



CARRERA DE ESPECIALIZACIÓN EN INTERNET DE LAS COSAS

MEMORIA DEL TRABAJO FINAL

Sistema de monitoreo y control de ambientes a distancia

Autor:

César Javier Fanelli

Director:

Ing. Fernando Lichtschein (FIUBA)

Codirector:

Ing. María Celeste Corominas (FIUBA)

Jurados:

Esp. Ing. Pedro Rosito (FIUBA)

Ing. Marcelo Edgardo Romeo (UNSAM)

MG. Lic. Leopoldo Zimperz (FIUBA)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires,
entre febrero de 2023 y diciembre de 2023.*

Resumen

En la presente memoria se describe el diseño de un prototipo de sistema integral de domótica de índole académica. El propósito es aplicar los conocimientos adquiridos en el transcurso del posgrado. En esta primer etapa se implementan dos tipos de nodos y se agregarán a futuro nuevos elementos con otras funcionalidades.

En el desarrollo del trabajo se utilizaron conocimientos de diseño de páginas web, desarrollo de código backend para el servidor, implementación de bases de datos, desarrollo de sistemas embebidos, ciberseguridad y diseño electrónico para el hardware adicional.

Agradecimientos

Agradezco a docentes, alumnos y no docentes que conforman el posgrado y a los directores que me acompañaron en el desarrollo de este trabajo. En especial agradezco a mi esposa e hija, quienes fueron mi apoyo en todo momento y sin quienes no habría podido lograr la hazaña de emprender este viaje arduo y satisfactorio.

Índice general

Resumen	I
1. Introducción general	1
1.1. Hogares inteligentes	1
1.1.1. Aplicaciones comunes	1
1.2. Motivación	2
1.3. Estado del arte	2
1.4. Objetivos y alcance	3
2. Introducción específica	5
2.1. Protocolos de comunicación	5
2.1.1. Modelo OSI	5
2.1.2. Protocolo HTTP	6
2.1.3. Protocolo MQTT	7
Calidad de servicio	8
Protocolos SSL y TLS	8
2.2. Componentes de hardware	8
2.2.1. Raspberry Pi	8
2.2.2. Sistema en chip	9
Familia ESP32	10
2.3. Herramientas de software	11
2.3.1. Visual studio code	11
2.3.2. Marco de desarrollo ESP	11
2.3.3. Lenguajes JavaScript y TypeScript	11
2.3.4. Angular y Ionic	12
2.3.5. Bases de datos relacionales MySQL y MariaDB	12
2.3.6. Contenedores con Docker	12
3. Diseño e implementación	15
3.1. Arquitectura del sistema	15
3.1.1. Especificaciones técnicas del servidor	16
3.2. Modelo de datos	16
3.2.1. Tabla Dispositivos	17
3.2.2. Tabla Usuarios	18
3.2.3. Tabla Mediciones	18
3.3. Desarrollo del frontend	19
3.3.1. Rutas y páginas más importantes	20
Pantalla de <i>login</i>	20
Pantalla principal <i>home</i>	21
3.4. Desarrollo del backend	21
3.5. Nodos, sensores y actuadores	21
3.6. Comunicación del sistema	23

4. Ensayos y resultados	25
5. Conclusiones	27
Bibliografía	29

Índice de figuras

1.1. Ejemplo de sistema de domótica	2
1.2. Matter y Home Assistant	3
1.3. Esquema básico	4
2.1. Modelo OSI	5
2.2. Arquitectura MQTT	7
2.3. Raspberry Pi	9
2.4. Familia ESP32	10
2.5. Docker	13
3.1. Arquitectura cliente-servidor	15
3.2. Docker base datos	17
3.3. Tabla dispositivos	17
3.4. Tabla usuarios	18
3.5. Tabla mediciones	19
3.6. Configuración Ionic	19
3.7. Estructura página	20
3.8. Pantalla login	21
3.9. Nodo de temperatura	23
3.10. Nodo dimer	23

Índice de tablas

1.1. Mercado nacional	3
3.1. Raspberry Pi 400	16
3.2. Rutas	21
3.3. Módulos ESP32	22

Dedicado a mi hija Sofía.

Capítulo 1

Introducción general

1.1. Hogares inteligentes

La domótica es la aplicación de la tecnología a la automatización del hogar que utiliza para controlar y gestionar diferentes sistemas y dispositivos, con el fin de aportar seguridad, bienestar y confort. Estos sistemas pueden incluir iluminación, calefacción, aire acondicionado, sistemas de seguridad y cámaras de vigilancia, sistemas de entretenimiento y otros dispositivos domésticos [1].

Este sector, como muchos otros que utilizan tecnología ha estado creciendo, al punto que algunas de las viviendas modernas son concebidas como hogares inteligentes desde su edificación, llegando a tener edificios completos con este tipo de soluciones instaladas.

