de conexión para indicar al programa de NodeRed que la placa se ha conectado.

Tras terminar de ejecutarse la función de setup, se pasa a la función de loop.

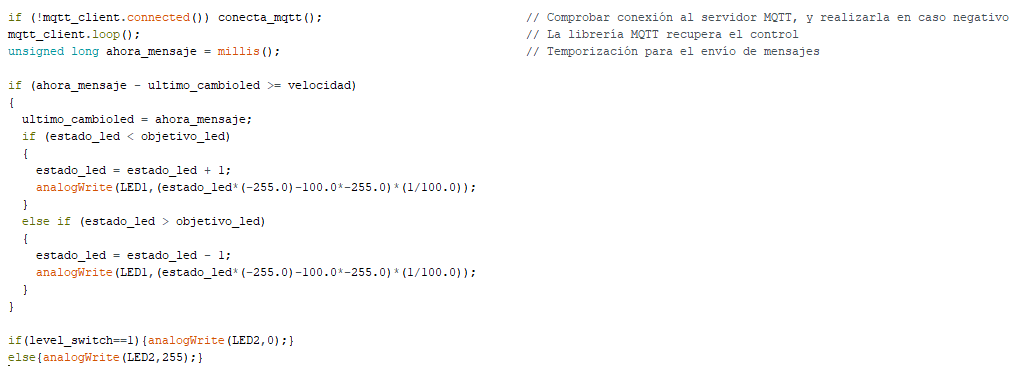
Lo primero que hace el loop es comprobar si se ha pulsado el botón de la placa, en cuyo caso asigna a **pulsacion** un valor distinto dependiendo de si la pulsación fue simple, doble o prolongada.

A continuación, en función del valor de **pulsación**, se pone el LED al valor indicado por NodeRed, al valor máximo, o se intenta actualizar la placa.

Lo siguiente es comprobar que el servidor MQTT siga conectado, y en caso negativo se intenta la reconexión.

Después, unas líneas de código se encargan de la variación del LED, ya que el LED tiene que tender poco a poco hacia la luminosidad que le indica NodeRed, en función de una variable de **velocidad** que también indica NodeRed.

Justo a continuación, se efectua el control del LED de switch, que sólo tiene que estar o al máximo o al mínimo.



Por último, cada 30 segundos se toma una medida de todos los sensores y variables a comunicar, y se almacenan en Json para a continuación ser enviados por los topics pertinentes. Un detalle a destacar es que el programa tiene algo de robustez implementada, ya que avisa por un topic de error al programa de NodeRed si los valores de los sensores caen fuera de unos ciertos rangos, lo que podría indicar que hay algún problema.

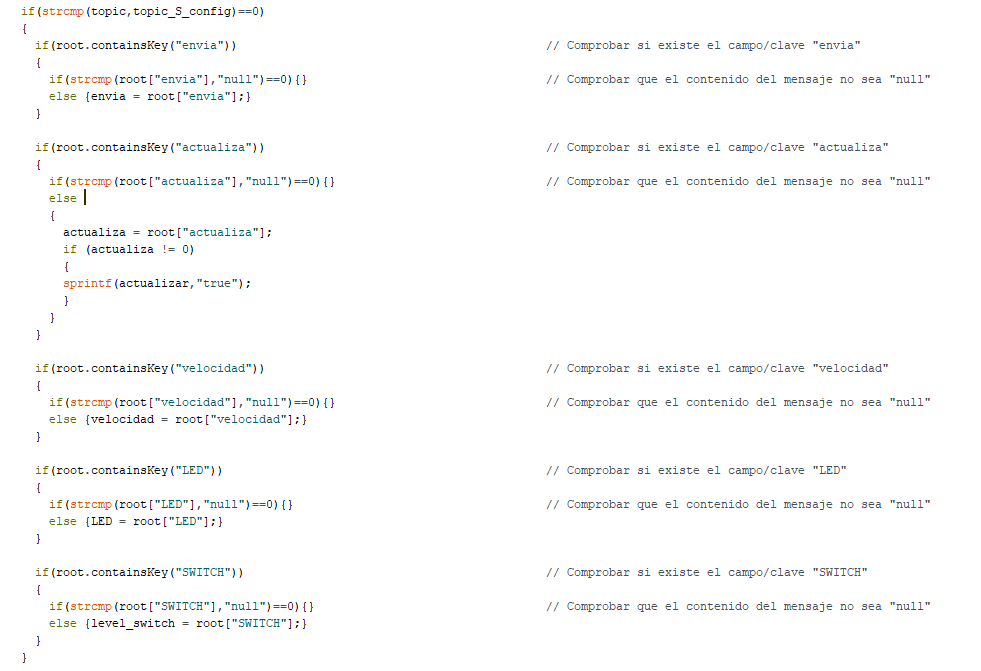
Al final del código se pueden ver unas últimas líneas que comprueban si hay una actualización cada **actualiza** segundos, suponiendo que el valor de la variable sea distinto de 0, ya que un valor 0 significaría que se le ha mandado no actualizarse automáticamente.



