Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DESARROLLO DE UNA RED DE SENSORES PARA AUDIOVIGILANCIA

Alumno: Juan Diego García Barranco

Tutor 1: Juan Carlos Cuevas Martínez
Depto.: Ingeniería de Telecomunicación

Tutor 2: Antonio Jesús Yuste Delgado
Depto.: Ingeniería de Telecomunicación

Septiembre, 2021

Resumen

Este Trabajo Fin de Grado describe el desarrollo de una red de sensores inalámbricos para audio vigilancia basada en la detección de eventos sonoros. También presenta un análisis sobre qué tecnología para redes de sensores es la más adecuada para las comunicaciones y para el hardware de los nodos sensores. Este trabajo también incluye el desarrollo de una plataforma web para gestionar la red de sensores basada en una base de datos y una API REST para recibir datos desde los nodos. La plataforma web proporciona un mecanismo de autenticación y la interfaz gráfica para monitorear las alarmas recibidas desde la red de sensores.

Abstract

This Degree thesis describes the development of a wireless sensor network for audio surveillance based on the detection of sound events. It also presents an analysis about which sensor network technology is the most suitable for the communications and for the hardware of the sensor nodes. This work also includes the development of a web platform to manage the sensor network based on a database and a REST API to receive data from the nodes. The web platform provides an authentication mechanism and the graphical interface to monitor the alarms thrown from the sensor network.

Índice de contenido

Índice de imágenes

Índice de tablas

1. Introducción

En los últimos años ha habido un auge en las tecnologías de redes inalámbricas y más aún en las redes de sensores para su aplicación en diferentes campos, entre ellos en el campo de la seguridad, ya que su ventaja en cuanto a su versatilidad y precio respecto a otras soluciones es más que evidente.

Una red de sensores inalámbricos se puede definir como una red que consta de varios nodos de sensores, conectados entre sí por medios inalámbricos, que se encargan de recopilar y procesar información y mandarla a un nodo central, el cual se encarga de realizar las operaciones pertinentes. Una de las ventajas principales de las redes de sensores es la facilidad que ofrecen para poder desplegarlas en gran cantidad de emplazamientos.

Esta tecnología también se puede incluir en el Internet de las coas (IoT), ya que las redes de sensores se pueden conectar a Internet mediante el nodo central, estación base, o Coordinador, el cual envía todos los datos a un servidor para poder acceder a ellos desde cualquier lugar.

Este trabajo presenta el análisis de la tecnología y dispositivos más apropiados a usar para una red de sensores para audio vigilancia y su desarrollo, tanto la comunicación entre los nodos como el sistema de detección de sonido y que la red de sensores se comunique con un servidor con una API REST para que también se pueda monitorear todos los datos enviados desde la red de sensores en una aplicación web para que se pueda ver en todo momento si algún sensor ha detectado ruido, a la vez que se pueda añadir más sensores a la red y/o modificar los ya existentes.

1.2. Objetivos

Los objetivos de este TFG son los siguientes:

1. Análisis de las tecnologías aplicables a las redes de sensores.
2. Análisis de los distintos dispositivos para la detección de eventos provocados por fuentes acústicas.
3. Diseño de los elementos a emplear y la red de soporte.
4. Diseño de la aplicación para la gestión de los eventos de interés.
5. Realización de pruebas.
6. Estudio de resultados.

2. Estado del arte

En esta sección se exponen algunos de los proyectos presentes hoy en día de IoT con redes de sensores y de esta manera ver la mejor forma en la que realizar el proyecto.

2.1. Libelium

Libelium (1) es una empresa la cual diseña y fabrica gran variedad de soluciones tecnológicas para IoT. Tiene gran cantidad de productos para todo tipo de proyectos y comercializan desde los sensores, hasta los distintos dispositivos de comunicación.

También disponen de un servicio de consultoría con el cual puede realizar cualquier proyecto.

Figura 1 Logo Libelium

Fuente: https://www.libelium.com

Esta empresa ofrece sobre todo módulos de comunicación y conjuntos de sensores, los cuales vienen con un gran acabado y permiten una instalación muy sencilla y rápida. También cuentan con gran variedad de soluciones desarrolladas que se gestionan en la nube con gran posibilidad de personalización.

Destaca para este trabajo sobre todo el producto Waspmote, ya que es un dispositivo sensor que está pensado para el desarrollo de proyectos IoT y cuenta con gran variedad de sensores disponibles, gran capacidad de conectividad con distintas tecnologías inalámbricas y un muy bajo consumo energético.

En cuanto a las tecnologías inalámbricas que incluye el producto antes mencionado, se puede destacar las siguientes tecnologías:

- 4G.
- LoRaWAN.
- ZigBee.
- WiFi.
- Bluetooth Low Energy.

Un punto a tener en cuenta es que únicamente nos permite usar una tecnología inalámbrica a la vez, a no ser que usemos el módulo “Expansion Radio Board” el cual permite poder tener dos interfaces inalámbricas simultáneamente.

Sin embargo, todo lo que puede ofrecer esta empresa tiene un coste demasiado elevado para este trabajo.

2.2. Home Security Alarm Usign Wemos D1 and HC-SR501 Sensor Bases Telegram Notification

Este es un Proyecto (2) que ha sido desarrollado por Refni Wahyuni, Aditya Rickyta, Uci Rahmalisa y Yuda Irawan, y trata también de un sistema de alarma, pero a diferencia de este proyecto en vez de ser la alarma por sonido, es mediante un sensor de movimiento. Para ello usan sensor de movimiento, en concreto el HC-SR501, y la placa de desarrollo Wemos D1, la cual es una placa de desarrollo del microcontrolador Nodemcu ESP8266.

En este proyecto usan Wifi como tecnología de comunicación y lo interesante de este proyecto es que usan Telegram como sistema de notificaciones del sensor.

Telegram es una aplicación de mensajería móvil, que también cuenta con una API para desarrollar aplicaciones las cuales puedan enviar mensajes.

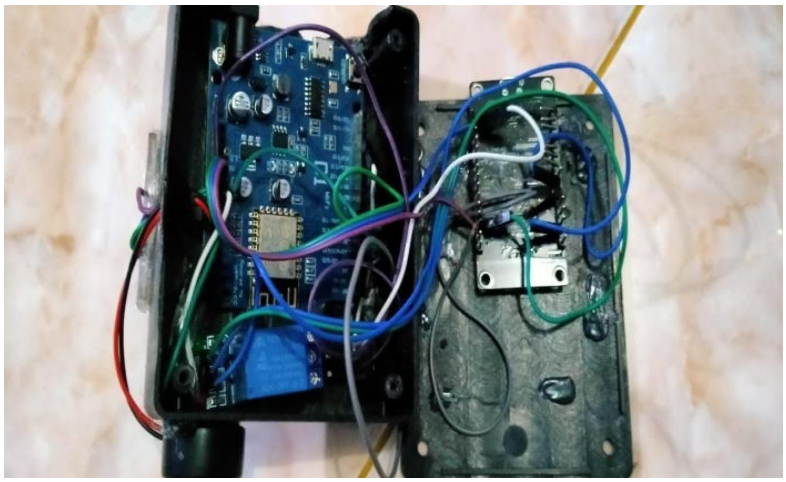


Figura 2 Home Security Alarm Using Wemos D1 and Sensor HC-SR501

Fuente: https://journal.umy.ac.id/index.php/jrc/article/view/10013

2.3. Illegal Logging Detection Based on Acoustic Surveillance of Forest

En este proyecto (3) se busca solucionar el problema de la tala ilegal de árboles con un sistema de redes de sensores para vigilancia acústica con pequeños sensores de bajo coste y así poder detectar la tala de árboles en grandes áreas como los bosques. En este proyecto también se realiza un procesado de audio para poder eliminar el ruido de fondo que hay normalmente y diferenciarlo del ruido generado por herramientas mecánicas.

Lo más interesante de este proyecto es el procesado de audio que se realiza para poder diferenciar el ruido normal de un bosque con el ruido de herramientas mecánicas.

Cabe mencionar que para el ámbito de este TFG no sería necesario el procesado del sonido ya que la alarma saltará con cualquier tipo de ruido superior al umbral de dB establecido.

3. Metodología

En este apartado se va a analizar las tecnologías que se podrían usar en este trabajo desde tres vertientes distintas:

- Las comunicaciones, se analizará que tecnología es la que mejor se adapta a este trabajo.
- El hardware, se analizará que microcontrolador y sensores son más interesantes para el ámbito de este trabajo.
- Análisis de las tecnologías de programación más interesantes para la gestión de las alarmas y plataforma web.

3.1. Análisis de comunicación

Se va a exponer los principales protocolos de comunicación para redes de sensores y cuál es el más adecuado para el ámbito de este trabajo.

3.1.1. Bluetooth

Bluetooth es una tecnología inalámbrica la cual se creó en un inicio para el ámbito industrial. Esta tecnología, realmente es un conjunto de protocolos, posee dos especificaciones diferentes:

- Bluetooth clásico
- Bluetooth de baja energía (BLE)

Las características generales son:

- Distancia de comunicación de hasta 1 km (en línea recta sin obstáculos).
- Uso de la frecuencia libre de 2,4 GHz.
- Transmisión de alta fiabilidad, esto lo consigue mediante el uso de canales de transmisión redundantes.
- Tiempo de retardo reducido (5-10 ms).
- Capacidad para funcionar en entornos con gran cantidad de dispositivos debido a su aprovechamiento de la frecuencia.

En cuanto a seguridad (4), Bluetooth ofrece características de seguridad similares a otros protocolos inalámbricos. Los aspectos de seguridad en Bluetooth pueden aplicarse bajo tres modos de operación:

- Sin seguridad.
- Seguridad en el nivel de servicios
- Seguridad en el nivel de enlace

3.1.2. WiFi

WiFi (5) es una tecnología inalámbrica que es en realidad la familia de normas inalámbricas IEEE 802.11. Esta tecnología permite la conexión inalámbrica entre dispositivos inalámbricos.

WiFi usa dos frecuencias distintas dependiendo del estándar que use dentro de la familia IEEE 802.11. Puede usar 2,4 GHz y 5 GHz, cada una de ellas tiene un punto fuerte.

El estándar a 2,4 GHz es capaz de llegar a mayor distancia, pero con unas prestaciones inferiores, sin embargo, el estándar a 5 GHz tiene mayores prestaciones, pero menor alcance.

3.1.3. Zigbee

Zigbee (6) es otra especificación de comunicación inalámbrica que se utiliza transmisiones inalámbricas de bajo consumo. Está basada en el estándar IEEE 802.15.4.

El objetivo principal de esta tecnología es utilizarla en aquellas aplicaciones que requieran comunicaciones seguras con bajo consumo y baja tasa de envío de datos.

Las características principales de esta tecnología son:

- Bajo consumo.
- Topología de red en malla, la cual permite alcanzar grandes zonas.
- Fácil integración en cualquier tipo de proyecto.

Zigbee utiliza la banda Industrial, Científico y Medico (ISM).

Si se compara el consumo de Zigbee contra Bluetooth se ve que Zigbee tiene un consumo menor con 30 mA en transmisión y 3 µA en reposo y Bluetooth 40 mA transmitiendo y 0,2 mA en reposo.

3.1.4. LoRaWAN

LoRaWAN es una especificación de una tecnológica creada especialmente para dispositivos de bajo consumo y que esta ideada especialmente para conexiones a grandes distancia y redes IoT. Las ventajas de esta tecnología (7) son:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos
- Bajo consumo
- Largo alcance de 10 a 20 km
- Baja transferencia de datos
- Conexión punto a punto
- Las frecuencias de trabajo son: 868 MHz en Europa, 915 en América y 433 MHz en Asía.

Normalmente las redes con LoRaWAN (8) suelen tener una topología en estrella, siendo el nodo central de la estrella, la puerta de enlace de los dispositivos finales, es decir,

la red de sensores se comunica con el nodo central mediante otra tecnología y el nodo central usa LoRaWAN para comunicarse con un servidor que también tenga LoRaWAN. Esto permite poder desplegar la red de sensores lejos de cualquier conexión a Internet tradicional.

3.2. Análisis de materiales

En esta sección se va a exponer las opciones de hardware posibles y cuál de las opciones es la más adecuada para este trabajo.

3.2.1. Microcontrolador

Cuando se habla de un microcontrolador (9) se habla de un circuito integrado, el cual incluye sistemas para controlar elementos de entrada/salida, procesador y memoria. La función principal de un microcontrolador es automatizar procesos y/o procesar información.

Un microcontrolador tiene al menos un procesador, unidades de entrada/salida y memoria, la memoria consta tanto de memoria RAM como de memoria FLASH que es donde se guarda el programa y también tiene una memoria EEPROM.

Raspberry Pi Zero W

Este dispositivo destaca por su pequeño tamaño, ya que es la mitad que una Raspberry Pi Model A+, esto hace posible que se pueda adaptar para prácticamente cualquier proyecto, además ofrece muchas opciones de conectividad.

Las características técnicas de este dispositivo (10) son:

- CPU de un solo núcleo a 1GHz
- 512MB de RAM
- Mini HDMI y USB
- Entrada de corriente Micro USB
- 40 GPIO
- Conector de cámara CSI

En cuanto a conectividad:

- 802.11b/g/n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)

Dimensiones:

- 66 mm x 30.5 mm x 5 mm

Una de las grandes ventajas de este modelo en concreto de Raspberry Pi es que tiene la misma cantidad de GPIO que sus versiones superiores, a un menor precio, menor tamaño y menor consumo.

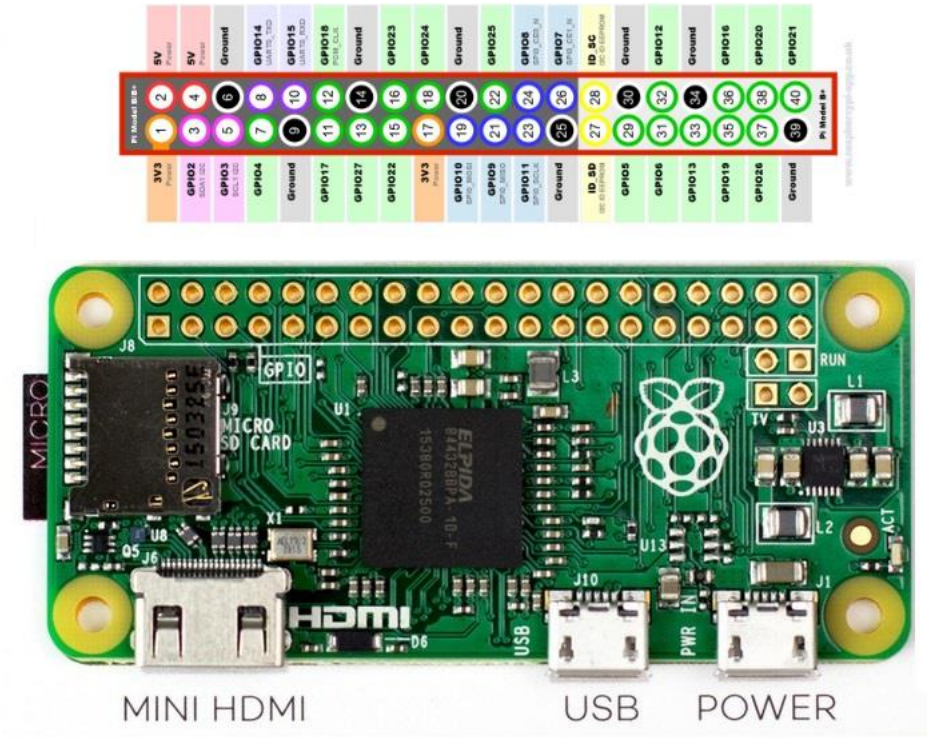


Figura 3 Raspberry Pi Zero W Datasheet

Fuente: https://tublogen3d.com/raspberry-pi/zero-w/

En cuanto a consumo (11) se puede ver que la Raspberry Pi Zero W en reposo consume unos 120 mA y a pleno rendimiento 230 mA (se entiende pleno rendimiento como grabación de video en 1080p, ya que es de las tareas que más energía consume).

	Zero	**Zero W**	**A+**	**A**	**B+**	**B**	**Pi2B**	**Pi3B**
	mA	mA	mA	mA	mA	mA	mA	mA
En Espera	100	120	100	140	200	360	230	230
Cargando LXDE	140	160	130	190	230	400	310	310
Viendo Video 1080p	140	170	140	200	240	420	290	290
Grabando video 1080p	240	230	230	320	330	480	350	350

Tabla 1 Consumos Raspberry Pi

Fuente: https://www.redeszone.net/2017/03/02/energia-consumida-raspberry-pi/

Arduino Uno R3

Arduino Uno (12) es una placa basada en el microcontrolador ATmega328P, consta de 14 pines de entrada/salida digital y 6 de ellas son *Pulse-Width Modulation* (PWM), también cuenta con 6 entradas analógicas y un oscilador de 16 MHz. Se pueden encontrar todas las especificaciones técnicas en la web del fabricante.

