UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

PROYECTO IOT

Informática Industrial

Antonio Jesús Pérez Bazuelo
Manuel Valle Delgado
Mathias Lofeudo Clinckspoor
Pablo Roldán Pérez
Pablo Vera Soto

GIERM
Fecha: 26/01/2021

Contenido

1 Introducción

Se ha desarrollado un software que aplica "internet de las cosas" (IOT) como trabajo final de la asignatura de Informática Industrial. Se implementan diferentes tecnologías estudiadas durante el curso. Entre ellas cabe destacar el uso de comunicaciones MQTT [1], el diseño de flujos Node-RED [2] y la programación en Arduino [3].

1.1. Especificaciones del proyecto

Este proyecto consiste en el desarrollo de un software capaz de realizar la comunicación entre una placa ESP8266 [4] y un servidor de Node-RED mediante MQTT. Se proponen los siguientes mínimos:

- Recibir y mostrar en el Dashboard los datos de los sensores (temperatura, humedad, etc.).
- Controlar las salidas de los LEDs integrados en la ESP tanto desde Dashboard como desde Telegram [5] como desde el botón Flash de la misma placa.
- Comunicar todos los cambios de los LEDs mediante mensajes MQTT.
- Regular la configuración de los LEDs: la velocidad de encendido del GPIO2 y la lógica con la que se manejan.
- Comprobar si hay actualizaciones FOTA (Firmware Over The Air) [6] de diversas maneras: al arrancar el dispositivo, con una pulsación larga del GPIO0, con un mensaje desde Node-RED, y con una periodicidad regulable desde Node-RED.
- Ser capaz de manejar varias ESP simultáneamente.
- Mostrar gráficamente en el Dashboard los datos recibidos desde la ESP, teniendo la posibilidad de descargarlos en un fichero .CSV.
- Almacenar los datos en una base de datos de MongoDB [7] y ser capaz de recuperarlos.
- Poder ver el historial o log de los cambios de estado de las interacciones relevantes.
- Poder establecer unos valores de temperatura y humedad máximos y mínimos. Al salir de estos, se activaría una alarma.
- Poder conversar con un bot de Telegram para interactuar con la ESP, ya sea para recibir información, como para actuar sobre ella.

1.2. Ampliaciones

Una vez obtenido el prototipo inicial, en el que el sistema controle diferentes dispositivos básicos, extendemos la solución obtenida para controlar un robot de tipo PIERO [8]. En este caso el robot ya está controlado por un módulo Arduino [9], por lo que estableceremos una conexión mediante el puerto serie entre el mismo y la ESP8266 que actúa como enlace con el servidor. De esta forma, trataremos al robot PIERO como a cualquier otro sensor u actuador del que se lean datos, o al que se le envíen comandos de actuación. Los datos de los sensores del robot podrán ser recogidos y almacenados en nuestra base de datos, y a su vez podremos enviar órdenes de actuación al robot. La monitorización de los sensores del robot y el control del mismo se podrán hacer, tanto desde el interfaz gráfico desarrollado (Dashboard), como desde el interfaz de Telegram. La programación del comportamiento del robot y su comunicación con la ESP8266 se ha hecho con Matlab, usando de Simulink [10].

2 Desarrollo del proyecto

El trabajo se ha repartido entre los miembros del grupo, generalmente se ha asignado tareas concretas a cada uno, aunque también se han realizado tareas entre varios miembros. La organización se ha llevado a cabo en las sesiones de clase, utilizadas como momento de reunión

para comunicar avances, dudas, propuestas y asignación de tareas. Se ha documentado la evolución del proyecto mediante una nota de texto. En ella se organizaban tanto las tareas pendientes como las ya realizadas.

Se ha utilizado la plataforma GitHub como repositorio de código, donde se ha depositado el código de la placa, el de Node-RED y el de Arduino.

Por otro lado, para los documentos de texto se ha utilizado una carpeta compartida entre todos los miembros en OneDrive. Se ha hecho así por la facilidad que se aporta para que diferentes miembros puedan trabajar en los documentos de forma simultánea.

3 Diseño hardware

El desarrollo del trabajo implica el uso de una placa ESP8266 (referida en adelante como ESP), esta se encarga de recoger la información del sensor DHT11 [11] y de iluminar sus leds incorporados para verificar el comportamiento del código propuesto. Además, la ESP es la responsable de realizar la comunicación WiFi, característica fundamental del proyecto.

El esquema de conexionado de la ESP se muestra en la Ilustración 1.

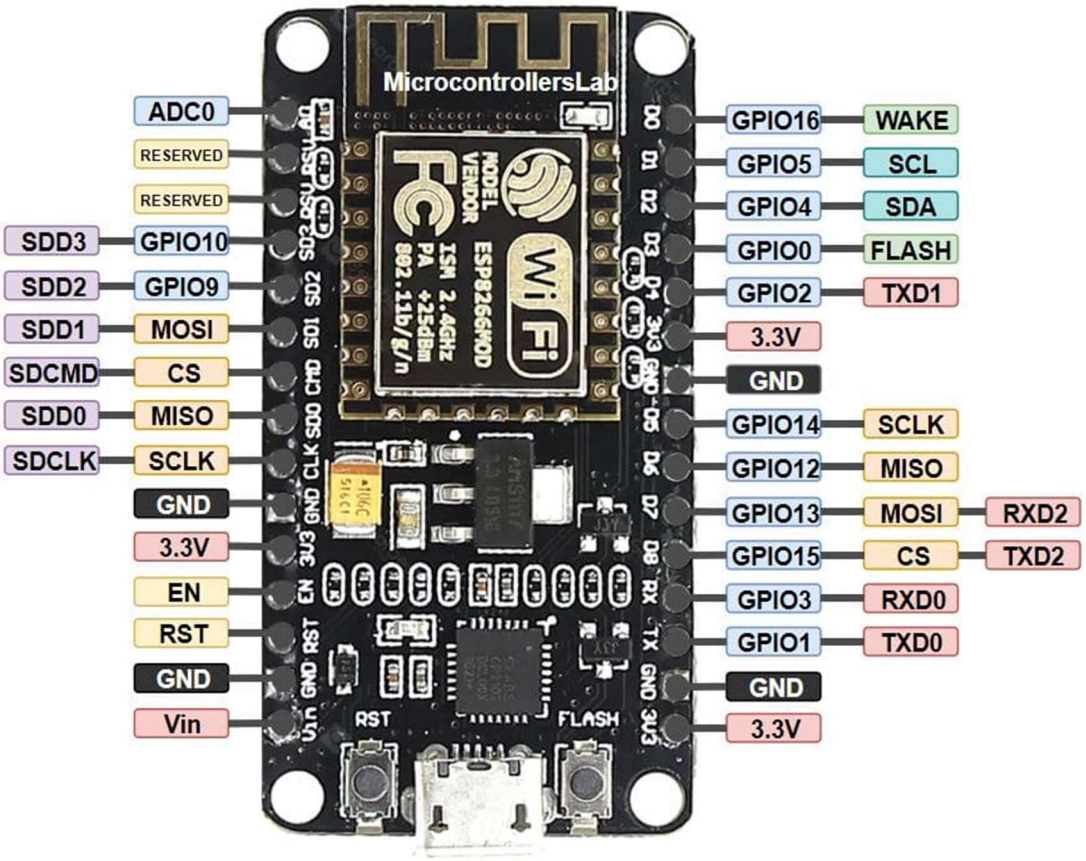

Ilustración 1 Placa ESP

En cuanto a la ampliación del proyecto, se ha conectado la ESP a un robot PIERO, desarrollado en la asignatura Laboratorio de Robótica. Se realiza la comunicación entre la ESP y el Arduino Mega 2560 que controla el PIERO mediante el puerto serie de ambas placas. Desde el punto de vista del presente proyecto, solo importa el cableado que permite la comunicación serie. aun así, por interés y la posibilidad de reproducir este proyecto, se observa en la Ilustración 2 el esquema de conexión de los elementos que constituyen el PIERO.

