UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DE COMPUTADORES

AUTOMATIZACIÓN DE UN SISTEMA DOMÓTICO PARA UNA VIVIENDA INTELIGENTE

AUTOMATION OF A HOME AUTOMATION SYSTEM FOR A SMART HOME

Realizado por

Daniela López Salazar

Tutorizado por

Francisco De Asis Rus Mansilla

Departamento

Lenguajes y Ciencias de la Computación

MÁLAGA, septiembre 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

AUTOMATIZACIÓN DE UN SISTEMA DOMÓTICO PARA UNA VIVIENDA INTELIGENTE

AUTOMATION OF A HOME AUTOMATION SYSTEM FOR A SMART HOME

Realizado por

Daniela López Salazar

Tutorizado por

Francisco De Asis Rus Mansilla

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2023

Fecha defensa: septiembre de 2023

RESUMEN

A pesar de la revolución tecnológica en curso en muchos campos, el avance en la domótica del hogar ha sido limitado. La mayoría de los métodos para automatizar una casa son costosos de implementar, lo que resulta en una baja demanda y poca competencia. Esto dificulta la automatización de una vivienda en la actualidad.

Sin embargo, la domótica puede ser más importante de lo que parece. Además de simplificar las tareas del hogar, puede aumentar la seguridad contra robos o incendios. Nos permite tener un mayor control sobre los sistemas de nuestra casa desde un único punto, y automatizar tareas simples como encender y apagar las luces. También puede ser útil para las personas mayores, para quienes tareas como abrir una persiana pueden ser una tarea difícil. Este proyecto tiene como objetivo ofrecer un sistema de software y hardware adaptable a cualquier entorno, escalable y económico. Se demostrará mediante una maqueta a escala real con sensores y placas correspondientes a los que se usarían en una vivienda real.

También puede ser utilizado en el ámbito educativo esto facilitará el aprendizaje y motivará a los estudiantes a seguir explorando el mundo de la informática. Además, el uso de la maqueta permitirá a los estudiantes comprender de manera más clara cómo funciona la domótica y cómo puede ser aplicada en la vida cotidiana al observar el funcionamiento de los sensores. Esto, a su vez, les ayudará a desarrollar habilidades en el campo de la domótica, lo que les será de gran utilidad en su futuro profesional.

Con este sistema, se espera ofrecer un nivel mayor de seguridad en la vivienda a la vez que mejorar la calidad de vida en la misma. Además, al ser fácilmente escalable y económico de implementar, se espera que este sistema pueda ser adoptado por un mayor número de personas y contribuir al avance en el campo de la domótica del hogar.

Palabras clave: revolución tecnológica, domótica del hogar, automatización, sensores, seguridad, escalable, económico, calidad de vida, ámbito educativo, aprendizaje.

ABSTRACT

Despite the ongoing technological revolution in many fields, progress in home automation has been limited. Most home automation methods are expensive to implement, resulting in low demand and little competition. This makes it difficult to automate a home today.

However, home automation may be more important than it seems. In addition to simplifying household chores, it can increase security against theft or fire. It allows us to have greater control over our home systems from a single point and automate simple tasks like turning lights on and off. It can also be useful for older people, for whom tasks like opening a blind can be a difficult task. This project aims to offer a software and hardware system adaptable to any environment, scalable and economical. It will be demonstrated by means of a full-scale model with sensors and plates corresponding to those that would be used in a real home.

It can also be used in the educational field; this will facilitate learning and motivate students to continue exploring the world of computing. In addition, the use of the model will allow students to understand more clearly how home automation works and how it can be applied in everyday life by observing the operation of the sensors. This, in turn, will help them develop skills in the field of home automation, which will be very useful in their professional future.

With this system, it is expected to offer a higher level of security in the home while improving the quality of life in it. In addition, being easily scalable and cheap to implement, it is expected that this system can be adopted by a greater number of people and contribute to the advancement in the field of home automation.

Keywords: technological revolution, home automation, automation, sensors, security, scalable, economical, quality of life, educational environment, learning.

INDICE

Resumen ---- 3
Abstract ---- 5
iNDICE ---- 7

1. Introducción ---- 17
1.1 Motivación ---- 17
1.2 Objetivos ---- 18
1.3 Estructura de la memoria ---- 18
1. Introducción ---- 19
2. Diseño y construcción de la maqueta ---- 19
3. Materiales usados ---- 19
4. Diseño de la estructura y funcionamiento de los sensores ---- 19
5. Programación de las placas ESP32 y sensores ---- 19
6. Tecnologías usadas y procesos de instalación ---- 19
7. Desarrollo del Backend de la aplicación web ---- 20
8. Desarrollo del Frontend de la aplicación web ---- 20
9. Creación de la API Rest ---- 20
10. Instalación servidor en raspberry pi ---- 20
11. Aplicación accesible desde cualquier lugar: configuración de puertos ---- 20
12. Resultado final ---- 21
13. Conclusiones y trabajo a futuro ---- 21
14. Referencias ---- 21

2. Diseño y construcción de la maqueta ---- 23
2.1 Pasos para la realización y fotos del proceso ---- 23
2.1.1 Esquemas y realización de la maqueta ---- 24

3. Materiales usados ---- 27
3.1 Sensores y placas ---- 27
3.1.1 Placas ESP 32 ---- 27
3.1.2 Sensor de temperatura y humedad ---- 28
3.1.3 Sensor de movimiento ---- 29
3.1.4 Sensor de distancia (proximidad) ---- 29
3.1.5 Sensor de luminosidad (fotoresistor) ---- 30
3.1.6 Leds de colores ---- 30
3.1.7 Cables ---- 30
3.1.8 Modulo zumbador activo ---- 31

4. Diseño de la estructura y funcionamiento de los sensores ---- 33
4.1 Disposición de los sensores dentro de la vivienda ---- 33

5. Programación de las placas ESP32 y sensores ---- 35
5.1 Pruebas individuales ---- 35
5.1.1 Sensor de temperatura y humedad ---- 35
5.1.2 Sensor de movimiento con led ---- 36
5.1.3 Sensor de distancia (proximidad) ---- 37
5.1.4 Sensor de luminosidad (fotoresistor) con un led ---- 39
5.1.5 Zumbador ---- 40
5.1.5.1 Zumbador con botón ---- 40
5.1.5.2 Zumbador con sensor de movimiento ---- 41
5.2 Código y conexiones sensores con Arduino ---- 42

6. Tecnologías usadas y proceso de instalación. ---- 45
1. ¿Qué es node.js? ---- 45
2. Instalación de Node JS. ---- 45
3. ¿Qué es Visual Studio Code? ---- 46
4. Instalación de Visual Studio Code. ---- 46
5. ¿Qué es Postman? ---- 46
6. Instalación de Postman. ---- 47
7. Introducción a las bases de datos y selección del tipo adecuado para nuestro proyecto ---- 48
8. ¿Qué es MySQL Workbench? ---- 50
9. Instalación de MySQL Workbench ---- 51
10. ¿Qué es MongoDB? ---- 54
11. Instalación de MongoDB ---- 55
12. Instalación de Angular ---- 59
13. ¿Qué es Angular CLI? ---- 60
14. Arduino ---- 60
15. Git ---- 61

7. Desarrollo del Backend de la aplicación web. ---- 63
7.1 Creación del Backend ---- 63
7.2 Instalación framework, middleware, bases de datos y módulos. ---- 67

8. Desarrollo del Frontend de la aplicación web. ---- 69
8.1 Creación del Frontend ---- 69
8.2 Creación de la estructura del proyecto ---- 70
8.3 Angular Material ---- 71

9. Creación de la API Rest. ---- 75
9.1 Introducción a la creación de una API REST ---- 75
9.2 API desarrollada para cada sensor ---- 76
9.3 Documentación de las APIs mediante Swagger ---- 77

10. Instalación servidor en raspberry pi. ---- 83

10.1 Configuración e instalación raspberry pi ---------- 83
10.2 Configuración interfaz gráfica raspberry pi ---------- 87
10.3 Configuración Backend y Frontend en raspberry pi ---------- 90
10.4 Configuración MongoDB en raspberry pi ---------- 94

11. Aplicación accesible desde cualquier lugar: configuración de puertos ---------- 97
11.1 Configuracion dentro del router ---------- 97
11.2 Instalacion no-ip raspberry pi ---------- 97

12.Resultado final ---------- 99

13. Conclusiones y trabajo a futuro ---------- 101

14. Referencias ---------- 103

Índice de figuras

Figura 1: Casa madera, vista desde arriba, sin pintar 24
Figura 2: Casa madera, vista desde arriba pintada 24
Figura 3: Casa madera frontal sin pintar 24
Figura 4: Casa madera frontal pintada 24
Figura 5: Casa madera con muebles 25
Figura 6: Casa madera con ventanas 25
Figura 7: Casa madera frontal con ventanas y puertas 25
Figura 8: Placa ESP 32 (Lobo, 2020) 27
Figura 9: Sensor de temperatura y humedad (Modulos arduino, s.f.) 28
Figura 10: Sensor de movimiento (Watson, 2020) 29
Figura 11: Sensor de distancia (HW libre, s.f.) 29
Figura 12: Funcionamiento sensor de distancia (arduino, 2023) 29
Figura 13: Sensor de luminosidad (fotoresistor) (Eepower, s.f.) 30
Figura 14: Leds de colores 30
Figura 15: Cables de conexión 30
Figura 16: Modulo zumbador activo (LuisLlamas, 2016) 31
Figura 17: Foto de la casa con los sensores y placas situados (vista desde arriba) 33
Figura 18: Circuito montado, sensor temperatura y humedad 36
Figura 19: Diagrama del circuito del sensor de temperatura y humedad 36
Figura 20: Funcionamiento del circuito del sensor de movimiento con el led 36
Figura 21: Diagrama del circuito del sensor de movimiento con led 36
Figura 22: Ejemplo de código del circuito del sensor de movimiento con led 37
Figura 23: Circuito montado del sensor movimiento con led 37
Figura 24: Funcionamiento del circuito de sensor de distancia con led 38
Figura 25: Diagrama del circuito del sensor de distancia con led 38
Figura 26: Circuito montado del sensor de distancia con led 38
Figura 27: Ejemplo de código del sensor de distancia con led 38
Figura 28: Funcionamiento del circuito del sensor de luminosidad con un led 39
Figura 29: Diagrama del circuito del sensor de luminosidad con un led 39
Figura 30: Ejemplo de código del sensor de luminosidad con led 39
Figura 31: Montaje del circuito del sensor de luminosidad con led 39
Figura 32: Diagrama del circuito de un zumbador con botón 40

Figura 33: Ejemplo de código del circuito de un zumbador con botón 40
Figura 34: Montaje del circuito de un zumbador con un botón 40
Figura 35: Diagrama del circuito del zumbador con sensor de movimiento y un led 41
Figura 36: Montaje del circuito del zumbador con sensor de movimiento y un led... 41
Figura 37: Ejemplo de código del circuito del zumbador con sensor de movimiento 41
Figura 38: Icono NODE JS (TheOpenJSFundation, 2023) 45
Figura 39: Icono visual studio Code (Microsoft, 2023)..... 46
Figura 40: Icono Postman (Postman, 2023) 47
Figura 41: Ejemplo de uso de Postman..... 47
Figura 42: Icono MySQL..... 51
Figura 43: Proceso de instalación MySQL 51
Figura 44: Proceso instalación MySQL comprobación puertos 52
Figura 45: Proceso instalación MySQL establecemos usuario y contraseña 52
Figura 46: Proceso instalación MySQL establecemos el nombre del servicio..... 53
Figura 47: Proceso instalación MySQL aceptamos permisos..... 53
Figura 48: Proceso instalación MySQL comprobamos la conexión..... 54
Figura 49: Icono MongoDB..... 54
Figura 50: Descarga mongo DB (MongoDB, 2023) 55
Figura 51: Instalación MongoDB ventana de inicio..... 56
Figura 52: Instalación MongoDB completa o personalizada..... 56
Figura 53: Instalación MongoDB configuración del servicio 57
Figura 54: Instalación MongoDB compass 57
Figura 55: Instalación MongoDB aceptamos la instalación 58
Figura 56: Pantalla inicio compass MongoDB 58
Figura 57: Icono Angular (Google, Angular, 2023) 59
Figura 58: Instalación Angular..... 59
Figura 59: Icono Arduino (AndProf, 2021) 60
Figura 60: Icono GIT (Git, s.f.)..... 61
Figura 61: Consola en modo escucha 65
Figura 62: Consola en ejecución comando nodemon..... 66
Figura 63: Método listen del servidor..... 68
Figura 64: Angular Material (Google P. b., 2010-2023)..... 72
Figura 65: Paleta colores de angular material (Google, Google, 2023)..... 72
Figura 66: Ejemplo del swagger asociado al sensor de luminosidad 81

Figura 67: Raspberry Pi Imager, elegimos OS 84
Figura 68: Elección sistema operativo 84
Figura 69: Configuración opciones avanzadas preinstalación 84
Figura 70: Escritura del SO en la raspberry pi 85
Figura 71: Conexión mediante ssh a la rapsberry pi 86
Figura 72: Configuración de ip estática en nuestra raspberry 86
Figura 73: Instalación vcn raspberry pi 87
Figura 74: Accedemos a la interfaz basada en texto de la rapsberry pi 87
Figura 75: Configuración gráfica raspberry pi 88
Figura 76: Activación VCN en raspberry pi 88
Figura 77: Instalación RealVNC Viewer 88
Figura 78: Acceso RealVNC Viewer desde nuestro ordenador personal 89
Figura 79: Interfaz Raspberry pi desde nuestro ordenador personal 89
Figura 80: Instalación FileZilla 90
Figura 81: Conexión con FileZilla a nuestra raspberry pi 90
Figura 82: Arquitectura hardware de nuestra raspberry pi 91
Figura 83: Instalación nodejs en la raspberry pi 91
Figura 84: Version nodejs en la raspberry pi 92
Figura 85: Instalación Angular en nuestra raspberry pi 92
Figura 86: MongoDB Atlas 94
Figura 87: MongoDB Atlas, nuestra base de datos 95
Figura 88: Estado final de la maqueta, con sus placas y sensores 99
Figura 89: Login página web, DOMOCDANI 100
Figura 90: Página principal, DOMOCDANI 100
Figura 91: Habitación con sus sensores disponibles 100
Figura 92: Tabla ejemplo, de los datos registrados 100

A mis padres, Martha e Ivan,

que me han aconsejado durante toda mi vida.

Gracias por creer en mí.

Por ser mi inspiración, gracias, J.A.

1. Introducción

1.1 Motivación

La motivación detrás de este trabajo es la creación de un proyecto real y práctico que pueda ayudar a un amplio colectivo de personas a un costo asequible. Para lograr este objetivo, se han seleccionado cuidadosamente placas y sensores que ofrecen una buena funcionalidad a un precio razonable.

La domótica de la vivienda ofrece numerosas ventajas.

En primer lugar, nos brinda una mayor comodidad al permitirnos programar el encendido y apagado de luces y otros dispositivos para crear un ambiente acogedor al llegar a casa después del trabajo.

En segundo lugar, nos permite ahorrar energía al programar el encendido de luces y otros dispositivos en función de la hora del día o la cantidad de luz natural disponible. En tercer lugar, mejora la seguridad de nuestra vivienda al permitirnos instalar cámaras y sensores de movimiento que pueden detectar cualquier actividad sospechosa.

Este proyecto tiene dos enfoques: primero en ayudar a personas mayores que pueden tener dificultades para realizar tareas cotidianas, y segundo puede ser utilizado en el ámbito educativo para facilitar el aprendizaje y motivar a los estudiantes a seguir explorando el mundo de la informática y la domótica.

La aplicación web recopila datos de los distintos sensores y estadísticas de la vivienda y se ubica en algún lugar de la casa. Las funciones incluyen el encendido y apagado automático de luces en las habitaciones al anochecer, la detección de intrusos mediante sensores de movimiento, el monitoreo de la humedad y temperatura en la vivienda, y la detección de proximidad en el garaje para avisarnos mediante un sonido cuando estamos cerca al entrar con el coche. Todos estos datos se almacenan en una base de datos para su posterior consulta. Por ejemplo, podemos consultar las veces que se ha encendido una bombilla o las temperaturas registradas en días anteriores.

En resumen, la domotización y automatización de nuestra vivienda nos brinda numerosas mejoras en términos de comodidad, ahorro de energía, seguridad y aprendizaje.

1.2 Objetivos

Como se mencionó en la introducción de nuestro proyecto, uno de los objetivos es mejorar la comodidad, el ahorro energético y la seguridad para aumentar la calidad de vida. Todo esto a un bajo costo y con un proyecto escalable y accesible para cualquier persona que desee incorporarlo en su hogar. Además, siempre existe la posibilidad de agregar más sensores y placas para expandir las funciones de la casa, o para mejorarlas. Y el otro objetivo es introducirlo en el ámbito educativo para facilitar el aprendizaje y motivar a los estudiantes a seguir explorando el mundo de la informática y la domótica.

Este documento describe el desarrollo de una aplicación web que recopila datos de una vivienda equipada con diferentes sensores. Estos sensores recogen información y la almacenan en una base de datos. Los sensores están conectados a placas ubicadas en cada habitación, baño o cocina donde se instalen. Esto permite la automatización de la vivienda.

Todo esto se puede visualizar en una maqueta a escala real que representa una vivienda con sus habitaciones, baños, garaje, salón y cocina correspondientes. En esta maqueta se instalarán diferentes sensores para demostrar su funcionamiento.

1.3 Estructura de la memoria

La estructura de esta memoria se ha diseñado cuidadosamente para reflejar los distintos pasos que se han seguido en el desarrollo completo de la aplicación web y la automatización de la vivienda en la maqueta a escala real. Cada sección de la memoria detalla un paso específico del proceso, desde la planificación inicial hasta la implementación final.

1. **Introducción**

   Presentamos el contexto en el que se desarrollará nuestro proyecto. Hemos identificado los puntos clave y establecidos objetivos claros para lograr el resultado que queremos.

2. **Diseño y construcción de la maqueta**

   Este apartado detalla el proceso de construcción de la maqueta, incluyendo las decisiones tomadas y su justificación. Se proporcionan fotografías para ilustrar el progreso y permitir al lector visualizar el resultado final.

3. **Materiales usados**

   Este apartado describe las herramientas que han sido esenciales para el desarrollo y correcto funcionamiento de la aplicación. Se describen en detalle para proporcionar una comprensión completa de cómo se han utilizado en el proyecto.

4. **Diseño de la estructura y funcionamiento de los sensores**

   En este apartado se plantea la ubicación de los sensores y las placas correspondientes dentro de la vivienda. Se describe en detalle cómo funcionarán en conjunto para lograr la automatización de la casa.

5. **Programación de las placas ESP32 y sensores**

   Este apartado presenta una investigación detallada de cada sensor, analizando sus características y funcionalidades para determinar su uso óptimo en el proyecto.

6. **Tecnologías usadas y procesos de instalación**

   Se explican en detalle las diferentes tecnologías y softwares que se han utilizado en el desarrollo del proyecto, así como los procesos de instalación y configuración necesarios para su correcto funcionamiento. Se incluyen instrucciones detalladas y capturas de pantalla para guiar al usuario en el

proceso de instalación y configuración de cada una de las herramientas utilizadas.

7. **Desarrollo del Backend de la aplicación web**

   Aquí se describe el proceso de desarrollo del Backend de la aplicación web, incluyendo fragmentos de código relevantes para ilustrar el trabajo realizado.

8. **Desarrollo del Frontend de la aplicación web**

   Este apartado detalla el desarrollo del Frontend de la aplicación web, proporcionando ejemplos de código para mostrar el progreso y resultado final.

9. **Creación de la API Rest**

   En este apartado se explica el procedimiento seguido para crear la API Rest y asegurar su correcto funcionamiento, describiendo los pasos y decisiones tomadas durante el proceso.

10. **Instalación servidor en raspberry pi**

    En el desarrollo de nuestro proyecto, se ha llevado a cabo el proceso de instalación del sistema operativo Raspberry Pi para poder utilizarlo como servidor. En este servidor, hemos instalado VNC Viewer para tener una interfaz gráfica y facilitar su uso. De esta manera, podemos tener nuestro proyecto en el servidor y acceder a él de manera remota a través de VNC Viewer.

11. **Aplicación accesible desde cualquier lugar: configuración de puertos**

    Además de la instalación del sistema operativo Raspberry Pi y VNC Viewer, también se ha llevado a cabo la configuración del router y el servicio No-IP para poder tener acceso remoto a nuestro servidor. Esto nos permite interactuar con la página web alojada en el servidor desde cualquier lugar, lo que facilita el acceso y la gestión de nuestro proyecto.

12. Resultado final

Podemos observar cómo la maqueta, con sus placas y sensores, ha quedado completamente montada y funcional. Además, podemos apreciar en una fotografía el resultado final de la página en la que registramos los datos y con la que podemos interactuar.

13. Conclusiones y trabajo a futuro

Se exponen las conclusiones de proyecto, posibles mejoras y el trabajo a futuro.

14. Referencias

Referencias recogidas durante el desarrollo del proyecto.

2. Diseño y construcción de la maqueta

2.1 Pasos para la realización y fotos del proceso

Para el diseño de la maqueta, se comenzó con un pequeño dibujo para planificar el número de habitaciones y los tipos y cantidad de sensores que se incorporarían para lograr los objetivos previamente descritos.