Las dimensiones del dispositivo son:

- 68.6 mm x 53.4 mm

En cuanto al consumo (13), podemos ver que el Arduino Uno R3 consume de media 46,5 mA.

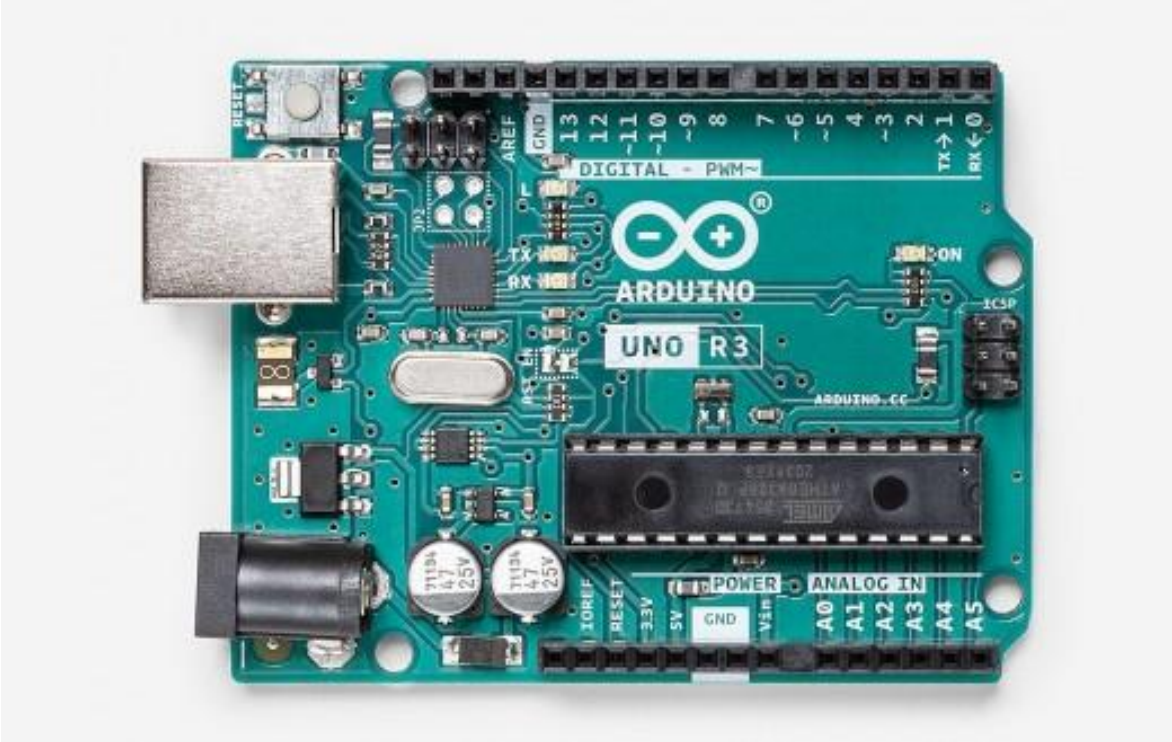


Figura 4 Arduino Uno R3

Fuente: https://store.arduino.cc/arduino-uno-rev3

NodeMCU V3

Cuando se habla de NodeMCU, se habla de una placa de desarrollo del *System on a Chip* (SoC) ESP8266. Las características principales de esta placa de desarrollo son:

- Procesador: ESP8266
- 4MB de memoria FLASH
- WiFi 802.11 b/g/n
- Regulador 3.3V integrado
- 9 pines GPIO con I2C y SPI
- 1 entrada analógica

Las dimensiones de este dispositivo son:

- 58 mm x 31 mm x 13 mm

NodeMCU (14) permite ser programado en Arduino. En cuanto a consumo (15), este dispositivo tiene un consumo medio de 80 mA.

Figura 5 NodeMCU V3

Fuente: https://tienda.bricogeek.com/wifi/1033-nodemcu-v3-wifi-esp8266-ch340.html

3.2.2. Sensores

Para este trabajo se necesita un sensor de sonido o micrófono.

Micrófono KY-038

Este dispositivo es una placa la cual incorpora un micrófono junto a un comparador LM393. Este sensor se suele usar sobre todo para detectar cuando el sonido supera un umbral, el cual es regulado mediante un potenciómetro que está ubicado en la placa.

Este sensor (16) no es recomendado para leer su salida analógica ya que no tiene amplificación y lo que envía por su salida analógica es una estimación del volumen que ha registrado.

Se puede alimentar con una entrada de 5V o 3.3V y tiene una sensibilidad de aproximadamente entre 48 y 66 dB.

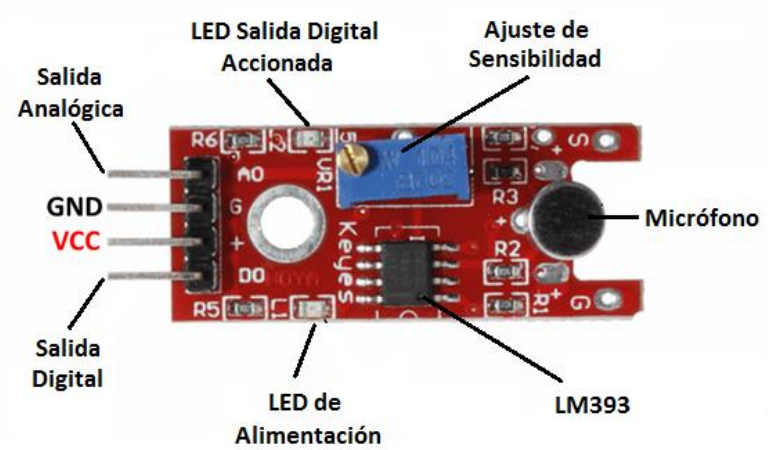


Figura 6 Componentes micrófono KY-038

Fuente: https://aprendiendoarduino.wordpress.com/2018/10/16/modulo-microfono-arduino/

Módulo Sensor de Sonido FC-109

Este dispositivo (17) es una placa que incorpora un micrófono con un amplificador de sonido, esto hace que se pueda detectar la intensidad de sonido ambiental.

El FC-109 necesita de una alimentación de 3.3V a 5V.

Figura 7 Modulo Sensor de Sonido FC-109

Fuente: https://www.bigtronica.com/sensores/sonido/121-tarjeta-sensor-de-sonido-mini-5053212001216.html

Módulo Amplificador de micrófono MAX4466

Este dispositivo (18) incorpora un micrófono y un amplificador que lo hace ideal para usarlo como un micrófono con preamplificador. Para alimentarlo simplemente hay que conectarlo a una entrada de 2.5V a 5 V y a tierra y tiene una salida lineal analógica.

Figura 8 Micrófono MAX4466

Fuente: https://www.amazon.es/ARCELI-Amplificador-micr%C3%B3fono-GY-MAX4466-Ajustable/dp/B07MY293KD/ref=asc_df_B07MY293KD/

Modulo Micrófono DFR0034

Este dispositivo también es capaz de detectar el sonido de manera lineal y se alimenta mediante una entrada entre 3.3V a 5 V

Figura 9 Micrófono DFR0034

Fuente: https://es.farnell.com/dfrobot/dfr0034/sensor-anal-gico-sonido-placa/dp/2946099?cjevent=59e51d0bc6b811eb819460200a180510&cjdata=MXxZfDB8WXww&CMP=AFC-CJ-ES-7947610&gross_price=true&source=CJ

3.3. Análisis de tecnologías

3.3.1. Base de datos

MySQL

MySQL (19) es un sistema para la gestión de bases de datos relacional, este sistema está desarrollado por Oracle. Este es uno de los sistemas de bases de datos más usado de código abierto. Es uno de los sistemas más sencillo de implementar para cualquier tecnología. Características destacas:

- Facilidad para escalar el sistema.
- Buen rendimiento.
- Gran flexibilidad para gestionar gran cantidad de datos.
- Debido a su popularidad, hay una gran cantidad de información en la red.

MariaDB

MariaDB (20) es un sistema para la gestión de datos y es una bifurcación de MySQL. Tiene algunas ventajas respecto a MySQL:

- Gran variedad de motores de almacenamiento, esto hace que pueda almacenar distintos tipos de datos de forma más efectiva.
- Incorpora un almacenamiento en caché que hace que los INSERT sean más rápidos que con MySQL
- Sistema de complementos el cual permite agregar funcionalidades al servidor MariaDB en caliente, sin que se tenga que rehacer el código fuente.

MongoDB

MongoDB (21) es un sistema de base de datos noSQL que está basado en documentos y cada vez más va aumentando su popularidad. Una de las funciones principales es la opción *insertMany()* la cual le permite insertar datos de manera más rápida. Características:

- Escalabilidad.
- Destinada al desarrollo móvil debido a la gran cantidad de datos que suelen producir estos sistemas.
- Estructura dinámica.

3.3.2. Back-end

En esta sección se va a exponer distintos *framework* a poder usar para crear la API REST con la conexión a la base de datos.

Symfony

Symfony (22) es un *framework* para el desarrollo de aplicaciones Modelo Vista Contralador (MVC). Este *framework* está desarrollado en PHP y es compatible con la gran mayoría de bases de datos. Proporciona herramientas para reducir el tiempo de desarrollo de una aplicación web.

NodeJS Express

NodeJS (23) es un entorno de programación el cual trabaja en tiempo de ejecución, este entorno de programación permite crear todo tipo de aplicaciones en JavaScript. Desde el punto de vista del desarrollo NodeJS tiene muchas ventajas:

- Buen rendimiento y escalabilidad.
- Gestor de paquetes propio el cual proporciona acceso a gran cantidad de paquetes.
- Fácil despliegue.
- Gran comunidad en Internet, esto hace que haya gran cantidad de información y recursos.

Express (24) es el *middleware* más popular de NodeJS. Este *middleware* proporciona:

- Manejadores de peticiones con distintos métodos de petición y rutas.
- Ajustes generales para la aplicación como puerto a utilizar.
- Capacidad de agregar procesamiento de peticiones mediante *middleware.*

3.3.3. Front-end

En esta sección se expondrán los *framework* para el desarrollo de la interfaz gráfica que verá el usuario final en el navegador.

AngularJS

AngularJS (25) es un *framework* desarrollado con JavaScript, que es usado para crear aplicaciones web de página única y contenido dinámico. Una de las ventajas principales de AngularJS es que esta desarrollado por Google y eso hace que sea muy usado a nivel mundial por la gran experiencia que tiene Google en este sector.

Vue.js

Vue.js (26) es de los *framework* que se ha creado en los últimos años y más usado. La gran ventaja que presenta Vue.js frente a otros *framework* para front-end, es que es muy ligero. Gracias a que sea tan ligero hace que sea muy rápido y flexible. También es muy sencillo de aprender y tiene librerías para extender sus funciones.

React

React (27) es un *framework* desarrollado en JavaScript y fue desarrollado primeramente en Facebook, de manera interna, para ser más efectivos. Hoy en día es también de los *framework* para *front-end* más usado. React está basado en componentes reutilizables, esto hace que sea muy rápido el crear interfaces de usuario complejos.

3.4. Conclusión

3.4.1. Comunicaciones

Para este trabajo se necesita que haya un bajo consumo, ya que los nodos van a funcionar mediante una batería y también se necesita un gran alcance entre los distintos nodos.

Teniendo esto en cuenta, la solución más adecuada para desplegar la red seria usar un nodo puente conectado mediante LoRaWAN a un equipo conectado a Internet, y el resto de nodos conectados al nodo principal mediante Zigbee, ya que permite tener una red en malla y no es necesario que todos los nodos estén conectados directamente al nodo puente.

Esto proporcionaría la capacidad de poder desplegar la red de sensores a gran distancia de cualquier punto urbano, ya que LoRaWAN tiene un alcance de 10 a 20 Km de distancia. Pero cuando se tiene en cuenta el precio reducido que debe tener este trabajo, esta opción queda descartada.

Finalmente se desplegará la red de sensores mediante Zigbee por las ventajas que ofrece por el bajo consumo y gran alcance, y la red Zigbee estará conectada a un servidor en Internet mediante un nodo puente que usará WiFi y Zigbee simultáneamente.

3.4.2. Microcontrolador

Para el ámbito de este trabajo se necesita un microcontrolador que consuma poca energía y sea compatible con las tecnologías mencionadas en el punto anterior. Si se tienen en cuenta los consumos el microcontrolador que menos consume, y lleva incorporado WiFi en sus comunicaciones, es el NodeMCU V3. También se ve que es el más barato entre los microcontroladores mencionados.

3.4.3. Sensores

En cuanto a los sensores de sonido, se ha decidido usar el KY-038, ya que es más sencillo cambiar su sensibilidad para cualquier persona, sin necesidad de saber nada de dB, simplemente girando un tornillo.

3.4.4. Base de datos

En este trabajo para la implementación de la base de datos se va a usar MySQL, ya que es la más popular a nivel mundial y permite una fácil y rápida aplicación web.

3.4.5. Back-end

Después de exponer las ventajas de las tecnologías de *back-end* expuestas en el apartado anterior. En este trabajo se usará NodeJS con el *middleware* Express, ya que, es especialmente indicado para realizar una API REST porque usa JavaScript al igual que el *front-end.*

3.4.6. Front-end

Para el *front-end* se expondrá Vue.js por ser ligero y sencillo de aprender.

4. Desarrollo

En la primera sección se va a ver un esquema general del sistema, después de este se va a detallar el desarrollo de la red de sensores que es la que va a enviar la notificación cuando uno de los sensores detecte sonido, el servicio API REST el cual es el encargado de registrar dicha alarma y el *front-end* que desde el cual el usuario gestionará el sistema.

4.1. Esquema general

Este sistema podrá ser usado como alarma por detección de sonido, la detección de sonido se realizará mediante la red de sensores y desde la aplicación web se gestionaran las alarmas.

Como se puede apreciar en el esquema general del sistema en la Figura 10, en este trabajo se diferencia claramente la parte que va a visualizar el usuario (*Front-End*), la parte que realiza toda la lógica del servicio (*Back-End*) y la red de sensores.

Los nodos de la red de sensores se comunican entre sí usando la tecnología Zigbee, todos los datos que envían los nodos sensores los recibe el nodo coordinador y es éste el que se comunica con el *Back-End* mediante HTTP.

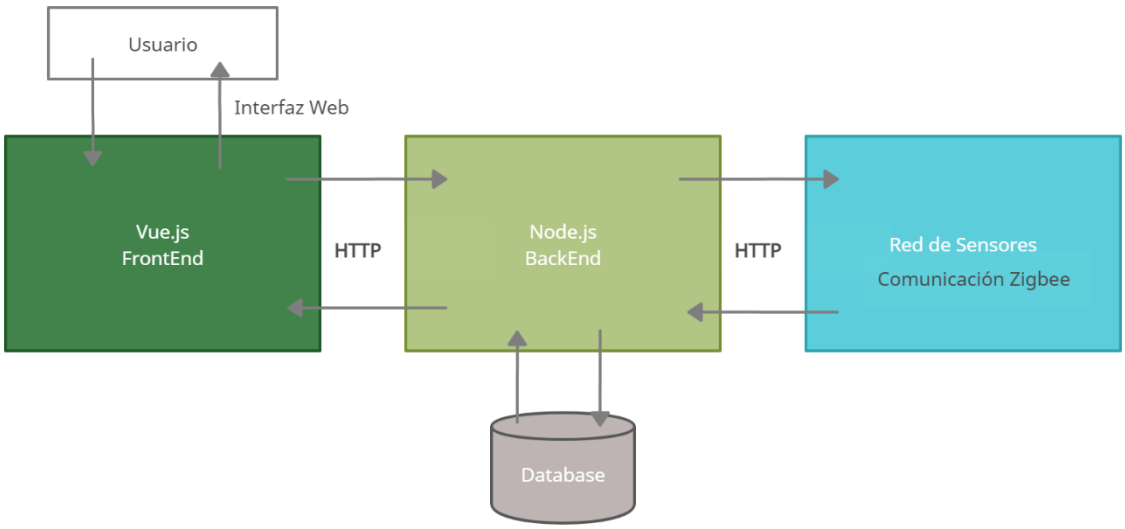


Figura 10 Esquema general del sistema

Todos los bloques de este sistema se comunican entre sí mediante una API REST la cual está en el *Back-End.*

4.2. Hardware

En este trabajo al usar un dispositivo NodeMCU hay que conectar físicamente mediante cableado distintos pines del módulo Zigbee, micrófono y batería.