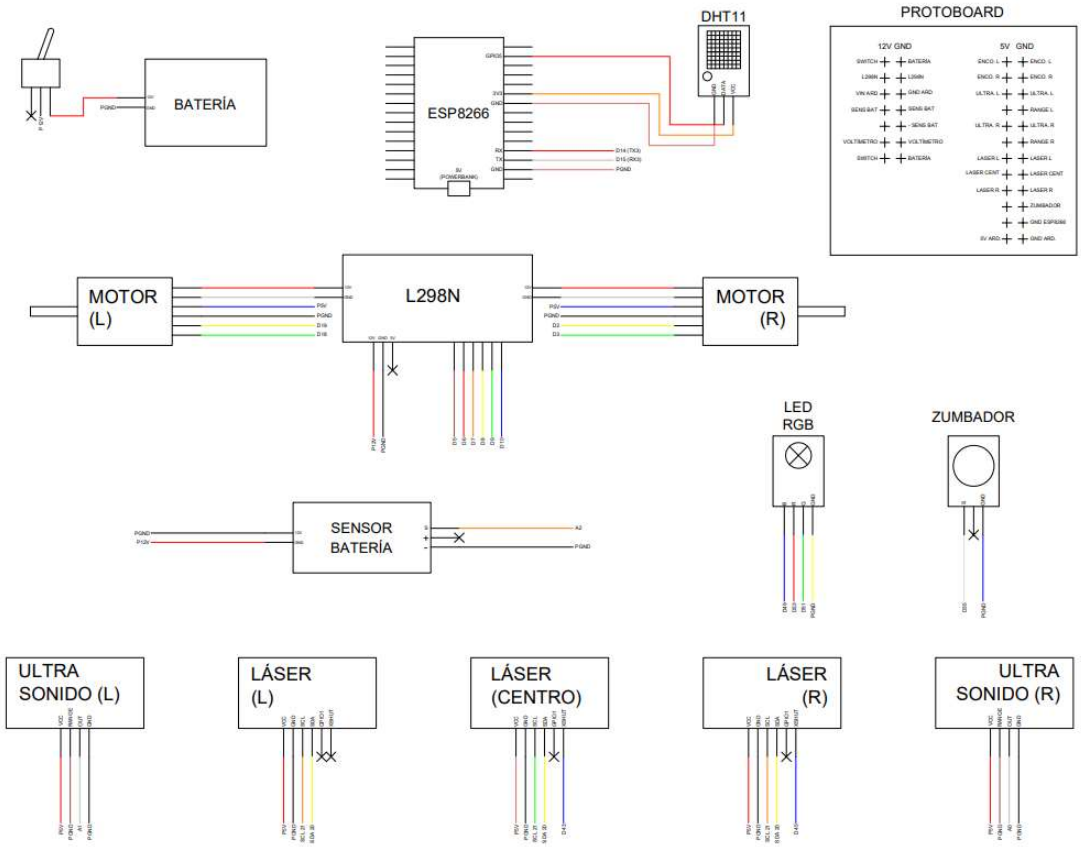

Ilustración 2Esquema de conexionado

4 Diseño software

El software implementado relaciona entre sí diferentes tecnologías que se ejecutan de forma simultánea para lograr el funcionamiento del proyecto. En primer lugar, se parte de un flujo (programa) Node-RED, que se ejecuta en un servidor localizado en una máquina virtual de nuestro ordenador. Este programa se comunica con tres servicios distintos: Telegram (usado como interfaz de mensajes de texto), un servidor MongoDB (donde almacenaremos datos) y un servidor MQTT (usado como método de comunicación). Por su parte, el Servidor MQTT se comunica con la ESP y sirve como canal de información bidireccional entre ésta y el flujo ejecutado en Node-RED. Finalmente, la ESP se comunica con el Arduino Mega 2560 que controla el robot PIERO. La comunicación entre estos dispositivos también es bidireccional. Se muestra el diseño descrito en la Ilustración 3.

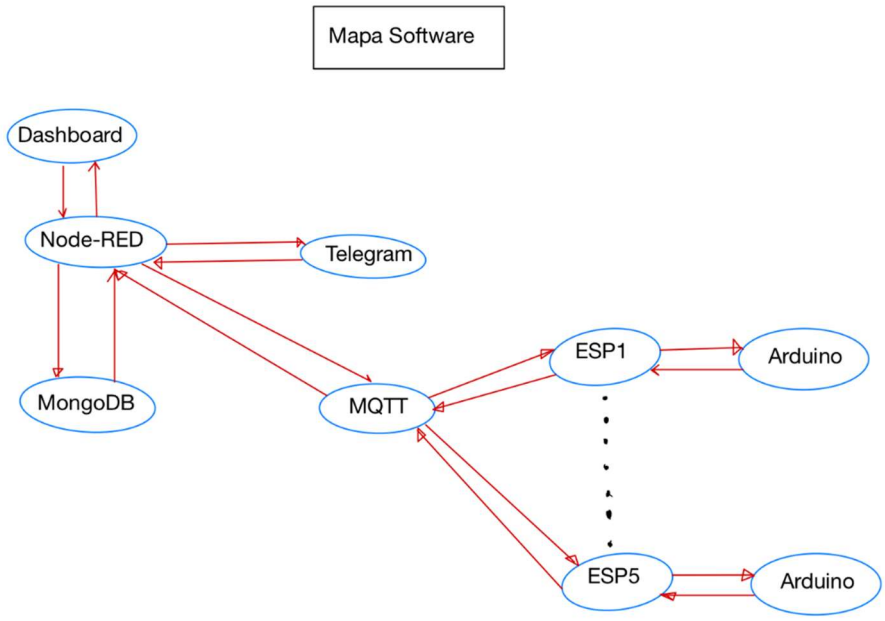

Ilustración 3 Mapa Software

5 Código IDE

El código que hace funcionar el ESP se ha dividido en diferentes programas que funcionan de manera simultánea. Se utiliza un programa "principal" y otros módulos que pueden conectarse tanto con éste como entre ellos. Cualquier programa puede usar tanto las funciones como las variables globales que contengan los otros. Para programar de esta manera, se necesita que cada programa tenga una "cabecera", que controla como lo ve el resto del programa. De esta manera cada programa (cuya extensión es .cpp), tendrá asociado otro con el mismo nombre, pero con la extensión .h que es la máscara.

A continuación, se explican las funcionalidades de los diferentes programas, haciendo hincapié en características importantes que éstos puedan contener. En primer lugar, se explica en detalle el contenido del programa Global_G1. Este programa es el que se ha calificado previamente como "principal". El resto de los programas se explican luego de forma más breve.

5.1 Global_G1

Este programa se encarga de administrar las diferentes partes del proyecto y servir de nexo entre ellas, logrando así una mejor organización. Se divide en tres partes, una sección declarativa, otra para inicializar procesos (setup) y una iterativa (loop) que contiene los procesos que se repiten mientras dure la ejecución.

Este programa se encarga de administrar el estado de los leds incorporados en la placa, de realizar actualizaciones de firmware cuando corresponda y de enviar datos de la placa y sensores al servidor MQTT.

5.1.1 Parte declarativa:

En primer lugar, se encuentran las librerías necesarias para la ejecución de este programa. A continuación, se declaran las variables globales necesarias y finalmente, se añade el resto de los programas que van a usarse, esto se hace tratándolos como librerías (funciones ".h").

5.1.2 Setup:

Esta sección se encarga de lo siguiente:

- Inicializa el ESP, declara la interrupción, el modo de funcionamiento de los pines a utilizar y realiza la conexión WiFi.
- Inicializa la conexión con el servidor MQTT y genera los topics de acuerdo a la configuración de config.
- Inicializa el sensor DHT11, sensor de temperatura y humedad.
- Avisa si se ha activado el modo debug, para ello se pone debug a "true" en config.

5.1.3 Loop:

El código de esta sección se ejecuta de manera cíclica. Cada vez que termina de ejecutarse vuelve a empezar.
Se distinguen distintas secciones dentro del bucle:

FOTA

Las actualizaciones se van a realizar vía FOTA mediante el servidor iot.ac.uma.es. Este servidor funciona con el protocolo https.

Cada vez que se enciende una placa, y una vez que se ha conectado al servidor MQTT, se comprueba si hay actualizaciones disponibles. Si hay una actualización disponible, se envía un mensaje al servidor MQTT para dejar registro de que se ha realizado dicha actualización. Para realizar esta comprobación inicial, se recurre a dos variables booleanas: primera_FOTA y prim_F. Primera_FOTA únicamente es "true" al iniciar la placa y está a "false" el resto del tiempo. Esta variable activa tanto temporizador como la variable prim_F. Cuando se cumple el tiempo previsto, se desactiva la variable y se comprueba la actualización mediante la función FuncionActualizacion().

Posteriormente, comprueba periódicamente si hay alguna actualización disponible. Este intervalo de tiempo se guarda en minutos en la variable fotaSampRate. Si dicha variable contiene un cero, no se comprobarán periódicamente las actualizaciones.

Finalmente, se comprueba si hay actualizaciones disponibles cuando se recibe un mensaje por el topic /infind/GRUPO1/ESPX/FOTA que lo indique o si se pulsa el botón flash de la placa durante 2 segundos o más. La llegada del mensaje de actualización modifica el valor de la variable "actualiza", que se comprueba continuamente en el loop. La pulsación se controla con una interrupción.

Datos

Se envían datos periodicamente, el intervalo de tiempo de envío viene determinado por "dataSampRate". Esta variable se comprueba continuamente. Si por algún motivo esta variable tiene un valor menor que cero, se acota el tiempo de envío a un segundo. Cada vez que se tiene que realizar un evío, primero se generan los datos que se desean enviar por mqtt. Estos se organizan en una estructura creada en la función tomaDatos(), localizada en el programa Datos. Se serializa el mensaje y se envían en el topic previsto.

Led y switch

A continuación, se controla el funcionamiento del led. Para ello se comprueba si el led se ha modificado. Esto se verifica mediante la variable ready_led. Esta valdrá true solo cuando se haya detectado una pulsación del botón. Cuando esto ocurra, se registra el momento de la pulsación y se envían los datos haciendo uso de la función led_mqtt(). Se procede de igual manera con el switch, que usará la variable ready_switch y la función switch_mqtt().

Además, también se controla el cambio de la intensidad del led, esto se consigue aplicando una condición que actualiza el valor de la intensidad cada cierto tiempo. El intervalo se puede configurar desde el Dashboard, modificando así la velocidad de actualización del led hasta que llegue al valor de intensidad deseado.

Durante el envío de datos se trabaja con valores del led entre 0 y 100, mientras que para dar valores directamente a la placa se utilizan valores entre 0 y 1023 además de tener en cuenta si se está trabajando con lógica positiva o negativa.

Sensores

Por último, se llama a la función sensores_arduino() para recoger los datos de los sensores de Piero enviados desde Arduino hasta la ESP. En caso de estar recibiendo datos, se serializan sus valores y se mandan por MQTT gracias a la función sensores_mqtt().

5.2 Config

Se encarga de declarar y gestionar todas las variables que tienen que ver con la configuración del programa, la conexión WiFi, la distribución de pines en la placa, modo debug y configuración de topics.

Con respecto a los topics, se han preparado para ser flexibles. Se forma una cadena de caracteres a partir de diferentes variables de manera que cambiar un topic solo significa cambiar el valor de una variable de este programa. Se evita así tener que modificar todas las llamadas a dicho topic. En el Setup del programa global se llama a la función mqttTopics() que se encarga de construir las cadenas de caracteres completas.

5.3 Cont_vel

Aquí se encuentran las funciones control_modo() y velocidad(). Estas se encargan de convertir los comandos de modo y velocidad, recibidos mediante MQTT, al formato que se está usando para la comunicación serie. Los datos de estos comandos se envían a la placa con la siguiente estructura:

- El modo 0 (reposo) se formatea a los caracteres ASCII “XR”.
- El modo 1 (manual) se formatea a los caracteres ASCII “XM”.
- El modo 2 (automático) se formatea a los caracteres ASCII “XE”.