Luego, se creó un boceto en cartón de la primera idea.

Finalmente, se construyó la maqueta en madera con algunos cambios y ampliaciones en comparación con la primera idea.

Una vez terminada la construcción, se pintó la maqueta.

Al final de este punto se pueden ver fotos de la maqueta que muestran todos los detalles añadidos para que fuera lo más realista posible.

Por ejemplo, se añadieron ventanas, puertas, escaleras, cocina y algunos muebles. Esto se hizo con la intención de poder instalar el proyecto en cualquier tipo de vivienda, como se mencionó anteriormente en la introducción y los objetivos.

A continuación, mostraremos el proceso de creación de la maqueta en fotos hasta llegar al resultado final.

2.1.1 Esquemas y realización de la maqueta

Figura 1: Casa madera, vista desde arriba, sin pintar

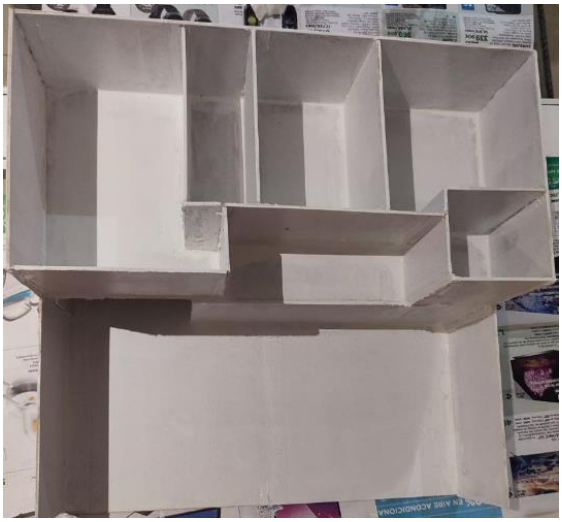


Figura 2: Casa madera, vista desde arriba pintada

Figura 3: Casa madera frontal sin pintar

Figura 4: Casa madera frontal pintada

Figura 6: Casa madera con ventanas

Figura 5: Casa madera con muebles

Figura 7: Casa madera frontal con ventanas y puertas

3. Materiales usados

3.1 Sensores y placas

3.1.1 Placas ESP 32

Figura 8: Placa ESP 32 (Lobo, 2020)

Las placas principales ESP32 serán las encargadas de conectar los sensores correspondientes a cada habitación. Esta placa de desarrollo de microcontroladores está basada en el chip ESP32, un microcontrolador de doble núcleo y 32 bits con conectividad inalámbrica Wifi y Bluetooth. Ofrece una amplia variedad de características, incluyendo 34 pines de entrada/salida (GPIO), 3 puertos UART, 2 puertos I2C y SPI, ADC con resolución de 12 bits, PWM, interrupciones externas, reloj en tiempo real (RTC), y soporte para pantalla OLED y tarjeta microSD. A continuación, describiremos brevemente los puertos que nos proporciona esta placa:

El chip ESP32 cuenta con 34 pines de entrada/salida GPIO (GPIO0 ~ GPIO19, GPIO21 ~ GPIO23, GPIO25 ~ GPIO27 y GPIO32 ~ GPIO39) que pueden ser configurados como entrada o salida según la necesidad del dispositivo que se quiera conectar. Los puertos UART, I2C y SPI son protocolos de comunicación serie utilizados en la electrónica digital para la comunicación entre dispositivos. El puerto UART (Universal Asynchronous Receiver-Transmitter) es un protocolo de comunicación serie asíncrono utilizado para la transmisión y recepción de datos entre dispositivos digitales, como microcontroladores, sensores, módems y otros dispositivos electrónicos. Utiliza dos cables de comunicación, TX (transmitir) y RX (recibir). El puerto I2C (Inter-Integrated Circuit) es un bus serie de datos que define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. Incluye dos cables de comunicación, SDA y SCL.

SDA (Serial Data) y SCL (Serial Clock) son los dos cables de comunicación utilizados en el puerto I2C. El cable SDA se encarga de enviar datos entre dispositivos, mientras que el cable SCL sincroniza la transferencia de datos. El I2C está diseñado como un bus maestro-esclavo, donde la transferencia de datos es siempre iniciada por un maestro y el esclavo responde. En un modo multimaestro, es posible tener varios maestros que pueden comunicarse entre sí, con uno de ellos actuando como esclavo.

El puerto SPI (Serial Peripheral Interface) es un estándar de comunicación utilizado para la transferencia de información entre circuitos integrados en equipos electrónicos. Permite la comunicación dúplex entre un dispositivo y dispositivos periféricos o entre varios microcontroladores. Los periféricos del SPI pueden configurarse como maestro o secundario para permitir la comunicación entre microcontroladores. Un ADC (Convertidor Analógico-Digital) es un dispositivo que convierte una señal analógica en una señal digital. Su resolución se refiere al número de bits utilizados para representar la señal digitalizada y determina la precisión con la que se puede medir una señal analógica. Por ejemplo, un ADC de 12 bits tiene una resolución de 2^12 = 4096 niveles. La modulación de ancho de pulso (PWM) es una técnica ampliamente utilizada para la entrega de energía.

3.1.2 Sensor de temperatura y humedad

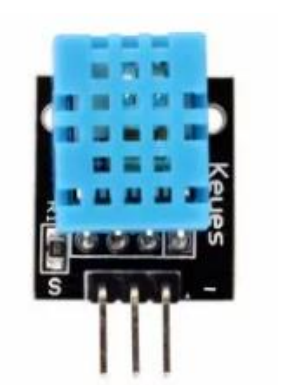


Figura 9: Sensor de temperatura y humedad (Modulos arduino, s.f.)

El módulo de sensor de temperatura y humedad KY-015 proporciona una interfaz serial digital para medir la humedad y la temperatura del ambiente. Es compatible con varios microcontroladores como Arduino, Raspberry Pi y ESP32.

Este módulo consta de un sensor digital de humedad y temperatura DHT11, una resistencia de 1 kΩ y 3 pines macho. (Modulos arduino, s.f.)

3.1.3 Sensor de movimiento

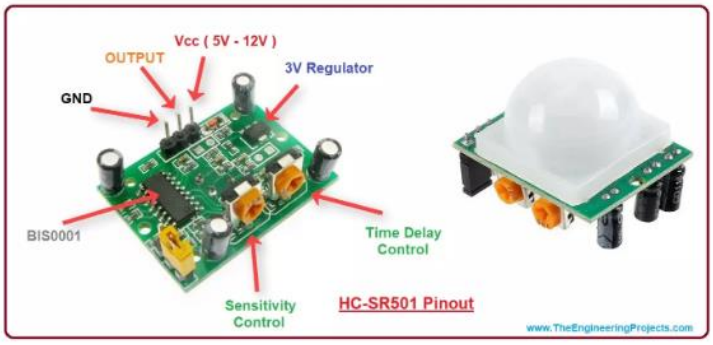


Figura 10: Sensor de movimiento (Watson, 2020)

El HC-SR501 es un sensor de movimiento altamente sensible y confiable que utiliza ondas infrarrojas para detectar objetos (Watson, 2020)

Cuando un objeto que emite radiaciones infrarrojas, como un ser humano o un animal, pasa por el campo de visión del sensor, este detecta el cambio de temperatura y puede ser utilizado para detectar movimiento (rjrobotics007, 2023).

3.1.4 Sensor de distancia (proximidad)

Figura 11: Sensor de distancia (HW libre, s.f.)

El HC-SR04 es un sensor de distancia de baja precisión basado en ultrasonidos. (HW libre, s.f.) Proporciona una medición precisa y alta estabilidad con un rango de medición de 2 a 400 cm. (arduino, 2023) La distancia se mide mediante ultrasonidos: se emite un sonido, hay un rebote y se recibe. (HW libre, s.f.)

3.1.4.1 Funcionamiento del sensor de distancia

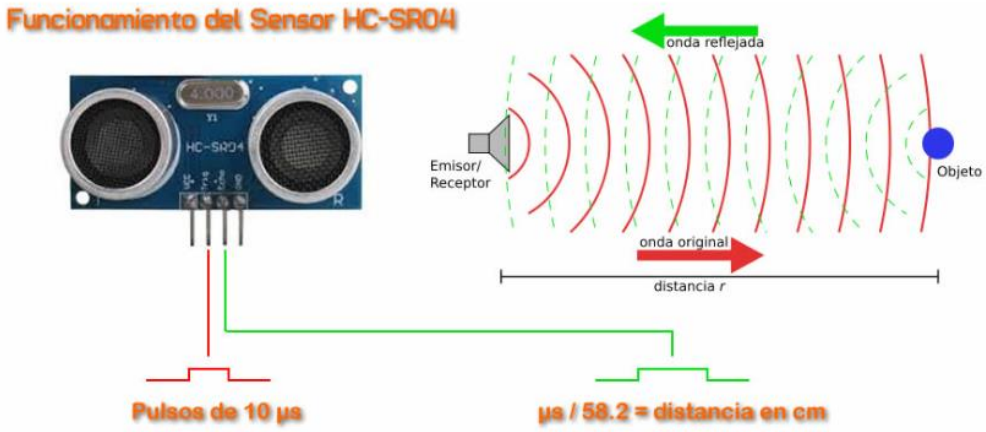


Figura 12: Funcionamiento sensor de distancia (arduino, 2023)

3.1.5 Sensor de luminosidad (fotoresistor)

Figura 13: Sensor de luminosidad (fotoresistor) (Eepower, s.f.)

Un fotoresistor (también conocido como célula fotoeléctrica o resistor dependiente de la luz) es un componente pasivo cuya resistencia disminuye con relación a la luminosidad (luz) recibida en la superficie sensible del componente.

En otras palabras, cuanto mayor sea la intensidad de luz incidente, menor será la resistencia del fotoresistor. (Eepower, s.f.)

3.1.6 Leds de colores

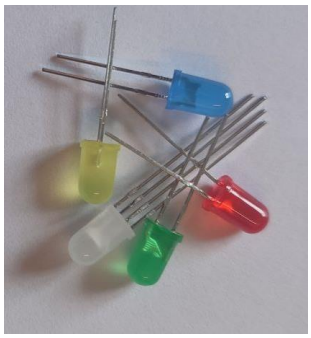


Figura 14: Leds de colores

Un LED es un diodo que nos emitirá luz, simulando las luces de nuestra casa.

3.1.7 Cables

Figura 15: Cables de conexión

Cables para todas las conexiones entre las placas y los

Sensores.

3.1.8 Modulo zumbador activo

Figura 16: Modulo zumbador activo (LuisLlamas, 2016)

Un zumbador activo es un dispositivo que genera un sonido de una frecuencia determinada y fija cuando se conectan a una tensión. Incorpora un oscilador simple por lo que únicamente es necesario suministrar corriente al dispositivo para que emita el sonido, si fuera un zumbador pasivo si necesitara recibir una da de la frecuencia. (LuisLlamas, 2016)

4. Diseño de la estructura y funcionamiento de los sensores

4.1 Disposición de los sensores dentro de la vivienda

Figura 17: Foto de la casa con los sensores y placas situados (vista desde arriba)

5. Programación de las placas ESP32 y sensores

5.1 Pruebas individuales

Se realizarán pruebas con los diferentes sensores que hemos definido en el punto 3 de esta memoria, incluyendo las pruebas con el sensor de temperatura y humedad, el sensor de movimiento con un LED, el sensor de proximidad con un LED, el sensor de luminosidad con un LED, el zumbador con un botón y el zumbador con un sensor de movimiento y un led. Para cada uno de estos sensores se proporcionará su código y fotografías para ilustrar su funcionamiento y resultados. Y comprobar el funcionamiento correcto que queremos aplicar en la vivienda.

5.1.1 Sensor de temperatura y humedad

El sensor que vamos a utilizar está definido en el punto 3.1.2.

Este sensor es una casa automatizada puede ser muy útil y vamos a ver en que vamos a utilizarla.

Vamos a recoger la información de temperatura y humedad del cuarto en el que se instale e ir recogiendo esa información en una base de datos.

Esto podría ser útil por ejemplo si se muestra mucha o poca humedad avisar al usuario de que debería usar humificadores o deshumificadores para ajustar esa humedad. Al igual podemos hacer con la temperatura para que se pudiera regular la temperatura.

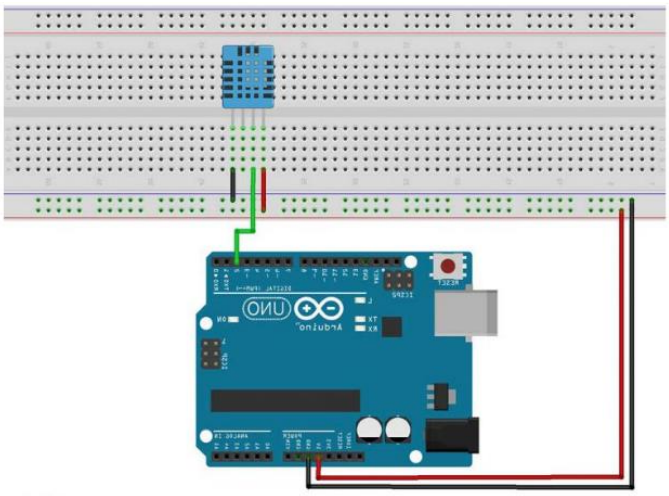


Figura 19: Diagrama del circuito del sensor de temperatura y humedad

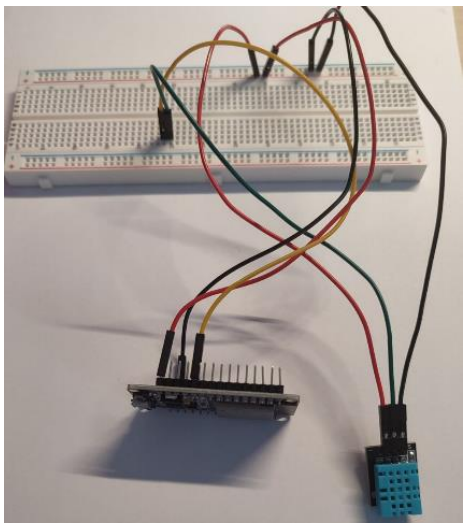


Figura 18: Circuito montado, sensor temperatura y humedad

5.1.2 Sensor de movimiento con led

El sensor que vamos a utilizar está definido en el punto 3.1.3.

Este sensor con el led nos será muy útil en una casa ya que nos puede servir para ahorrar energía ya que cuando detecte el movimiento se encenderá la luz, que en este caso esa luz será representada por el led mencionado.

Otra posible utilidad es para evitar robos, es decir, cuando detecte movimientos extraños fuera de la casa, esto lo veremos con más detalle más adelante con el zumbador.

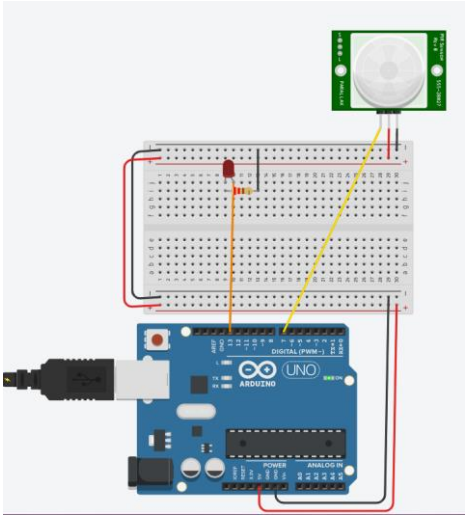


Figura 21: Diagrama del circuito del sensor de movimiento con led

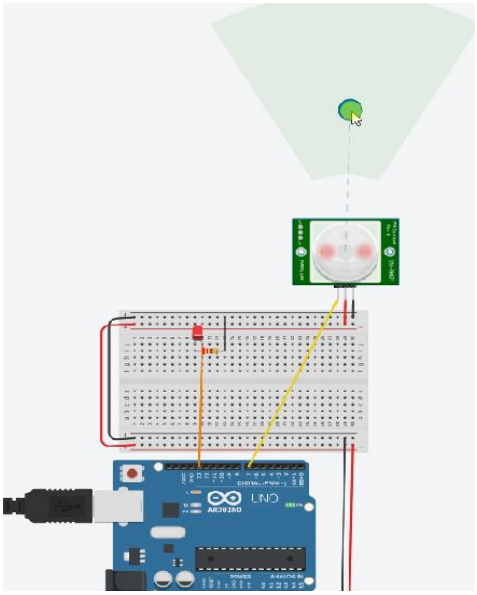


Figura 20: Funcionamiento del circuito del sensor de movimiento con el led

```
1
2   int led = 13;
3   int sensor = 7;
4   int sensordato;
5   void setup()
6   {
7     pinMode(led, OUTPUT);
8     pinMode(sensor,INPUT);
9   }
10  void loop()
11  {
12    sensordato = digitalRead(sensor);
13    if (sensordato == HIGH)
14      {
15      digitalWrite(led, HIGH);
16      }
17    else
18      {
19      digitalWrite(led, LOW);
20      }
21    delay(10);
22  }
```

Figura 22: Ejemplo de código del circuito del sensor de movimiento con led

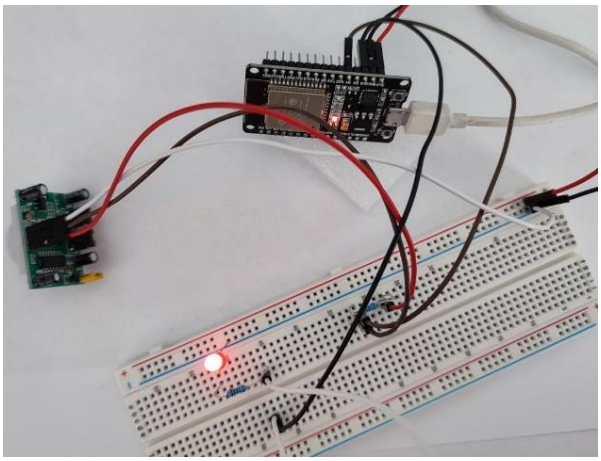


Figura 23: Circuito montado del sensor movimiento con led

5.1.3 Sensor de distancia (proximidad)

El sensor que vamos a utilizar está definido en el punto 3.1.4.

Este sensor podemos utilizarlo de casi la misma forma que hemos definido las utilidades del punto 5.1.2, es decir, puede ser útil para detectar la presencia de objetos o personas y llevar a cabo una determinada acción. Como encender la luz, en nuestro caso el led, cuando detecte una presencia y apagarla automáticamente cuando ya no la detecte por tanto nos sirve para ahorra energía también, al igual que detectar algún intruso, por ejemplo, fuera de casa.

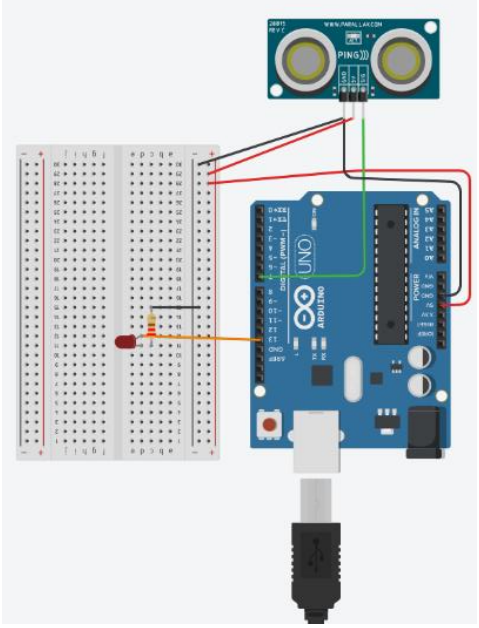


Figura 25: Diagrama del circuito del sensor de distancia con led

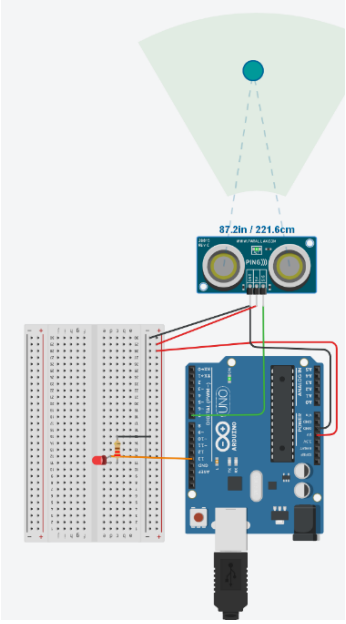


Figura 24: Funcionamiento del circuito de sensor de distancia con led

```
int cm = 0;
int led = 13;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);
  pinMode(led, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);

  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);

  return pulseIn(echoPin, HIGH);
}
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  cm = 0.01723 * readUltrasonicDistance(7, 7);

  if( cm <= 8 || cm > 300){
    digitalWrite(led, LOW);
  }else{
    digitalWrite(led,HIGH );
  Serial.print(cm);
  Serial.println("cm");
  }
  delay(100);
}
```

Figura 27: Ejemplo de código del sensor de distancia con led

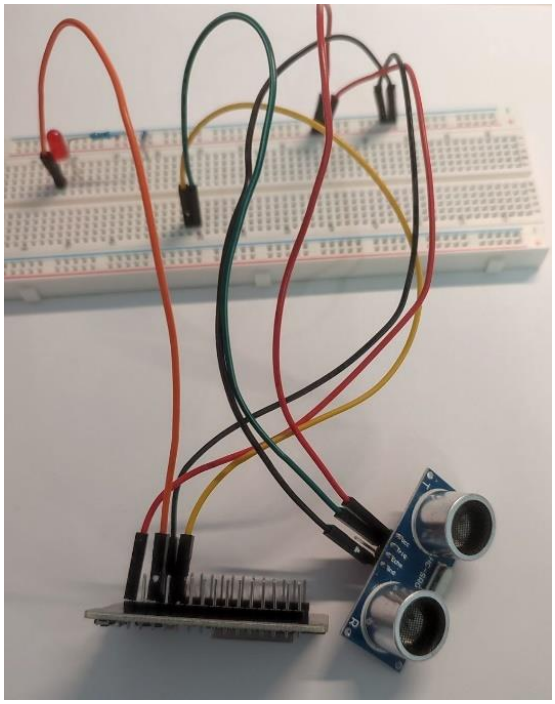


Figura 26: Circuito montado del sensor de distancia con led

5.1.4 Sensor de luminosidad (fotoresistor) con un led

El sensor que vamos a utilizar está definido en el punto 3.1.5.