El patillaje del NodeMCU es el siguiente:

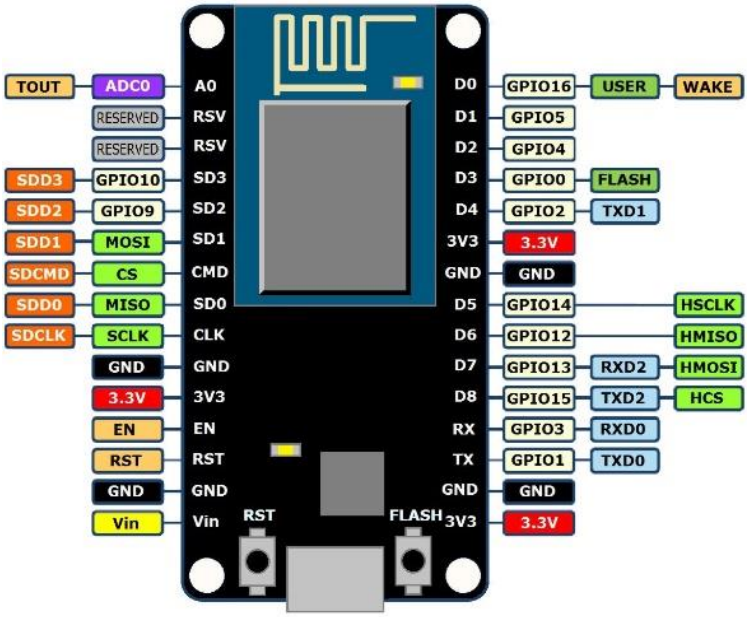


Figura 11 Patillaje NodeMCU

Fuente: https://ssdielect.com/cb/wifi-sistemas-de-desarrollo-esp8266-wifi/768-nodemcu-lolin-ch340.html

El patillaje del módulo Zigbee:

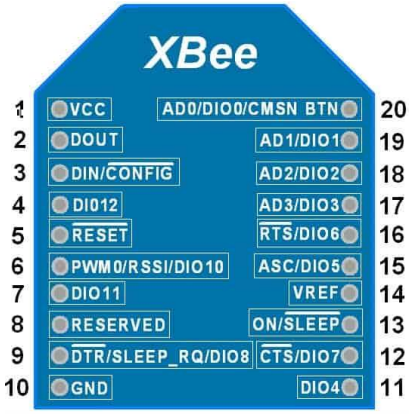


Figura 12 Patillaje módulo Zigbee

Fuente: https://microcontrollerslab.com/xbee-s2c-module-pinout-applications-programming-datasheet/

El patillaje del micrófono se ha visto en el apartado de análisis de materiales y se usará la salida digital.

4.2.1. NodeMCU a módulo Zigbee

Para el conexionado del módulo Zigbee se usarán los pines del módulo 1, 2, 3 y 10. Los pines 1 y 10 para la alimentación del módulo y los pines 2 y 3 para la comunicación serie con el módulo.

El conexionado será el siguiente en todos los nodos:

Xbee	NodeMCU
1	3V
2	D1
3	D2
10	GND

Tabla 2 Conexionado Xbee con NodeMCU

4.2.2. NodeMCU a micrófono

Para el conexionado del módulo micrófono se realizará mediante un pin digital del NodeMCU.

El conexionado será el siguiente:

Micrófono	NodeMCU
G	GND
+	3V
D0	D0

Tabla 3 Conexionado Micrófono con NodeMCU

4.2.3. Imagen del Hardware

Como se han indicado en los apartados anteriores, en la Figura 13 se puede ver un nodo con todas sus conexiones indicadas anteriormente. Incluida una batería que va conectada al pin VIN del NodeMCU V3 el positivo de la batería y al pin G del NodeMCU V3 el negativo de la batería.

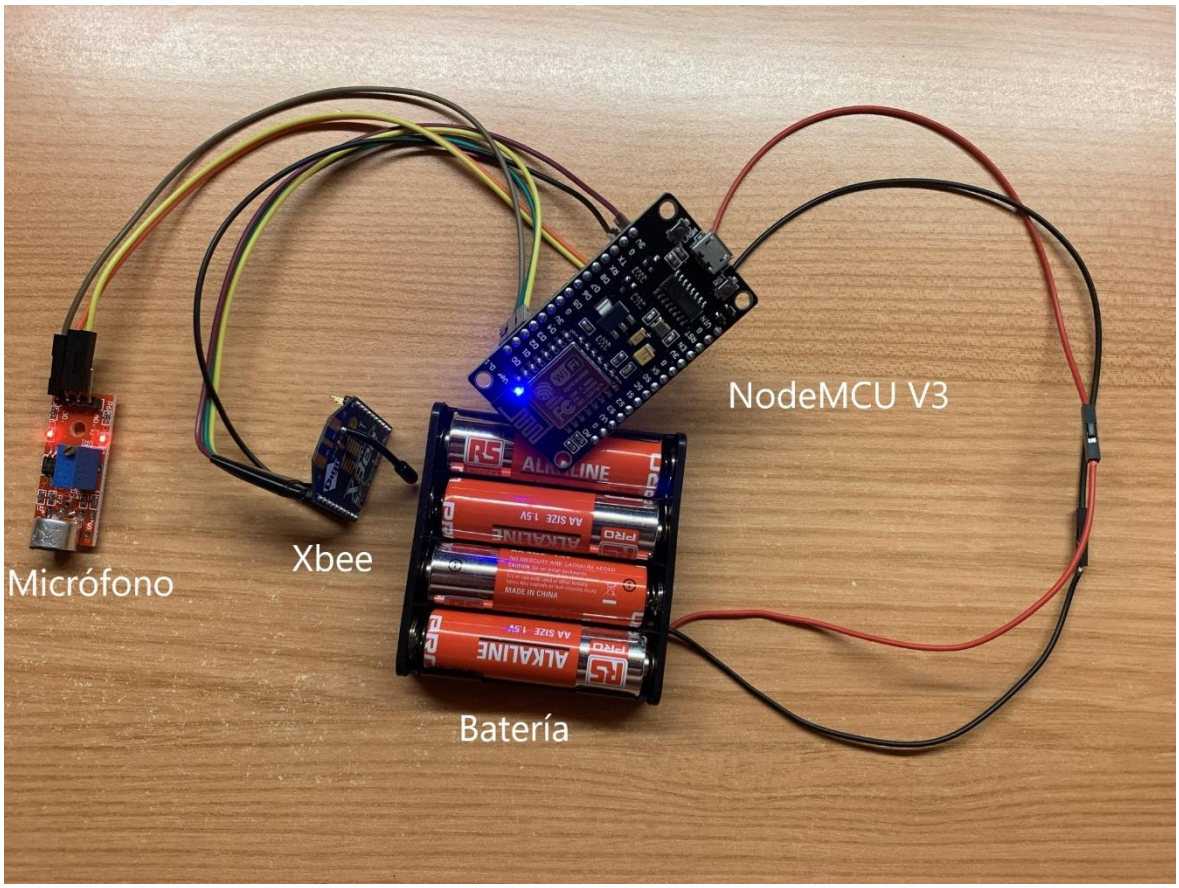


Figura 13 Imagen de un nodo

4.3. Red de sensores

4.3.1. Introducción

En este trabajo se ha realizado el desarrollo de la red de sensores mediante Arduino y Zigbee, para ello se han usado dos softwares principales, el IDE Arduino y X-CTU.

El primero IDE Arduino es en el cual se desarrolla la lógica del nodo sensor, mediante el cual se controla la detección de sonidos mediante un micrófono conectado analógicamente y los datos se envían al nodo central mediante Zigbee conectado mediante conexión serie.

El segundo X-CTU nos sirve para la configuración de Zigbee, conectando al PC el módulo Zigbee mediante el dispositivo adaptador a *micro-usb*.

El nodo principal deberá tener conexión Wi-Fi y recibirá los datos del resto de sensores y los enviará mediante HTTP y la API REST para que queden registrados, así mismo también servirá de nodo sensor, enviando sus propios datos.

4.3.2. Programación en IDE Arduino

Como ya se ha indicado en la introducción habrá dos tipos de nodos: los nodos sensores que se comunican mediante Zigbee, y el nodo sensor central, el cual recibe los datos y los envía al servidor mediante la API REST.

Para acometer esta dualidad se han creado dos programaciones distintas, una para los nodos sensores, la cual solamente se comunica mediante Zigbee con el nodo central y otra para el nodo central, la cual se comunica con el resto de los nodos y también con el servidor mediante WiFi.

Para la configuración inicial de cada uno de los nodos habrá que conectarlos a un PC mediante un cable USB a *microUSB* y cargar el *script* de configuración, cambiando en cada uno de estos nodos la línea de código la cual contiene la definición de la macro (“define”) “sensorId” que hay al comienzo del código ya que es este el que le indica al nodo que id tiene para el registro y envió de datos.

También habrá que configurar en el nodo coordinador el SSID del WiFi al que se conectará mediante la variable “ssid” y la contraseña de este WiFi mediante la variable “password”.

Una vez hechos los cambios mencionados habrá que subir el código a cada uno de los nodos mediante IDE Arduino.

4.3.3. Configuración Zigbee

4.3.3.1. Introducción

Para la configuración de Zigbee se realizará mediante el software X-CTU, para ello habrá que conectar el adaptador de Zigbee mediante un cable USB a *microUSB* y colocar el módulo Zigbee en dicho adaptador, una vez hecho y después de escanear en busca de módulos Zigbee en el programa, nos aparecerá el módulo Zigbee conectado y todas sus opciones.

Figura 14 Xbee Explorer y módulo Xbee

Fuente: https://xbee.cl/xbee-serie-2-configuracion/

Para la configuración de los módulos será igual para todos los módulos de nodos sensores *endpoints*[1]. En este trabajo los nodos *endpoints* van a ser *routers* dentro de la red zigbee, la diferencia entre un nodo *endpoint* y un nodo *router*, es que el nodo *endpoint* se pone en modo inactivo hasta que se activa un pin configurado mediante X-CTU, cuando un módulo Zigbee está en modo inactivo es incapaz de recibir datos y en este sistema es necesario que los nodos *endpoints* reciban datos, por eso se han configurado como *routers* en la red Zigbee.

Cabe destacar que, al ser todos los nodos *routers*, un nodo sensor se puede conectar a otro nodo sensor para enviar los datos al nodo coordinador, esto hace que sea una red en malla con un nodo central.

[1] *Endpoints: dentro de una red Zigbee son los nodos finales, estos envían los datos al nodo Coordinador*

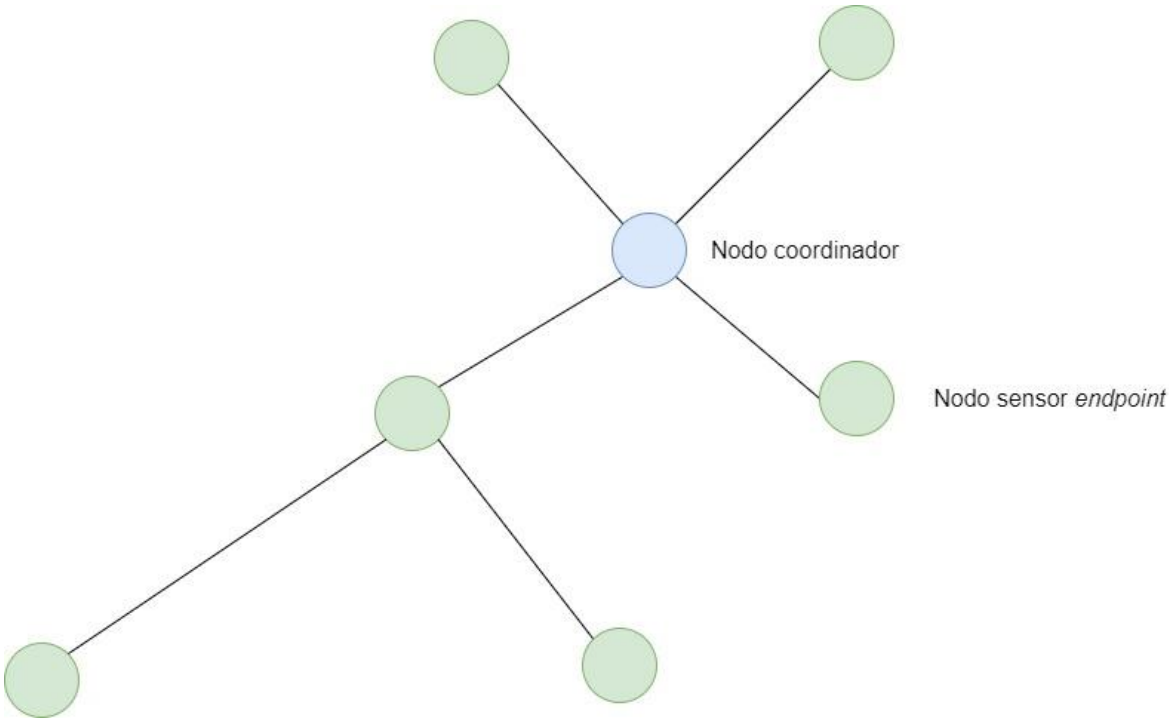


Figura 15 Topología de la red

En la Figura 15 se puede observar la topología de la red, los círculos verdes se corresponden con los nodos sensores *endpoints* y el nodo azul se corresponde al nodo coordinador.

En los siguientes apartados se explicará con más detalle la configuración de dichos módulos.

4.3.3.2. Nodo Coordinador

Este nodo será único en la red y por ello llevará una configuración única para que todos los datos que se envíen en la red lleguen a él y que los datos que él envié lleguen al resto de los nodos.

Una vez conectado Xbee Explorer con el módulo Xbee y abierto X-CTU, seleccionamos el módulo y saldrá a la derecha las opciones del módulo repartidas en varias secciones.

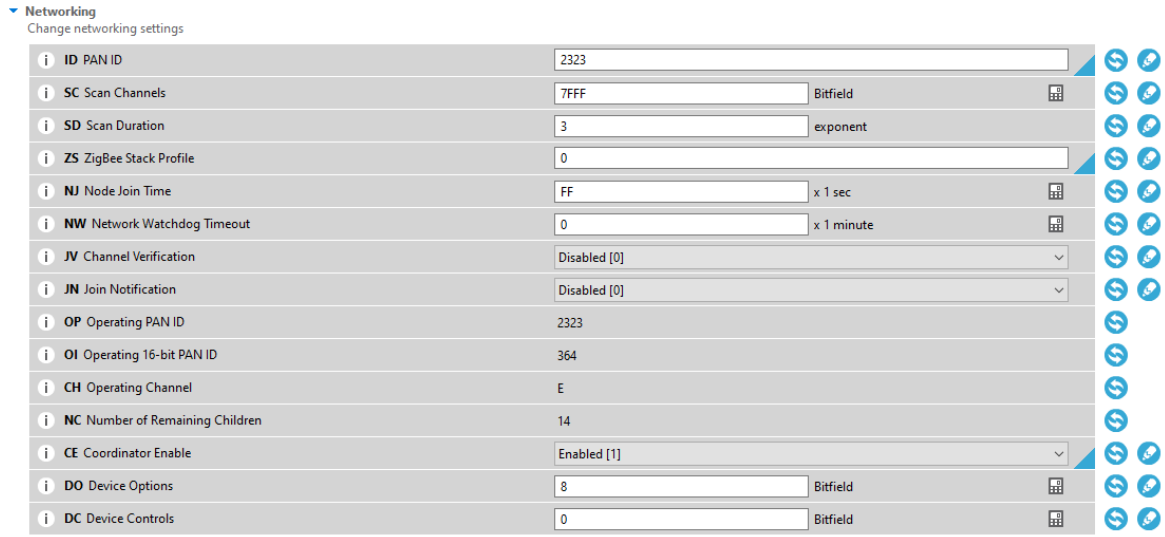


Figura 16 Vista del apartado Networking del nodo Coordinador del software X-CTU

En este apartado las opciones más importantes, y las cuales se tiene que configurar, son:

- PAN ID: Este es el ID de la red, todos los nodos deben tener el mismo ID de red.
- Coordinator Enable: Esta opción tiene que estar en “Enabled” en el nodo coordinador, ya que es la que indica cual va a ser el nodo coordinador de la red.