Los comandos de velocidad sólo se envían en caso de que se esté en el modo manual. La estructura que siguen es la siguiente:

- La orden 8 (adelante) se formatea a los caracteres ASCII “XW”.
- La orden 4 (izquierda) se formatea a los caracteres ASCII “XA”.
- La orden 6 (derecha) se formatea a los caracteres ASCII “XD”.
- La orden 2 (atrás) se formatea a los caracteres ASCII “XS”.
- La orden 0 (stop) se formatea a los caracteres ASCII “X ”.

Además, si se detecta un obstáculo, sólo se procesarán comandos de velocidad que le permitan al PIERO evitarlo.

Para concluir, los modos y órdenes indicados se procesan en el Arduino, explicado en detalle en el apartado 8.

5.4 Datos

Contiene las funciones para gestionar los datos que se manejan en el programa. Serializa_datos_JSON() se encarga de crear la estructura de datos que contiene la información que se desea enviar por el topic "datos". Esta información se asigna en la función tomaDatos().

Además de enviar los datos recogidos, en esta pestaña también se encuentran las funciones que se encargan de actualizar el estado del led y switch, las que se encargan de enviar los valores de los sensores del robot y la que se encarga de informar sobre la actualización FOTA.

5.5 Debug

Esta función se emplea para mostrar mensajes por el puerto serie de la placa ESP con el objetivo de depurar las funcionalidades del programa. La ventaja de usar esta función en lugar de un simple Serial.print() es que es posible decidir cuándo mostrar mensajes y cuando no, solo modificando una variable del aparatado de config, llamada "debug".

5.6 FOTA

Se encarga de ejecutar la actualización FOTA. Para ello contiene FuncionActualizacion() y otras funciones relativas a la impresión de mensajes de estado mediante el puerto serie.

Cuando se detecta una actualización se llama a la función actualiza_mqtt(), que envía un mensaje JSON del estilo {"ultimaFOTA": 1, "ESP_": {"placa": 1, "grupo": 1}} por MQTT a Node-RED. Ya en Node-RED, se le añade la fecha al mensaje y se guarda en la base de datos. De esta forma, queda registro de las actualizaciones de las placas.

5.7 Interrupciones

Contiene la rutina de tratamiento de interrupción, declarada previamente en el setup de Global_G1. Dicha función se encarga de procesar la pulsación del botón, identificando cuándo está en nivel bajo y cuando en alto; se obtiene así la duración del pulso. Este dato es procesado en la función pulsos. Cabe destacar que la interrupción contiene un filtro para eliminar los posibles rebotes de la pulsación. Se ha tenido en cuenta que, al ser una interrupción, debe de procesarse durante un tiempo mínimo y no ejecutar muchas funciones o variables dentro de la misma, para ejecutarse en un tiempo óptimo.

5.8 MQTT

Aquí se encuentran las funciones correspondientes a la configuración de la conexión de la ESP al servidor MQTT, configuración de los topics que van a utilizarse en todo el programa y recepción de información mediante suscripción a distintos topics.

La función callback se encarga de recibir y procesar la información que se lee en los distintos topics a los que se ha realizado una suscripción. La función reconnect() establece la comunicación con el servidor y mqttTopics() construye los topics que se configuran en el apartado config. Estos topics se generarán en función del topic raíz "infind/GRUPO**y**/ESP**x**/**topic**", en donde de forma dinámica se introduce y=número de grupo, x=número de placa y el topic correspondiente. En algunos envíos de datos se envía además el campo "id", para realizar correctamente las peticiones del cambio de estado de los actuadores realizadas por Telegram.

5.9 Pulsos

Este programa contiene dos funciones relativas a la identificación de las pulsaciones del botón Flash (corta, larga o doble) y la respuesta de la placa ante estos estímulos. Si la pulsación

es corta (la duración de la pulsación menor a 2 segundos), se encienden o apagan el LED y el switch. Si la pulsación es doble (pulsaciones menores a 2 segundos y el tiempo entre ambas pulsaciones menor a 300 milisegundos), se encienden el LED al 100% y el switch. Si la pulsación es larga (pulsación mayor a 2 segundos), se comprueba si hay una actualización FOTA.

5.10 Robot_5sens

Este programa se encarga de leer los valores de todos los sensores recibidos desde Arduino por el puerto serie. Para identificarlos se utilizan etiquetas. De esta forma se recibe la etiqueta del sensor y a continuación, su lectura. La estructura de estas etiquetas está explicada en el apartado 8.

Debido a que existen dos modelos diferentes de robot (uno equipado con 2 sensores y otro con 5 sensores) se debe de introducir previamente de qué modelo se trata para no hacer lecturas falsas en un robot de 2 sensores.

Al detectar un obstáculo a menos de 50 centímetros, se manda una señal de “stop” a Arduino, para evitar que el robot impacte con algún objeto o pared. Además de eso, se manda un mensaje por MQTT (se envía por el topic “infind/GRUPO1/ESPX/PIERO/Obstaculo”), indicando por qué sensor ha detectado el objeto y mostrarlo en el Dashboard y por el bot de Telegram (si se está controlando desde ahí).

5.11 WiFi

Para finalizar, esta función se encarga de establecer la conexión WiFi con un router. Se distinguen tres funciones:

- setup_wifi()

Se conecta al router asignado, con su nombre y contraseña (previamente descritos en “config”). Si no logra conectarse en un máximo 5 intentos, la propia ESP crea su punto de acceso llamando a la función WiFiConfig(). En caso de establecer conexión, se obtiene la dirección IP.

- WiFiConfig()

La ESP se comporta como un punto de acceso: al conectarse con la ESP introduciendo la contraseña (**ESP8266_12**) configurada en “config.cpp” (Ilustración 4), entra en una página web con todas las redes WiFi que es capaz de captar, como se muestra en la Ilustración 5.

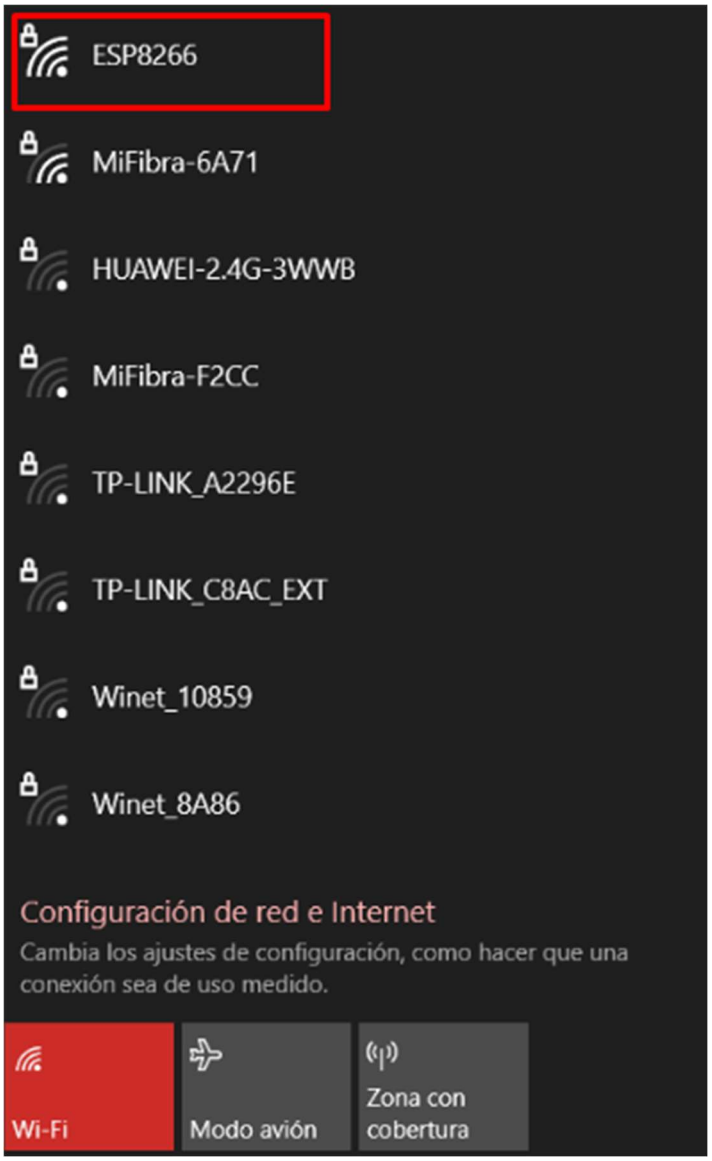

Ilustración 4 Se muestre el punto de acceso ESP creado por el propio microcontrolador

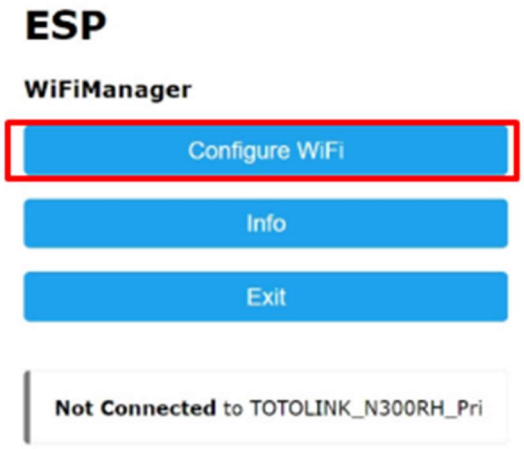

Ilustración 5 Interfaz página web para elegir configuración WiFi

Se selecciona el router al que se desea conectar introduciendo la contraseña (red Wifi a acceder) (Ilustración 6). Acto seguido, la ESP se conectará a esa red WiFi.