Este nos puede ser útil para detectar el nivel de luz que recibe, y asi poder controlar que se encienda automáticamente la luz, en nuestro caso el led.

También puede darnos como utilidad para regular las persianas de la casa y conseguir que suban o baje según la intensidad de luz que se detecte.

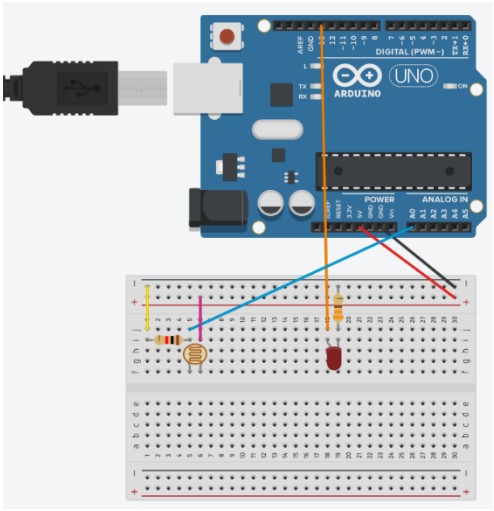


Figura 29: Diagrama del circuito del sensor de luminosidad con un led

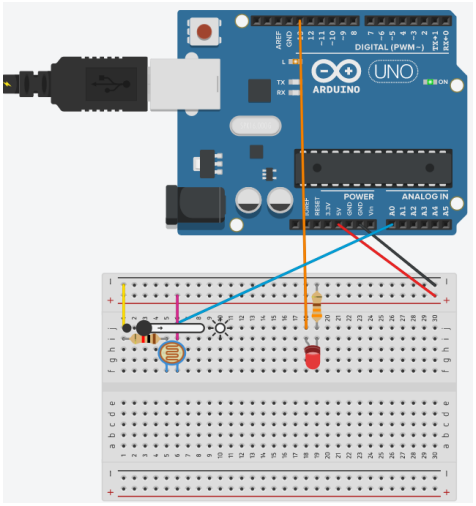


Figura 28: Funcionamiento del circuito del sensor de luminosidad con un led

```
int sensorluz = 0;
int umbral = 400;

void setup()
{
 Serial.begin(9600);

  pinMode(13, OUTPUT);
}

void loop()
{
  sensorluz = analogRead(A0);
  Serial.println(sensorluz);

 if (sensorluz < umbral){
    digitalWrite(13, HIGH);
  } else{
    digitalWrite(13, LOW);
  }

   delay(100);
```

Figura 30: Ejemplo de código del sensor de luminosidad con led

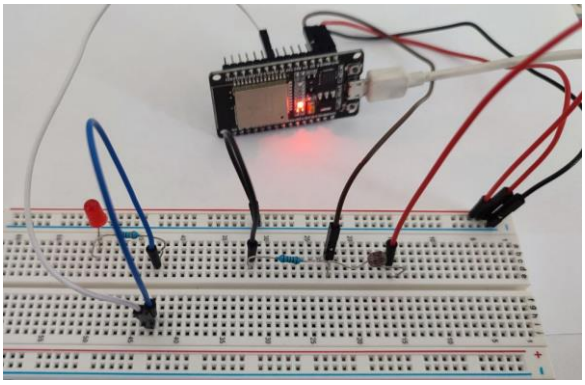


Figura 31: Montaje del circuito del sensor de luminosidad con led

5.1.5 Zumbador

5.1.5.1 Zumbador con botón

El siguiente que vamos a utilizar está definido en el punto 3.1.8.

Aquí vamos a ver la utilidad de un zumbador con un botón, puede ser utilizado como avisa de emergencia en algún lugar de la casa, o como timbre de la casa.

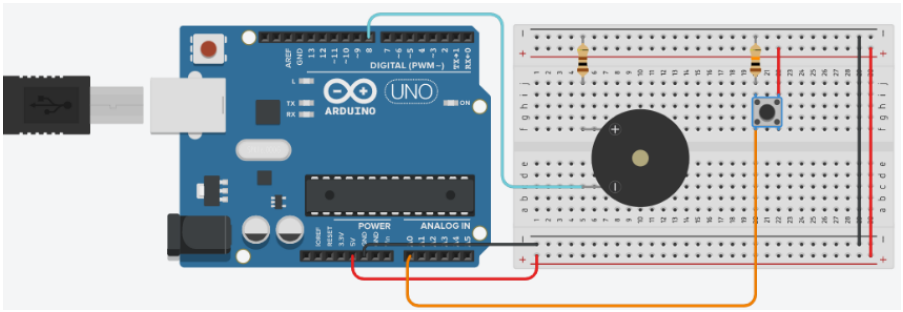


Figura 32: Diagrama del circuito de un zumbador con botón

```
1  int zumbador = 8;
2  int sensorDato;
3
4  void setup(){
5    pinMode(A0, INPUT);
6    pinMode(zumbador, OUTPUT);
7  }
8
9  void loop(){
10   sensorDato = digitalRead(A0);
11
12   if (sensorDato == HIGH) {
13     digitalWrite(zumbador, HIGH);
14       tone(8, 440, 100);
15       // play tone 57 (A4 = 440 Hz)
16
17   }else{
18     digitalWrite(zumbador, LOW);
19
20   }
21     delay(10);
22   }
```

Figura 33: Ejemplo de código del circuito de un zumbador con botón

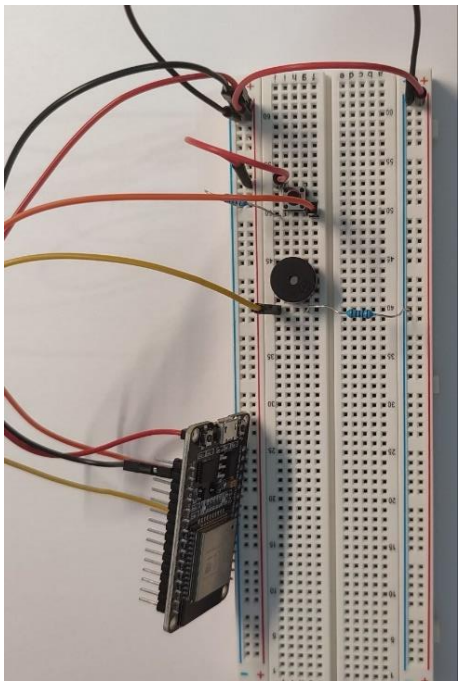


Figura 34: Montaje del circuito de un zumbador con un botón

5.1.5.2 Zumbador con sensor de movimiento

El siguiente que vamos a utilizar está definido en el punto 3.1.8, mezclado con el sensor que está definido en el punto 3.1.3.

Las utilidades que podemos ver con estos juntos y un led, es por ejemplo un sistema de alarma, que cuando detecte un movimiento suene el zumbador y a su vez encenderse el led, a forma de luz de emergencia.

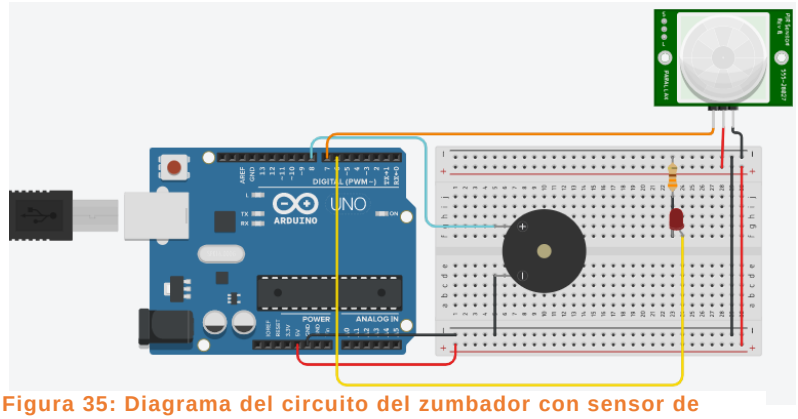


Figura 35: Diagrama del circuito del zumbador con sensor de movimiento y un led

```
int sensor = 7;
int led = 6;
int zumbador = 8;
int sensorDato;

void setup(){
  pinMode(sensor, INPUT);
  pinMode(led, OUTPUT);
  pinMode(zumbador, OUTPUT);
}

void loop(){
  sensorDato = digitalRead(sensor);

  if (sensorDato == HIGH) {
    digitalWrite(led, HIGH);
    digitalWrite(zumbador, HIGH);
    Serial.println("Intrusos");
  }else{
    digitalWrite(led, LOW);
    digitalWrite(zumbador, LOW);
    Serial.println("Modo vigilante");
  }
    delay(10);

  }
```

Figura 37: Ejemplo de código del circuito del zumbador con sensor de movimiento

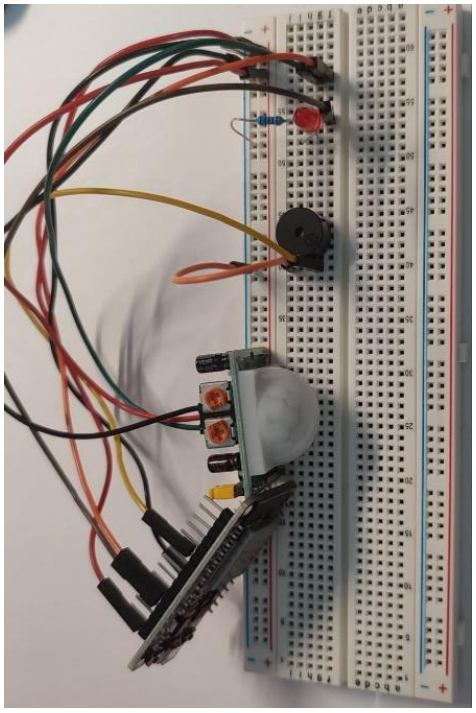


Figura 36: Montaje del circuito del zumbador con sensor de movimiento y un led

5.2 Código y conexiones sensores con Arduino

A continuación, se muestra un fragmento de código que ilustra la configuración más importante para la comunicación entre los sensores y la base de datos. Este código es un programa escrito en el IDE de Arduino para una placa ESP32 con Wifi, que se conecta a una red y envía datos a un servidor. El programa utiliza las librerías **WiFi.h** y **HTTPClient.h** para conectarse a una red Wifi y realizar peticiones **HTTP**. Se definen varias constantes para almacenar los datos de la red Wifi y las URLs de los servidores a los que se enviarán los datos.

El programa también define varios pines para conectar sensores y Leds. En el método setup, el programa se conecta a la red Wifi y configura los pines como entrada o salida. En el método loop, el programa lee los valores de los sensores y envía los datos al servidor utilizando peticiones **HTTP POST**.

En resumen, este código es un ejemplo de cómo utilizar una placa ESP32 con Wifi en el IDE de Arduino para enviar datos de sensores a un servidor utilizando peticiones HTTP.

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>
HTTPClient http;

void setup() {
Serial.begin(115200);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Conectando a WiFi...");
}
Serial.println("Conectado a WiFi");

void loop() {
}
if (WiFi.status() == WL_CONNECTED) {
http.begin(lightServerUrl);
http.addHeader("Content-Type", "application/x-www-form-
urlencoded");
bool ledOn = (lecturaSensorLuminosidad < 300);
String postData = "value=" + String(lecturaSensorLuminosidad) +
"&ledOn=" + ledOn;
int httpResponseCode = http.POST(postData);
if (httpResponseCode > 0) {
String response = http.getString();
Serial.println(response);
} else {
Serial.print("Error en la petición HTTP: ");
Serial.println(httpResponseCode);
}
http.end();
} else {
Serial.println("Error en la conexión WiFi");
}
delay(5000);
```

Además, nuestro código utiliza la librería **WebServer.h** que se utiliza para crear un servidor web en el puerto 80 que maneja las solicitudes **HTTP** y permite interactuar con el proyecto a través de una página web.

La función handleGaraje() maneja las solicitudes HTTP para abrir o cerrar la puerta del garaje y envía una respuesta al cliente.

```
#include <WebServer.h>

WebServer server(80);
boolean puertaGarajeAbierta = 0;

void handleGaraje() {
  server.sendHeader("Access-Control-Allow-Origin", "*"); //
Permitir solicitudes CORS desde cualquier dominio
  String state = server.arg("state");
  if (state == "1") {
    puertaGarajeAbierta = true;
    digitalWrite(yellowLedPin, HIGH);
    digitalWrite(blueLedPin, LOW);
    server.send(200, "text/plain", "Puerta del garaje abierta");
  } else if (state == "0") {
    puertaGarajeAbierta = false;
    digitalWrite(yellowLedPin, LOW);
    digitalWrite(blueLedPin, HIGH);
    server.send(200, "text/plain", "Puerta del garaje cerrada");
  } else {
    server.send(400, "text/plain", "Estado no válido");
  }
```

En nuestro Frontend, tenemos una función llamada cambiar estado puerta garaje que se encarga de cambiar el estado de la puerta del garaje. Esta función recibe como parámetro la dirección IP de la Raspberry Pi con el puerto 80 y, cuando se pulsa el botón en el Frontend, envía una solicitud **HTTP GET** a esta dirección para que la placa reaccione y cambie el estado del LED o del sensor correspondiente. La función invierte el valor de la variable puertaGarajeAbierta y envía una solicitud HTTP GET a una URL específica con un parámetro que indica si la puerta del garaje debe estar abierta o cerrada.

La respuesta del servidor se maneja en el método subscribe de la solicitud HTTP.

```
 cambiarEstadoPuertaGaraje() {
      this.puertaGarajeAbierta = !this.puertaGarajeAbierta;