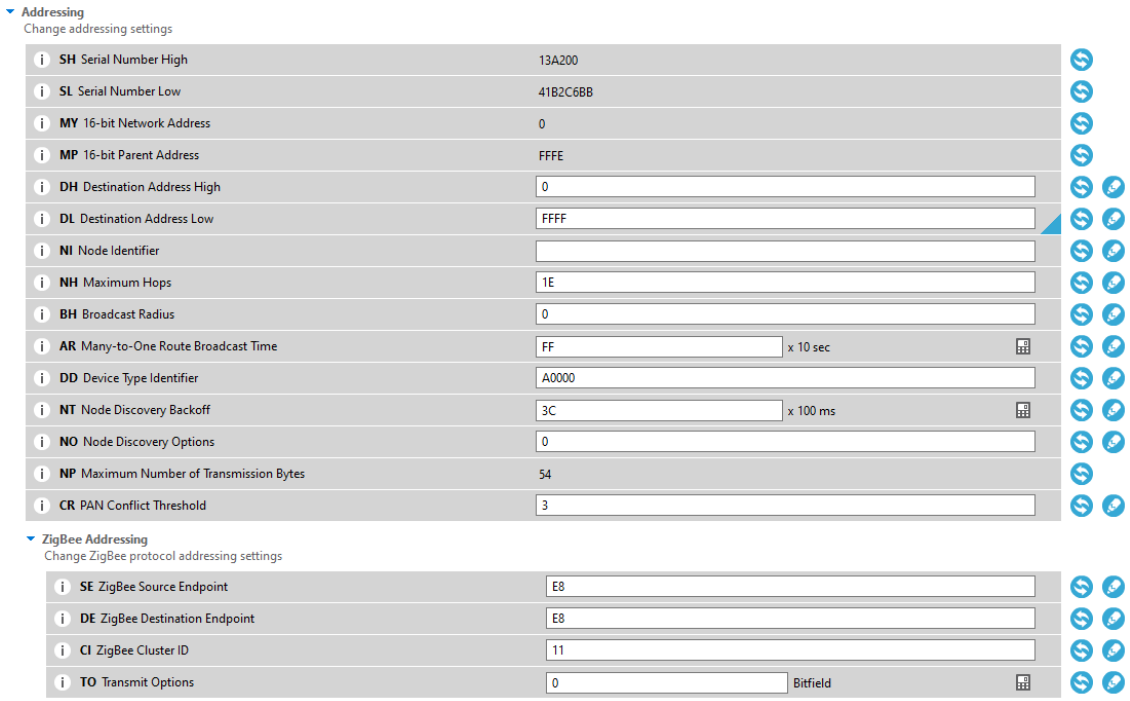


Figura 17 Vista del apartado Addressing del nodo Coordinador del software X-CTU

En este apartado se tienen que apuntar los campos “Serial Number High” y “Serial Number Low”, ya que forman el número de serie del nodo coordinador y habrá que configurarlo en el resto de nodos. En el este caso, como es el nodo coordinador, los campos “Destination Address High” y “Destination Address Low” están configurados para que todos los mensajes que reciba el módulo por la entrada serie se envíen a la dirección de difusión y así lleguen los mensajes al resto de nodos. Los mensajes que se envían por Zigbee están definidos en el apartado 4.3.4. Protocolo de comunicación entre nodos Routers y nodo Coordinador.

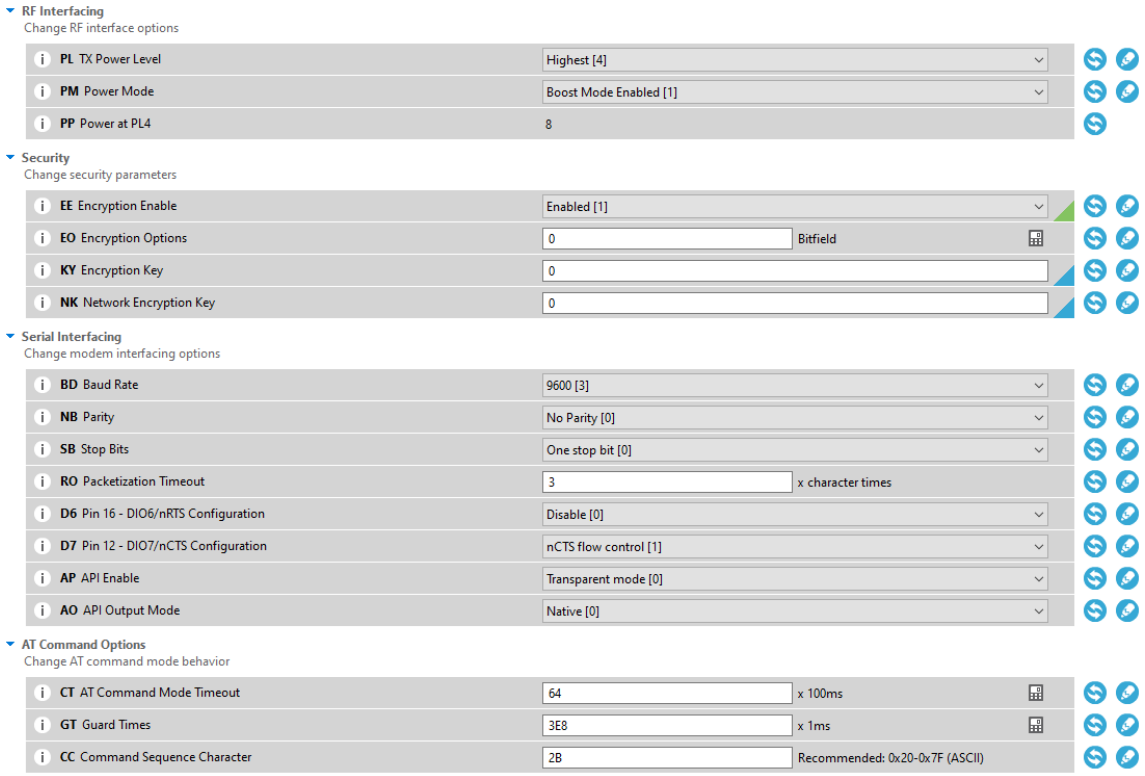


Figura 18 Vista de los apartados de configuración del nodo Coordinador del software X-CTU

Como se puede ver en la imagen anterior, se tiene la posibilidad de configurar la potencia de transmisión de los módulos y añadir seguridad. En el apartado de “Serial Interfacing” se puede ver que la opción “Baud Rate” es 9600, esto se usara en la programación Arduino para la comunicación serie con el módulo. El resto de opciones que ofrece no han sido usadas en este trabajo.

4.3.3.3. Nodo Sensor Router

Los nodos sensor son en realidad nodos *router* en la red Zigbee. Estos módulos enviaran toda la información que les llegue al nodo coordinador.

Se conecta el módulo al PC mediante XBee Explorer y se carga a la configuración igual que en el nodo coordinador.

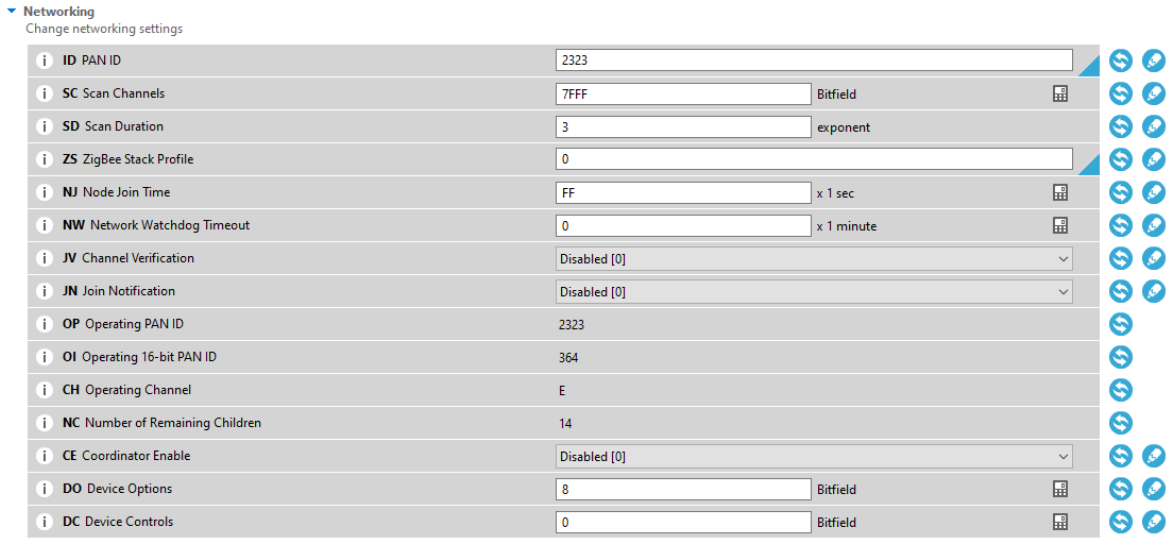


Figura 19 Vista del apartado Networking de los nodos sensores del software X-CTU

En este apartado las opciones más importantes son igual que en el nodo coordinador:

- PAN ID: Se tiene que poner el mismo ID que se ha puesto en el coordinador.
- Coordinator Enable: En este caso como es un nodo *Router* esta opción está marcada como “Disabled”.

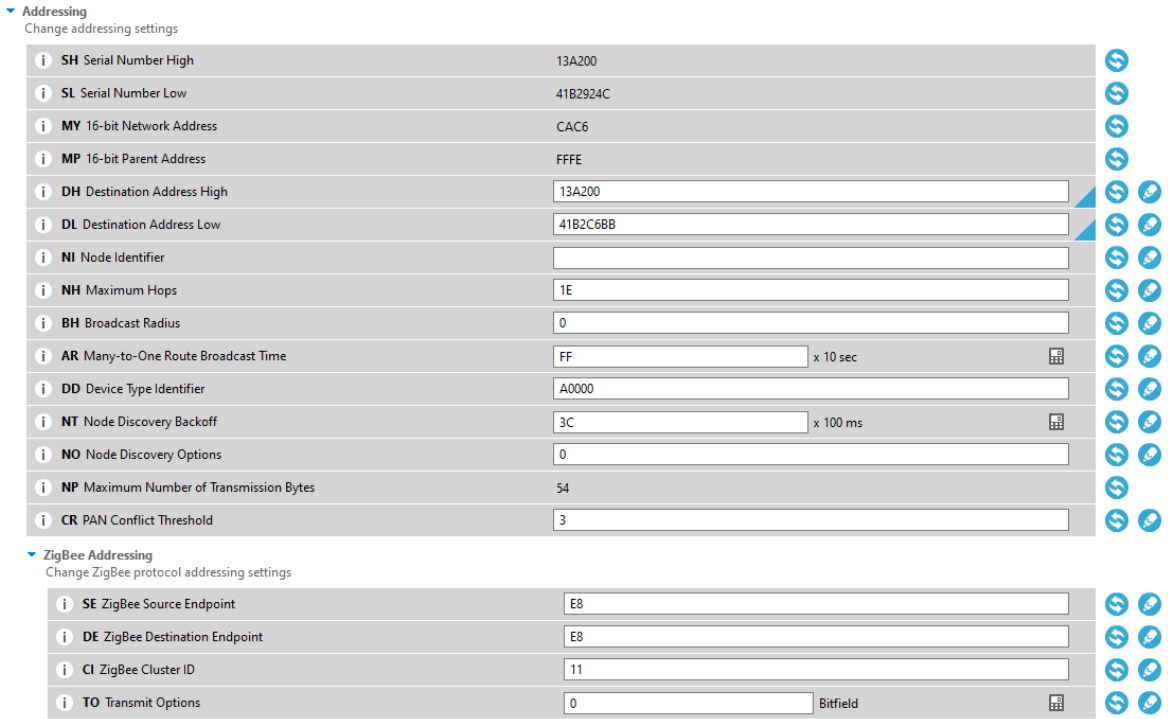


Figura 20 Vista del apartado Addressing de los nodos sensores del software X-CTU

En este apartado en los nodos sensores, se tiene que modificar las siguientes opciones:

- Destination Address High: Aquí se tiene que poner la parte alta del número de serie del módulo del nodo coordinador.
- Destination Address Low: Aquí se tiene que poner la parte baja del número de serie del módulo del nodo coordinador.

En la siguiente imagen se puede ver la misma configuración que hemos visto anteriormente para el módulo del nodo coordinador.

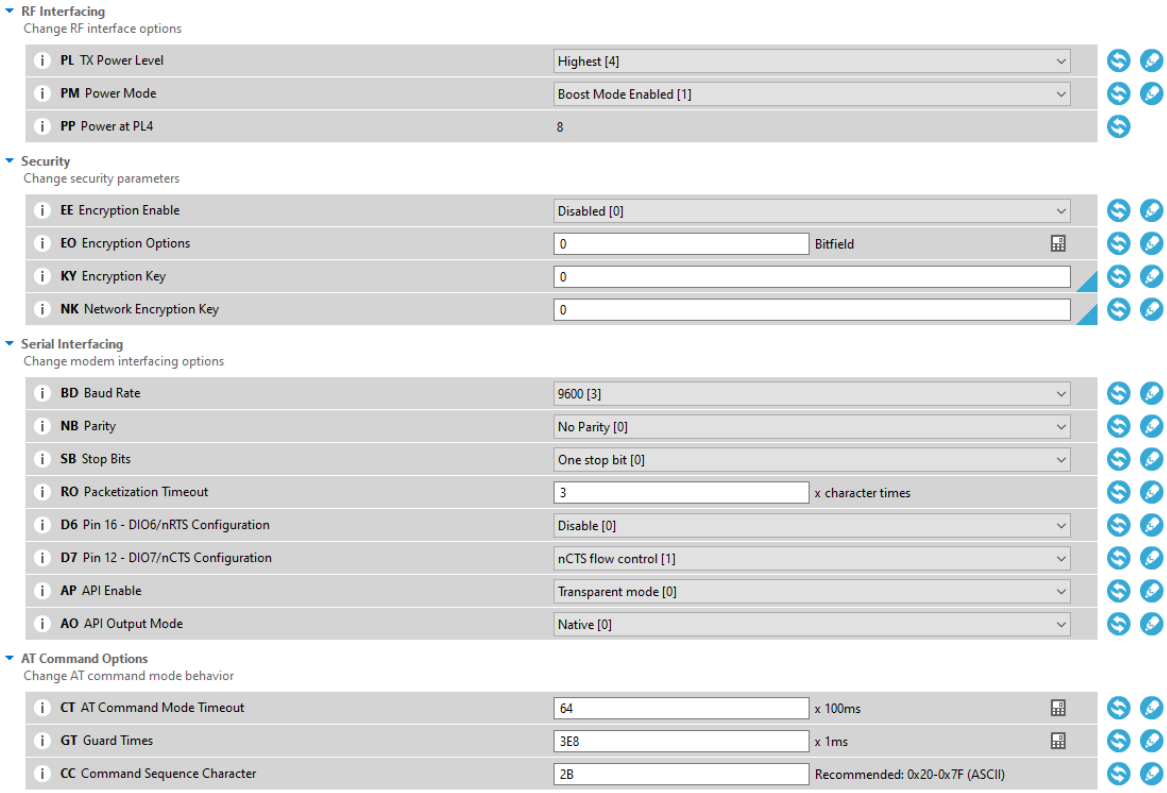


Figura 21 Vista de los apartados de configuración de los nodos sensores del software X-CTU

En el caso de los módulos de los nodos sensores que hacen de *Router* dentro de la red Zigbee, si se tiene que configurar el apartado “Sleep Modes” en concreto la opción:

- Sleep Mode: Esta opción le indica al módulo, si tiene habilitada la opción de dormir. Como en este trabajo se han usado todos los nodos como *Router* menos el principal. Tendrá que estar seleccionada la opción “No Sleep (Router)”

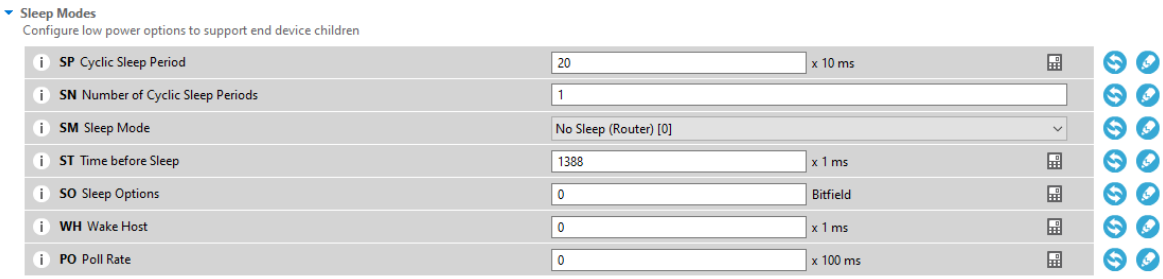


Figura 22 Vista del apartado Sleep Modes de la configuración de los nodos Sensores del software X-CTU

4.3.4. Protocolo de comunicación entre nodos Routers y nodo Coordinador

En este trabajo se usan los módulos Zigbee mediante comunicación serie, entonces todo lo que reciben los módulos mediante la comunicación serie lo envían mediante la red Zigbee. Por este motivo se necesita un pequeño protocolo de comunicación.