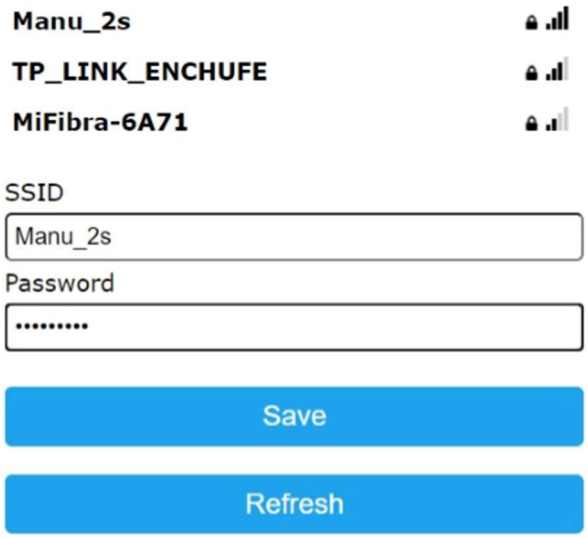

Ilustración 6 Selección de red WiFi para conectar el microcontrolador

- ssidReq()

Se encarga de obtener el indicador de fuerza de la señal recibida (RSSI: Received Signal Strength Indicator) para posteriormente enviarlo por MQTT junto a los demás datos recogidos.

6 TELEGRAM

6.1 Introducción

Se ha desarrollado un bot de Telegram, implementado en Node-RED, con una interfaz bastante asequible para el usuario. Esto permite obtener con cualquier smartphone la información del sistema, ya sea conocer los datos de los sensores, poder controlar el estado del led y switch de las placas, e incluso poder controlar el robot PIERO con una sencilla interfaz de mensajes.

Relacionado con lo anterior, se ha buscado una interfaz de usuario intuitiva. Para ello, se han agrupado los distintos comandos por categorías en función del tipo de acción que realiza. Disponemos principalmente de tres de ellas, las cuales son: información de los sensores, control de los actuadores (led y switch) y comandos relacionados con el control del robot. Para cada categoría, se ha generado un comando explícito que, tras ser pulsado por el usuario, devolverá en otro mensaje los distintos comandos relacionados en esa categoría, para así no mostrar de golpe toda la lista de comandos. Sin embargo, si el usuario ya es más experimentado o desea conocer a fondo todos los comandos de los que dispone el robot, también dispone de una petición para que se le muestre en un listado todos los comandos, además de una breve explicación al lado de ellos.

6.2 Generación de teclados auxiliares y robustez ante errores.

Como se ha comentado anteriormente, se ha enfocado el desarrollo del bot mediante comandos, y en aquellos en los que el usuario tenga que enviar la respuesta (por ejemplo, elegir

el led de la placa que desea apagar), generalmente se ha optado por generar un teclado para que sea más cómodo y evitar errores de escritura. No obstante, en comandos donde se espera una respuesta del usuario (como ajustar temperaturas de aviso), se verifica que el usuario introduce valores que cumplen la lógica del programa, y en caso contrario, se le manda un mensaje de aviso indicándole que ha ocurrido un error. En la Ilustración 7 puede verse una sección del flujo que establece estos comandos.

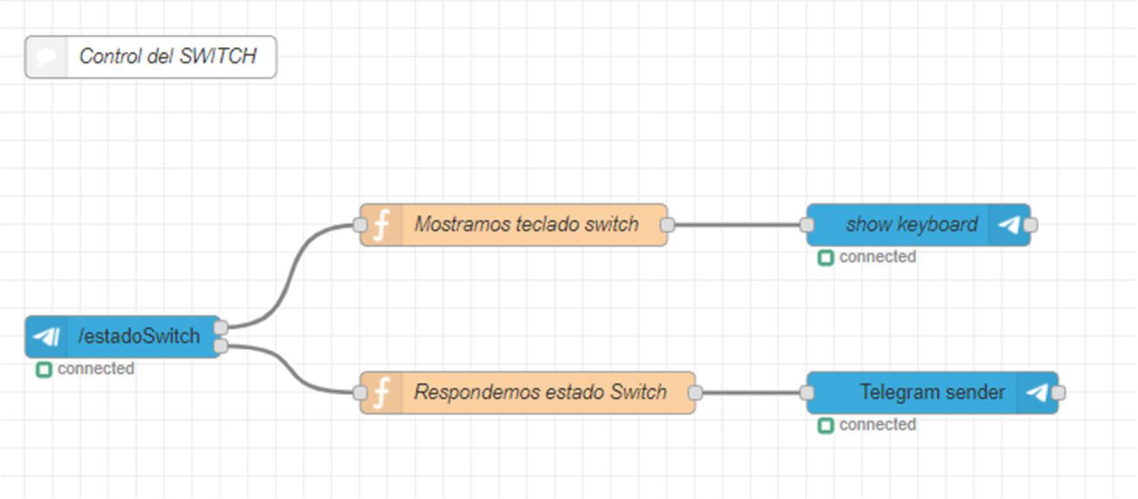

Ilustración 7 Ejemplo de flujo TELEGRAM establecimiento de comando /estadoSwitch

Cuando el usuario pulse /estadoSwitch, se desplegará un teclado como el de la Ilustración 8.

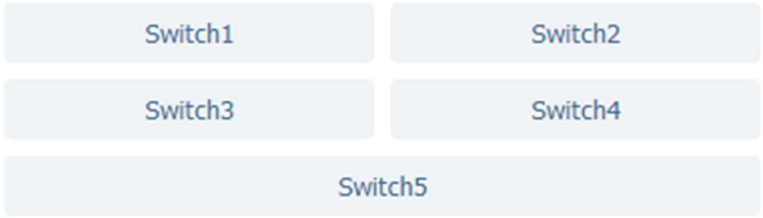

Ilustración 8 Ejemplo teclado para /estadoSwitch

Tras pulsar alguna de dichas opciones, el usuario tendrá como respuesta el último estado registrado en dicho switch. Si hubiera algún problema de conexión con respecto a ello, también se tiene en cuenta, devolviendo al usuario un mensaje de "No disponible".

Respecto a lo mencionado con los mensajes de errores, en esta la Ilustración 9 se muestra un ejemplo de ello.

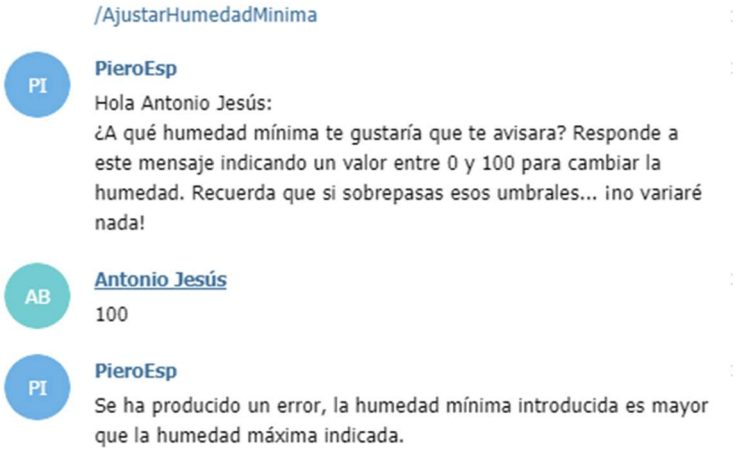

Ilustración 9 Ejemplo respuesta mensaje erróneo

Como se ha introducido una humedad mínima mayor que la máxima actual, se produce un error y se le notifica al usuario, para que introduzca un dato coherente para el sistema.

6.3 Control de actuadores

Con respecto al control de los actuadores, para las peticiones del usuario destinadas a variar el estado de sus dispositivos, se desarrollaron flujos similares al de la Ilustración 10.

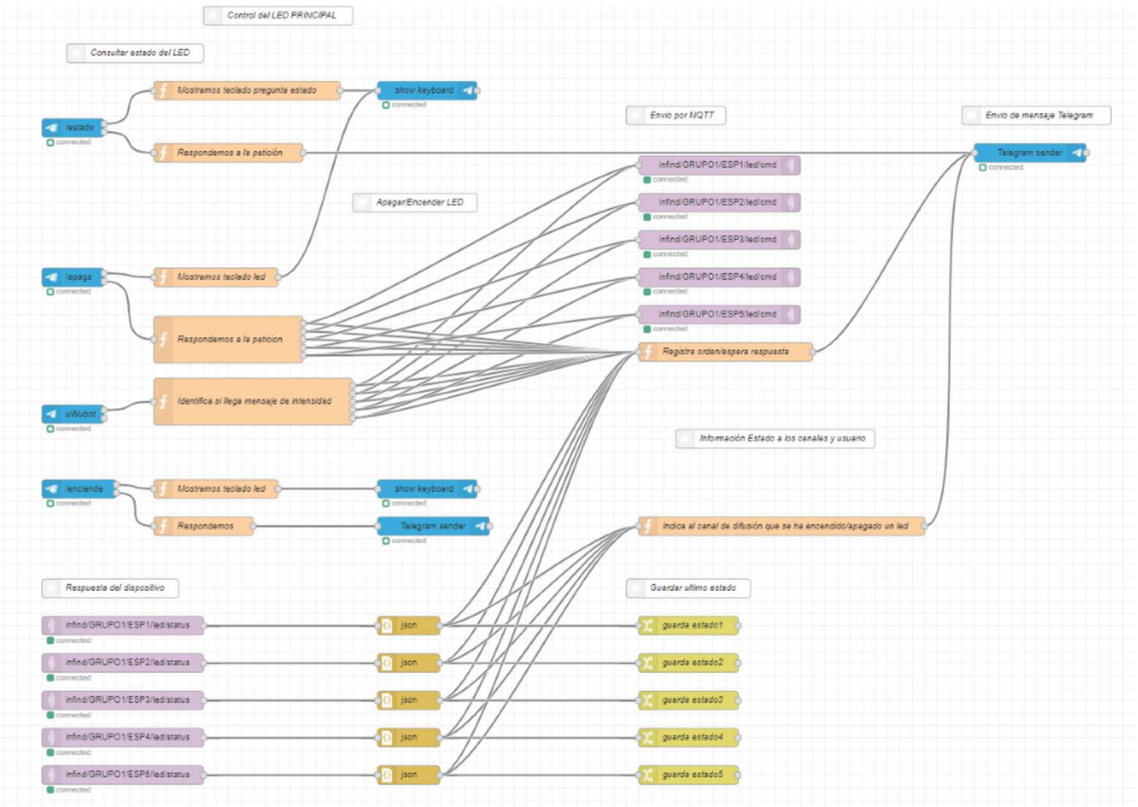

Ilustración 10 Flujo desarrollado para el control del estado del led de las placas en TELEGRAM

Dependiendo de la petición del usuario (ya sea apagar o encender el LED a una intensidad deseada), se publica por MQTT la petición correspondiente, para luego ser procesada por la ESP. Tras realizar la petición, se le envía un mensaje individual donde se indica que se ha realizado esa orden. Para luego recibir por el topic infind/GRUPO1/EXPx/led/status (siendo x el número de placa correspondiente a la petición) de MQTT la confirmación del estado del led (donde se corresponde la intensidad pedida con el estado real), se le vuelve a mandar un mensaje al usuario indicándole que la orden enviada se ha procesado correctamente e informándole sobre el estado del led. Además, también se publica por el canal de difusión de Telegram la confirmación de la respuesta, para informar al resto de usuarios que ha cambiado el estado del LED.