      const url = 'http://192.168.100.72:80';
        const data = {
            state: this.puertaGarajeAbierta ? 'on' : 'off',
        };
        // @ts-ignore
        this.http.get(url, {params: data }).subscribe(response => {
            // Manejar la respuesta del servidor aquí
        });
    }
```

6. Tecnologías usadas y proceso de instalación.

1. ¿Qué es node.js?

Es un entorno multiplataforma que permite a los desarrolladores crear toda tipo de clase de herramientas de servidor y aplicaciones en JavaScript.

Hemos elegido node ya que su forma de trabajar optimiza el rendimiento y la escalabilidad en aplicaciones web.

El hecho de que emplee JavaScript nos permite perder menos tiempo.

Además, nos instalaremos el framerwork ***express*** ya que nos permitirá manejar peticiones con HTTP en diferentes rutas, URL.

Para la instalación de esta dependencia hemos utilizado la línea de comando en la CMD, ***npm i -D express.***

Una vez creado nuestro servidor, para no tener que pararlo e iniciarlo cada vez que hacemos algún cambio, nos instalamos la dependencia ***nodemon,*** que nos permite lanzarlo una vez y cada que vez que realizamos cambios se reinicia sola, para ello escribimos en la ventana de comandos, ***npm i -D nodemon***.

La página que consultamos para lo que se necesitemos es (Nodemon, s.f.)

2. Instalación de Node JS.

Para la instalación hemos descargado el instalador Node.js en la página (TheOpenJSFundation, 2023), seguidamente ejecutar el instalador y avanzar con los pasos que este nos indica.

Figura 38: Icono NODE JS (TheOpenJSFundation, 2023)

Una vez que termine esta instalación comprobamos que todo se ha realizado correctamente abriendo la consola de Windows (CMD), escribimos ***node -v*** , y debe aparecer la versión correspondiente que se ha instalado en nuestra máquina.

Y de la misma forma comprobamos la versión NPM (node Packet manager) nos proporcionara acceso a muchos paquetes reutilizables, con el comando ***npm -v,*** debería igual que antes mostrarnos la versión instalada correspondiente.

3. ¿Qué es Visual Studio Code?

Es (IDE) el editor con el que vamos a trabajar, que tiene grandes extensiones que nos van a permitir trabajar con muchas más funcionalidades, como programar con lenguajes diferentes si asi lo necesitamos o agregar alguna extensión propia del lenguaje para que lo reconozca y sea más visible a la hora de desarrollar.

4. Instalación de Visual Studio Code.

Para la instalación del entorno de programación que vamos a utilizar, hemos descargado el instalador de la página (Microsoft, 2023),seguidamente ejecutar el instalador y avanzar con los pasos que este nos indica. Una vez instalado podremos hacer uso de él, e ir agregando extensiones según lo vayamos necesitando.

Figura 39: Icono visual studio Code (Microsoft, 2023)

Además, vamos a tener la sincronización con git para tener nuestros cambios al día y no correr riesgos de perderlos.

5. ¿Qué es Postman?

Es una herramienta que permite realizar peticiones de una manera simple para testear APIs de tipo REST. Postman sirve para múltiples tareas dentro de las cuales destacaremos en esta oportunidad las siguientes:

- Testear colecciones o catálogos de APIs tanto para Frontend como para Backend.
- Organizar en carpetas, funcionalidades y módulos los servicios web.
- Permite gestionar el ciclo de vida (conceptualización y definición, desarrollo, monitoreo y mantenimiento) de nuestra **API**.
- Generar documentación de nuestras APIs.
- Trabajar con entornos (calidad, desarrollo, producción) y de este modo es posible compartir a través de un entorno cloud la información con el resto del equipo involucrado en el desarrollo.

Los métodos más utilizados, los cuales nos permiten realizar los testeos son;

- GET: obtener información
- POST: agregar información
- PUT: reemplazar información
- DELETE: Borrar información

Todos estos detalles son de (OpenWebinars, 2023)

6. Instalación de Postman.

Figura 40: Icono Postman (Postman, 2023)

Para la instalación de Postman, hemos descargado el instalador de la página (Postman, 2023),seguidamente ejecutar el instalador y avanzar con los pasos que este nos indica.

Una vez instalado abriremos y podremos ir haciendo pruebas según requiramos una opción u otra

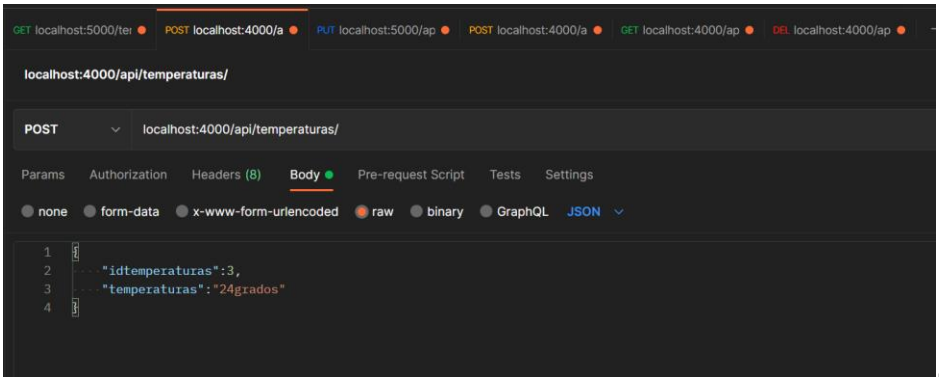


Figura 41: Ejemplo de uso de Postman

7. Introducción a las bases de datos y selección del tipo adecuado para nuestro proyecto

Antes de decidir qué tipo de base de datos utilizaremos para nuestro proyecto, es importante comprender qué es una base de datos y los diferentes tipos disponibles. Una base de datos es un conjunto organizado de datos que se almacena y se accede electrónicamente. Hay varios tipos de bases de datos, incluyendo bases de datos relacionales y no relacionales. Cada tipo tiene sus propias fortalezas y debilidades y puede ser adecuado para diferentes situaciones. En nuestro caso, evaluaremos cada tipo y elegiremos la opción que mejor se adapte a las necesidades específicas de nuestro proyecto.

Veamos la definición de los tipos de bases de datos que hemos mencionado:

- **Bases de datos relacionales:** Organizan los datos en tablas con filas y columnas y utilizan el lenguaje de consulta estructurado (SQL) para manipular los datos. Las tablas están relacionadas entre sí mediante claves foráneas.

- **Bases de datos no relacionales:** No utilizan el modelo relacional y no se basan en tablas con relaciones entre ellas. Utilizan una variedad de modelos de datos, como el almacenamiento basado en documentos, en columnas, en clave-valor o en grafos.

Vamos a ver las principales características de cada una:

1. **Bases de datos relacionales:**
   - Organiza los datos en tablas con filas y columnas.
   - Las tablas están relacionadas entre sí mediante claves foráneas.
   - Utiliza el lenguaje de consulta estructurado (SQL) para manipular los datos.
   - Garantiza la integridad y la consistencia de los datos mediante restricciones y transacciones.
   - Ejemplos: MySQL, Oracle, SQL Server.

2. **Bases de datos no relacionales:**
    - ❖ No utiliza el modelo relacional y no se basa en tablas con relaciones entre ellas.
    - ❖ Utiliza una variedad de modelos de datos, como el almacenamiento basado en documentos, en columnas, en clave-valor o en grafos.
    - ❖ No utiliza SQL para manipular los datos y tiene su propio lenguaje de consulta.
    - ❖ Puede manejar grandes cantidades de datos no estructurados y es altamente escalable.
    - ❖ Ejemplos: MongoDB, Cassandra, Redis.

Para la elección entre estos dos tipos de base de datos va a depender de las necesidades específicas de nuestro proyecto.

- **Base de datos relacional:**
    - Si necesitamos garantizar la integridad y la consistencia de datos, una base de datos relacional puede ser una opción ya que ofrece restricciones y transacciones para garantizar la coherencia de los datos.
    - Si necesitamos que realice consultas complejas y relaciones entre múltiples tablas, una base de datos relacional puede ser más fácil de usar y optimizar.
    - Si estamos familiarizados con el lenguaje de consultas estructurados (SQL), puede ser más fácil de aprender y utilizar.

- **Base de datos no relacional:**
    - Si necesitamos recopilar y almacenar grandes cantidades de datos no estructurados, una base de datos no relacional podría ser entonces una buena opción ya que es altamente escalable y puede manejar datos no estructurados.
    - Si necesitamos un alto rendimiento para operaciones de lectura y escritura en tiempo real, esta puede ofrecernos un rendimiento más rápido.
    - Si nuestro esquema de datos es flexible y cambia con frecuencia, esta puede ser más fácil de usar ya que no requiere un esquema fijo.

Después de analizar las características de ambas bases de datos relacionales y no relacionales, consideramos las necesidades específicas de nuestro proyecto. Este implica la recopilación y el almacenamiento de datos de sensores en tiempo real para la domotización de una casa. Hemos determinado que una base de datos no relacional como MongoDB o Redis podría ser una buena opción. Ambas son conocidas por su capacidad para manejar grandes cantidades de datos no estructurados y por su escalabilidad.

MongoDB es una base de datos basada en documentos que es fácil de usar, mientras que Redis es una base de datos en memoria basada en clave-valor que es conocida por su velocidad y su capacidad para manejar datos en tiempo real.

En los dos siguientes puntos de nuestro TFG, explicaremos cómo instalar tanto una base de datos relacional como una no relacional. Aunque finalmente trabajaremos solo con una de ellas en nuestro proyecto, creemos que es importante entender cómo instalar y configurar ambas opciones. Esto nos permitirá tomar una decisión informada sobre qué base de datos utilizar y también nos brindará la flexibilidad de cambiar de opción en el futuro si es necesario.

8. ¿Qué es MySQL Workbench?

MySQL Workbench es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y administradores de bases de datos (DBA). MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración completas para la configuración del servidor, administración de usuarios, copias de seguridad y mucho más. MySQL Workbench está disponible en Windows, Linux y Mac OS X. (mysql, 2023)

9. Instalación de MySQL Workbench

Como base de datos relacional vamos a utilizar MySQL Workbench.

Para la instalación de MySQL, hemos descargado el instalador de la página (MYSQL, 2023) , seguidamente ejecutar el instalador y avanzar con los pasos que este nos indica. En esta instalación se han tenido que seguir unos pasos específicos y como requisito tener visual studio instalado. Veamos algunos de los pasos más importantes que hemos tenido que aplicar para la creación de nuestra base de datos:

Figura 42: Icono MySQL

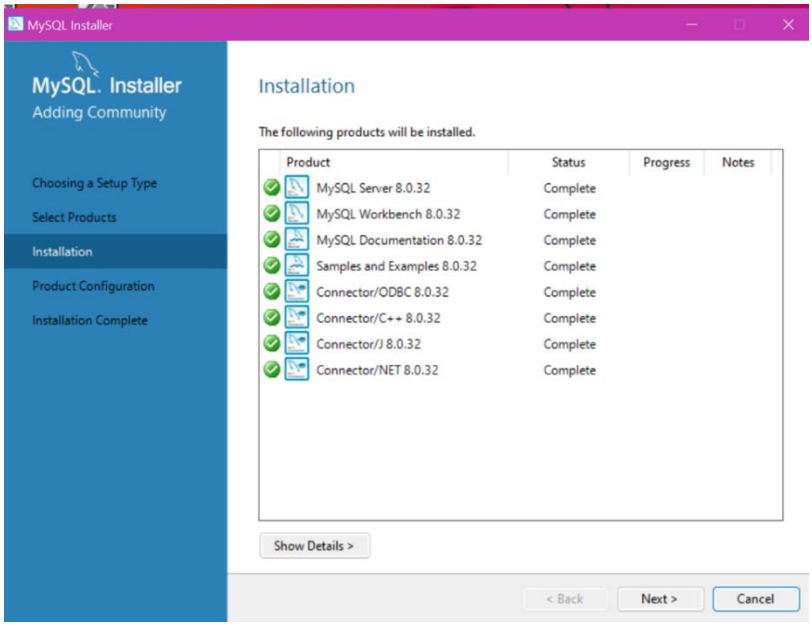


Figura 43: Proceso de instalación MySQL

Una vez seleccionado los productos que queremos que se instalen, debemos esperar a que salga completado y en verde como podemos ver en la figura N.º 43.

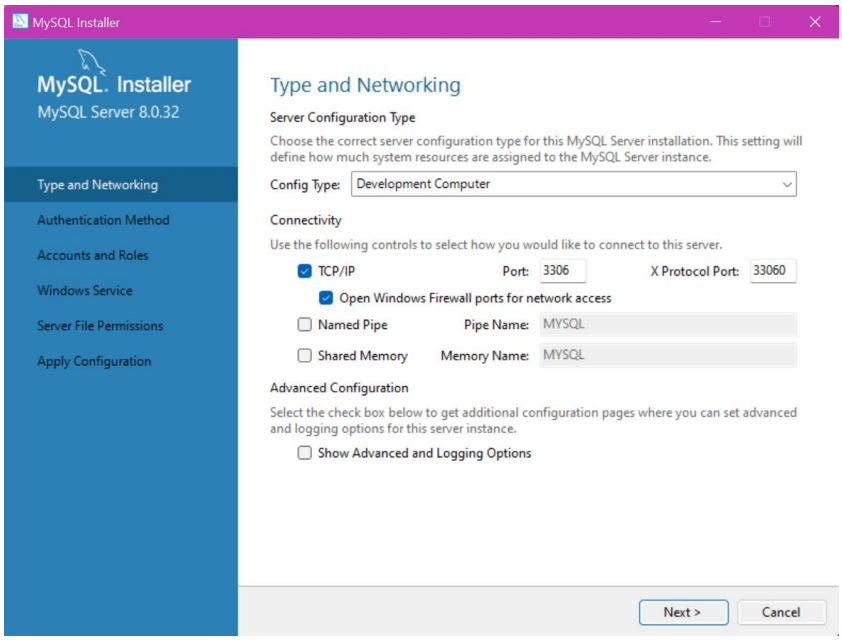


Figura 44: Proceso instalación MySQL comprobación puertos

En este paso comprobamos los puertos que se van a utilizar y mediante que protocolo, que en nuestro caso es TCP/IP. Tal y como vemos en la figura N.º 44.

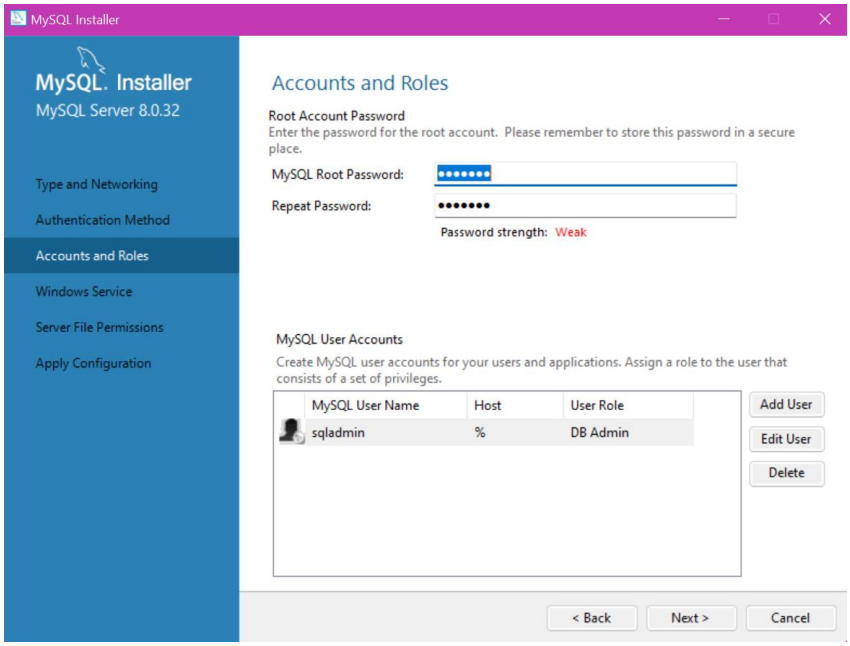


Figura 45: Proceso instalación MySQL establecemos usuario y contraseña

Como vemos en la figura N.º 45, debemos establecer un usuario y una contraseña para poder acceder a nuestra base de datos. Podemos tener varios usuarios no tiene que ser sola y exclusivamente uno.

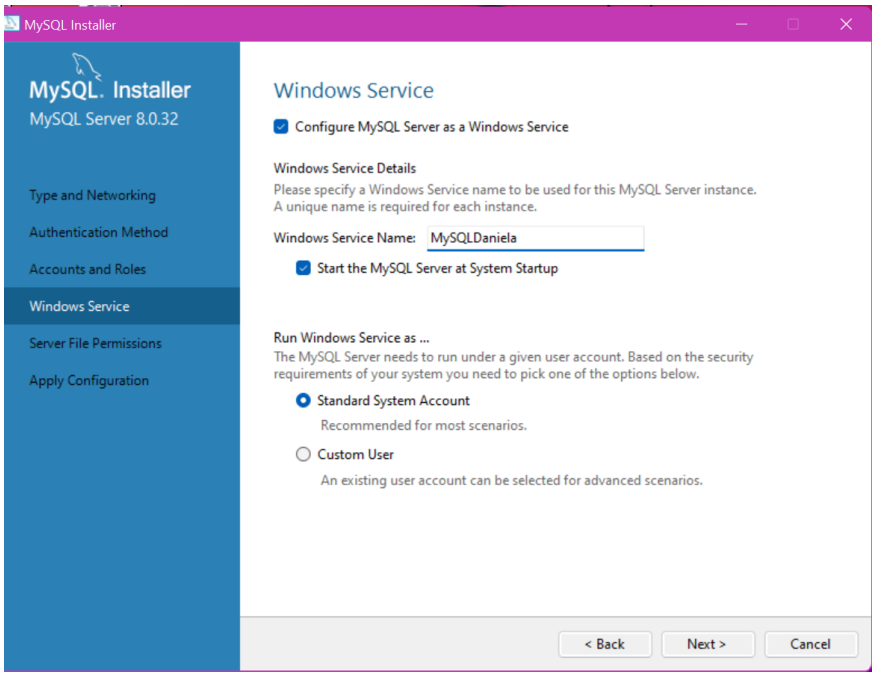


Figura 46: Proceso instalación MySQL establecemos el nombre del servicio

En este punto de la figura 46, establecemos el nombre con el que queremos que se quede nuestro servicio.

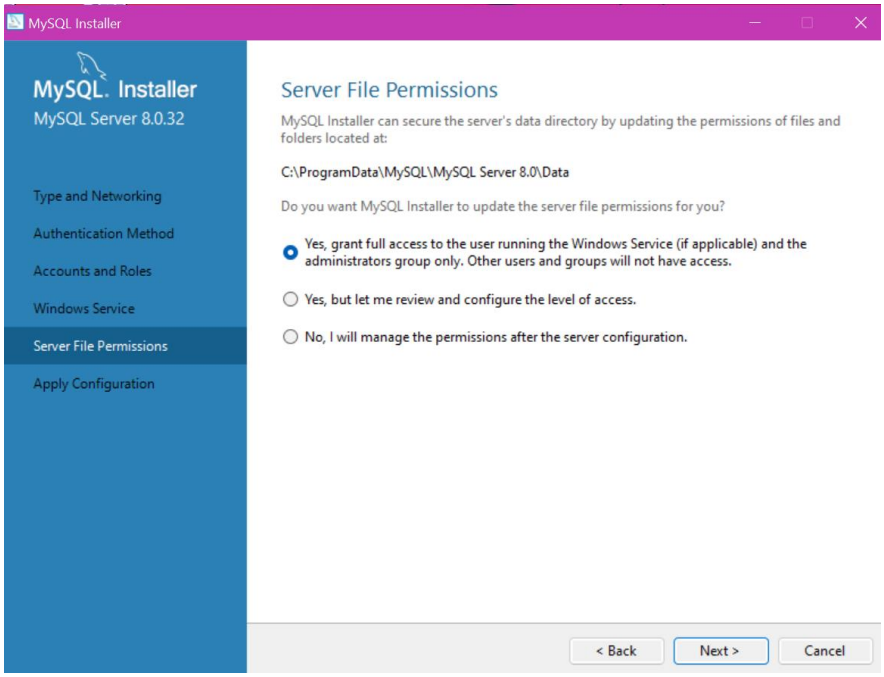


Figura 47: Proceso instalación MySQL aceptamos permisos

Casi finalizando, figura N.ª 47, nos pregunta: ¿Quieres que el instalador de MYSQL actualice los permisos del archivo del servidor por ti?, y en nuestro caso le hemos dicho que sí.

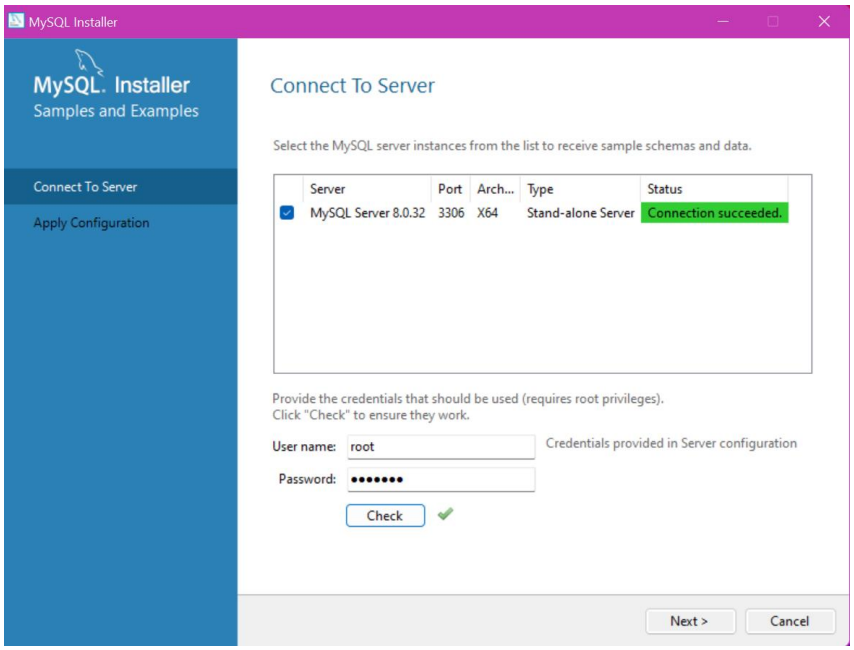


Figura 48: Proceso instalación MySQL comprobamos la conexión

Por último, como vemos en la figura N.º 48, comprobamos que nuestra conexión sea correcta después de toda la configuración anterior, y finalizamos.

10. ¿Qué es MongoDB?

Figura 49: Icono MongoDB

MongoDB es una base de datos NoSQL orientada a documentos. Esto significa que en lugar de guardar los datos en registros y tablas como en las bases de datos relacionales tradicionales, MongoDB guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON. Una de las ventajas de este enfoque es que no es necesario seguir un esquema rígido: los documentos de una misma colección pueden tener esquemas diferentes. Esto proporciona una gran flexibilidad y escalabilidad a la hora de trabajar con datos.

MongoDB está diseñada para ser distribuida en su núcleo, lo que significa que la alta disponibilidad, la escalabilidad horizontal y la distribución geográfica están integradas y son fáciles de usar. Además, ofrece un modelo de consultas e indexación avanzado que permite acceder y analizar los datos de manera potente y eficiente.

Es una excelente opción para trabajar con grandes volúmenes de datos y para aplicaciones que requieren alta disponibilidad y escalabilidad horizontal. Por esta razón, hemos decidido utilizar esta base de datos en nuestra aplicación.

(MongoDB, MongoDB, 2023)

11. Instalación de MongoDB

Como hemos mencionado anteriormente, para nuestro proyecto utilizaremos MongoDB como nuestra base de datos no relacional. A continuación, vamos a explicar el proceso de instalación de MongoDB en nuestro equipo. Para instalarlo, debemos dirigirnos a la página oficial de MongoDB (MongoDB, 2023) y descargar el instalador de la versión más reciente. Una vez que hayamos completado la descarga, ejecutamos el instalador y seguimos los pasos indicados para completar la instalación de MongoDB en nuestro equipo. Es importante seleccionar la versión más reciente para asegurarnos de tener acceso a las últimas funcionalidades y mejoras de seguridad.

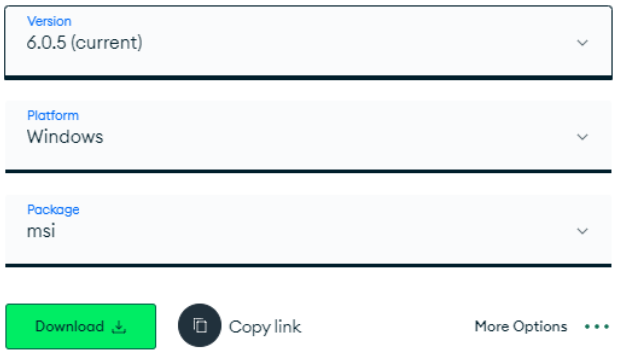


Figura 50: Descarga mongo DB (MongoDB, 2023)

Figura 51: Instalación MongoDB ventana de inicio

Pulsamos siguiente y aceptamos las condiciones hasta la figura N.º 51 que se muestra a continuación.

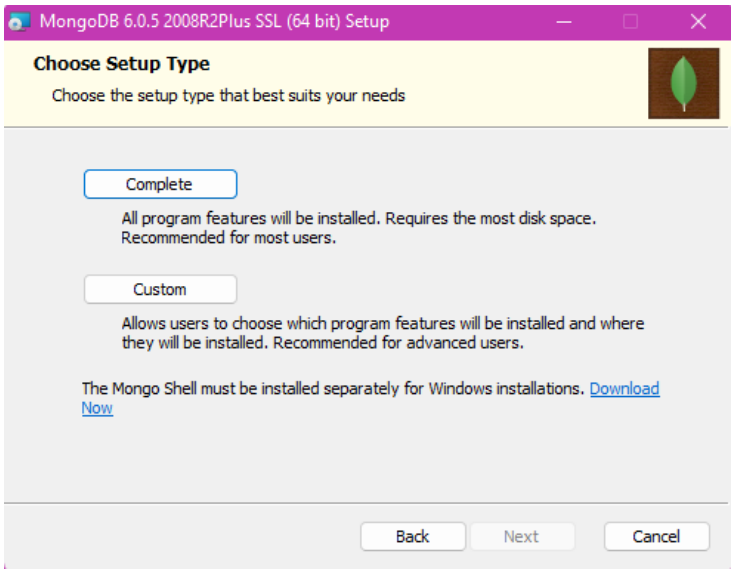


Figura 52: Instalación MongoDB completa o personalizada

En la figura N.º 52 Durante el proceso de instalación de MongoDB, se nos presentará la opción de elegir entre una instalación completa o personalizada. En este caso, es recomendable elegir la opción de instalación completa para asegurarnos de que se instalen todos los componentes necesarios para el correcto funcionamiento de MongoDB.

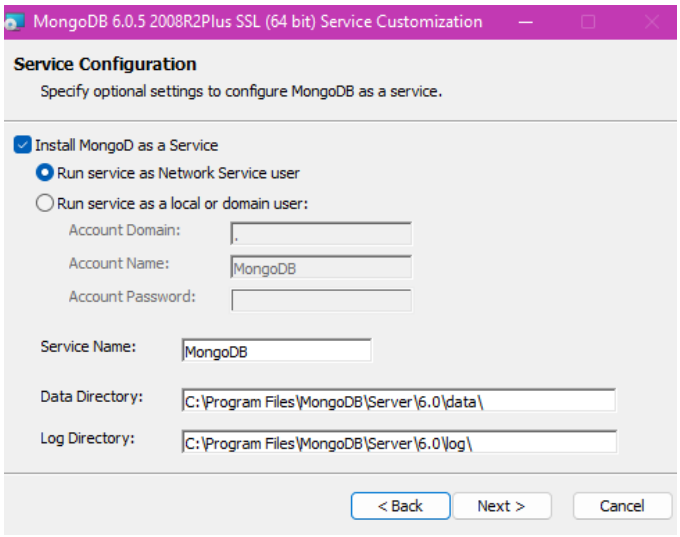


Figura 53: Instalación MongoDB configuración del servicio

Durante la instalación de MongoDB, se presentan algunos ajustes preconfigurados. Si se dejan estos ajustes por defecto, MongoDB se instalará como un servicio en nuestro ordenador y se iniciará automáticamente cada vez que encendamos el equipo. Sin embargo, en nuestro caso no es necesario que MongoDB se inicie automáticamente, por lo que podemos desmarcar esta opción y así tendremos que iniciar MongoDB manualmente cada vez que lo necesitemos.

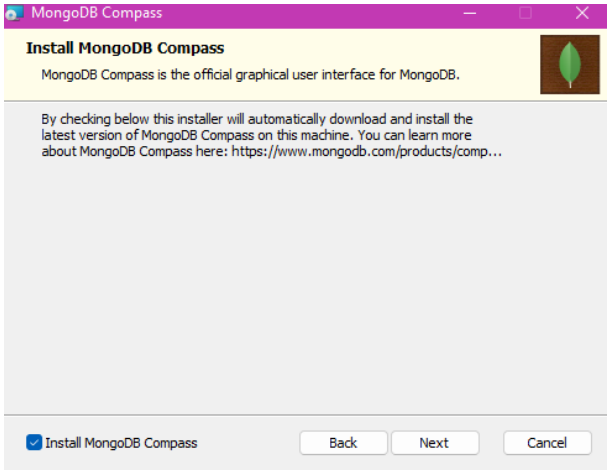


Figura 54: Instalación MongoDB compass

Durante el proceso de instalación de MongoDB, concretamente en la figura N.º 53 se nos presenta la opción para instalar MongoDB Compass. Esta opción viene marcada por defecto y es recomendable dejarla seleccionada. MongoDB Compass es una herramienta gratuita que proporciona una interfaz gráfica de usuario (GUI) para interactuar con nuestra base de datos. Con ella, podemos consultar, optimizar y analizar nuestros datos de manera interactiva. Además, Compass nos permite obtener

información clave sobre nuestros datos y desarrollar canales de agregación mediante su función de arrastrar y soltar. (MongoDB, MongoDB, 2023)

Por último, le damos instalar tal y como se muestra en la figura N.º 54.

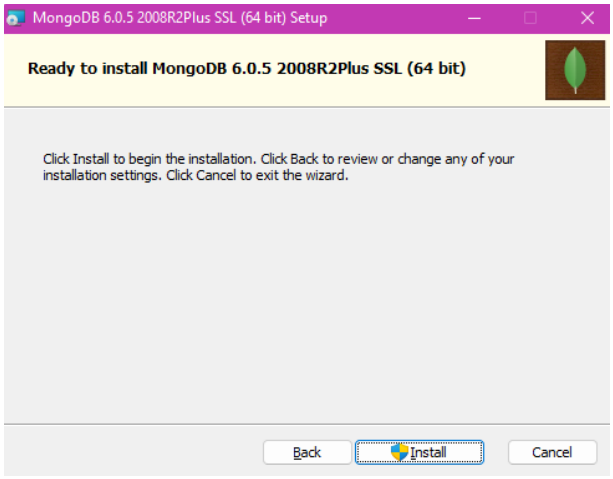


Figura 55: Instalación MongoDB aceptamos la instalación

El proceso de instalación puede tardar unos minutos en completarse. Una vez finalizado, se abrirá automáticamente una pantalla de MongoDB Compass con una conexión preconfigurada a localhost.

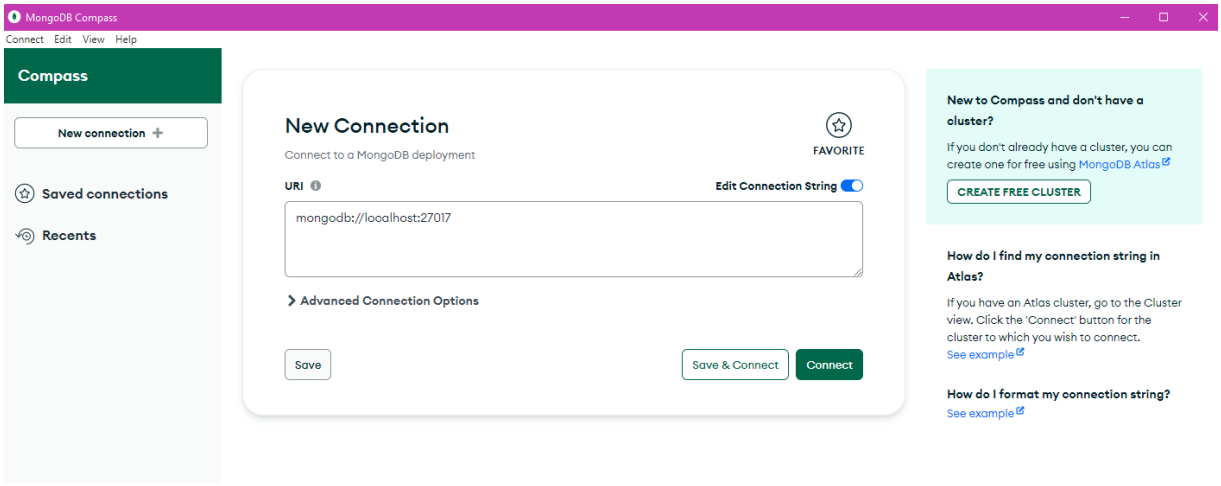


Figura 56: Pantalla inicio compass MongoDB

Para ejecutar MongoDB correctamente, debemos seguir los siguientes pasos: Primero, debemos crear una carpeta llamada ‘data’ en el directorio raíz de nuestro disco local C. Dentro de esta carpeta ‘data’, debemos crear otra carpeta llamada ‘db’. Esto es necesario para que MongoDB pueda almacenar sus datos correctamente. Luego, debemos dirigirnos a la carpeta **‘Archivos de programa/MongoDB/bin’** y ejecutar el servidor de MongoDB y el cliente. Si estamos utilizando una versión 6 o

superior de MongoDB, es necesario instalar el 'Mongo Shell' ya que ya no viene incluido en el bin por defecto. Una vez que hayamos arrancado el servidor y el cliente de MongoDB, podremos configurar nuestro servidor para permitir conexiones.

12. Instalación de Angular

Para instalar Angular en nuestro sistema, podemos encontrar toda la información necesaria en la página oficial de Angular (Google, Angular, 2023).Para llevar a cabo la instalación, utilizaremos la línea de comandos (CMD) y ejecutaremos las siguientes instrucciones:

Figura 57: Icono Angular (Google, Angular, 2023)

- **1.** Abrimos el terminal (CMD) y ejecutamos la siguiente línea de comando para la instalación de Angular CLI:
  - ✓ ***npm install -g @angular/cli***
- ***2.*** Una vez que hayamos instalado Angular CLI, comprobamos la version y asi vemos que se instaló correctamente, para eso ejecutamos la siguiente línea de comando:
  - ✓ ***ng version***

  Y veremos el siguiente resultado:

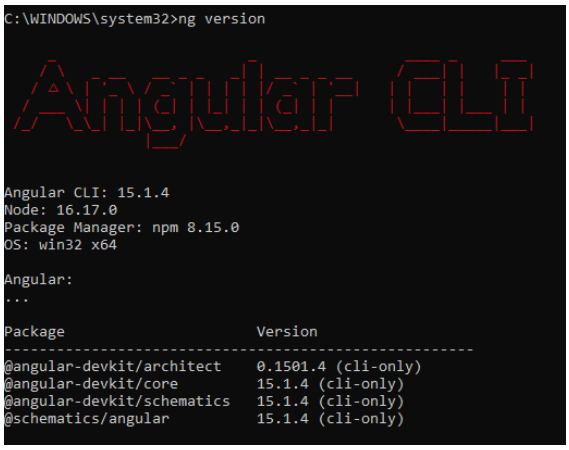


Figura 58: Instalación Angular

- **3.** Creamos el espacio de trabajo con el siguiente comando:
  - ✓ ***ng new my-app***
  - ✓ Seguidamente entramos en el con el siguiente comando:
  - ✓ ***cd my-app***
- **4.** Para ejecutar la aplicación una vez dentro, ejecutamos el siguiente comando:
  - ✓ ***ng serve***

13. ¿Qué es Angular CLI?

Angular CLI es una herramienta de línea de comandos desarrollada por el equipo de Angular para facilitar el proceso de creación y mantenimiento de aplicaciones Angular. Esta herramienta nos permite realizar una amplia variedad de tareas, desde la creación de un nuevo proyecto hasta la generación de componentes y servicios, así como la ejecución de pruebas y la preparación de nuestra aplicación para su despliegue en producción.

Una de las principales ventajas de utilizar Angular CLI es que nos permite mantener una estructura y organización consistente en nuestros proyectos, lo que facilita su mantenimiento y escalabilidad. Además, al automatizar muchas tareas comunes, podemos ahorrar tiempo y reducir la posibilidad de errores.

En resumen, Angular CLI es una herramienta muy útil que utilizaremos como nuestro framework para desarrollar la parte del front de nuestra aplicación. Esto incluye elementos como botones, lógica, tablas y otros aspectos estéticos.

14. Arduino

Arduino IDE es un software de código abierto diseñado por Arduino.cc para escribir, compilar y cargar código en las placas de Arduino. Es el software oficial de Arduino y hace que la

2021)

compilación de código sea muy fácil, incluso para personas sin conocimiento técnicos previos.

El IDE de Arduino proporciona un entorno integrado para escribir código en el lenguaje de programación de Arduino y cargarlo en las placas de Arduino. También incluye una serie de bibliotecas y ejemplos que pueden ayudarte a empezar con tus proyectos.

En resumen, el IDE de Arduino es una herramienta esencial para cualquier persona que quiera trabajar con diferentes placas y desarrollar proyectos con ellas. (AndProf, 2021).

15. Git

Git es un sistema de control de versiones distribuido que se utiliza para rastrear cambios en cualquier conjunto de archivos de computadora. Es comúnmente utilizado para coordinar el trabajo entre programadores que desarrollan código fuente de manera colaborativa durante el desarrollo de software.

Figura 60: Icono GIT (Git, s.f.)

Git permite a los desarrolladores ver la línea de tiempo completa de sus cambios, decisiones y progreso en cualquier proyecto en un solo lugar. Al utilizar Git, los desarrolladores pueden trabajar juntos en un proyecto y mantener un historial completo y detallado de todos los cambios realizados en el código fuente.

En resumen, Git es una herramienta esencial para cualquier persona que trabaje en el desarrollo de software y quiera mantener un control riguroso sobre el código fuente de sus proyectos. (Git, s.f.).

7. Desarrollo del Backend de la aplicación web.

7.1 Creación del Backend

La realización de nuestro Backend la vamos a realizar con nodejs, y trabajaremos con el IDE visual studio code, y además tendremos nuestro git sincronizado para que podamos ir haciendo commits y guardando nuestros cambios en nuestro repositorio.

Para la creación de nuestro Backend nos situamos en la carpeta donde vamos a tener ubicados el Backend y el Frontend, en nuestro caso no situamos en la carpeta Automatización y crearemos una llamada SERVER, que será nuestro Backend.

En primer lugar, para inicializar nuestro proyecto, abrimos una consola de comandos, puede ser la que viene integrada en el visual studio code, o la misma del sistema, y dentro de la carpeta creada, en nuestro caso SERVER, tecleamos el siguiente comando:

- ✓ ***npm init --y***,
  - Este comando se utiliza para crear un archivo **package.json** con valores predeterminados en un proyecto de Node.js. El archivo **package.json** es un archivo importante en cualquier proyecto de Node.js ya que contiene información sobre el proyecto y sus dependencias. Al ejecutar este comando con la opción **--y**, se aceptan automáticamente los valores predeterminados para la creación del archivo, lo que agiliza el proceso de configuración del proyecto.

A continuación, creamos el archivo index.js. Este archivo generalmente maneja el inicio de la aplicación, el enrutamiento y otras funciones de la aplicación. Para ejecutar este archivo recién creado, debemos ingresar lo siguiente en nuestra consola:

- ✓ ***node index.js***

Nuestra aplicación se desarrollará en TypeScript en lugar de JavaScript. Antes de continuar, veamos en qué se diferencian. TypeScript es un superconjunto de JavaScript que agrega características adicionales como el tipado estático. Esto permite a los desarrolladores definir tipos de variables y detectar errores en tiempo de compilación en lugar de en tiempo de ejecución. Aunque TypeScript tiene una curva de aprendizaje más difícil que JavaScript, es conocido como un lenguaje de programación orientado a objetos y ofrece soporte para módulos, lo que facilita la organización del código en diferentes archivos y carpetas para su mantenimiento y reutilización.

Para su instalación tecleamos en nuestra consola el siguiente comando:

- ✓ ***npm install typescript --save-dev.***
  - save-dev indica que solo la vamos a utilizar desarrollando, entonces indica dependencia desarrollo.

Esta instalación creará una carpeta llamada **node_modules** y agregará automáticamente TypeScript como una dependencia de desarrollo en nuestro archivo **package.json**.

Para inicializar TypeScript tecleamos en consola lo siguiente:

- ✓ ***tsc –init***
  - Se utiliza para inicializar un proyecto de TypeScript. Este comando crea un archivo ***tsconfig.json*** en el directorio actual, que contiene la configuración predeterminada para el compilador de TypeScript.

Para trabajar con TypeScript, creamos un archivo llamado **index.ts** y escribimos nuestro código en él.

Sin embargo, este archivo no se ejecutará directamente como lo hace **index.js.** En su lugar, el código que escribimos en **index.ts** se transcribirá a JavaScript y se ejecutará

en **index.js**. Para hacer esto, ejecutamos el comando **tsc**, que transcribe nuestro código TypeScript a JavaScript.

Cada vez que realicemos cambios en nuestro código TypeScript, debemos ejecutar el comando **tsc** y luego ejecutar **index.js.** Sin embargo, para evitar tener que hacer esto manualmente cada vez que introduzcamos cambios, podemos hacer lo siguiente en la consola:

- ✓ **tsc --watch**
  - Este comando se utiliza para ejecutar el compilador en TypeScript en modo observación , esto quiere decir que el compilador se queda observando los cambios en nuestros archivos y cada modificación ejecuta la transpilación.
  - La transpilación es un caso particular de la compilación, es decir, todo transpilador es también un compilador.
    - Hay compiladores que no se pueden considerar transpiladores.
    - La característica que convierte un compilador en transpilador es la relación entre los lenguajes origen y destino de la traducción. Si el compilador traduce código entre dos lenguajes que están al mismo nivel de abstracción entonces, estamos ante un transpilador si no, no lo es.

Por tanto, al ejecutar el comando se quedará asi:

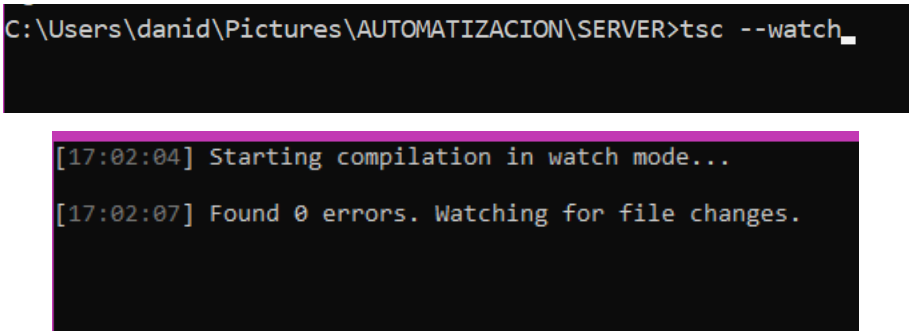


Figura 61: Consola en modo escucha

A continuación, abriremos otra en la que seguir trabajando y únicamente ejecutar nuestro fichero index.js.

Pero para no tener que estas ejecutando esto cada vez, podemos teclear el siguiente comando:

- ✓ **nodemon index.js** (o la ruta que sea en tal caso, con ***nodemon*** delante)
  - Nodemon es una herramienta útil para el desarrollo de aplicaciones en Node.js. Su función principal es reiniciar automáticamente la aplicación cuando se detectan cambios en los archivos del directorio. Esto facilita el proceso de desarrollo ya que no requiere ningún cambio adicional en el código.

Tendremos entonces por un lado escuchando los cambios y por el otro ejecutando:

```
C:\Users\danid\Pictures\AUTOMATIZACION\SERVER>nodemon ./dist/index.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/index.js`
```

Figura 62: Consola en ejecución comando nodemon

A partir de aquí debemos abrirlo una nueva terminal, si necesitamos realizar más instalaciones.

Para organizar mejor nuestro código y hacerlo más legible, podemos crear una carpeta llamada **dist** y mover nuestro archivo **index.js** allí.

Para indicar dónde debe transcribirse nuestro código TypeScript, podemos ir al archivo **tsconfig.json** y agregar la ruta a la opción **outDir** de la siguiente manera**: "outDir": "./dist".**

También podemos crear una carpeta llamada **src** y mover nuestro archivo **index.ts** allí. Como resultado de estos cambios, para volver a ejecutar nuestro archivo, debemos ingresar el siguiente comando en la consola:

- ✓ **node dist/index.js**

El siguiente paso es crear una carpeta dentro de src llamadas ***models*** a su vez dentro un archivo llamado ***server.ts*** en el cual crearemos una clase llamadas Server , y al final de esta hacemos un ***export default Server,*** esto es para poder utilizar esta clase en las que queramos haciendo un:

import Server from "./models/server" , o simplemente al escribir Server ya el mismo IDE nos hacemos el import.

7.2 Instalación framework, middleware, bases de datos y módulos.

A continuación, vamos a instalar cuatro paquetes de Node.js:

- ***Express:*** es un ***framework*** para construir aplicaciones web en Node.js.
- ***Dotenv*** es un módulo que carga variables de entorno desde un archivo ***.env*** en el proyecto.
- ***Cors*** es un paquete que proporciona un ***middleware*** para habilitar CORS (intercambio de recursos de origen cruzados) en nuestra aplicación, ya que nuestro Backend y nuestro Frontend se van a ejecutar en dominios diferentes y para que estos dos se puedan comunicar vamos a tener que configurarlos.
- ***MySQL*** o ***MongoDB*** es un paquete que nos permitirá conectar y trabajar con bases de datos MySQL o MonogDB desde nuestra aplicación Node.js

Para todo esto mencionado , ejecutamos en la consola:

- ✓ **npm install express dotenv cors mysql** (si usamos mysql)
- ✓ **npm install express dotenv cors mongodb** (si usamos mongodb)

Esta instalación nos actualizará automáticamente nuestro package.json con las nuevas dependencias.

Importamos en nuestro fichero server.ts, express de la siguiente forma:

- **import express from “express”**

Y en la consola debemos importar el tipo de la siguiente forma:

- **npm -i --save-dev @type/express.**

Esto instala el paquete ***@types/express*** como una dependencia de desarrollo en tu proyecto de Node.js. Este paquete proporciona definiciones de tipos para el paquete ***express***, lo que permite a los desarrolladores que utilizan TypeScript obtener ayuda de autocompletado y verificación de tipos al usar express en su código.

- Creamos una variable privada dentro de nuestra clase del fichero server.ts
- private app: express.Application
- Y en el constructor hacemos lo siguiente: this.app = express()

- Creamos otra que será el puerto, private port: string;
- Y en el constructor this.port = 'nuestroPuerto'
- Creamos otro método llamado listen, para que escuche nuestro puerto, le pasamos el puerto y el callback.