Cuando se inicia el coordinador, se envía mediante la red Zigbee el mensaje START, una cadena de texto que llega a todos los nodos sensores. Este es para la configuración del tiempo que el sensor estará sin enviar alarmas desde que envié una alarma. El formato en ABNF del mensaje es el siguiente:

```
START = "min:" SP 1*DIGIT / 2*DIGIT SP CRLF
```

Cuando un nodo sensor se inicia más tarde que el coordinador, el nodo sensor envía a la red el mensaje INIT, que es una cadena de texto con el siguiente formato en ABNF:

```
INIT = "min" CRLF
```

y el coordinador vuelve a enviar a todos los nodos el mismo mensaje mencionado anteriormente.

Cuando un nodo sensor detecta sonido, envía la siguiente cadena de texto para informar al nodo coordinador de que ha detectado una alarma y su id, este es el formato del mensaje en ABNF:

```
ALARM = "id:" 1*DIGIT / 2*DIGIT CRLF
```

4.4. Servicio API REST

4.4.1. Introducción

El servicio API REST está realizado en el entorno de desarrollo Node.js usando el *middleware* Express y cuenta con una base de datos en MySQL con la cual se comunica usando un *Object-Relational Mapping (ORM)*. Un ORM lo que hace es convertir las tablas de una base de datos en entidades, en un lenguaje de programación orientado a objetos.

Esta parte del servicio se encarga de manejar los datos de los sensores y su comunicación, a la vez de la comunicación con el *Front-End* para facilitarle los datos necesarios.

También se encarga del sistema de autenticación del servicio mediante *Json Web Token (JWT)* (28). El funcionamiento de JWT se comentará más adelante en el apartado 4.4.3. Autenticación JSON Web Token.

4.4.2. Esquema general

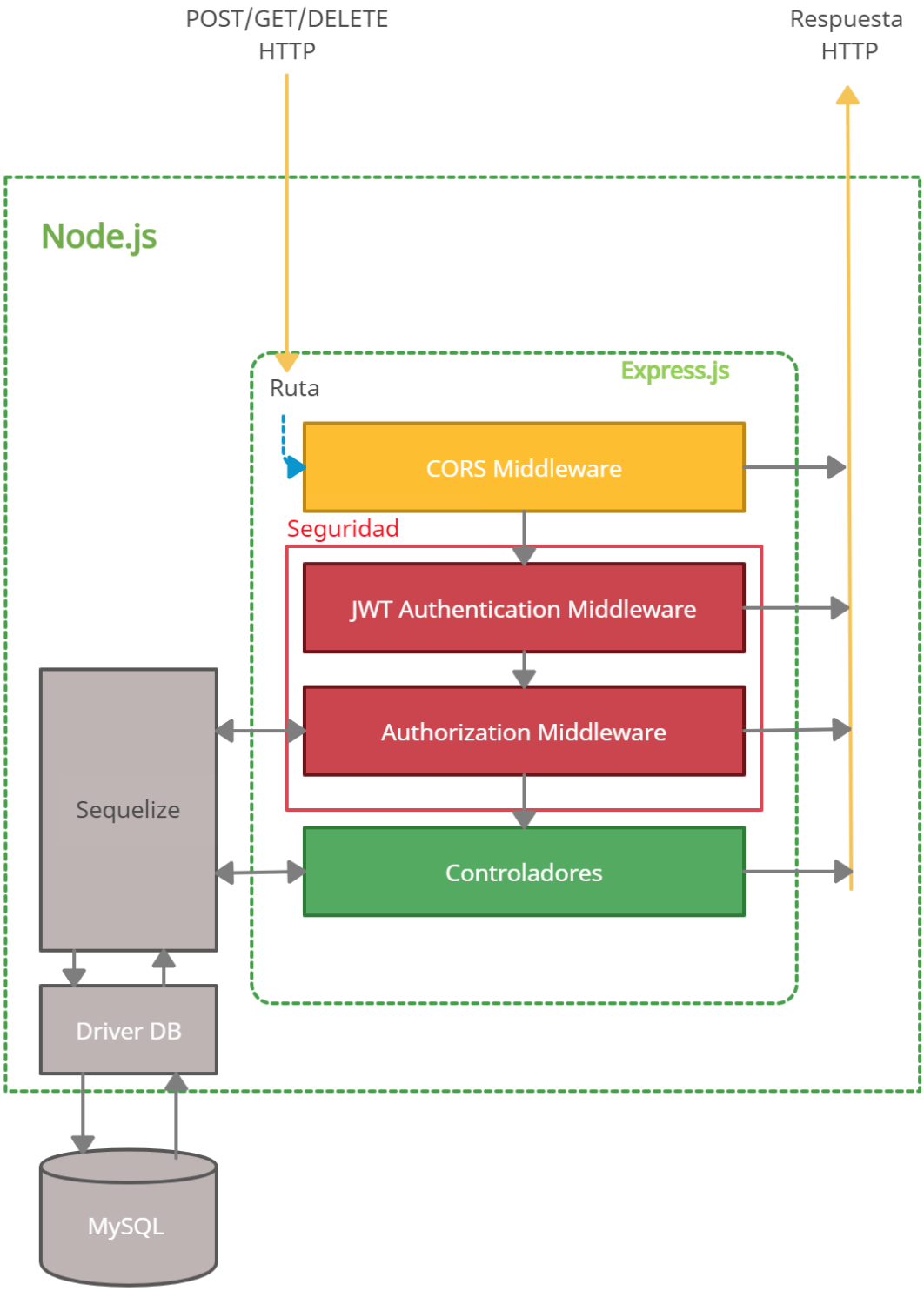


Figura 23 Esquema general de la API REST Node.js Express

4.4.2. Base de datos

Para que el desarrollo de este trabajo sea más rápido y eficaz, se ha realizado usando un ORM.

En este trabajo en concreto se va a usar el ORM Sequelize (29) para NodeJS.

Mediante el uso de Sequelize, las tablas de la base de datos las creara el ORM siguiendo los modelos creados dentro de la implementación del servidor.

Se han declarado los siguientes modelos dentro de Sequelize NodeJS:

- Alarm.model.js
- Config.model.js
- Nodos.model.js
- Role.model.js
- User.model.js

También se ha definido un archivo index.js en el cual se ha definido el conector con la base de datos MySQL y creadas las claves foráneas usadas en la base de datos para relacionar las distintas tablas.

Una vez desplegado el sistema, la base de datos tiene la estructura Entidad-Relación que se puede ver en la Figura 24:

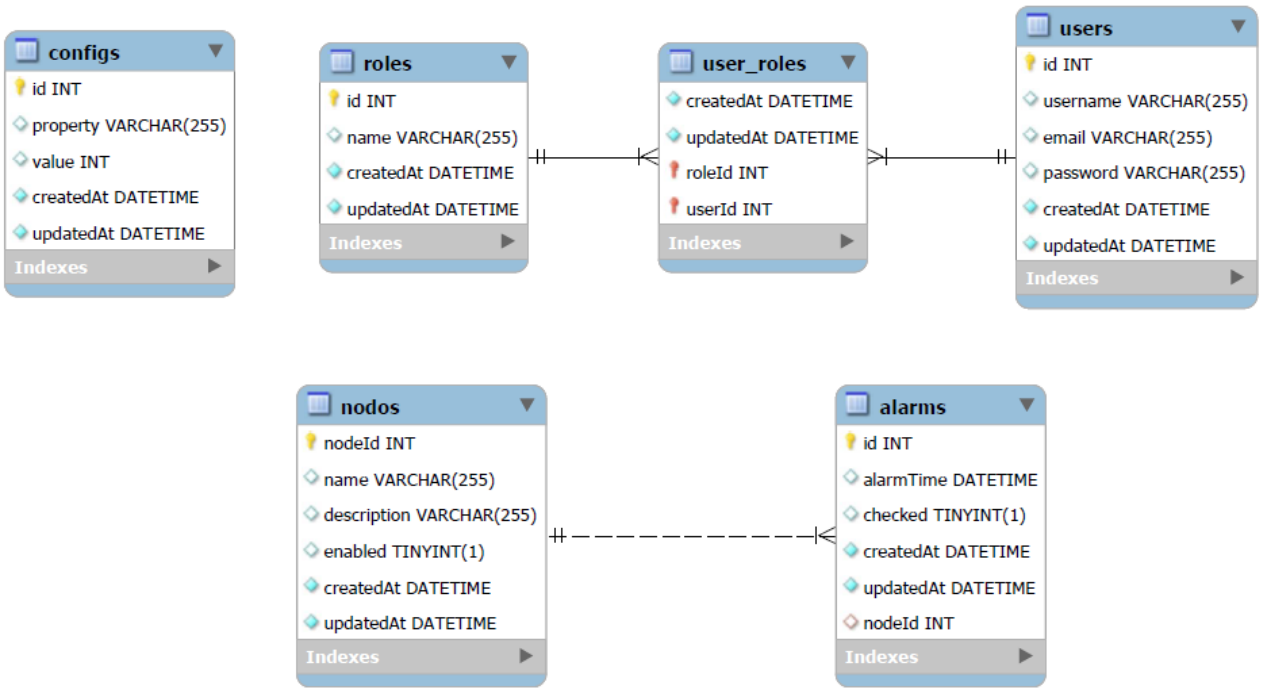


Figura 24 Diagrama Entidad-Relación

4.4.3. Autenticación JSON Web Token

Para el sistema de autenticación de este trabajo se va a usar dos middlewares de NodeJS Express llamado JWT (28) Authentication Middleware y Authorization Middleware.

Un middleware es un bloque de código que está presente entre la petición desde el Front-End y cuando esta petición llega al servidor de Express, en concreto estos middlewares se encargan de:

- JWT Authentication Middleware: Verificación del registro y verificación del token.
- Authorization Middleware: Verificación del rol de usuario con el registro de la base de datos.

Para la implementación de este sistema de autenticación se ha agregado distintas clases JavaScript para su correcto funcionamiento. En el sistema de rutas de NodeJS se ha añadido tres clases las cuales son:

- Auth.routes.js: Se encarga de las peticiones HTTP POST para registro e inicio de sesión.
- User.routes.js y sensor.routes.js: Se encargan de manejar el resto de las peticiones HTTP y su cabecera con el testigo (token).

Desde las rutas se llama al Authorization Middleware y éste sí comprueba si corresponde el rol del usuario según su token, si tiene acceso a la ruta solicitada y si tiene acceso pasa a la parte del controlador la cual es la encargada de la respuesta HTTP.

Todas las peticiones que necesiten una autenticación para que se logren realizar en el servidor por seguridad, tendrán que llevar la cabecera *"x-acces-token"* con el JWT token.

También en este trabajo se han incluido roles para el acceso a distintos apartados según el grado de autorización que tenga cada usuario.

En concreto existen tres roles: Usuario, Moderador y Administrador, lo cuales se comentan brevemente a continuación:

- Los usuarios con el rol Usuario, solo tendrán acceso a la pantalla para ver las alarmas en tiempo real.
- Los usuarios con el rol Moderador podrán agregar sensores y modificar la configuración y datos de los sensores.
- Los usuarios con el rol Administrador podrán hacer lo mismo que los usuarios con el rol Moderador, además de poder cambiar el rol de los usuarios registrados.

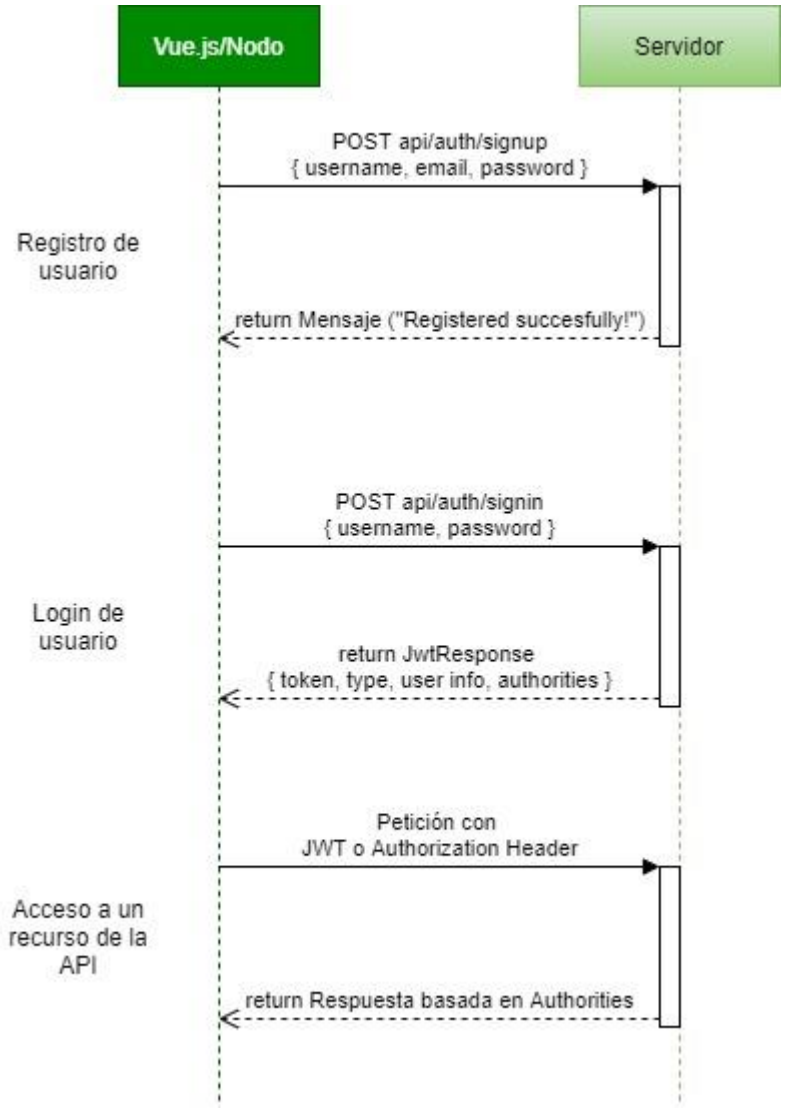


Figura 25 Diagrama secuencial de autenticación

4.4.4. Controladores

Los controladores dentro de NodeJS Express se encargan de devolver las peticiones HTTP y hacer las modificaciones o insertar los datos necesarios en la base de datos según la petición que se haya hecho o devolver los datos solicitados mediante JSON en el cuerpo de la respuesta HTTP.

Para cada ruta definida hay un controlador asociado que se encarga de la lógica de la ruta.

4.4.5. Peticiones API REST

En esta sección del trabajo se van a indicar todas las rutas API REST usadas y los datos que son enviados y recibidos con cada una de las peticiones.

En todas las peticiones habrá una respuesta común si el servidor falla

Código HTTP: 500

Body Json:

{

“message”: “Error ocurrido”

}

4.4.5.1. Registro

Descripción

Mediante esta petición se realiza el registro en la plataforma. Cuando se realiza la petición, se comprueba que el e-mail y/o usuario no se encuentre ya registrado en la plataforma.

Tipo de petición HTTP: POST

Ruta: /auth/signup

Cabecera: Content-Type: application/json

Body:

{

"username": "admin",

"email": "admin@admin.com",

"password": "admin1234",

"roles": ["admin", "moderator", "user"]

}

Esta petición se hará para registrar al primer usuario administrador. Las peticiones de registro desde la plataforma en cuestión no incorporan el array “roles”.

Respuesta OK

Código HTTP: 200

Body Json:

{

"message": "User was registered successfully!"

}

Respuesta Nombre de usuario duplicado

Código HTTP: 400

Body Json:

{

"message": "Failed! Username is already in use!"

}

Respuesta e-Mail duplicado

Código HTTP: 400

Body Json:

{

"message": "Failed! Email is already in use!"

}

4.4.5.2. Log In

Descripción

Esta petición se realiza durante el inicio de sesión y en ella se devuelve la información de usuario.

Tipo de petición HTTP: POST

Ruta: /auth/signin

Cabecera: Content-Type: application/json

Body:

{

"username": "admin",

"password": "admin1234"

}

Respuesta OK

Código HTTP: 200

Body Json:

{
"id": 1,
"username": "admin",
"email": "admin@admin.com",
"roles": [
"ROLE_USER",
"ROLE_MODERATOR",
"ROLE_ADMIN"
],
"accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNjI0NjQxMTYwLCJleHAiOjE2MjQ3Mjc1NjB9.Q_gHKi42mVjtTEhDk0QcKW2JX8fGzPL7biLDCBjgzUg"
}

Respuesta Contraseña Incorrecta

Código HTTP: 401

Body Json:

{
"accessToken": null,
"message": "Invalid Password!"
}

Respuesta Usuario Incorrecto

Código HTTP: 404

Body Json:

{
"message": "User Not found"
}

4.4.5.3. Agregar nodo

Descripción

Petición para agregar un nuevo nodo al sistema.