En caso de no recibir respuesta, significaría que ha habido un problema con las placas, ya sea de conexión o por algún tipo de avería; esto también se ha tenido en cuenta, donde tras esperar un tiempo considerable (10 segundos tras la petición) se le envía al usuario un mensaje diciendo que no se ha podido procesar la orden. Finalmente, cabe destacar en este apartado que se han tenido en cuenta las peticiones paralelas, almacenado la ID del usuario para que así el bot no respondiera erróneamente a otro usuario.

Si se realizara una petición de encender o apagar todos los leds simultáneamente, se enviará un mensaje por cada led, para así poder identificar, en caso de avería, cual ha sido la placa que ha fallado, para poder localizar errores en el sistema de una forma más sencilla.

Finalmente, para los switch se ha seguido la misma mecánica, donde se envía un mensaje al usuario indicando que se ha recibido la orden y tras recibir por MQTT el estado del switch por el topic correspondiente, se le envía posteriormente otro mensaje para confirmar que se ha procesado correctamente la orden.

6.4 Sensores

Cambiando un poco de ámbito y volviendo a los sensores, el usuario puede consultar el último estado de la humedad y/o la temperatura medida por cada DHT-11 o, si así lo deseara, puede realizar consultas de hasta una semana atrás, en donde se le informará de distintos valores, como la temperatura o la humedad media del intervalo consultado. Además, si los valores medidos por los sensores superan unos umbrales establecidos, se enviará por el canal de difusión un aviso, indicando qué sensor ha detectado dicha medida. Se ha añadido la opción de que el usuario pueda configurar los valores mínimos y máximos de aviso, y además se ha verificado que no se pudieran introducir temperaturas mínimas de aviso mayores que las máximas, para evitar fallos. Si el usuario realizara dicha petición, se le enviaría un mensaje de aviso indicándole que se ha producido un error (mostrado anteriormente en la ilustración 9). Para finalizar, se ha añadido una opción para restablecer los valores de aviso de temperatura y humedad a los valores establecidos por defecto, para asemejarlo más a un sistema real.

6.5 Control PIERO

Respecto al control del robot, se han generado distintos comandos relacionados con la ESP que controla a nuestro robot. El usuario dispone de la posibilidad de activar su robot en un modo de conducción automática, donde el robot comenzará a moverse libremente y sorteando obstáculos, o también dispondrá de la posibilidad de controlar manualmente su robot. En el caso de esta última opción, se despliega un teclado y se le explica al usuario el funcionamiento del teclado desplegado. Durante el control manual, si se encontraran obstáculos, el robot se pararía e indicaría con un mensaje al usuario la localización del obstáculo, solo pudiendo moverse con una acción adecuada (por ejemplo, si detecta un obstáculo delante, únicamente acepta que el usuario le indique marcha atrás). Si el usuario deseara salir del modo de control manual, automáticamente el robot se pararía, esperando de nuevo una nueva petición del usuario (modo manual o automático).

7 Node-RED

7.1 Conexión ESP

Este flujo se encarga de la recepción de los mensajes enviados desde el ESP, mediante el topic "infind/GRUPO1/ESPx/conexión", donde x indica el número de placa utilizado (1-5). Cada mensaje recibido se convierte a un objeto JSON y se le añade la fecha. Estos objetos son almacenados en la base de datos (mongoDB) en una colección llamada "_LOGS". A su vez se guarda el estado de la conexión (true/false) en una variable de tipo global, para su posterior uso en otros flujos de Node-RED.

7.2 Logs ESP

Compuesto del flujo "LOGs" y cuya pestaña en Dashboard recibe el mismo nombre. Su función principal es permitir la visualización de un histórico de conexiones, así como de las distintas ESP registradas.

Por una parte, se permite al usuario consultar mediante Dashboard el historial de registros de conexiones, donde se indica el estado (ONLINE/OFFLINE), la fecha, el número de placa utilizado y el "CHIPID". Además, se pueden establecer una serie de filtros, que restringen los parámetros de la búsqueda, mediante una operación de agregación a la base de datos (el nombre de la colección es "LOGs") por etapas: primero se restringen los resultados que coincidan con el intervalo temporal seleccionado. Acto seguido, se limita el número de resultados y realiza una proyección (cambio del nombre de los parámetros devueltos). El resultado finalmente se ordena en función del parámetro escogido (Fecha, Conexión, Placa o CHIP ID), en sentido ascendente o descendente (configurable por el usuario).

La segunda función que se presenta es la de almacenar un listado con los usuarios (ESP) registrados. Se tiene una colección (_USERS) en donde existe una entrada por cada usuario y en dicha entrada se guarda el "CHIPID", el número de placa asignado y la fecha de la última modificación. Al realizarse una nueva conexión de un dispositivo ESP, por defecto este se conecta como placa 0, publicando en el topic "infind/GRUPO1/ESP0/conexión", el cual es un canal común a cualquier dispositivo que se conecte. Esto genera una consulta a la colección "_USERS" devolviendo el número de placa asignado a ese determinado CHIPID y publicándolo en el topic de broadcast "infind/GRUPO1/ESP0/config/placa" para ser procesado por la ESP y conectarse al topic correspondiente a la placa establecida ("…/GRUPO1/ESPx/…"). De no encontrar una entrada para ese CHIPID se creará automáticamente una entrada nueva en la base de datos y se asignará el siguiente número disponible al parámetro placa. En caso de que la ESP tenga ya un numero de placa asignado y este no coincida con el dato registrado, se actualizará dicha entrada al valor establecido en la ESP.

El usuario puede consultar los usuarios registrados (Ilustración 11) mediante el Dashboard (pestaña "LOGs"), así como añadir o modificar usuarios existentes introduciendo el CHIPID y el número de placa deseados. También se permite borrar usuarios simplemente introduciendo el número de placa a borrar.

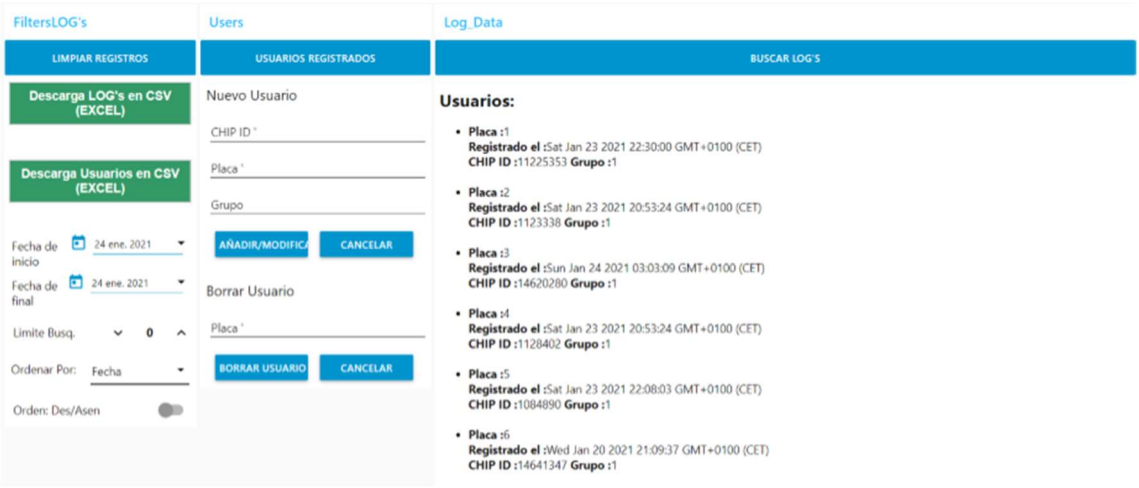

Ilustración 11 Ejemplo de búsqueda "Usuarios Registrados" en la pestaña LOGS

Ambas funciones comentadas, LOGs y Usuarios registrados, se podrán descargar en formato CSV los resultados de las búsquedas realizadas.

7.3 Recepción de datos ESP

Es el flujo encargado de recibir por MQTT el último dato recibido desde las ESP por el topic "infind/GRUPO1/ESPx/datos" (x indica el número de placa), le añade la fecha de recepción, genera una nueva entrada en la base de datos (colección "_DATOS") y muestra en el Dashboard (en la pestaña "UltimosDatos") los datos recibidos (Conexión, CHIPID, Uptime, Vcc, temperatura, humedad, actuadores y parámetros de la conexión Wifi). Generará una alerta en el Dashboard y en Telegram cuando la humedad o la temperatura sobrepasen un umbral configurable (pestaña "Control_ESP"). Por otro lado, recibirá por el topic "infind/GRUPO1/ESPX/FOTA/actualizado" la confirmación de la actualización FOTA y la almacenará en la base de datos, así como mostrarlo por el Dashboard. Finalmente, en la ilustración 12 se muestra dicha pestaña al visualizarla en el Dashboard.