```
listen(){
    this.app.listen(this.port, () =>{
        console.log ('Aplicacion andando por el puerto ', this.port);
    });
}
```

Figura 63: Método listen del servidor

Para gestionar la configuración de nuestra aplicación en diferentes entornos de manera más sencilla, podemos utilizar dotenv. Este módulo de Node.js nos permite cargar variables de entorno desde un archivo ***.env*** en ***process.env.*** De esta manera, podemos evitar tener que especificar valores como el puerto directamente en el código fuente, lo que puede ser propenso a errores.

Después de la instalación comentada anteriormente, creamos un archivo .env en la raíz de nuestro proyecto y especificamos las variables de entorno utilizando la sintaxis ***NOMBRE_VARIABLE=valor***.

Finalmente, para cargar las variables de entorno desde el archivo ***.env***, importamos dotenv en nuestro archivo ***index.ts*** y ejecutamos su método ***config***. Esto cargará las variables de entorno en ***process.env***, permitiéndonos acceder a ellas desde cualquier parte de nuestro código utilizando la sintaxis ***process.env.NOMBRE_VARIABLE.***

8. Desarrollo del Frontend de la aplicación web.

8.1 Creación del Frontend

Para crear nuestro Frontend, nos ubicamos en la carpeta previamente creada para el Backend llamada “automatización”. Una vez allí, escribimos en nuestra consola de comandos ***new ng Frontend***.

Posteriormente, se nos consultara si queremos agregar el enrutamiento de Angular. En este caso, seleccionamos la opción de no agregarlo (esto se explicará más adelante). Finalmente, se nos preguntará qué estilo queremos usar y elegiremos CSS.

Con estos pasos, se prepara automáticamente toda la estructura de nuestro proyecto. Una vez este creada la carpeta con todo lo comentado, nos situamos dentro de ella para comenzar a trabajar.

Ahora debemos abrir nuestro IDE , en nuestro caso Visual Studio Code. Una vez dentro, abrimos una consola de comandos, ya sea la integrada en Visual Studio Code o una consola de comandos de Windows, siempre y cuando no situemos en nuestra carpeta para poder continuar trabajando.

Para ejecutar nuestro proyecto, escribimos en la CMD ***ng serve - -o.*** Esto significa que ***ng serve*** inicia el servidor, observa los archivos y reconstruye la aplicación a medida que realizamos cambios en esos archivos, y ***- - o*** es el diminutivo de open ,es decir, abre automáticamente nuestro navegador en ***http://localhost:4200*** .

Una vez hecho esto , nos preguntara si queremos compartir datos de forma anónima con el equipo Angular, en nuestro caso elegimos la opción “sí”.

La primera vez tardará un poco más en procesar la compilación. Esperamos un poco y se nos abrirá automáticamente la pantalla del navegador con lo que ya teníamos configurado de manera automáticamente.

8.2 Creación de la estructura del proyecto

Para empezar, crearemos una carpeta llamada ***"components"*** dentro de ***"app".***

Luego, en nuestra CMD introducimos el comando "***ng g c components".***

Este es un comando en angular que se utiliza para generar un nuevo componente llamado "components".

Este comando ***"ng"*** es la abreviatura de angular , ***"g"*** es la abreviatura de "generate" (generar) y ***"c"*** es la abreviatura de "component" (componente).

Si a nuestro comando le agregamos al final ***/list-"miComponente",*** se crearán los archivos ***.html, .css*** y ***.ts*** para nuestra componente. Además, crearemos otro componente llamado ***"agregar-editar-miComponente*** que reutilizaremos como modal para agregar o editar nuestra componente.

También crearemos una carpeta llamada "***services"*** al mismo nivel que la carpeta "components".

Esta carpeta se utiliza principalmente para tres cosas:

1. realizar peticiones al servidor mediante HTTP,
2. reutilizar código entre componentes y
3. compartir datos entre componentes

En nuestro caso utilizaremos "services" para la primera opción.

Podemos utilizar el comando ***ng g - - help*** para obtener información sobre cómo usar el comando ***ng generate*** y sus subcomandos. Este comando resulta de gran utilidad cuando trabajamos con angular y necesitamos generar o modificar archivos basados en un esquema. Además, el comando ***ng help*** muestra una lista de los comandos disponibles y sus descripciones breves en consola. Para obtener ayuda con comandos individuales, se puede usar la opción ***- - help o -h*** con el comando en cuestión.

A continuación, vamos a generar entonces el servicio de la siguiente forma, tecleamos en nuestra consola de comandos ***ng g s services/nuestroNombre.*** Este comando es una abreviatura de ***ng generate service services/nuestroNombre.***

Este comando se utiliza para generar un servicio en Angular. Un servicio es una clase con propósito bien definido que proporciona funcionalidad específica para una aplicación. Los servicios pueden ser inyectados en componentes y otros servicios, lo que permite reutilizar código y mantener la separación de responsabilidades. Al generar un servicio con este comando, se crea automáticamente un archivo de servicio con el nombre especificado en la carpeta ***services*** y agrega una clase decorada con ***@Injectable()*** que permite que el servicio sea inyectado en otras partes de la aplicación. Este decorador se utiliza para marcar una clase como disponible para ser proporcionada e inyectada como una dependencia. Esto significa que la clase puede ser inyectada en otras partes de la aplicación donde se necesite, como en componente o en otros servicios.

A continuación, creamos dos carpetas al mismo nivel que la carpeta services: una llamada ***shared*** y otra llamada ***interfaces.*** En la carpeta shared, crearemos más adelante un módulo en el que configuraremos todo lo relacionado con Angular Material. En la carpeta interfaces, como su nombre indica, almacenaremos las interfaces correspondientes a nuestro proyecto.

Por otro lado, vamos a necesitar Angular Material. Veamos qué es y para qué nos servirá.

8.3 Angular Material

Angular Material es una biblioteca de componentes de interfaz de usuario para Angular que implementa patrones de interacción comunes según la especificación de Material Design. Ofrece componentes de alta calidad, internacionalizados y accesibles para todos. Está bien probado para garantizar el rendimiento y la fiabilidad. Se integra perfectamente con Angular y se puede agregar a nuestro proyecto utilizando esquemas para generar rápidamente vistas con componentes de Material Design.

Angular Material ofrece una amplia gama de componentes que implementan patrones de interacción comunes según la especificación de Material Design. Algunos de los componentes que ofrece son: controles de formulario que recopilan y validan la entrada del usuario, navegación como menús, barras laterales y barras de herramientas que organizan su contenido, diseño como bloques de construcción esenciales para presentar su contenido, botones e indicadores como botones,

interruptores, indicadores de estado y progreso, ventanas emergentes y modales como componentes flotantes que se pueden mostrar u ocultar dinámicamente y tabla de datos para mostrar e interactuar con datos tabulares.

Para instalar Angular Material, nos dirigimos a su página oficial (Google P. b., 2010-2023). Una vez alli, podemos ver todos los pasos para su instalación haciendo clic en el botón ***"Get started"*** tal y como podemos ver en la siguiente figura:

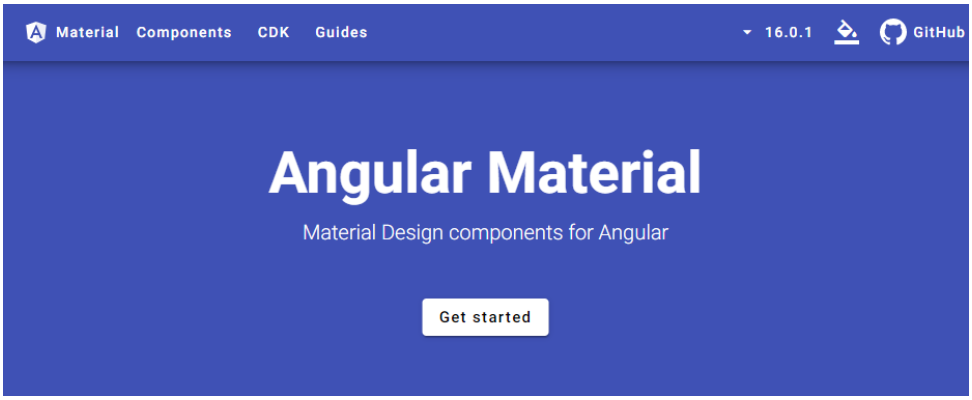


Figura 64: Angular Material (Google P. b., 2010-2023)

Para instalar, Angular Material en nuestro proyecto, debemos ejecutar el siguiente comando ***ng add @angular/material*** en nuestra consola.

Al hacerlo, nos harán tres preguntas :

1- Se nos preguntará la versión que deseamos instalar.
2- Se nos preguntara el tema que queremos elegir. Tenemos cuatro opciones por defecto , pero podremos personalizarlo como nosotros lo queramos:

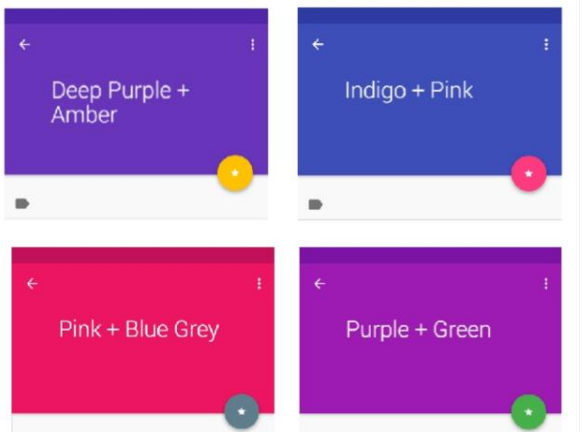


Figura 65: Paleta colores de angular material (Google, Google, 2023)

3- Si queremos configurar la tipografía estilos de Angular Material
4- Si queremos habilitar las animaciones

En nuestro caso a estas cuatro preguntas hemos respondido que sí

Como resultado, se han actualizado los archivos module.ts , angular.json, index.html y style.css.

Podemos encontrar más detalle sobre este proceso en la página mencionada anteriormente.

9. Creación de la API Rest.

9.1 Introducción a la creación de una API REST

Para crear una API REST, primero debemos definir los recursos y modelos que nuestra API va a exponer. Los recursos son las entidades que nuestra API va a manejar, como temperaturas, movimientos etc. Los modelos son las representaciones de esos recursos en nuestro código.

Una vez que definamos los recursos y modelos, debemos utilizar los métodos HTTP para definir las operaciones que se pueden realizar en esos recursos.

Por ejemplo, el método GET para obtener un recurso o una colección de recursos, el método POST para crear un nuevo recurso, el método PUT para actualizar un recurso existente y el método DELETE para eliminar un recurso.

Después de definir las operaciones en nuestros recursos, debemos diseñar y documentar las solicitudes y respuestas de nuestra API. Esto incluye definir los parámetros de entrada y salida de cada operación, así como los códigos de estado HTTP que nuestra API nos va a devolver.

Una vez que hayamos completado este diseño de nuestra API, podemos implementarla utilizando nuestro framework de desarrollo web, Express.js.

Finalmente, debemos probar nuestra API para asegurarnos de que funciona correctamente. Como hemos comentado anteriormente utilizaremos la herramienta Postman para enviar solicitudes a nuestra API y verificar las respuestas.

También es importante que realicemos pruebas automatizadas para asegurarnos de que nuestra API sigue funcionando correctamente a medida que realizamos cambios en el código.

9.2 API desarrollada para cada sensor

Nuestra API REST se encarga de gestionar los datos de los sensores en una base de datos ***MongoDB***. La API define una serie de rutas y operaciones para interactuar con la base de datos y permitir la comunicación entre los sensores y el sistema.

La comunicación con la base de datos MongoDB se realiza a través del módulo ***mongodb*** de Node.js. Este módulo proporciona una serie de métodos y clases para interactuar con nuestra base de datos MongoDB desde una aplicación Node.js.

La conexión a la base de datos se establece mediante una instancia de la clase ***MongoClient***, que recibe como parámetro la URI de conexión a la base de datos. Una vez establecida la conexión, nuestra API utiliza diferentes métodos del módulo mongodb para interactuar con la base de datos.

Cada vez que un sensor recoge nuevos datos, estos se envían a la API a través de una solicitud ***POST*** a una ruta específica para ese tipo de sensor (por ejemplo, /luminosidad para datos de luminosidad). La API procesa la solicitud y guarda los datos en la base de datos MongoDB utilizando el método ***insertOne*** de la colección correspondiente al tipo de sensor. Los datos se almacenan en colecciones específicas para cada tipo de sensor. Este método recibe como parámetro un objeto con los datos del registro y los guarda en la base de datos. Para recuperar todos los registros de un tipo de sensor, nuestra API utiliza el método ***find*** de la colección correspondiente para obtener un cursor que permite iterar sobre todos los documentos de la colección.

La API también permite **recuperar**, **actualizar** y **eliminar** registros de la base de datos a través de diferentes rutas y operaciones. Por ejemplo, para recuperar todos los registros de un tipo de sensor, se puede enviar una solicitud ***GET*** a una ruta específica para ese tipo de sensor (por ejemplo, /luminosidad para datos de luminosidad). La API procesa la solicitud y devuelve un conjunto de datos que contiene todos los registros almacenados en la base de datos para el tipo de sensor especificado.

En resumen, esta API REST nos permite gestionar los datos de los sensores en una base de datos MongoDB, permitiendo **agregar**, **recuperar**, **actualizar** y **eliminar** registros. La comunicación entre los sensores y el sistema se realiza a través de

solicitudes ***HTTP*** a la API y la comunicación con la base de datos se realiza utilizando el módulo ***mongodb*** de Node.js.

9.3 Documentación de las APIs mediante Swagger

Swagger es una herramienta de código abierto que permite diseñar, construir, documentar y consumir APIs RESTful. Con Swagger, podemos generar una documentación interactiva y fácil de usar para nuestras APIs que muestra información detallada sobre cada ruta, incluyendo los parámetros de entrada, las respuestas esperadas y los códigos de estado. Además, Swagger nos permite probar nuestras APIs directamente desde la página de documentación, lo que facilita la depuración y el desarrollo de las aplicaciones.

Para instalar Swagger en nuestro proyecto Node.js y comenzar a utilizarlo, hemos seguido estos pasos:

1. Instalamos el paquete ***swagger-ui-express*** y ***swagger-jsdoc*** en nuestro proyecto utilizando npm:

- ***npm i swagger-jsdoc swagger-ui-express***

```
C:\Users\danid\Pictures\AUTOMATIZACION\SERVER>npm i swagger-jsdoc swagger-ui-express

added 34 packages, and audited 147 packages in 16s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

2. Creamos un archivo llamado ***swagger. ts*** en nuestro proyecto y agregamos el siguiente código para configurar Swagger:

```
import swaggerJSDoc from 'swagger-jsdoc';
import swaggerUi from 'swagger-ui-express';

const options = {
    definition: {
        openapi: "3.0.0",
        info: {
            title: 'API Automatizacion de una casa',
            version: '1.0.0'
        },
    },
    apis: ['./src/models/luminosidad.ts',
'./src/models/proximidad.ts' , './src/models/botonPuerta.ts'
,  './src/models/temperaturaHumedad.ts'],
};

const swagger = swaggerJSDoc(options);

export const swaggerDocs = (app: any) => {
    app.use('/docs', swaggerUi.serve, swaggerUi.setup(swagger));
    app.get('/docs.json', (req: any, res: any) => {
        res.setHeader('Content-Type', 'application/json');
        res.send(swagger);
    });

    console.log('Version 1 docs api')
};
```

3. Añadimos la dependencia de swagger introduciendo en la consola el siguiente comando :

- ***npm i –save-dev @types/swagger-jsdoc.***

El comando npm i --save-dev @types/swagger-jsdoc instala el paquete @types/swagger-jsdoc como una dependencia de desarrollo en nuestro proyecto de Node.js. Este paquete proporciona los tipos TypeScript para el paquete swagger-jsdoc, que nos permite generar documentación para nuestra API utilizando comentarios en el código fuente. Por otro lado, el paquete @types/swagger-ui-express nos proporciona los tipos TypeScript para el paquete swagger-ui-express,

que nos permite agregar la interfaz de usuario de Swagger a nuestra aplicación Express para visualizar y probar la documentación de API generada con swagger-jsdoc. Cuando ejecutamos estos comandos, npm descarga los paquetes ***@types/swagger-jsdoc*** y ***@types/swagger-ui-express*** y los agrega a la sección ***devDependencies*** de nuestro archivo ***package.json***. Esto indica que estos paquetes son una dependencia de desarrollo, lo que significa que solo se utilizan durante el desarrollo y no se incluyen en el paquete final de nuestra aplicación.

```
C:\Users\danid\Pictures\AUTOMATIZACION\SERVER>npm i --save-dev @types/swagger-jsdoc

added 1 package, and audited 148 packages in 3s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\danid\Pictures\AUTOMATIZACION\SERVER>
```

```
C:\Users\danid\Pictures\AUTOMATIZACION\SERVER>npm i --save-dev @types/swagger-ui-expres

added 1 package, and audited 149 packages in 3s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

4. Agregamos los comentarios especiales en nuestros archivos de rutas para documentar nuestra API utilizando la sintaxis de Swagger. Por ejemplo, para la api luminosidad , el get:

El uso del comentario ***tags:*** en una API permite agrupar visualmente las diferentes APIs que tienen el mismo tag. Esto mejora la organización y presentación de las APIs, facilitando su uso y comprensión.

```
/**
         * @swagger
         * /luminosidad:
         *   get:
         *     tags:
         *       - Luminosidad
         *     summary: Recupera todos los registros de luminosidad
         *     responses:
         *       200:
         *         description: Una lista de registros de luminosidad.
         *         content:
         *           application/json:
         *             schema:
         *               type: array
         *               items:
         *                 type: object
         *                 properties:
         *                   _id:
         *                     type: number
         *                     description: El ID del registro.
         *                   date:
         *                     type: string
         *                     format: date-time
         *                     description: La fecha y hora del registro.
         *                   value:
         *                     type: number
         *                     description: El valor de luminosidad del
registro.
         *                   ledOn:
         *                     type: boolean
         *                     description: Indica si el LED estaba
encendido o apagado en el momento del registro.
         */
        // GET
        this.app.get('/luminosidad', async (req, res) => {
            try {
                const db = this.client.db('AUTOMATIZACION');
                const collection = db.collection('LUMINOSIDAD');
                const data = await collection.find({}).toArray();
                res.status(200).json(data);
            } catch (error) {
                res.status(500).send('Error al recuperar los datos');
            }
        });
```

5. Importamos el archivo ***swagger.ts*** en nuestro archivo principal ***index.js*** y llamamos a la función exportada para habilitar Swagger en nuestra aplicación:

```
const { startServer } = require ("./models/server");
const { swaggerDocs } = require('./swagger/swagger');


async function main() {
    const app = await startServer();