Tipo de petición HTTP: POST

Ruta: /sensor/addNew

Cabecera: Content-Type: application/json, x-access-token: {json token}

Body:

```
{
    "name": "Nodo 1",
    "description": "Esta es la descripción del Nodo 1",
    "enabled": false
}
```


Respuesta OK

Código HTTP: 201

4.4.5.4. Modificar nodo

Descripción

Petición con la cual se modifican datos de un nodo, el nodeId se usa para encontrar el nodo en la base de datos, ya que el nodeId es único.

Tipo de petición HTTP: POST

Ruta: /sensor/modify

Cabecera: Content-Type: application/json, x-access-token: {json token}

Body:


```
{
    "nodeId": 1,
    "name": "Sensor 1. Modificado",
    "description": "Se ha modificado este sensor",
    "enabled": false
}
```


Respuesta OK

Código HTTP: 200

Respuesta NodeId No encontrado

Descripción

Cuando no se ha encontrado un nodo con el nodeId especificado en la petición, se recibe esta respuesta.

Código HTTP: 404

Body:


```
{
  "message": "Nodo no encontrado"
}
```

4.4.5.5. Borrar nodo

Descripción

Petición en la cual se envía el nodeId del nodo que se quiera borrar en la ruta. Las alarmas asociadas al nodo borrado también son borradas.

Tipo de petición HTTP: DELETE

Cabecera: x-access-token: {json token}

Ruta: /sensor/delete/{nodeId}

Respuesta OK

Código HTTP: 200

Respuesta nodeId no encontrado

Código HTTP: 404

4.4.5.6. Obtener todos los nodos

Descripción

Petición mediante la cual el servidor devuelve todos los nodos que hay en el sistema con sus datos.

Tipo de petición HTTP: GET

Cabecera: x-access-token: {json token}

Ruta: /sensor/all

Respuesta OK

Código HTTP: 200

Body:

```
{
  "nodos": [
    {
      "nodeId": 1,
      "name": "Sensor 1.",
      "description": "Este es el sensor 1",
      "enabled": true,
      "createdAt": "2021-05-31T17:05:46.000Z",
      "updatedAt": "2021-05-31T17:21:34.000Z"
    },
    {
      "nodeId": 2,
      "name": "Sensor 2",
      "description": "Este es el sensor 2",
      "enabled": false,
      "createdAt": "2021-05-31T17:08:27.000Z",
      "updatedAt": "2021-05-31T17:08:27.000Z"
    }
  ]
}
```


4.4.5.7. Obtener nodos habilitados

Descripción

Petición mediante la cual el servidor devuelve todos los nodos en los cuales la propiedad “enabled” está a “true”.

Tipo de petición HTTP: GET

Ruta: /sensor/enabled

Cabecera: x-access-token: {json token}

Respuesta OK

Código HTTP: 200

Body:

```
{
 "nodos": [
  {
   "nodeId": 1,
   "name": "Sensor 1.",
   "description": "Este es el sensor 1",
   "enabled": true,
   "createdAt": "2021-05-31T17:05:46.000Z",
   "updatedAt": "2021-05-31T17:21:34.000Z"
  }
 ]
}
```


4.4.5.8. Nueva alarma

Descripción

Esta petición crea una nueva alarma asociada a un nodeId y guarda la hora en la cual ha ocurrido la alarma.

Tipo de petición HTTP: POST

Ruta: /sensor/create/newEntry

Cabecera: Content-Type: application/json, x-access-token: {json token}

Body:


```
{
        "nodeId": 1,
        "alarmTime": "2021-05-31T19:00:30.996+0000"
}
```


Respuesta OK

Código HTTP: 201

Respuesta nodeId no encontrado

Código HTTP: 404

4.4.5.9. Obtener alarmas que no han sido comprobadas

Descripción

Esta petición devuelve las alarmas en las cuales la propiedad “checked” está a “false”, con todas sus propiedades.

Tipo de petición HTTP: GET

Ruta: /alarms/notcheckeds

Cabecera: x-access-token: {json token}

Respuesta OK

Código HTTP:200

Body:


```
{
  "alarms": [
    {
      "id": 7,
      "alarmTime": "2021-05-31T19:00:30.000Z",
      "checked": false,
      "createdAt": "2021-07-03T11:37:33.000Z",
      "updatedAt": "2021-07-03T11:37:33.000Z",
      "nodeId": 1
    }
  ]
}
```


4.4.5.10. Obtener todas las alarmas

Descripción

Esta petición devuelve todas las alarmas que se hayan generado en el sistema.

Tipo de petición HTTP: GET

Ruta: /alarms/all

Cabecera: x-access-token: {json token}

Respuesta OK

Código HTTP: 200

Body:

```
{
  "alarms": [
    {
      "id": 1,
      "alarmTime": "2021-05-31T19:00:30.000Z",
      "checked": false,
      "createdAt": "2021-07-03T11:37:33.000Z",
      "updatedAt": "2021-07-03T11:37:33.000Z",
      "nodeId": 1
    }
  ]
}
```


4.4.5.11. Marcar alarma como checked

Descripción

Esta petición envía el nodeId de la alarma que se quiera marcar como revisada. El sistema marca como revisadas todas las alarmas de ese nodo.

Tipo de petición HTTP: POST

Ruta: /sensor/update

Cabecera: x-access-token: {json token}

Body:


```
{
        "nodeId": 5
}
```


Respuesta OK

Código HTTP: 200

Respuesta nodeId no encontrado

Código HTTP: 404

4.4.5.12. Obtener configuración

Descripción

Esta petición devuelve la configuración de los nodos.

Tipo de petición HTTP: GET

Ruta: /sensor/config/minutes

Cabecera: x-access-token: {json token}

Respuesta OK

Código HTTP: 200

Body:


```
{
  "property": "minutes",
  "value": 10
}
```


4.4.5.13. Modificar configuración

Descripción

Esta petición modifica la configuración de los nodos.

Tipo de petición HTTP: POST

Ruta: /sensor/config/minutes/update

Cabecera: Content-Type: application/json, x-access-token: {json token}

Body:


```
{
        "minutes": 5
}
```


Respuesta OK

Código HTTP: 200

4.5. Front-End Vue.js

4.5.1. Introducción

Para el desarrollo del *Front-End* de la aplicación se ha usado Vue.js, debido a la facilidad que da para el desarrollo de plataformas web y su velocidad, que es lo que más le caracteriza.

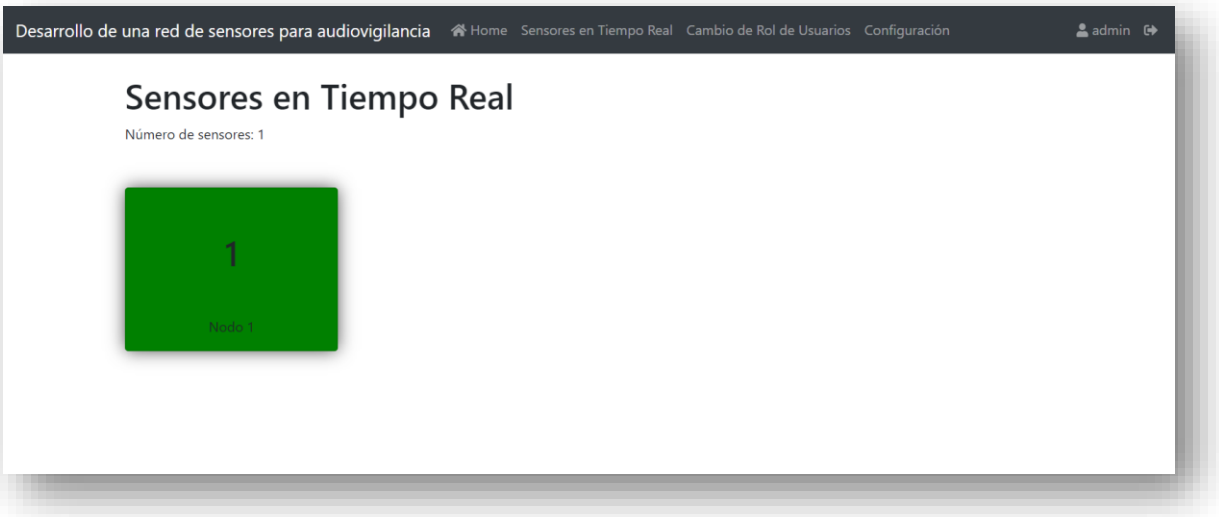


Figura 26 Plataforma web

4.5.2. Esquema general

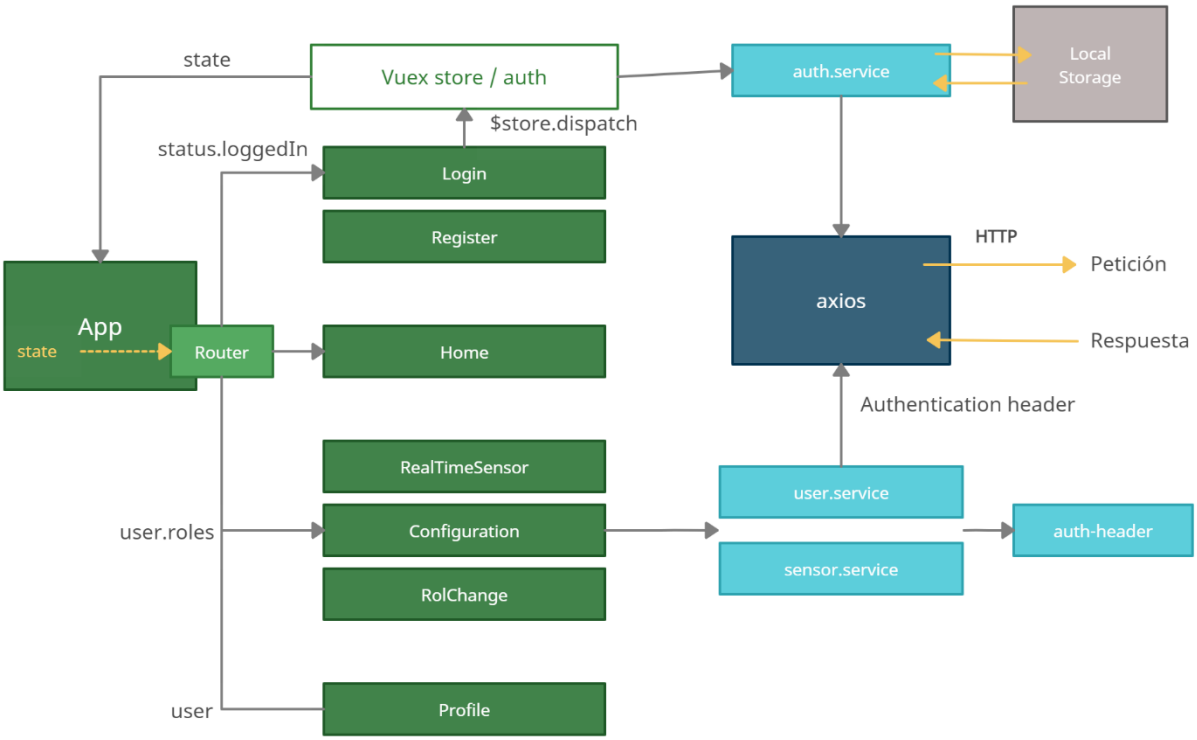


Figura 27 Esquema general de Vue.js

4.5.3. Desarrollo Vue.js

Durante el desarrollo del *Front-End* mediante Vue.js se puede diferenciar cuatro partes principales: Servicios, Vistas, Encaminador (*Router*) y Almacén (*Store*).

Servicios

Los servicios se encargan de las llamadas a la API REST mediante Axios (30), esto es una librería la cual realiza las llamadas HTTP.

En este trabajo se han creado 4 servicios distintos los cuales son:

- Auth-header.js: Se encarga de añadir la cabecera de autenticación a las llamadas HTTP a la API REST
- Auth.service.js: Se encarga de las llamadas HTTP a la API REST para autenticación y registro, también de la lógica para saber si hay una sesión iniciada.
- Node.service.js: Se encarga de las llamadas HTTP a la API REST que tienen que ver con los datos de los nodos.
- User.service.js: Se encarga de las llamadas HTTP a la API REST que tienen que ver con los datos de usuario.

Vistas

Las vistas se encargan de todos los elementos gráficos y de la lógica dinámica de la aplicación web.

Por cada pantalla de la aplicación hay una vista, la cual, según la lógica programada mediante JavaScript, hace llamadas a la API REST mediante los servicios.

En este trabajo las vistas que se han creado son las siguientes:

- Configuration.vue: Vista de configuración de nodos y minutos sin alarma.
- Home.vue: Vista de ventana principal.
- Login.vue: Vista que cuenta con el inicio de sesión de la aplicación.
- Profile.vue: Vista que cuenta con los datos de usuario una vez iniciada la sesión.
- RealTimeSensor.vue: Vista que muestra tarjetas de los sensores y mediante colores informa de cuando un sensor detecta ruido, se actualiza cada 5 segundos cuando se está visualizando.
- Register.vue: Vista que cuenta con el registro de usuario de la aplicación.
- Rol.vue: Vista mediante la cual se puede modificar los permisos de cada usuario registrado en la aplicación.
- App.vue: En esta vista es donde está la navegación y dónde se van cargando el resto de las vistas.

Router

Al ser una aplicación de pantalla única dinámica, hace falta un *router* el cual se encarga de ir llamando a la pantalla que corresponda.

Desde la vista App.vue se llama al resto de vista mediante un menú en la parte superior de la aplicación. Excepto cuando se inicia sesión desde la vista Login.vue que ésta llama a la vista Home.vue si se ha iniciado sesión de manera correcta y si no, vuelve a llamar a la vista Login.vue

Las rutas de la aplicación son las siguientes:

- /: Llama a la vista Home.vue.
- /home: Llama a la vista Home.vue.
- /register: Llama a la vista Register.vue.
- /login: Llama a la vista Login.
- /profile: Llama a la vista Profile.vue.
- /sensor/realtime: Llama a la vista RealTimeSensor.vue.
- /rol: Llama a la vista Rol.vue.
- /configuration: Llama a la vista Configuration.vue.

Store

Se encarga de almacenar los datos de usuario y redirigir según los roles y de gestionar el inicio de sesión de usuario.

Se ha creado un módulo dentro del *store* que se encarga de guardar el estado de inicio de sesión:

- Sesión iniciada y el usuario
- Sesión no iniciada

También cuenta con acciones las cuales son:

- Login: La cual se encarga de controlar el inicio de sesión de un usuario.
- Logout: Se encarga de cuando se realiza el cierre de sesión de un usuario.
- Register: Se encarga de cuando se produce un registro de un nuevo usuario.

También cuenta con mutaciones, las cuales se encargan de modificar los datos guardados en el *store*. Las mutaciones creadas son:

- loginSuccess
- loginFailure
- logout
- registerSuccess
- registerFailure

5. Presupuesto

En este apartado se presenta el presupuesto para la realización del TFG descrito en el presente documento, para ello se van a tener en cuenta el coste del material usado en la implementación de los prototipos de los nodos de la red y el coste de la mano de obra necesaria para el desarrollo del trabajo.

5.1. Prototipos

Cantidad	Dispositivo	Precio unitario	Total
2	NodeMCU V3	4,52€	9,04€
2	Micrófono KY-038	1,23€	2,46€
2	Módulo Xbee S2C	20,53€	41,06
2	Portapilas	1,23€	2,46€
1	Módulo Xbee USB	24,42€	24,42€
		Total	**79,44€**

Tabla 4 Presupuesto materiales

El coste total del material para el desarrollo del prototipo es de 79,44€ sin IVA.

5.2. Mano de obra

El coste de la mano de obra para el desarrollo del trabajo, se presenta en la siguiente tabla.

Horas	Tarea	Precio/hora	Total
52	Especificación de requisitos	30,00 €	1.560€
88	Diseño	30,00€	2.640€
120	Implementación	30,00€	3.600€
40	Pruebas	30,00€	1.200€
300		**30,00€**	**9.000€**

Tabla 5 Horas desarrollo

5.3. Resumen

Capítulo	Total
Prototipos	79,44€
Mano de obra	9.000€
Base imponible	9.079,44€
IVA (21%)	1.906,68€
Total	10.986,12€

Tabla 6 Presupuesto final del trabajo

El presupuesto final del trabajo es de DIEZ MIL NOVECIENTOS OCHENTA Y SEIS EUROS Y DOCE CÉNTIMOS (10.986,12€).

6. Conclusiones

En este trabajo como se ha podido ver, se ha empezado realizando un análisis de las distintas tecnologías posibles para las redes de sensores y se ha determinado el uso de una de ellas. También se han analizado los distintos dispositivos hardware a poder usar y determinado que dispositivo era el más idóneo para este trabajo siguiendo determinados criterios como coste, consumo energético y tecnología.

También se ha analizado los distintos *framework* y tecnologías para el desarrollo de la aplicación de gestión del sistema y se ha realizado su desarrollo.