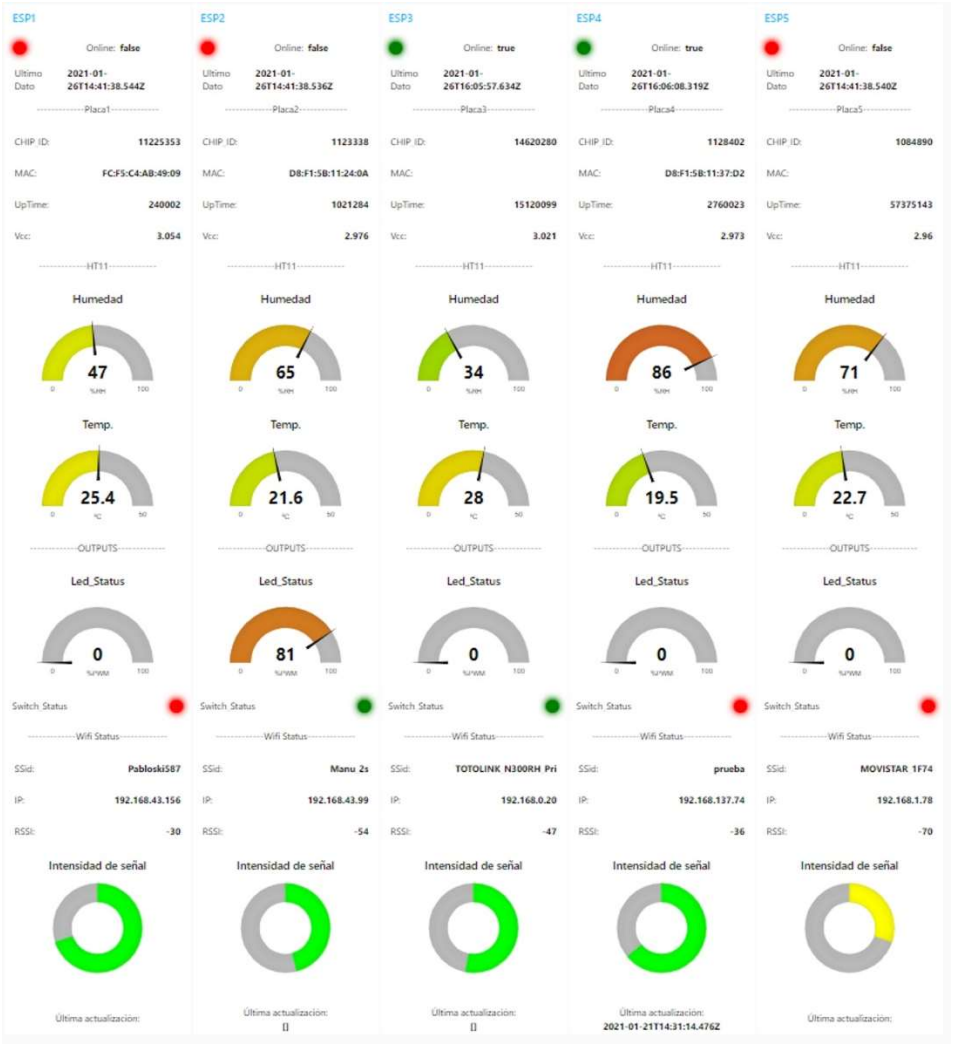

Ilustración 12 Ejemplo de "Ultimos Datos"

7.4 Control ESP

Implementa el control y la configuración de las distintas placas ESP. Para ello hay dos formas de hacerlo: la primera, es de forma individual. Cada placa tiene su propio panel de control "ESPxCtrl", en donde se podrá controlar el estado del led (PWM entre 0 y 100%), encender y apagar el switch, comprobar actualización (FOTA) y realizar las configuraciones necesarias en la placa. Estos parámetros son: "Lógica negativa/positiva" (permite invertir la lógica utilizada en los actuadores 0=apagado 1=encendido o viceversa); "Actualiza", donde se introduce el periodo (en minutos) en el cual se desea comprobar las actualizaciones; "Envia datos", en el cual se introduce (en segundos) el tiempo transcurrido entre envíos del paquete "datos" y, finalmente, el parámetro "Velocidad", que establece la velocidad de apagado del led (en 1%PWM/ms, el parámetro a introducir será de milisegundos).

Mediante los controles "ESP" y "MASTERControl" se permite al usuario controlar un grupo de ESP de forma simultánea. El procedimiento de control será el mismo que para cada placa individual. También se podrá configurar los valores de las alertas de temperaturas y humedad máximas y mínimas.

A modo de realimentación, se proporciona (para cada placa) el estado de la conexión (true/false), el estado del led y del switch. Estos datos provienen directamente de la respuesta (MQTT) del propio ESP al realizar algún cambio. El resultado final de esta pestaña aparece en la Ilustración 13.

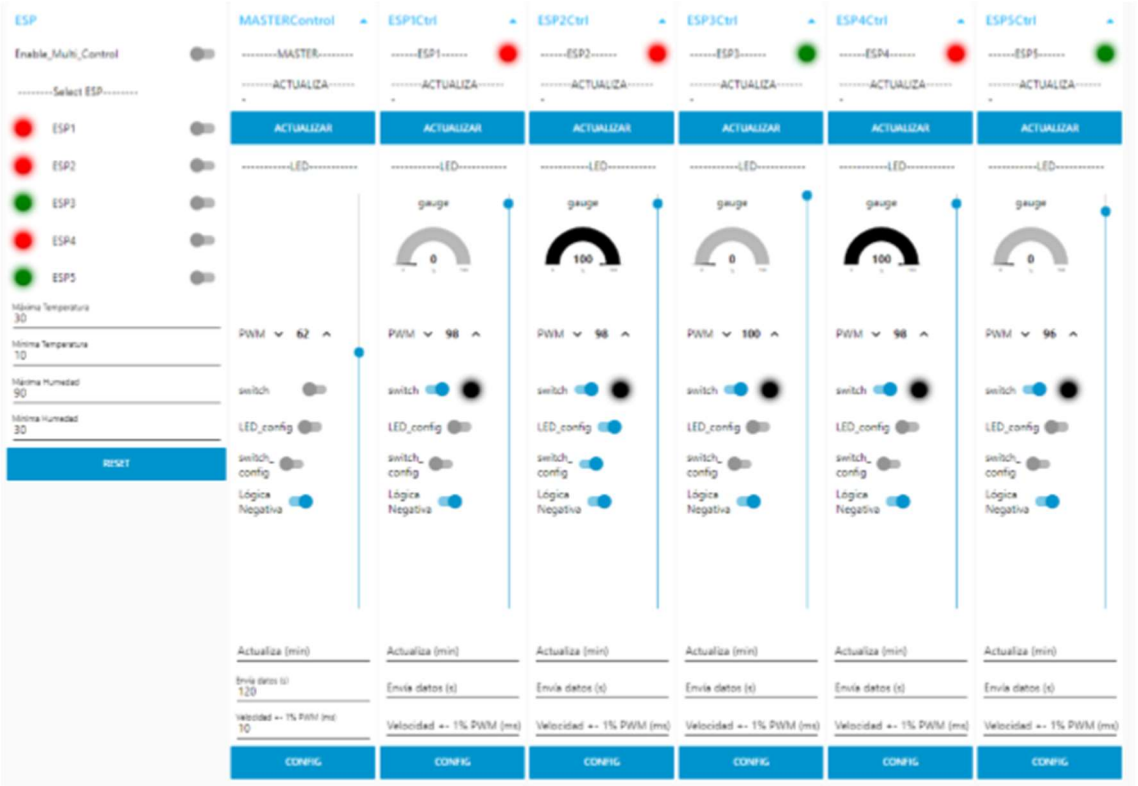

Ilustración 13 Ejemplo de "Control ESP"

7.5 Históricos

Esta sección se encarga de realizar consultas en la base de datos (colección _DATOS) en función de filtros establecidos desde el Dashboard. Los datos devueltos se representan en formato gráfico (chart) así como en texto en formato de lista. También se podrá descargar los resultados en formato CSV.

Los filtros o restricciones nos permiten seleccionar, por una parte, el tipo de dato a buscar (Humedad, Temperatura, Actuadores, Datos del Wifi y estadísticas). También se podrá seleccionar para qué placas se restringe la búsqueda (ESP1, ESP2…), así como escoger un periodo (fecha inicio/fecha fin), el parámetro en el que se ordenan los resultados (fecha, placa, CHIPID) y orden ascendente o descendente, junto al límite de entradas a consultar.

Se presentarán en formato de gráfico de líneas y puntos los parámetros de temperatura, humedad, voltaje de alimentación, led y switch (en caso de estar seleccionados en los filtros). En cada “chart” correspondiente se representarán de forma superpuesta (distintos colores) las distintas ESP.