1.1.1. Aplicaciones comunes

Los servicios que ofrece la domótica se pueden agrupar según cinco aspectos o ámbitos principales [2]:

- Programación y ahorro energético: el ahorro energético no es algo tangible, sino legible con un concepto al que se puede llegar de muchas maneras. En muchos casos no es necesario sustituir los aparatos o sistemas del hogar por otros que consuman menos energía sino una gestión eficiente de los mismos.
- Confort: conlleva todas las actuaciones que se puedan llevar a cabo que mejoren la comodidad en una vivienda. Dichas actuaciones pueden ser de carácter tanto pasivo, como activo o mixtas.
- Seguridad: consiste en una red encargada de proteger tanto los bienes patrimoniales, como la seguridad personal y la vida.
- Comunicaciones: son los sistemas o infraestructuras de comunicaciones que posee el hogar.
- Accesibilidad: bajo este mecanismo se incluyen las aplicaciones o instalaciones de control remoto del entorno que favorecen la autonomía personal de personas con limitaciones funcionales, o discapacidad.

El presente trabajo puede encuadrarse dentro de la programación y ahorro energético, confort y accesibilidad. En la figura 1.1 puede verse una imagen con los distintos tipos de dispositivos que pueden estar conectados a un sistema de domótica.



FIGURA 1.1. Ejemplo de sistema de domótica. ¹

1.2. Motivación

El uso de la tecnología para facilitar y mejorar la vida de las personas se está implementando en todo el mundo en diversos ámbitos, creando soluciones complejas e innovadoras. Los electrodomésticos e instalaciones se fabrican con posibilidades de conexión y capacidades cada vez más amplias, abarcando funcionalidades complejas que aportan al bienestar y confort de las personas.

Este proyecto nace como mejora y actualización de la tesis de grado de ingeniería, en la cual se creó un sistema similar que carecía de conectividad a Internet y utilizando tecnología que al día de hoy es obsoleta. Es por este motivo que se ha decidido hacer un proyecto académico con esta temática, pudiendo crear un sistema integral con una página web, base de datos que almacene mediciones y utilizando hardware actualizado.

1.3. Estado del arte

En el mercado actual existe una gran cantidad de sistemas de domótica, algunos privados o bajo licencias propietarias y otros con licencias de código abierto, nacionales e internacionales. Aquellas soluciones de código abierto tienen los repositorios públicos en *GitHub* al alcance de todos para ser descargados, implementados y hasta modificados.

Del mercado internacional, podemos nombrar 2 importantes variantes:

- **Matter:** es un estándar de conectividad de código abierto creado por la *Connectivity standards alliance*. Es de los más importantes a nivel global y participan empresas como Amazon, Apple, Google, Huawei entre muchas otras. Ha crecido y tomado mucha fuerza durante el corto tiempo de vida que tiene el proyecto [3].
- **Home assistant:** software de automatización de hogares de código abierto que prioriza el control local y la privacidad. Desarrollado por una comunidad mundial de entusiastas de autodidactas y profesionales con un sentido propio [4].

¹ Imagen tomada de: <https://intelligy.com/blog/2018/03/12/conoces-la-domotica/>

En la figura 1.2 se observa un kit de Matter a la izquierda y la pantalla de la aplicación de Home Assistant a la derecha.



FIGURA 1.2. Matter y Home Assistant.²

A nivel nacional existen empresas que se dedican al desarrollo de sistemas de domótica de forma privada:

- Domotic [5].
- Commax [6].
- Reactor [7].

Todos estos sistemas utilizan distintos tipos de conexión (Zigbee o Wi-Fi), mensajes entre dispositivos (HTTP, MQTT y otros) y tipos de alojamiento de las plataformas (servidor local o en la nube).

Pocos de estos sistemas tienen la posibilidad de funcionar con mandos desde el lugar o sin conexión con el servidor, es decir, que sean independientes y controlables de forma local. Este proyecto tiene tal posibilidad, ya que la página web es una forma más de cambiar los parámetros de los nodos y está diseñado para decidir si cada terminal posee comandos o no.

En la tabla 1.1 pueden verse las principales características de las empresas nacionales con soluciones similares.

TABLA 1.1. Comparativa entre las distintas opciones

Empresa	Productos	Compatibilidad
Domotic	Catálogo amplio	No especifica
Commax	Catálogo amplio	<i>Smart things</i> y <i>Apple Home Kit</i>
Reactor	2 alternativas	<i>Google Assistant</i> e <i>IFTTT</i>

1.4. Objetivos y alcance

El objetivo de este trabajo es aplicar todos los conocimientos adquiridos en el transcurso del posgrado, en un tema de interés particular como es la domótica. En especial, el principal propósito es hacer un prototipo que dé inicio a un sistema más grande y con mayores características, con la particularidad de que en un futuro cercano se pueda llegar a analizar la opción de integrarlo con sistemas como *Matter* o *Home Assistant*.