Una vez hecho todo lo anterior se han realizado las comprobaciones del sistema con todo el sistema desplegado y se ha visto que se han cumplido con todos los objetivos propuestos.

6.1. Líneas de futuro

6.1.1. Implementar Websockets

Se podría implementar websockets en la página web de alarmas para que el aviso de una alarma sea instantáneo una vez ha llegado al servidor. En la implementación de este trabajo, la página web recarga su información cada cinco segundos para mostrar la información actualizada.

6.1.2. Añadir historial de alarmas

Se podría añadir una página que muestre el historial de alarmas ocurridas y que diera la posibilidad de aplicar filtros de búsqueda como por ejemplo las alarmas ocurridas en un nodo o en una determinada fecha.

6.1.3. Añadir configuración para umbral de sonido

Para este cambio habría que cambiar los micrófonos usados por unos micrófonos lineales, cambiar la programación de los sensores, calibrar los micrófonos y añadir un apartado en la web para modificar dicho umbral.

6.1.4. Añadir LoraWan

Cambiar la conexión del nodo coordinador de WiFi a LoraWan para poder desplegar la red de sensores en situaciones alejadas de un punto de conexión WiFi.

Bibliografía

1. **S.L., Libelium Comunicaciones Distribuidas.** Libelium. *Libelium.* [En línea] Libelium Comunicaciones Distribuidas S.L. [Citado el: 12 de Mayo de 2021.] https://www.libelium.com/.

2. *Home Security Alarm Using Wemos D1 And HC-SR501 Sensor Based Telegram Notification.* **Wahyuni, Refni, y otros.** 3, s.l. : Journal of Robotics and Control (JRC), 2021, Vol. 2. 2715-5072.

3. *Illegal Logging Detection Based on Acoustic Surveillance of Forest.* **Mporas, Iosif, y otros.** 20, s.l. : Applied Sciences, 2020, Vol. 10.

4. **INCIBE.** incibe-cert. [En línea] 26 de Junio de 2016. [Citado el: 2021 de Mayo de 2021.] https://www.incibe-cert.es/blog/analizando-bluetooth.

5. **Zone, ADSL.** ADSL Zone. *ADSL Zone.* [En línea] 1 de Marzo de 2021. [Citado el: 25 de Mayo de 2021.] https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/.

6. **Wikipedia.** Wikipedia. *Wikipedia.* [En línea] [Citado el: 25 de Mayo de 2021.] https://es.wikipedia.org/wiki/Zigbee.

7. **CATSENSORS.** CATSENSORS Sensores e Intrumentacion Industrial . [En línea] [Citado el: 25 de Mayo de 2021.] https://www.catsensors.com/es/lorawan/tecnologia-lora-y-lorawan.

8. **Wikipedia.** Wikipedia. [En línea] [Citado el: 25 de Mayo de 2021.] https://es.wikipedia.org/wiki/LoRaWAN.

9. **E-Marmolejo, Rubén.** HETPRO. *HETPRO.* [En línea] [Citado el: 24 de Mayo de 2021.] https://hetpro-store.com/TUTORIALES/microcontrolador/.

10. **Foundation, Raspberry Pi.** Raspberry Pi. *Raspberry Pi.* [En línea] [Citado el: 24 de Mayo de 2021.] https://www.raspberrypi.org/products/raspberry-pi-zero-w/.

11. **Rubén, Velasco.** Redes Zone. *Redes Zone .* [En línea] Grupo ADSL Zone, 02 de Marzo de 2017. [Citado el: 24 de Mayo de 2021.] https://www.redeszone.net/2017/03/02/energia-consumida-raspberry-pi/.

12. **Arduino.** Arduino. *Arduino.* [En línea] [Citado el: 24 de Mayo de 2021.] https://store.arduino.cc/arduino-uno-rev3.

13. **Igor.** Gadget Makers Blog. *Gadget Makers Blog.* [En línea] 2 de Septiembre de 2013. [Citado el: 24 de Mayo de 2021.] http://www.gadgetmakersblog.com/arduino-power-consumption/.

14. **BricoGeek.** *BricoGeek.* [En línea] [Citado el: 24 de Mayo de 2021.] https://tienda.bricogeek.com/wifi/1033-nodemcu-v3-wifi-esp8266-ch340.html.

15. Llamas, Luis. COMPARATIVA ESP8266 FRENTE A ESP32, LOS SOC DE ESPRESSIF PARA IOT. [En línea] [Citado el: 24 de Mayo de 2021.] https://www.luisllamas.es/comparativa-esp8266-esp32/.

16. —. Luis Llamas. *Luis Llamas.* [En línea] 9 de Noviembre de 2016. [Citado el: 26 de Mayo de 2021.] https://www.luisllamas.es/detectar-sonido-con-arduino-y-microfono-ky-038/.

17. BIGTRONICA. BIGTRONICA. [En línea] [Citado el: 26 de Mayo de 2021.] https://www.bigtronica.com/sensores/sonido/121-tarjeta-sensor-de-sonido-mini-5053212001216.html.

18. Get Electronics. Arduino Learning. *Arduino Learnign.* [En línea] [Citado el: 06 de 06 de 2021.] http://arduinolearning.com/code/arduino-and-max4466-electret-module-example.php.

19. Oracle Corporation. MySQL. *MySQL.* [En línea] [Citado el: 17 de Junio de 2021.] https://www.mysql.com/.

20. MariaDB Foundation. MariaDB. [En línea] [Citado el: 17 de Junio de 2021.] https://mariadb.org/.

21. MongoDB, Inc. Mongo DB. [En línea] [Citado el: 17 de Junio de 2021.] https://www.mongodb.com/es.

22. Symfony SAS. Symfony. [En línea] [Citado el: 17 de Junio de 2021.] https://symfony.com/.

23. OpenJS Foundation. Node.js. [En línea] [Citado el: 17 de Junio de 2021.] https://nodejs.org/es/.

24. StrongLoop, Inc. Express. Infraestructura web rápida, minimalista y flexible para Node.js. [En línea] [Citado el: 12 de Junio de 2021.] https://expressjs.com/es/.

25. Google . AngularJS. [En línea] [Citado el: 17 de Junio de 2021.] https://angularjs.org/.

26. Evan You. Vue.js. [En línea] [Citado el: 17 de Junio de 2021.] https://vuejs.org/.

27. Facebook Inc. React. [En línea] [Citado el: 17 de Junio de 2021.] https://es.reactjs.org/.

28. Auth0. Npm jsonwebtoken. [En línea] [Citado el: 20 de Junio de 2021.] https://www.npmjs.com/package/jsonwebtoken.

29. Sequelize. Sequelize ORM. [En línea] [Citado el: 20 de Junio de 2021.] https://sequelize.org/.

30. Axios. Promise based HTTP client for the browser and node.js. [En línea] [Citado el: 20 de Junio de 2021.] https://github.com/axios/axios.

Anexo 1: Despliegue del sistema.

El despliegue del sistema se va a realizar en Ubuntu 20.04 y se va a realizar desde el momento en el cual se tiene acceso al sistema.

Todo el despliegue se va a realizar mediante el terminal de Ubuntu y se va a usar el repositorio de Github para la descarga del proyecto.

Se va a realizar el despliegue de la API REST y del *front-end* en el mismo servidor.

1. Instalar node.js

Primero se tiene que ejecutar el siguiente comando para actualizar el índice de paquetes locales:

```
user@juandi:~$ sudo apt update
[sudo] password for user:
Hit:1 http://es.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:4 http://es.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Fetched 328 kB in 1s (479 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
67 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Después de esto se tiene que ejecutar el siguiente comando para instalar node.js:

```
user@juandi:~$ sudo apt install nodejs
```

Una vez completada la instalación de node.js, es necesario instalar también el gestor de repositorios de node.js, npm, esto se realiza ejecutando el siguiente comando:

```
user@juandi:~$ sudo apt install npm
```

2. Instalar y configurar MySQL

Ahora se va a proceder a la instalación y configuración de la base de datos MySQL.

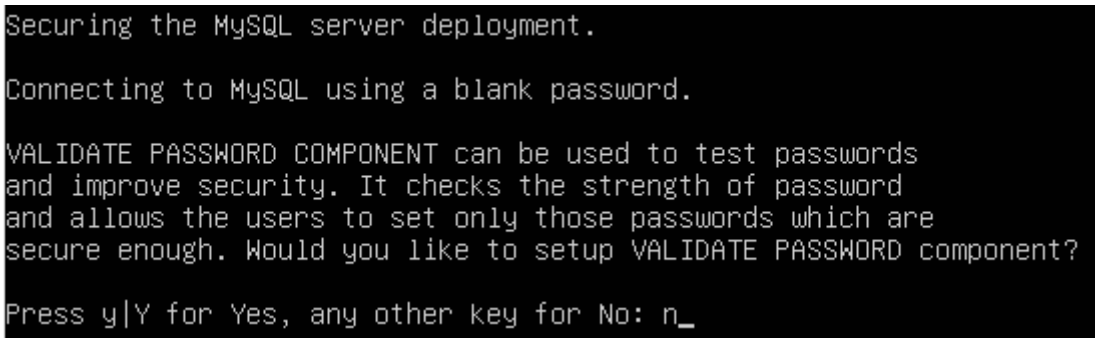
Para la instalación se realiza mediante el terminal con los siguientes comandos:

```
user@juandi:~$ sudo apt install mysql-server
```

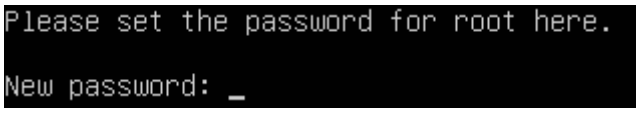
Una vez realizada la instalación de MySQL mediante el comando anterior, se tiene que ejecutar la secuencia de comando de configuración de MySQL mediante el siguiente comando:

```
user@juandi:~$ sudo mysql_secure_installation
```

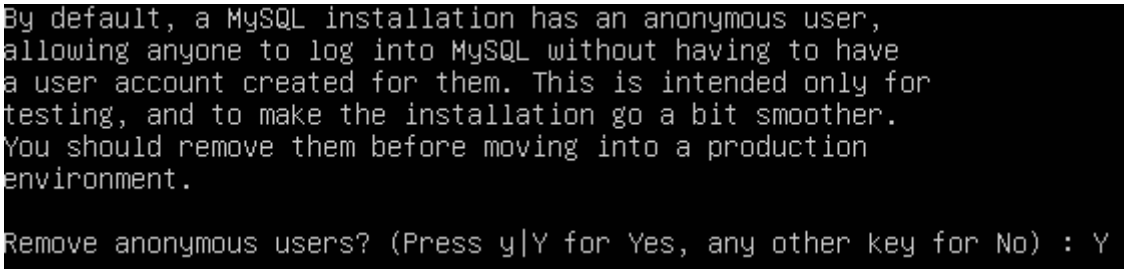
Este comando sirve para proteger la instalación, y va a ir preguntando configuraciones, lo primero pregunta si se quiere que se valide la contraseña elegida para el usuario root. En este caso se puede obviar esto y seleccionar No.



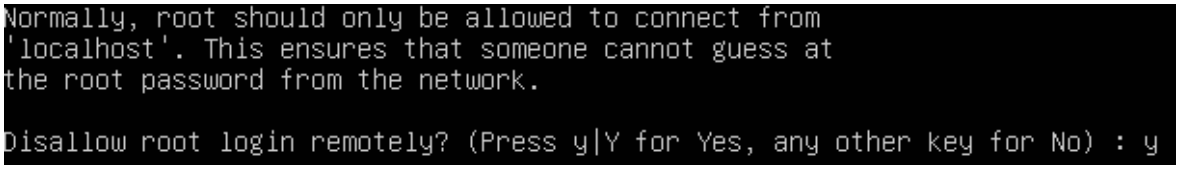
Ahora el sistema pedirá una contraseña para el usuario root de la base de datos.



Después de poner la contraseña volverá a pedir confirmación de la contraseña y preguntará que si se quiere deshabilitar el acceso mediante un usuario anónimo, como la aplicación tendrá su propio usuario, se selecciona sí.



Ahora pregunta si se quiere que el usuario root pueda acceder desde fuera de localhost, por seguridad se selecciona la opción no, ya que a la base de datos solo podrá acceder la aplicación y las modificaciones que se tengan que hacer, se harán desde la consola directamente desde el servidor.



Pregunta que, si se quiere borrar el acceso y las bases de datos test, se selecciona sí.

Por último, pregunta que, si se quiere recargar las bases de datos y acceso a ellas, se marca si:

```
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y_
```

Después de esto se tiene que acceder a la consola de MySQL y crear la base de datos de la aplicación y el usuario que va a usar la aplicación para acceder a ella.

Para acceder a la consola de MySQL se usa el siguiente comando:

```
user@juandi:~$ sudo mysql -u root -p_
```

Pedirá la contraseña del usuario *root* y saldrá la siguiente pantalla:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.25-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Una vez dentro de la consola de MySQL, se ejecuta el siguiente comando para crear la base de datos de la aplicación:

```
mysql> CREATE DATABASE alarmaporsonido;
Query OK, 1 row affected (0.02 sec)
```

Ahora se crea el usuario que va a usar la aplicación para el acceso a la base de datos y darle permisos.

```
mysql> CREATE USER 'alarmaporsonido'@'localhost' IDENTIFIED BY '1+Prueba+Del+TFG';
Query OK, 0 rows affected (0.02 sec)
```

Para darle todos los permisos para la base de datos de la aplicación, se ejecuta el siguiente comando:

```
mysql> GRANT ALL PRIVILEGES ON alarmaporsonido.* TO 'alarmaporsonido'@'localhost' WITH GRANT OPTION;

Query OK, 0 rows affected (0.05 sec)
```

Una vez hecho esto, ya se tiene creado el usuario y la base de datos para la aplicación.

3. Instalar Git y descargar proyecto de GitHub

Ahora se procede a la instalación de Git para la descarga del proyecto desde GitHub. Para ello se ejecuta el siguiente comando:

```
user@juandi:~$ sudo apt-get install git -y
```

Se ejecuta los siguientes comandos para configurar el usuario y correo electrónico de git:

```
user@juandi:~$ git config --global user.name "Juan Diego"
user@juandi:~$ git config --global user.email "jdgb0002@red.ujaen.es"
```

Una vez hecho esto vamos a realizar un clone del repositorio mediante el siguiente comando:

```
user@juandi:~$ git clone https://juandybarranco@github.com/juandybarranco/alarmaporsonido.com.local.
git
Cloning into 'alarmaporsonido.com.local'...
remote: Enumerating objects: 376, done.
remote: Counting objects: 100% (376/376), done.
remote: Compressing objects: 100% (254/254), done.
remote: Total 376 (delta 222), reused 261 (delta 107), pack-reused 0
Receiving objects: 100% (376/376), 363.34 KiB | 1.90 MiB/s, done.
Resolving deltas: 100% (222/222), done.
```

4. Despliegue de la API REST y Vue.js

Ahora se procede al despliegue de las aplicaciones, primero se tiene que descargar los paquetes necesarios para cada aplicación mediante los siguientes comandos:

```
user@juandi:~$ cd alarmaporsonido.com.local/
user@juandi:~/alarmaporsonido.com.local$ cd server/
user@juandi:~/alarmaporsonido.com.local/server$ npm install_
```

Después de esto tenemos que hacer lo mismo para la parte de vue.js:

```
user@juandi:~/alarmaporsonido.com.local/server$ cd ..
user@juandi:~/alarmaporsonido.com.local$ cd client/
user@juandi:~/alarmaporsonido.com.local/client$ npm install
```

Una vez hecho esto, ya se tiene todas las dependencias descargadas, ahora se tiene que modificar el archivo app.js de la carpeta */server*, para que genere las tablas de la base de datos.

Se usa el siguiente comando para editar el archivo:

```
user@juandi:~/alarmaporsonido.com.local/server$ sudo nano app.js
```

Y se cambia lo siguiente:

```
  GNU nano 4.8                                  app.js
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");

const app = express();
const http = require('http').Server(app);
const io = require('socket.io')(http);

var corsOptions = {
    origin: "*"
};

app.use(cors(corsOptions));

// parse requests of content-type - application/json
app.use(bodyParser.json());

// parse requests of content-type - application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({extended: true}));

//database
const db = require("./models");
const Role = db.role;
const Config = db.config;

db.sequelize.sync();
//Lo siguiente borra las tablas si existen y resincronizan las tablas
//Solo ejecutar la primera vez, ya que borra los usuarios
/*db.sequelize.sync({force: true}).then(() => {
    console.log('Drop an Resync Database with { force: true}');
    initial();
});*/


^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line M-E Redo
```

Por esto:

```
  GNU nano 4.8                                  app.js
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");

const app = express();
const http = require('http').Server(app);
const io = require('socket.io')(http);

var corsOptions = {
    origin: "*"
};

app.use(cors(corsOptions));

// parse requests of content-type - application/json
app.use(bodyParser.json());

// parse requests of content-type - application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({extended: true}));

//database
const db = require("./models");
const Role = db.role;
const Config = db.config;

db.sequelize.sync();
//Lo siguiente borra las tablas si existen y resincronizan las tablas
//Solo ejecutar la primera vez, ya que borra los usuarios
db.sequelize.sync({force: true}).then(() => {
    console.log('Drop an Resync Database with { force: true}');
    initial();
});

                          [ Read 75 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line M-E Redo
```

Una vez realizado esto se ejecuta el siguiente comando:

```
user@juandi:~/alarmaporsonido.com.local/server$ sudo node app.js _
```

Y una vez haya cargado se pulsa Ctrl+C para salir de la ejecución de la API REST.

Y se vuelve a modificar el archivo anterior a como estaba originalmente. Esto ha creado las tablas y contenido inicial en la base de datos del servidor.

Ahora se va a ejecutar en el servidor la API REST en segundo plano, para ello desde la carpeta *server* del proyecto se va a utilizar el siguiente comando:

```
user@juandi:~/alarmaporsonido.com.local/server$ forever start app.js
```

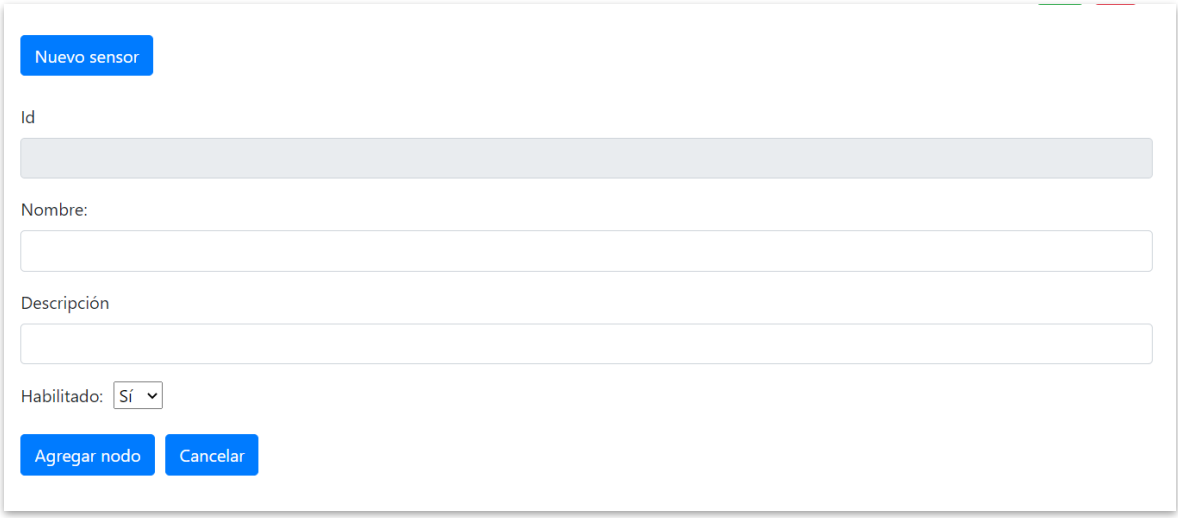
Una vez ejecutado se pone los siguientes comandos:

```
user@juandi:~/alarmaporsonido.com.local/server$ cd ..
user@juandi:~/alarmaporsonido.com.local$ cd client
user@juandi:~/alarmaporsonido.com.local/client$ npm run serve
```

Desde este momento ya estará desplegada la plataforma y será accesible desde cualquier ordenador conectado a la misma red, una vez ejecutado el comando anterior pondrá la dirección IP desde la cual se puede acceder a la plataforma.

5. Añadir nodos

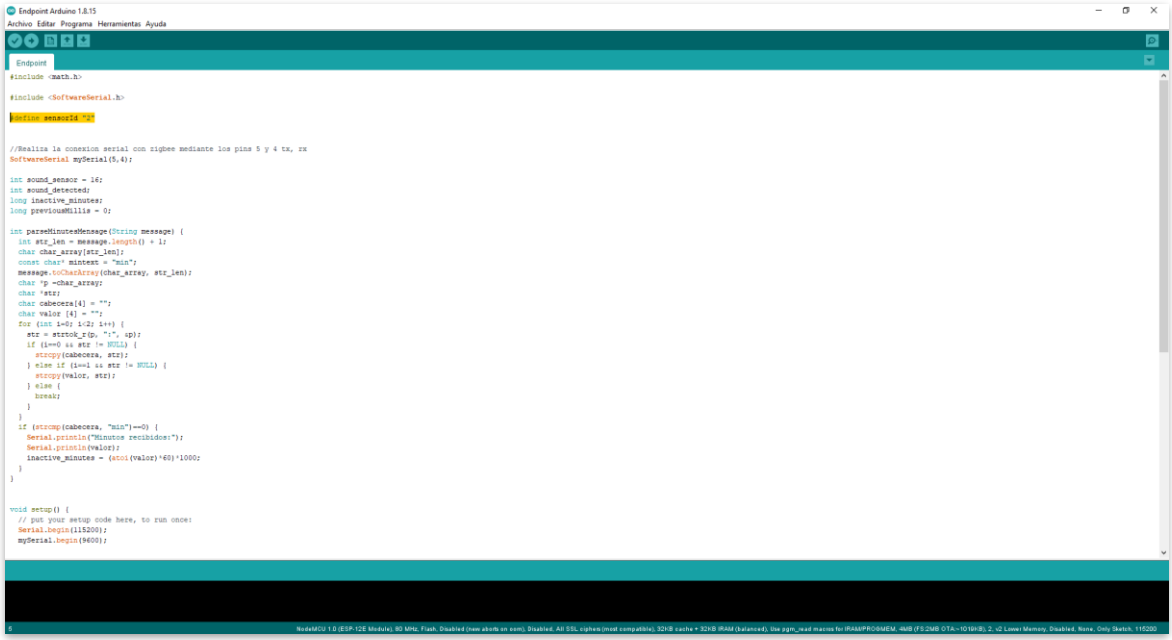
Para añadir un nuevo nodo, primero hay que conectarse a la plataforma con una cuenta que al menos sea Moderador, una vez dentro vamos a la ventana Configuración y le damos a Nuevo Sensor, rellenamos lo que lo que nos pide y pinchamos en Agregar Nodo.



Una vez hecho esto, se ve que id de nodo que le ha asignado el sistema. Ahora abrimos el programa Arduino IDE y cargamos el archivo .ine para el nodo y conectamos el NodeMCU al PC.

Se modifica el sensorId poniendo el que se ha asignado y se pulsa el botón de la flecha.

En la imagen inferior se puede ver una vista del programa Arduino IDE, señalado en amarillo donde se modifica el sensorId.

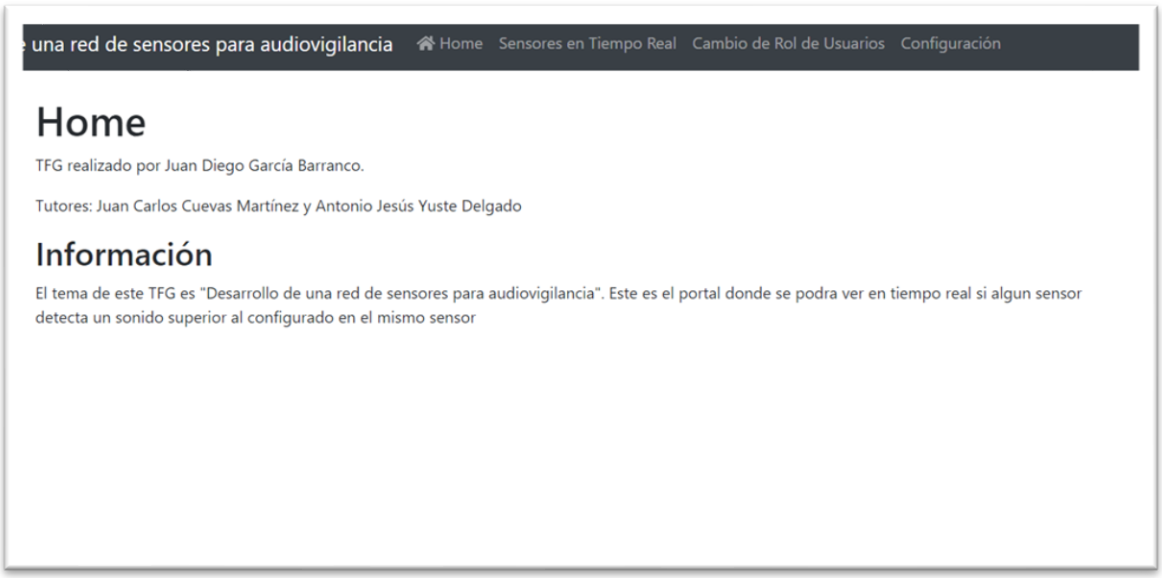


Anexo 2: Manual de usuario de la plataforma web

En este anexo se va a explicar el funcionamiento de la plataforma web, se va a contar con que ya se tenga un usuario administrador creado durante el despliegue.

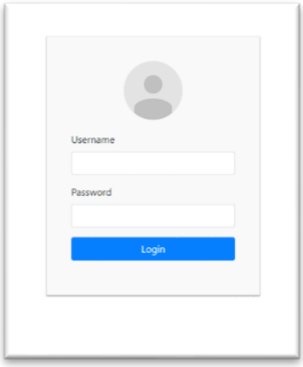
1. Acceso a la plataforma

Cuando se accede a la plataforma, se accede a la pantalla home en la cual hay información de la plataforma, en la parte superior derecha se encuentran dos enlaces, uno para el registro de usuarios y otro para el acceso de usuarios ya registrados.



2. Inicio de sesión de usuario ya registrado

Cuando accedemos a la página de inicio de sesión podemos ver un formulario en el cual se nos pide el usuario y la contraseña.



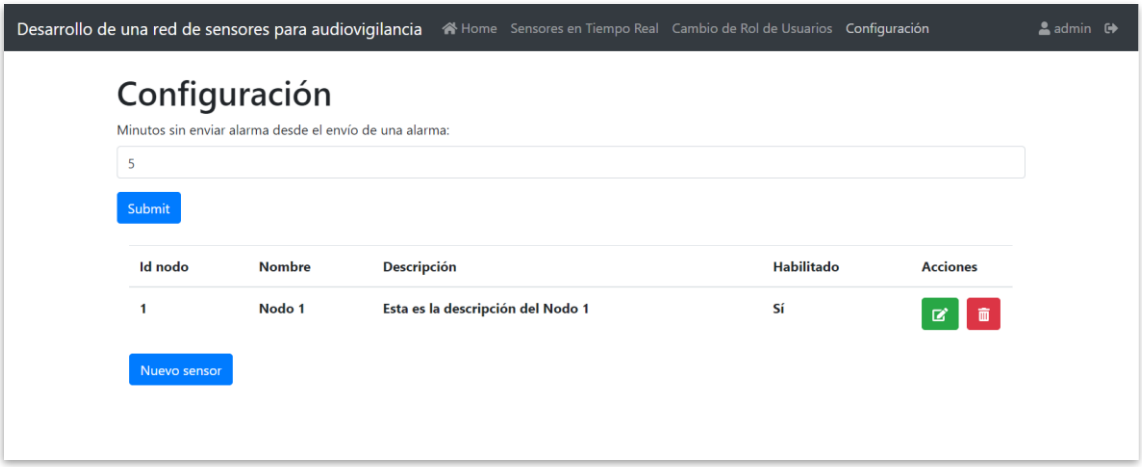
3. Sensores en tiempo real

Una vez se ha accedido con un usuario ya registrado, se puede ver la página en la cual se informa en tiempo real del estado de los sensores y si alguno de ellos ha superado el umbral de sonido. En la siguiente imagen se puede ver que el sensor está en verde, por lo tanto, no ha saltado la alarma.

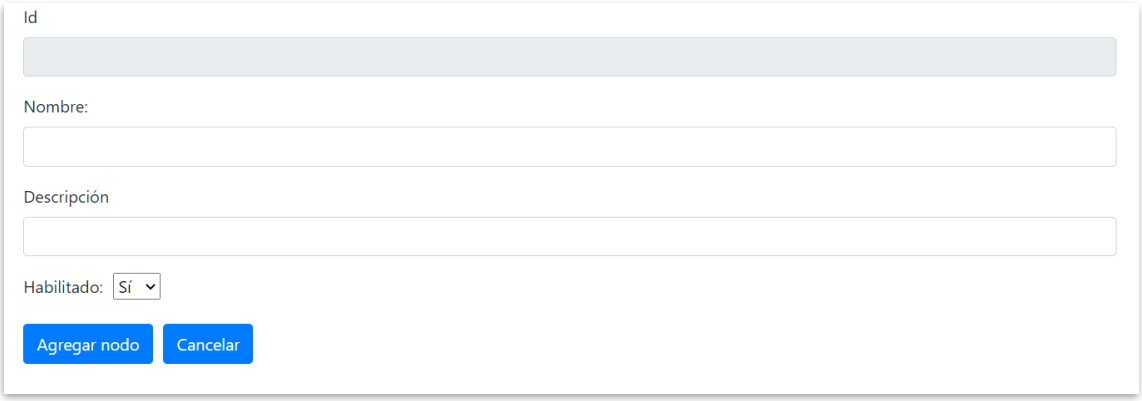
Cuando en un sensor salta la alarma, el recuadro de dicho sensor pasa a estar en rojo y si se pulsa sobre dicho sensor, el sistema entiende que ya se ha visto la alarma y lo devuelve a verde.

4. Configuración

Si el usuario que ha accedido a la plataforma cuenta con el rol “Administrador” o “Moderador” puede acceder a la página de configuración. En esta página se puede modificar el tiempo que está un nodo sensor sin enviar alarmas cuando ya ha enviado una. También se tiene la posibilidad de modificar la información de los nodos sensores, activarlos o desactivarlos, e incluso eliminarlos.



También se tiene la opción de agregar un nuevo sensor.

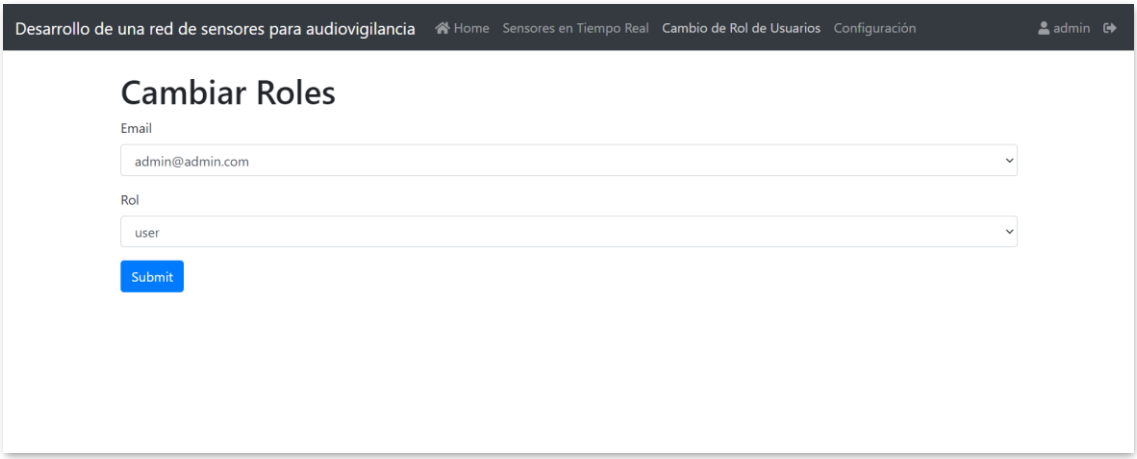


5. Administración de roles de usuario

La plataforma cuenta con distintos roles, los cuales les dan unos permisos u otros a los usuarios. Los permisos son los siguiente:

- Usuario: Solo tiene acceso a la pantalla de sensores en tiempo real.
- Moderador: Tiene acceso a la pantalla de sensores en tiempo real y a la de configuración.
- Administrador: Tiene acceso a todas las pantallas de la plataforma y puede modificar los permisos de otros usuarios mediante la pantalla “Cambio de Rol de Usuarios”

Cuando se accede a la pantalla de “Cambio de Rol de Usuarios” se puede ver un formulario en el cual se selecciona el usuario del cual se quiere modificar el rol y el rol que se le quiere dar.


Desarrollo de una red de sensores para audiovigilancia
Home
Sensores en Tiempo Real
Cambio de Rol de Usuarios
Configuración
admin
Cambiar Roles
Email
admin@admin.com
Rol
user
Submit