    // Agrega la documentación de Swagger a tu aplicación
    swaggerDocs(app);
}

main();
```

6. Por último , iniciamos nuestro servidor y navegamos a la ruta ***http://localhost:3000/docs*** en nuestro navegador para ver la página de documentación generada por Swagger.

Figura 66: Ejemplo del swagger asociado al sensor de luminosidad

10. Instalación servidor en raspberry pi.

10.1 Configuración e instalación raspberry pi

Para la instalación del servidor con nuestro Backend, Frontend y nuestra base de datos, vamos a utilizar una Raspberry Pi. Por ello, es importante explicar primero el proceso de instalación del sistema operativo en la Raspberry Pi para que todo funcione correctamente.

Para instalar el sistema operativo Raspbian en una Raspberry Pi, debemos seguir los siguientes pasos:

1. Descargar el software y los archivos necesarios desde la página oficial de Raspberry Pi.
2. Una vez descargado el sistema operativo Raspberry Pi OS (anteriormente llamado Raspbian), se puede utilizar la herramienta Raspberry Pi Imager para instalarlo en una tarjeta microSD. Para hacerlo, se debe insertar la tarjeta microSD en un lector de tarjetas y ejecutar Raspberry Pi Imager.
3. Luego, seleccionamos el sistema operativo que se desea instalar y se siguen las instrucciones en pantalla para escribirlo en la tarjeta microSD.
4. Una vez completado, se puede insertar la tarjeta microSD en la Raspberry Pi y encenderla para comenzar a usar el sistema operativo.

Figura 67: Raspberry Pi Imager, elegimos OS

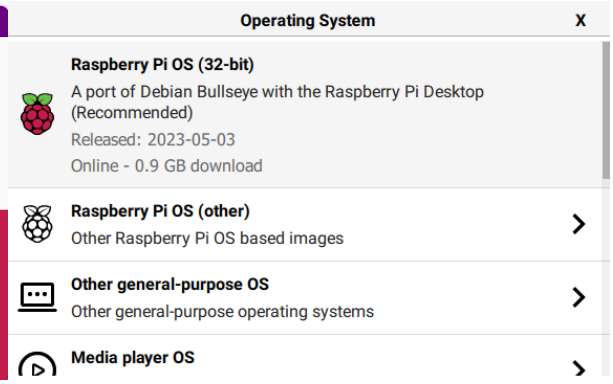


Figura 68: Elección sistema operativo

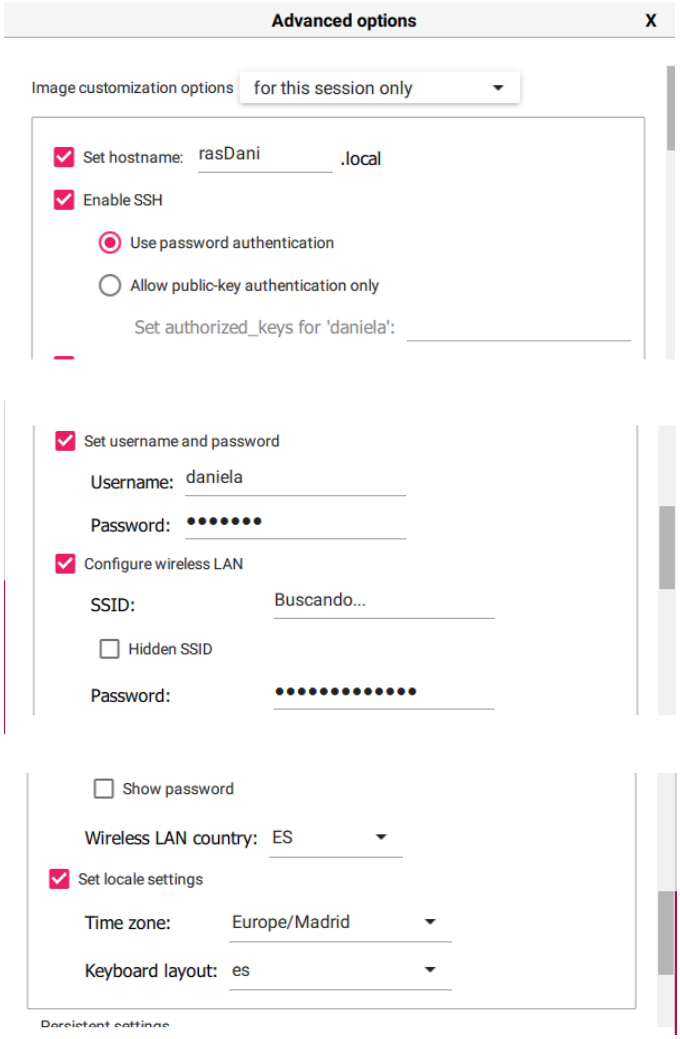


Figura 69: Configuración opciones avanzadas preinstalación

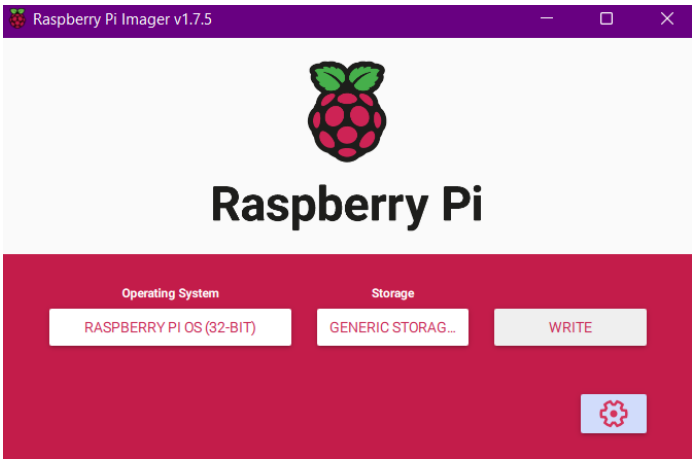


Figura 70: Escritura del SO en la raspberry pi

Una vez que se haya completado la configuración en las opciones avanzadas de Raspberry Pi Imager, debemos pulsar en la opción "WRITE", de la figura 70 para escribir el sistema operativo en la tarjeta microSD. Cuando el proceso haya finalizado, podremos acceder a la Raspberry Pi mediante una terminal utilizando el protocolo SSH.

Para hacerlo, debemos asegurarnos de que el servicio SSH esté habilitado en la Raspberry Pi.

Si no lo tenemos habilitado, tenemos dos opciones, a través de la aplicación de configuración gráfica de Raspberry Pi o desde la terminal utilizando el comando **“sudo raspi-config”**.

Una vez que habilitemos el servicio, es necesario saber la dirección IP o el nombre de host de la Raspberry Pi, lo cual se puede obtener ejecutando el comando **“hostname -I”** en la terminal, o como hemos configurado en la figura 69 sabemos cuál es el host, en nuestro caso **“rasDani”**.

Con esta información, podemos utilizar un cliente SSH para conectarnos a la Raspberry Pi desde otro dispositivo.

En nuestro caso al estar usando Windows debemos utilizar algún software adicional para poder conectarnos a la raspberry como PuTTY.

Una vez estemos conectados, podremos controlar y administrar la Raspberry Pi de forma remota, lo que nos permite realizar tareas de mantenimiento y administración sin necesidad de conectar un monitor y un teclado a la Raspberry Pi.

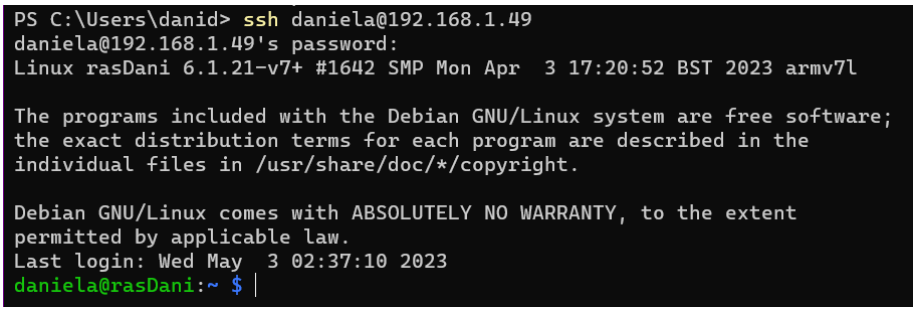


Figura 71: Conexión mediante ssh a la rapsberry pi

Para facilitar el acceso a nuestra Raspberry Pi y evitar que su dirección IP cambie constantemente, vamos a configurar una dirección IP estática. En este caso, vamos a utilizar la dirección IP **"192.168.1.2"**. Para hacerlo, necesitamos editar el archivo de configuración de DHCP, que se encuentra en la ruta **"/etc/dhcpcd.conf"**.

Este archivo se puede editar utilizando un editor de texto como **"nano"**, ejecutando el comando **"sudo nano /etc/dhcpcd.conf"**.

Figura 72: Configuración de ip estática en nuestra raspberry

Dentro del archivo, debemos agregar una línea que especifique la dirección IP estática que queremos utilizar, utilizando el formato **"static ip_address="** seguido de la dirección IP que hemos mencionado y el sufijo **"/24"**.

Por tanto, debemos agregar la línea **"static ip_address=192.168.1.2/24"**.

Una vez que hayamos agregado la línea y guardado los cambios en el archivo de configuración, debemos reiniciar la Raspberry Pi para que los cambios surtan efecto., utilizando el comando **"sudo reboot"**.

Después del reinicio, debemos conectarnos a la Raspberry Pi con la nueva dirección ip que acabamos de configurar, es decir , **"ssh daniela@192.168.1.2".**

10.2 Configuración interfaz gráfica raspberry pi

Para acceder a la Raspberry Pi de forma remota y con una interfaz gráfica, vamos a utilizar el software RealVNC.

RealVNC es un software que permite acceder y controlar de forma remota la interfaz gráfica de una computadora desde otro dispositivo. Para instalar RealVNC en una Raspberry Pi, vamos a hacerlo desde la terminal ejecutando los comandos **"sudo apt-get update"** y **"sudo apt-get install realvnc-vnc-server".**

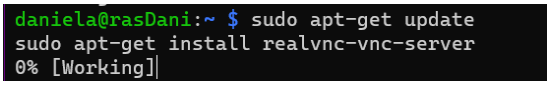


Figura 73: Instalación vcn raspberry pi

Una vez instalado, debemos asegurarnos de que el servidor VNC esté habilitado en la Raspberry Pi. Para ello debemos hacerlo a través de la aplicación de configuración gráfica de Raspberry Pi, seleccionando la opción "Interfaces" y asegurándote de que "VNC" esté activo.

Para ello debemos utilizar el comando **"sudo raspi-config"** que se utiliza para ejecutar la herramienta de configuración de Raspberry Pi desde la terminal.

Figura 74: Accedemos a la interfaz basada en texto de la rapsberry pi

Esta herramienta nos proporcionara una interfaz basada en texto para realizar cambios en la configuración del sistema, como habilitar o deshabilitar interfaces, cambiar la contraseña del usuario, configurar la localización y el idioma, entre otras opciones. Al ejecutar este comando, se nos abrirá la herramienta de configuración que podemos ver en la figura 75 y se puede navegar por las diferentes opciones utilizando las teclas de flecha y la tecla Enter.

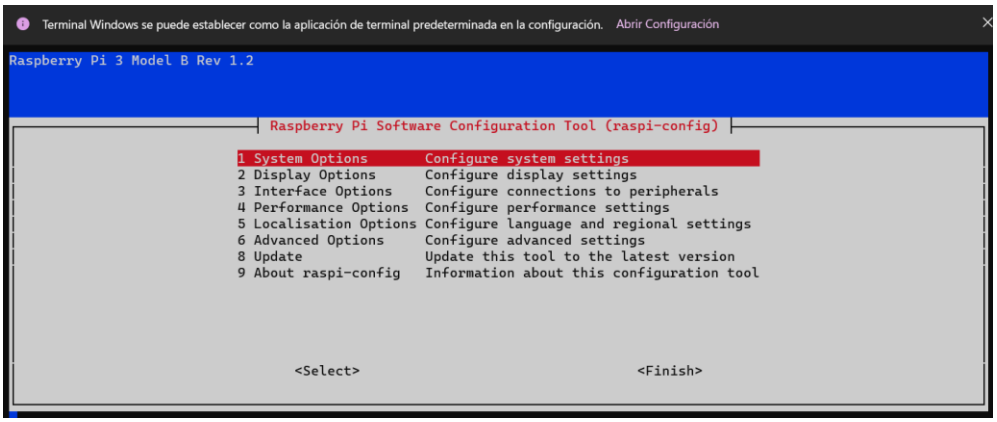


Figura 75: Configuración gráfica raspberry pi

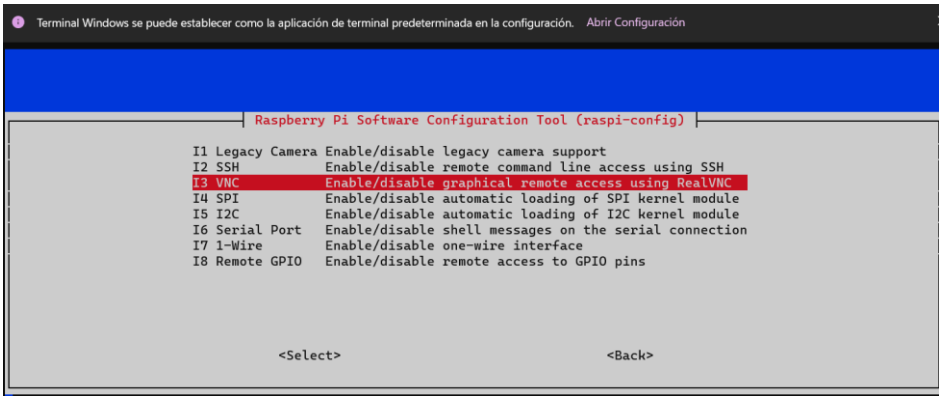


Figura 76: Activación VCN en raspberry pi

Para acceder a la Raspberry Pi desde Windows, debemos descargarnos e instalarnos un cliente VNC como RealVNC Viewer. Una vez instalado, podremos acceder a nuestra raspberry mediante la dirección IP de nuestra Raspberry Pi y controlarla de forma remota.

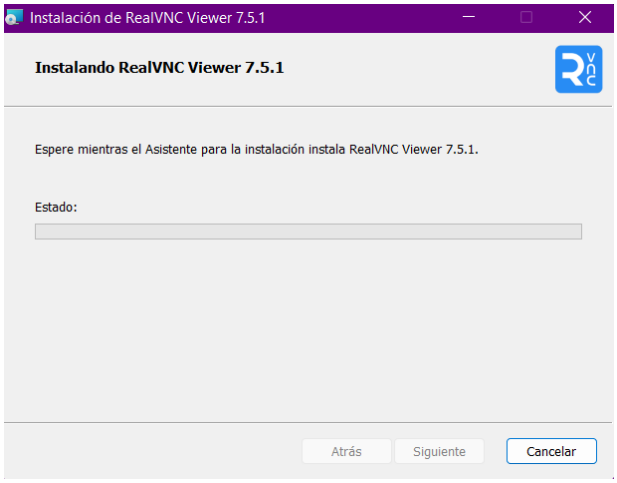


Figura 77: Instalación RealVNC Viewer

Al abrir RealVNC Viewer y en la barra de direcciones, debemos ingresar la dirección IP de nuestra Raspberry Pi en nuestro caso 192.168.1.2 y presionar Enter. Esto iniciará una conexión con el servidor VNC en la Raspberry Pi y nos pedirá las credenciales que configuramos en la figura 69, una vez que ingresemos nuestras credenciales, figura 78, podremos ver y controlar la interfaz gráfica de la Raspberry Pi de forma remota desde nuestro ordenador personal tal y como podemos ver en la figura 79.

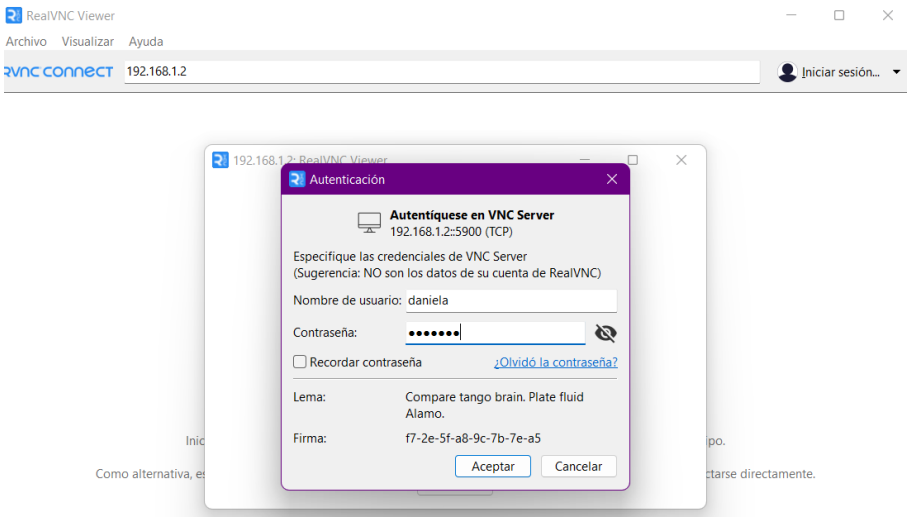


Figura 78: Acceso RealVNC Viewer desde nuestro ordenador personal

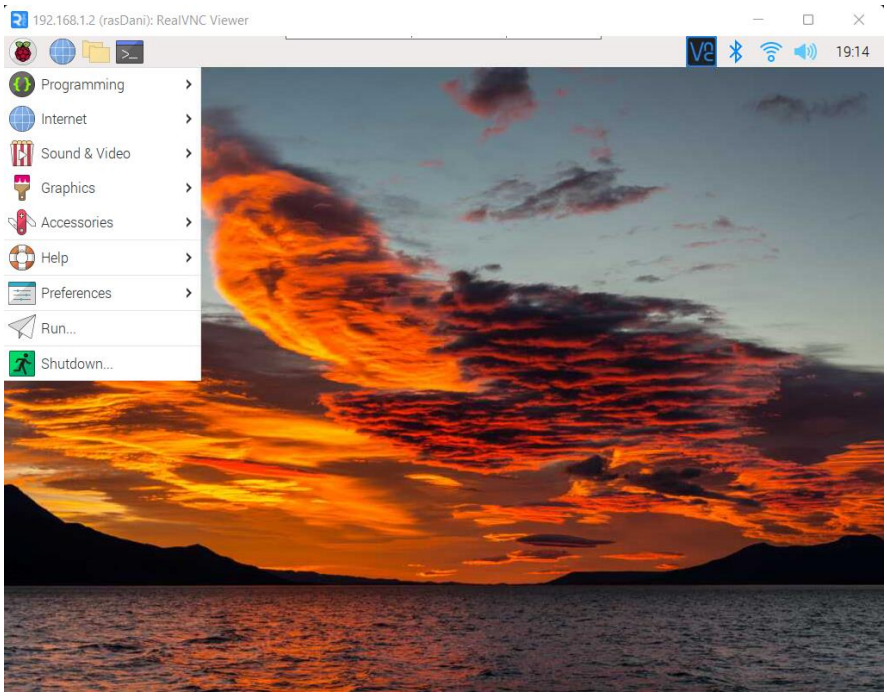


Figura 79: Interfaz Raspberry pi desde nuestro ordenador personal

10.3 Configuración Backend y Frontend en raspberry pi

A continuación, vamos a instalar FileZilla en nuestro ordenador para poder transferir archivos entre nuestra Raspberry Pi y nuestro ordenador de forma fácil y rápida. FileZilla es un programa de código abierto que permite transferir archivos entre dispositivos a través de varios protocolos, como FTP, SFTP y FTPS. Con FileZilla, podemos conectarnos a nuestra Raspberry Pi y transferir archivos entre nuestro ordenador y la Raspberry Pi. FileZilla también ofrece características avanzadas como la capacidad de reanudar transferencias interrumpidas, limitar la velocidad de transferencia y configurar el uso de proxy.

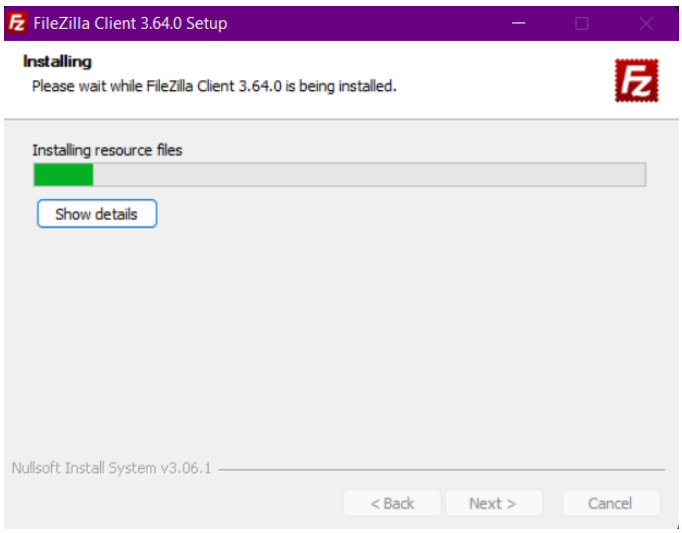


Figura 80: Instalación FileZilla

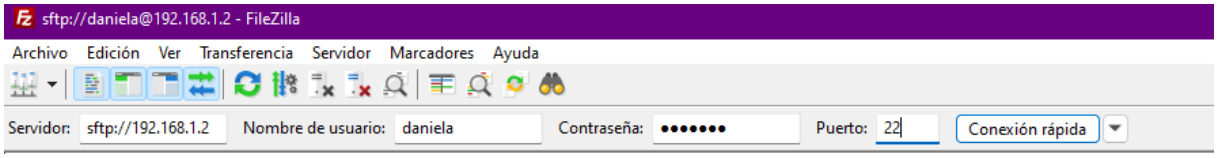


Figura 81: Conexión con FileZilla a nuestra raspberry pi

A través de FileZilla, vamos a transferir todo el código desarrollado para nuestro servidor desde nuestro ordenador a nuestra Raspberry Pi.

Esto nos permitirá tener una copia de nuestro código en la Raspberry Pi y ejecutarlo directamente desde allí.

Para instalar Node.js en nuestra Raspberry Pi, primero debemos determinar qué versión de Node.js es compatible. Para ello ejecutamos el comando **"uname -m"** en una terminal de la Raspberry Pi, lo que nos mostrará la arquitectura de hardware de nuestra Raspberry Pi. Con esta información, podemos visitar la página oficial de descargas de Node.js y buscar la versión de Node.js que sea compatible con nuestra arquitectura de hardware.

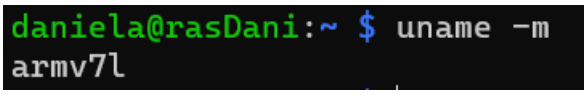


Figura 82: Arquitectura hardware de nuestra raspberry pi

Una vez que hayamos encontrado la versión adecuada, podemos hacer clic con el botón derecho del mouse sobre el enlace de descarga y seleccionar la opción **"Copiar dirección del enlace"**.

Luego, podemos abrir una terminal en nuestra Raspberry Pi y ejecutar el comando **"wget"** seguido de la URL que acabamos de copiar para descargar el archivo de instalación de Node.js en nuestra Raspberry Pi.

En nuestro caso el siguiente comando:

"wget https://nodejs.org/dist/v18.16.0/node-v18.16.0-linux-armv7l.tar.xz" para descargar el archivo.

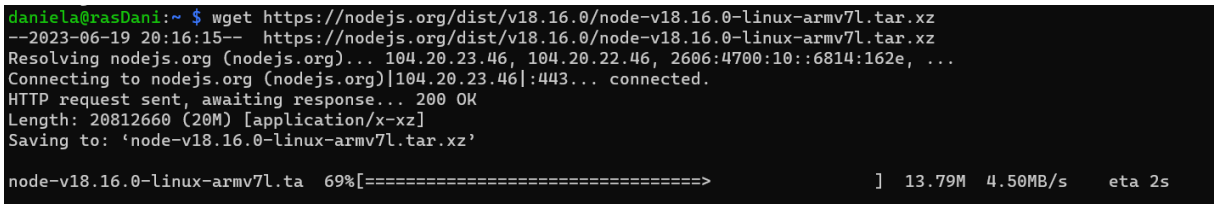


Figura 83: Instalación nodejs en la raspberry pi

Una vez que hayamos descargado el archivo de instalación de Node.js en nuestra Raspberry Pi, debemos extraerlo y copiarlo en nuestro directorio local. Para hacerlo, utilizamos los comandos **"tar -xf node-v18.16.0-linux-armv7l.tar.xz"** para extraerlo.

Y para copiarlo no ponemos dentro de la carpeta que acabamos de descomprimir y ejecutamos el siguiente comando **"cp -R * /usr/local"** , esto es que copia de forma recursiva todos los archivos y directorios del directorio actual al directorio **"/usr/local"**. La opción **"-R"** indica que la copia debe realizarse de forma recursiva, lo que significa

que se copiarán todos los archivos y subdirectorios del directorio actual, incluyendo sus contenidos. El asterisco **"*"** representa todos los archivos y directorios del directorio actual. Por lo tanto, este comando copiará todos los archivos y directorios del directorio actual al directorio **"/usr/local"**, preservando la estructura de directorios.

Una vez que hayamos completado este proceso, podemos verificar que Node.js se haya instalado correctamente ejecutando el comando **"node –version"**, que nos mostrará la versión de Node.js instalada en nuestra Raspberry Pi. Con Node.js instalado, ya podremos utilizarlo para ejecutar aplicaciones y scripts escritos en JavaScript en nuestra Raspberry Pi.

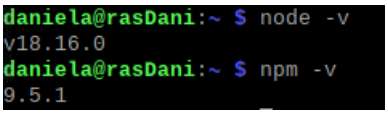


Figura 84: Version nodejs en la raspberry pi

Para instalar Angular en nuestra Raspberry Pi, es necesario tener previamente instalados Node.js y npm. Una vez que estos requisitos se cumplan, podemos proceder a instalar la interfaz de línea de comandos (CLI) de Angular utilizando el comando **"sudo npm install -g @angular/cli"**. Este comando instalará la CLI de Angular de forma global en nuestra Raspberry Pi, permitiéndonos utilizar el comando **"ng"** para crear y gestionar proyectos de Angular.

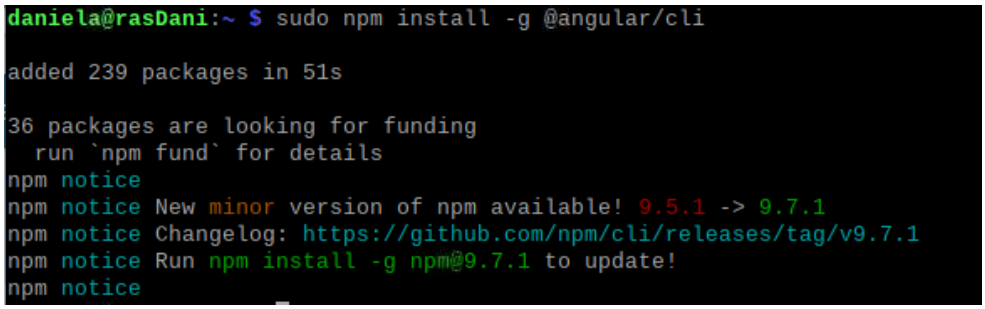


Figura 85: Instalación Angular en nuestra raspberry pi

Con la CLI de Angular instalada, ya podemos hacer uso de nuestro frontend.

Para poder acceder a nuestra aplicación Angular desde fuera de la Raspberry Pi, debemos ejecutar el comando **"ng serve"** indicando la dirección IP de nuestra Raspberry Pi. Por ejemplo, en nuestro caso la dirección IP de nuestra Raspberry Pi es **"192.168.1.2"**, podemos ejecutar el comando:

"sudo node --max_old_space_size=900 ./node_modules/@angular/cli/bin/ng serve --host 192.168.1.2" para iniciar un servidor de desarrollo local accesible desde fuera de la Raspberry Pi.

Vamos a explicar este comando:

- **"sudo"** ejecuta el comando con privilegios de superusuario, lo que puede ser necesario para acceder a ciertos recursos del sistema.
- **"node"** es el comando para ejecutar aplicaciones Node.js.
- **"--max_old_space_size=900"** es una opción que se pasa al motor de JavaScript V8 para aumentar el tamaño máximo del espacio de memoria antiguo a 900 MB. Esto es útil para evitar problemas de memoria al ejecutar aplicaciones grandes en dispositivos con recursos limitados, como una Raspberry Pi.
- **"./node_modules/@angular/cli/bin/ng"** es la ruta al ejecutable de la CLI de Angular en nuestro proyecto.
- **"serve"** es el comando de la CLI de Angular para iniciar un servidor de desarrollo local.
- **"--host 192.168.1.2"** es una opción que se pasa al comando **"ng serve"** para especificar la dirección IP en la que se debe escuchar el servidor de desarrollo local. En este caso, se está indicando que el servidor debe escuchar en la dirección IP **"192.168.1.2"**.

En resumen, este comando inicia un servidor de desarrollo local para una aplicación Angular en una Raspberry Pi, escuchando en la dirección IP especificada y con un tamaño máximo de espacio de memoria antiguo aumentado a 900 MB.

10.4 Configuración MongoDB en raspberry pi

En la raspebrry hemos usado MongoDB Atlas en la nube para almacenar y gestionar los datos del nuestro proyecto. Esta decisión se tomó debido a que la Raspberry Pi suele tener recursos limitados, como memoria RAM y almacenamiento, que pueden no ser suficientes para ejecutar MongoDB de forma eficaz.

Por defecto, MongoDB no proporciona paquetes de instalación para los sistemas operativos de Raspberry Pi y para poder instalar MongoDB en una Raspberry Pi, es necesario compilar el código fuente de MongoDB desde cero. Este proceso puede ser complejo.

MongoDB Atlas es un servicio de base de datos MongoDB totalmente administrado que se ejecuta en la nube. MongoDB Atlas ofrece una amplia gama de opciones de almacenamiento y rendimiento, lo que lo hace ideal para su uso en sistemas operativos de Raspberry Pi.

Para poder utilizar MongoDB Atlas en un Raspberry Pi y poder acceder a nuestros datos y almacenarlos como veniamos haciendo en nuestro ordenador personal ,es necesario cambiar la configuración del backend, que en lugar de apuntar al localhost, el backend debe apuntar a la dirección IP del servidor MongoDB Atlas.

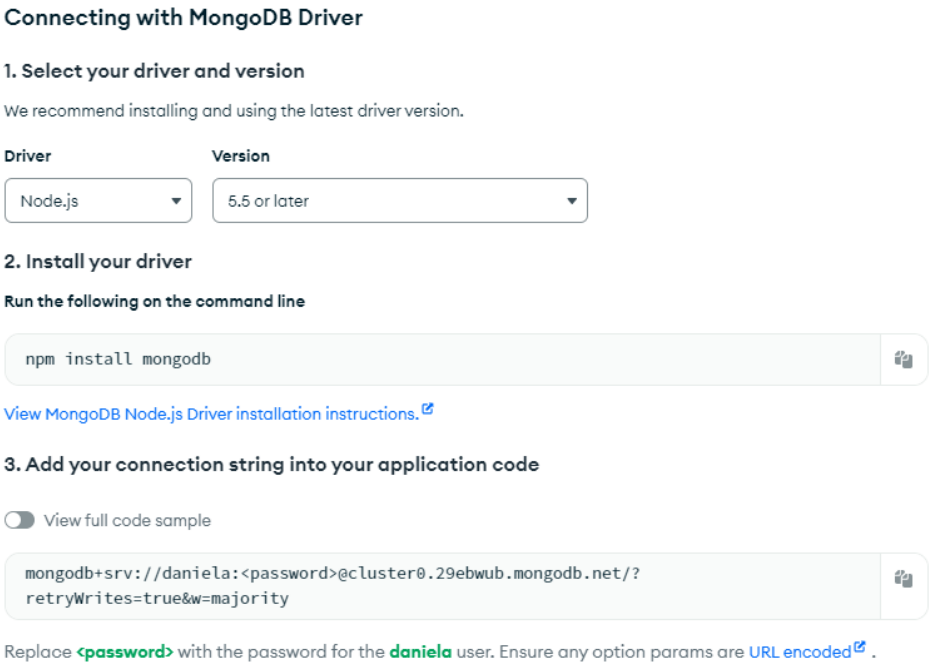


Figura 86: MongoDB Atlas

La linea que se ha tenido que tocar para cada sensor es:

```
'mongodb+srv://daniela:daniela@cluster0.29ebwub.mongodb.net/?retryWrites=true&w=majority';
```

En conclusión, el uso de MongoDB Atlas en la nube para almacenar y gestionar los datos del proyecto ha sido una decisión acertada. Esta solución ha permitido superar las limitaciones de la Raspberry Pi y garantizar un rendimiento óptimo de la aplicación.

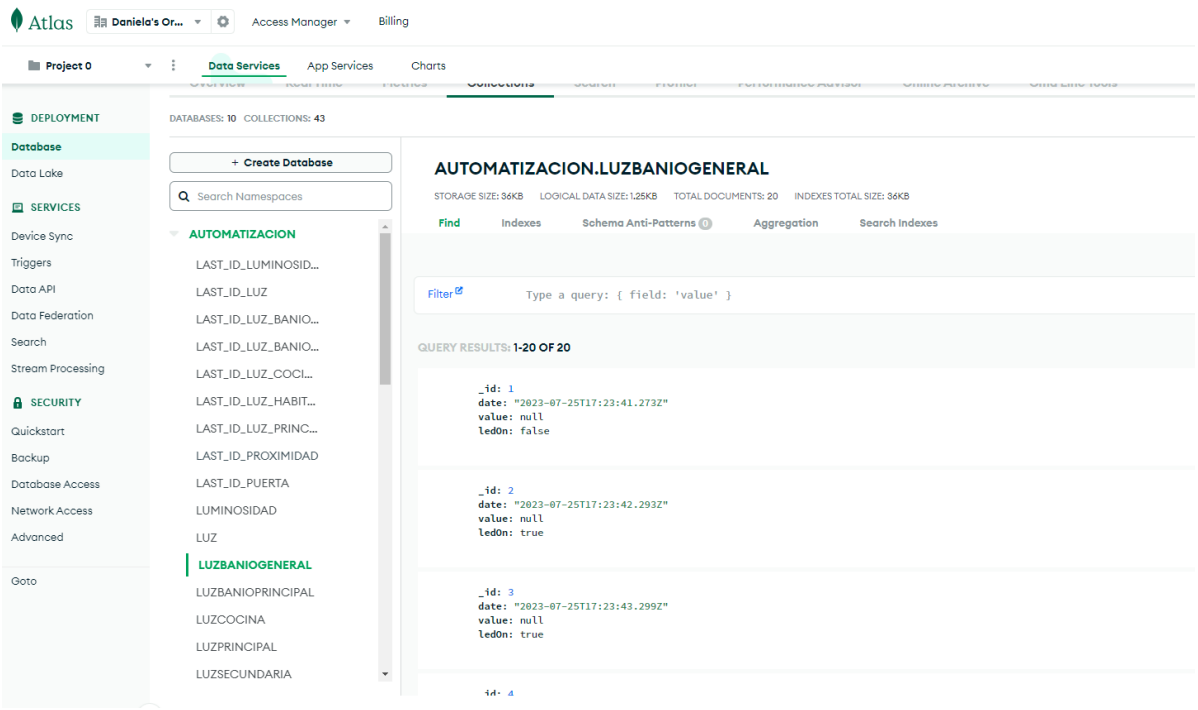


Figura 87: MongoDB Atlas, nuestra base de datos

11. Aplicación accesible desde cualquier lugar: configuración de puertos

11.1 Configuracion dentro del router

Para poder acceder a nuestra aplicación desde cualquier lugar o dispositivo, es necesario configurar los puertos en el router. Para hacerlo, antes de configurar los puertos en el router es importante comprobar la red local y los datos que tenemos en ella, así como el rango de direcciones IP disponibles.

11.2 Instalacion no-ip raspberry pi

Para descargar e instalar No-IP en una Raspberry Pi con Raspbian haremos los siguientes pasos:

1. Descargamos la última versión disponible de no-ip utilizando el siguiente comando **“wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz”**.

```
daniela@rasDani:~ $ wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz
--2023-07-05 17:45:31--  http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz
Resolving www.no-ip.com (www.no-ip.com)... 158.247.7.199
Connecting to www.no-ip.com (www.no-ip.com)|158.247.7.199|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.noip.com/client/linux/noip-duc-linux.tar.gz [following]
--2023-07-05 17:45:31--  https://www.noip.com/client/linux/noip-duc-linux.tar.gz
Resolving www.noip.com (www.noip.com)... 158.247.7.200
Connecting to www.noip.com (www.noip.com)|158.247.7.200|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 134188 (131K) [application/x-gzip]
Saving to: ‘noip-duc-linux.tar.gz’

noip-duc-linux.tar. 100%[===================>] 131.04K   185KB/s    in 0.7s

2023-07-05 17:45:33 (185 KB/s) - ‘noip-duc-linux.tar.gz’ saved [134188/134188]
```

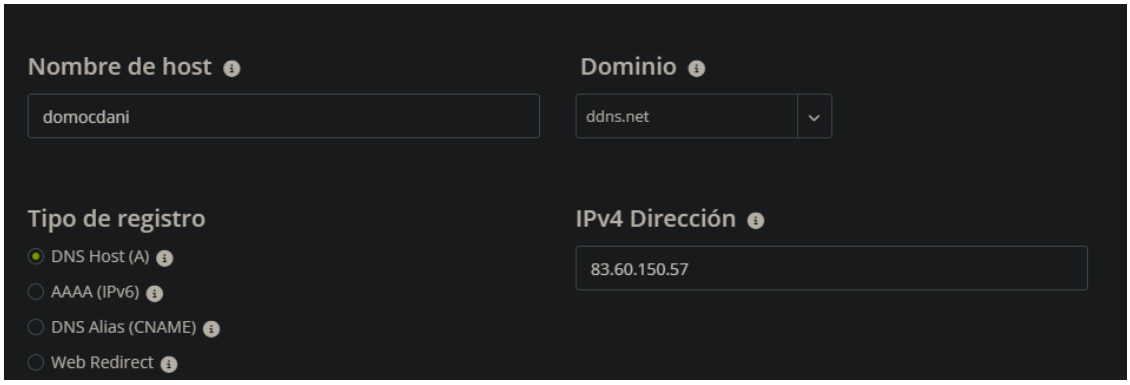
2. Una vez descargado descomprimimos el fichero utilizando el comando

“tar -zxvf noip-duc-linux.tar.gz”.

3. Accedemos a la carpeta que se ha creado y procedemos con la instalación.

4. Ejecutamos el comando **“make”** para compilar el programa.

5. Debemos tener nuestro host ya creado para que a la hora de hacer la instalación encuentra uno o más host ya registrados, en nuestro caso tendremos únicamente el que vamos a utilizar para nuestra aplicación.



6. Ejecutamos el comando **“sudo make install”** para instalar el programa.

12.Resultado final

Podemos observar cómo quedo finalmente la maqueta, con sus placas y sensores, cableada y funcional.

La instalación se ha realizado por habitaciones, lo que permite apreciar de manera clara qué sensores, botones y cables corresponden a cada placa y habitación. Esto facilita la comprensión del funcionamiento de la maqueta y su enseñanza a los alumnos.

Figura 88: Estado final de la maqueta, con sus placas y sensores

Por último, podemos apreciar el resultado final de la página en la que registramos los datos y con la que podemos interactuar.

Esta página nos permite visualizar la información recopilada por los sensores de la maqueta y manipularla de manera sencilla, de forma que podremos encender una bombilla o abrir la puerta del garaje pulsando el botón habilitado para ello.

Figura 89: Login página web, DOMOCDANI

Figura 90: Página principal, DOMOCDANI

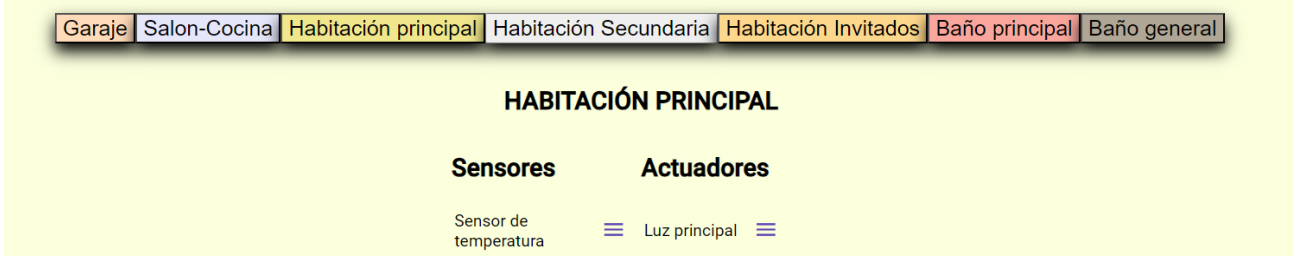


Figura 91: Habitación con sus sensores disponibles

ID	Fecha	Led On
1	2023-07-25T17:23:41.273Z	false
2	2023-07-25T17:23:42.293Z	true
3	2023-07-25T17:23:43.299Z	true

Figura 92: Tabla ejemplo, de los datos registrados

13. Conclusiones y trabajo a futuro

En conclusión, este proyecto ha demostrado la viabilidad de un sistema domótico escalable y económico para mejorar la seguridad y la calidad de vida en el hogar. A través de la implementación de una maqueta a escala real, se ha podido comprobar el correcto funcionamiento de los sensores y el software desarrollado. Además, el uso de una base de datos no relacional como MongoDB ha permitido una gestión eficiente de los datos recopilados por los sensores.

Este proyecto podría utilizarse como herramienta docente para enseñar a los alumnos sobre los sistemas domóticos. La maqueta a escala real podría utilizarse para mostrar a los alumnos cómo funcionan los diferentes componentes de un sistema domótico y cómo se pueden integrar para mejorar la seguridad y la calidad de vida en el hogar.

Sin embargo, aún hay margen para mejorar y expandir el sistema en el futuro, mediante la adición de nuevos sensores y la integración con otros sistemas domóticos. Algunas mejoras que se podrían realizar incluyen:

- ✓ Añadir nuevos sensores para monitorear más aspectos del hogar, como la calidad del aire o el consumo de energía. Esto permitiría al sistema ofrecer una mayor cantidad de información y control sobre el entorno del hogar.
- ✓ Integrar el sistema con otros sistemas domóticos para ofrecer una solución más completa y fácil de usar. Esto podría incluir la integración con sistemas de seguridad, sistemas de climatización o sistemas de entretenimiento.
- ✓ Trabajar en mejorar la facilidad de uso del sistema para personas mayores o con discapacidades, mediante el desarrollo de interfaces más intuitivas o el uso de tecnologías como el reconocimiento de voz.
- ✓ La siguiente guía proporcionada nos da una posible orientación sobre cómo utilizar el proyecto en el aula:
    - Introducir a los alumnos en los conceptos básicos de los sistemas domóticos.
    - Mostrar a los alumnos la maqueta a escala real del sistema domótico.

- Explicar a los alumnos cómo funcionan los diferentes componentes del sistema domótico.
- Dejar que los alumnos experimenten con el sistema domótico y lo modifiquen para que se adapte a sus propias necesidades.
- Evaluar el aprendizaje de los alumnos mediante preguntas, exámenes o proyectos.
- Esta guía proporciona una orientación general sobre cómo utilizar el proyecto en el aula. Los profesores pueden adaptar la guía a las necesidades específicas de sus alumnos.

A lo largo del desarrollo de este proyecto, se han adquirido una gran cantidad de conocimientos y habilidades. Hemos aprendido a utilizar tecnologías como Angular y Node.js para desarrollar el Frontend y el Backend del sistema, respectivamente. También a utilizar una base de datos no relacional como MongoDB para almacenar y gestionar los datos recopilados por los sensores. Además, se ha adquirido experiencia en el diseño y desarrollo de interfaces de usuario utilizando HTML y CSS. Todo esto ha permitido crear un sistema domótico completo y funcional, que integra hardware y software para mejorar la seguridad y la calidad de vida en el hogar.

El desarrollo de este proyecto ha sido una oportunidad valiosa para aprender y aplicar nuevas tecnologías y conceptos en un entorno práctico.

14. Referencias

AndProf. (2021). Obtenido de https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/

arduino, P. c. (2023). *Proyectos con arduino*. Obtenido de https://proyectosconarduino.com/sensores/sensor-de-distancia-hc-sr04/

Eepower. (s.f.). *Eepower*. Obtenido de https://eepower.com/resistor-guide/resistor-types/photo-resistor/

Git. (s.f.). Obtenido de https://www.git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

Google. (2023). *Angular*. Obtenido de https://angular.io/docs

HW libre. (s.f.). Obtenido de https://www.hwlibre.com/hc-sr04/

Lobo, J. (2020). Obtenido de https://jvlobo.com/esp32-clock-in-with-calamari-io-part-1/

LuisLlamas. (2016). Obtenido de https://www.luisllamas.es/arduino-buzzer-activo/

Microsoft. (2023). *VisualStudioCode*. Obtenido de https://visualstudio.microsoft.com/es/downloads/

Modulos arduino. (s.f.). Obtenido de https://arduinomodules.info/ky-015-temperature-humidity-sensor-module/

MongoDB. (2023). Obtenido de https://www.mongodb.com/try/download/community

MongoDB. (2023). *MongoDB*. Obtenido de https://www.mongodb.com/es/what-is-mongodb

mysql. (2023). Obtenido de https://www.mysql.com/products/workbench/

MYSQL. (2023). Obtenido de https://dev.mysql.com/downloads/workbench/

Nodemon. (s.f.). Obtenido de https://www.npmjs.com/package/nodemon.

OpenWebinars. (2023). Obtenido de https://openwebinars.net/blog/que-es-postman/

Postman. (2023). Obtenido de https://www.postman.com/downloads/

rjrobotics007. (2023). *Hackaday*. Obtenido de https://hackaday.io/page/8804-how-to-interface-hc-sr501-pir-sensor-with-raspberry-pi

TheOpenJSFundation. (2023). *NodeJS*. Obtenido de https://nodejs.org/es/download/

Watson, D. (2020). *The engineering projects*. Obtenido de https://www.theengineeringprojects.com/2019/01/introduction-to-hc-sr501.html