En cuanto a los datos en formato texto se permite la visualización en el Dashboard en formato lista (html) los datos de la respuesta generada por la base de datos, para los filtros establecidos. En caso de estar seleccionada la opción “Estadísticas” se generará una segunda lista en la cual para cada placa se calculan la media, mínima y máxima para los valores de temperatura humedad y voltaje de alimentación. Se podrá optar por descargar, en formato CSV, la respuesta de la base de datos para los filtros seleccionados. En la ilustración 14 aparece el resultado final de la pestaña comentada anteriormente:

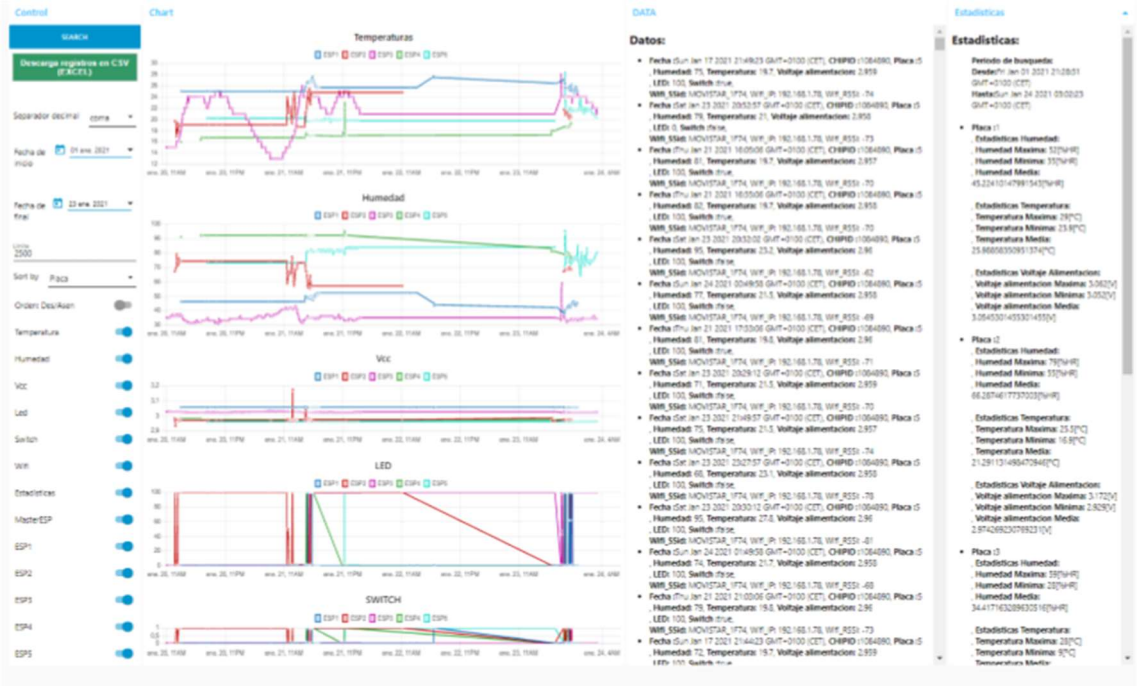

Ilustración 14 Ejemplo de búsqueda en la pestaña "Historicos"

7.6 Control PIERO

Esta pestaña de NodeRed se encarga del control del robot PIERO por medio del Dashboard. Se divide en dos partes: la recepción de los datos de los sensores de distancia y la emisión de comandos de control del PIERO, ambos realizados mediante MQTT.

En primer lugar, hay que tener en cuenta que se dispone de dos tipos de robots, uno con 2 sensores de distancia (láser), y otro con 5 (2 de ultrasonido y 3 láser). Por tanto, se dispone de dos flujos distintos para la recepción de los sensores y su interpretación:

- 2 sensores: recibe los valores de los sensores en un array y los divide en 2 mensajes: uno para cada valor, y los muestra en el Dashboard, siguiendo el flujo de la Ilustración 15. Además, si alguno de los sensores muestra un valor menor a 50 cm, se enciende un LED para indicar que hay un obstáculo.

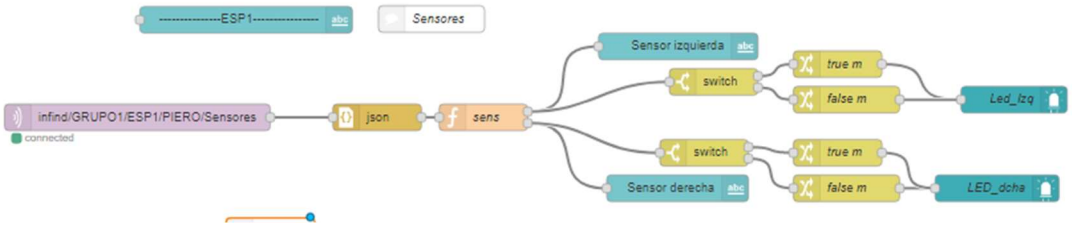

Ilustración 15 Flujo de recepción de datos para dos sensores

- 5 sensores: recibe los valores de los sensores en un array y los divide en 5 mensajes, uno para cada valor, y los muestra en el Dashboard, siguiendo el flujo de la Ilustración 16. Si alguno de los dos sensores de la izquierda tiene un valor inferior a 50 cm, se enciende el LED que indica un obstáculo por la izquierda. Ocurre de igual forma con los dos sensores de la derecha.

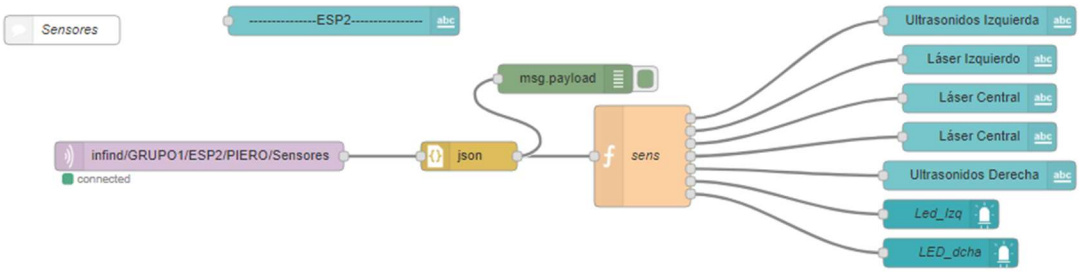

Ilustración 16 Flujo de recepción de datos para dos sensores

En cuanto al control del PIERO, éste se divide en dos partes: el control del modo de funcionamiento y el envío de comandos de velocidad (en caso de estar en modo Manual). PIERO tiene 3 modos de funcionamiento: Reposo, Manual y Automático.

- En el modo Reposo, el robot está parado y no admite comandos de movimiento.
- En el modo Manual, se puede comandar el robot con los botones W, A, S, D y Stop:
  - W: aumenta la velocidad lineal para ir hacia delante.
  - A: modifica la velocidad angular para girar a la izquierda.
  - S: disminuye la velocidad lineal para ir hacia atrás.
  - D: modifica la velocidad angular para girar a la derecha.
  - Stop: para el robot poniendo la velocidad lineal y angular a 0.

Estos botones son incrementales, lo que quiere decir que la velocidad es regulable en función de las veces que se pulsan (llegando hasta un cierto límite, ya que nuestros motores no alcanzan velocidades muy altas). Además, si se detecta un obstáculo, el robot parará automáticamente y sólo aceptará comandos que le permitan evitarlo.

- En el modo Automático, el robot se mueve de forma autónoma hacia adelante, evitando los obstáculos que se va encontrando.

Se controla el modo de comportamiento del robot con unos botones en el Dashboard y se guarda en una variable global. Se han configurado los modos para que sólo se pueda acceder a los modos Manual y Automático si se está previamente en el modo Reposo, como medida de seguridad en el control, puesto que el robot se detiene automáticamente al entrar en modo Reposo. Este modo de funcionamiento se le indica al PIERO mediante un mensaje MQTT y formateado en JSON, como se muestra en la Ilustración 17. Además, se indica en el Dashboard el modo de funcionamiento actual mediante dos LEDS, uno para el modo Manual y otro para el modo Automático.

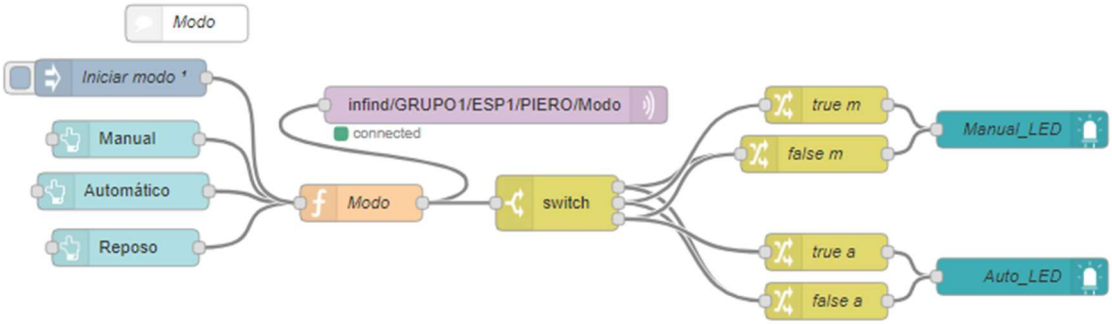

Ilustración 17 Flujo del control del modo de funcionamiento del robot

Por último, si el robot está en modo Manual, se envían al robot los comandos de velocidad W, A, S, D y Stop, mediante MQTT y formateados en JSON, según se vayan pulsando los botones en el Dashboard (como se muestra en la Ilustración 18).

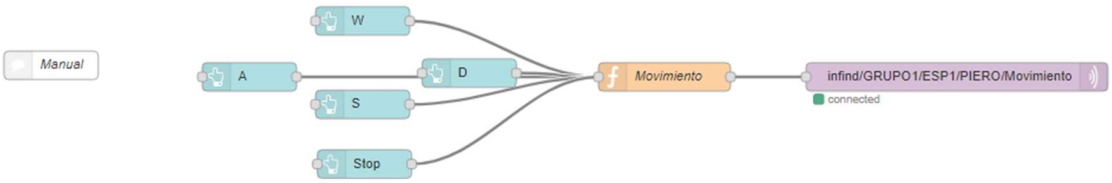

Ilustración 18 Flujo del envío de comandos de velocidad al robot

8 Simulink

8.1 Recepción de datos

La placa Arduino encargada de controlar el robot PIERO está programada en Simulink (Matlab). La comunicación entre la placa Arduino y la ESP se ha realizado mediante un puerto serie. Desde la parte de Simulink, existen bloques que se encargan de esta comunicación: Serial Receive y Serial Transmit. En la Ilustración 19 se puede observar la recepción de los comandos enviados desde la ESP. Éstos llegan a un bloque que contiene el diagrama de bloques o Stateflow de la Ilustración 20, donde se procesarán y darán a la salida el modo de funcionamiento del robot y la velocidad lineal que se le va a transmitir a cada rueda (en el caso de que se encuentre en modo Manual).