Sólo queda por mencionar a la función de Callback asignada a la recepción de mensajes MQTT.

La única alteración que ha sufrido con respecto a las tareas anteriores, es que ahora compara a todos los topics a los que el programa está suscrito con el topic de llegada del mensaje, y realiza una serie de “checks” más antes de almacenar la información en el mensaje.

Despues de comprobar que el topic de llegada es uno de los correctos, la función comprueba si el mensaje Json tiene las claves que debería, y luego almacena los datos en ellas suponiendo que dichos datos no sean la plabra **null**, que indica que no se quiere realizar ningún cambio en ese campo.



Técnicas que se han utilizado para hacer robusto el sistema. Por ejemplo, cómo se maneja el fallo de un sensor, cómo se asegura el paso de mensajes, etc.

En la programación de los ESP8266 mediante el código en Arduino, se ha evitado los comandos tipo wait para hacer más robusta la toma de datos, ya que de esta forma el sistema es podrá recibir un comando de entrada en cualquier momento.

Se ha implementado un cortafuegos para los topics, mediante un diagrama if-else de forma que solo se deserialicen y lean los topics que encajan exactamente con el requerido, de lo contrario se podrá leer el mensaje “Error: topic desconocido”.

### Estudio del tiempo de ejecución de distintas fases de funcionamiento y el respectivo

### consumo de energía. Posibles mejoras u optimizaciones introducidas y análisis de su

### impacto.

### Librerías utilizadas

NODE-RED:

Imagen que contiene Texto

Descripción generada automáticamente

ARDUINO:

#include <ESP8266WiFi.h>

#include <ESP8266httpUpdate.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

#include <ArduinoJson.hpp>

#include "DHTesp.h"

# RESULTADOS Y CONCLUSIONES

# MANUAL DE USUARIO

**DASHBOARD:**

Se ha programado un dashboard para la supervisión de los procesos, organizado en 4 tabs, a los que se puede acceder desde la pestaña situada en la barra azul superior, pinchando a la izquierda.

Los tabs son:

1. **ACTUADORES**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En él se puede configurar la señal enviada a los actuadores del sistema, están organizados en 2 grupos:

Nivel Led: Al seleccionar un nivel de luz y un ESP, el led del sistema se encenderá con la intensidad requerida. Es necesario primero seleccionar el nivel y a continuación el ESP elegido.

Switch: Mediante un switch es posible encender o apagar el led seleccionado. Es necesario primero seleccionar la posición del switch y a continuación el ESP elegido.

1. **CONFIG**

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Con el tab config se puede cambiar la configuración del sistema:

En el campo **envía** se puede escoger cada cuanto se envían mensajes con los datos actualizados de los ESPs.

El campo **actualiza** configura cada cuanto se comprueba la actualización.

**Velocidad** cambia la velocidad de salida del PWM en milisegundos (+-1%)

**LED** configura si los leds toman o no la salida.

**Switch** configura lo mismo que LED

1. **ACTUALIZACIÓN**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Este tab está pensado para visualizar de forma rápida la última actualización vigente.

El él podemos ver la fecha de la última actualización, el ESP que envió esa información, y los datos de temperatura, humedad y LED.

1. **HISTÓRICO\_DE\_DATOS**

En el tab config podemos ver los datos almacenados por la base de datos

**TELEGRAM:**

# LISTA Y DESCRIPCIÓN DE LOS FICHEROS ENTREGADOS

o Proyectos Arduino (\*.ino)

o Flujos Node-RED (\*.json)

ETC