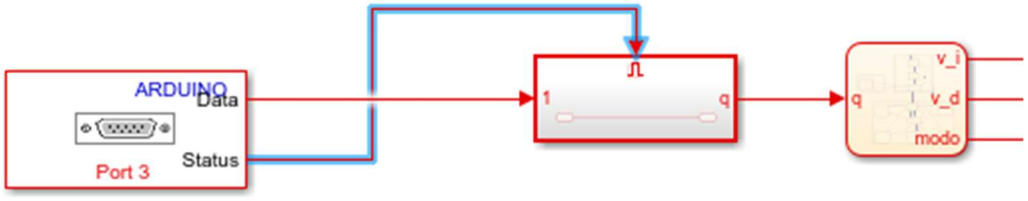

Ilustración 19 Recepción de los comandos del PIERO en Simulink

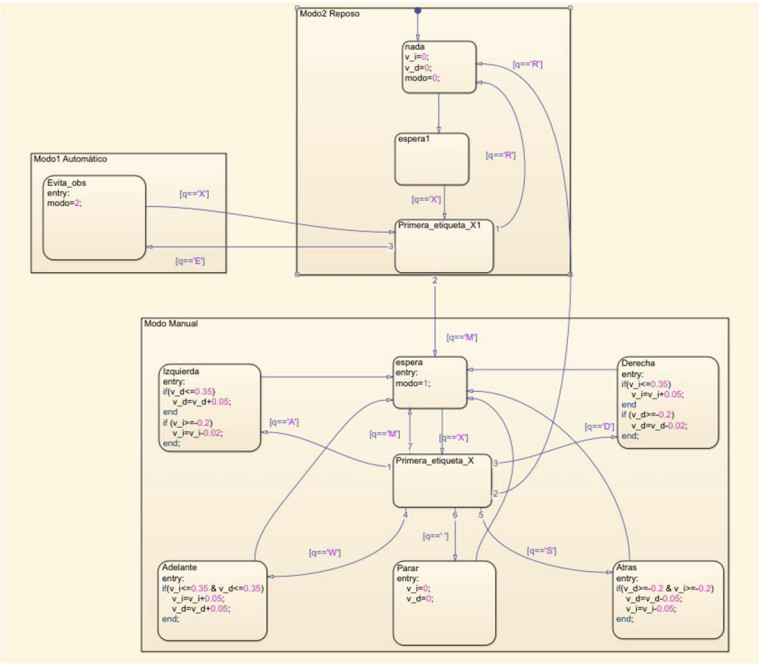

Ilustración 20 Procesamiento de los comandos recibidos

Los comandos que se reciben se dan en formato de dos bytes enviados de forma consecutiva: el primero es una 'X', una etiqueta que se utiliza para poder identificar que se trata de un comando; y el comando en sí, codificado de la siguiente manera:

- 'R': modo Reposo.
- 'M': modo Manual.
- 'E': modo Automático.
- 'W': hacia adelante.
- 'S': hacia atrás.
- 'A': hacia la izquierda.
- 'D': hacia la derecha.
- ' ' (un espacio): detenerse.

El diagrama de bloques (Ilustración 20) está dividido en tres zonas, una para cada modo. En cada una de ellas, cada vez que llega una 'X', el flujo de funcionamiento se dirige a un bloque donde queda a la espera del comando, según el cual se dirigirá a un bloque u otro para realizar distintas acciones.

8.2 Envío de datos

Finalmente, en la Ilustración 21, se muestra el envío de la información de los sensores de distancia del PIERO. La información se envía en un array de bytes donde se intercalan unos valores que se han reservado como etiquetas (del 251 al 255) y los valores de los sensores en centímetros, los cuales se han limitado a 250 cm para poder ser enviados en un solo byte. Se envían de esta forma para que la ESP pueda identificar el valor de cada sensor por la etiqueta que le precede. El bloque ZOH (en amarillo) permite controlar la frecuencia de envío de información a la ESP para no sobrecargarla. Actualmente está configurada para que se mande información cada 0.6 s, tiempo suficiente para que el PIERO, yendo a máxima velocidad, pueda detenerse sin llegar a chocar al detectar un obstáculo.

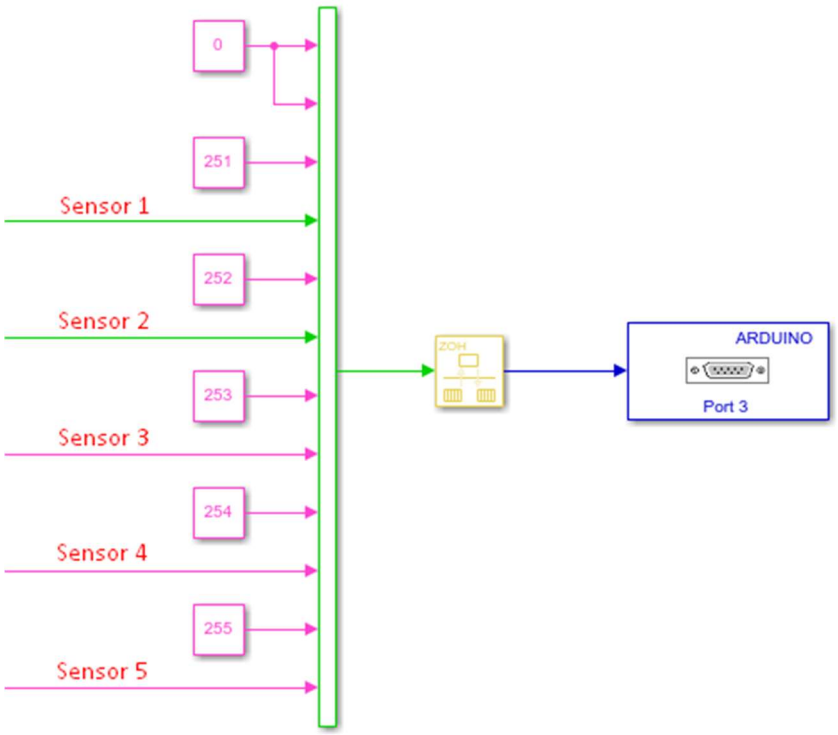

Ilustración 21 Envío de la información de los sensores por el puerto serie

9 Bibliotecas empleadas

- Arduino
  - ESP8266WiFi (https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi)
  - PubSubClient (https://github.com/knolleary/pubsubclient/releases/tag/v2.8)
  - DHTesp (https://github.com/beegee-tokyo/DHTesp)
  - ESP8266httpUpdate (https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266httpUpdate)
  - ArduinoJson (https://github.com/bblanchon/ArduinoJson)
  - DNSServer (https://github.com/esp8266/Arduino/blob/master/libraries/DNSServer/examples/DNSServer/DNSServer.ino)
  - ESP8266WebServer (https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServer)
  - WiFiManager (https://github.com/tzapu/WiFiManager)
- NodeRED
  - Node-red
  - Node-red-contrib-telegrambot
  - Node-red-contrib-chatbot
  - Node-red-contrib-ui-led
  - Node-red-dashboard
  - Node-red-node-mongodb
- MATLAB/Simulink
  - MATLAB Support Package for Arduino Hardware
  - Simulink Support Package for Arduino Hardware

10 Conclusiones

Al comenzar la asignatura, no conocíamos el concepto de “internet de las cosas” (IoT). Al finalizar la asignatura y terminar este proyecto hemos mirado con perspectiva con respecto al inicio de curso y hemos aprendido en muchísimos aspectos:

Hemos descubierto qué es Node-RED, cómo crear flujos y qué utilidades tiene. Además, hemos trabajado con bases de datos, haciendo consultas y agregaciones para obtener y administrar información. Hemos comprendido cómo funciona el protocolo MQTT, descubriendo qué es un “bróker”, un “topic” o el formato de texto JSON. Incluso hemos aprendido a crear un bot de Telegram capaz de interactuar con nosotros y asignarle distintos comandos directamente relacionados con aspectos de la asignatura.

Se ha profundizado en la utilización de distintos lenguajes de programación, como Java (usado para las funciones de Node-RED) o C++ (utilizado para la programación completa de la ESP8266). Hemos descubierto nuevas tecnologías con el fin de ser más completos, aprendiendo a usar la comunicación en serie entre 2 placas de prototipo (Arduino Mega 2560 y ESP8266).

Por último y no menos importante, hemos descubierto nuevas formas de trabajar en grupo, especialmente para trabajos donde se desarrolla código, utilizando plataformas como GitHub o plataformas para distribución de tareas, como Trello.

En definitiva, en esta asignatura hemos abordado multitud de temas y conocimientos que se pueden aplicar en la práctica y de una forma muy útil.

11 Referencias

[1] MQTT: https://mqtt.org/
[2] Node-Red: https://nodered.org/
[3] Arduino IDE: https://www.arduino.cc/en/software
[4] Placa ESP: https://hardzone.es/reportajes/tema/esp8266-2n2222-arduino/
[5] Telegram: https://web.telegram.org/
[6] FOTA: https://www.online-convert.com/es/formato-de-archivo/fota
[7] Mongodb: https://www.mongodb.com/es
[8] Robot
PIERO: https://riuma.uma.es/xmlui/bitstream/handle/10630/14766/Experiencia%20docente%20en%20Autom%C3%A1tica%20empleando%20un%20robot%20m%C3%B3vil%20como%20elemento%20motivador%20de%20metodolog%C3%ADas%20activas.pdf?sequence=3&isAllowed=y
[9] Arduino Mega 2560: https://proyectoarduino.com/arduino-mega-2560/
[10] Matlab/Simulink: https://la.mathworks.com/products/simulink.html
[11] Sensor DHT11: https://www.luisllamas.es/arduino-dht11-dht22/