²Imágenes tomadas de: <https://csa-iot.org/> y <https://www.home-assistant.io/>

En este desarrollo se incluyó:

- El esquema de conexiones de las placas de ESP32 con los módulos de display, encoder, driver LED y driver de calefacción.
- El diseño en una placa de desarrollo del circuito del driver de corriente continua de la iluminación LED.
- El diseño en una placa de desarrollo del circuito del driver de corriente alterna de la calefacción.
- La programación de un firmware de la placa ESP32 para la comunicación y manejo de conexiones.
- La creación del software frontend, backend y la base de datos que almacena toda la información dentro del servidor.
- Las conexiones, instalación y configuración del servidor montado en una Raspberry Pi.

En la figura 3.10 se puede ver un esquema básico del sistema desarrollado.

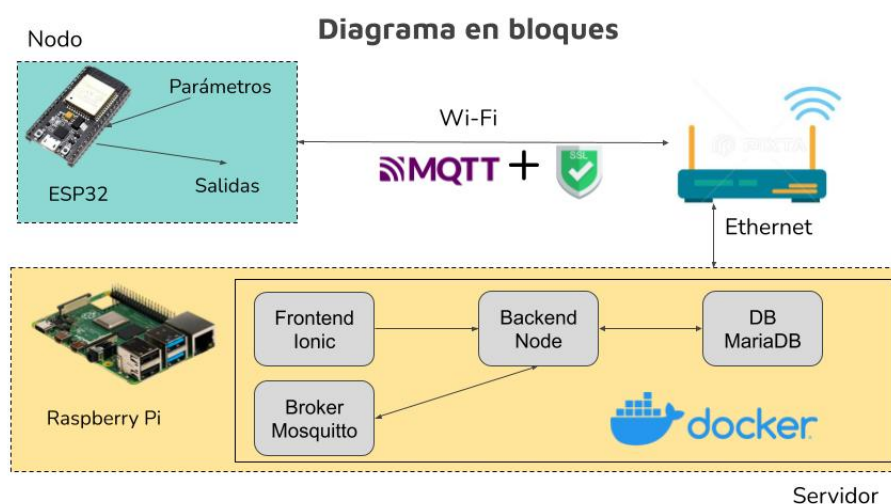


FIGURA 1.3. Esquema básico del sistema desarrollado.

Capítulo 2

Introducción específica

En el presente capítulo se introducen las tecnologías y herramientas de hardware y software utilizados en el desarrollo del trabajo.

2.1. Protocolos de comunicación

2.1.1. Modelo OSI

El modelo de interconexión de sistemas abiertos, conocido como modelo OSI (en inglés, *Open Systems Interconnection*), es un modelo de referencia para los protocolos de la red. Define un estándar que tiene por objetivo interconectar sistemas de distinta procedencia para que estos puedan intercambiar información sin ningún tipo de impedimentos. Está conformado por 7 capas o niveles de abstracción. Cada uno de estos niveles tiene sus propias funciones para que en conjunto sean capaces de poder alcanzar su objetivo final. Precisamente esta separación en niveles hace posible la intercomunicación de protocolos distintos al concentrar funciones específicas en cada nivel de operación [8].

En la figura 2.1 pueden verse de forma gráfica las distintas capas del modelo OSI.

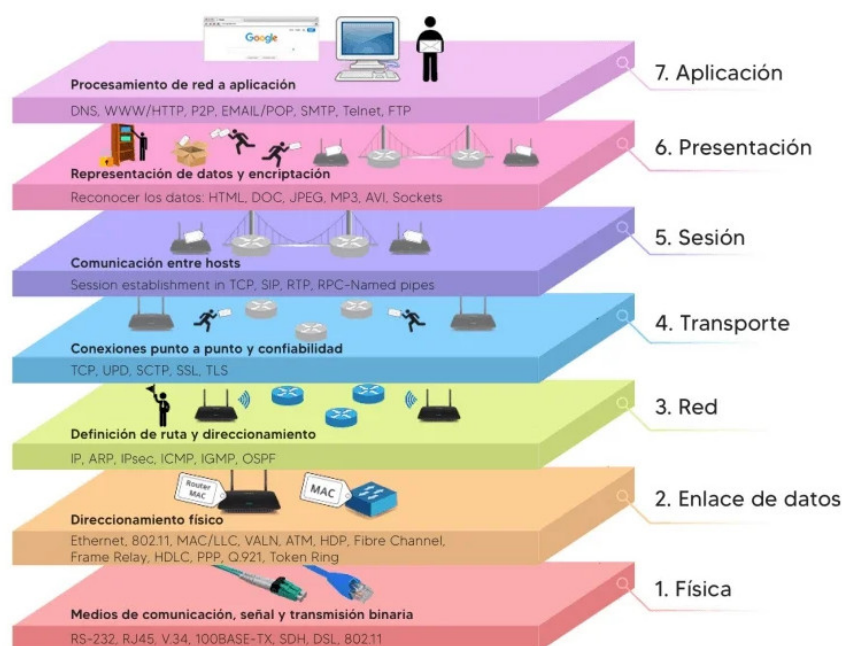


FIGURA 2.1. Modelo OSI. ¹

A continuación se presentan en detalle las funciones de cada capa:

- Capa 1 o física: define todas las especificaciones eléctricas y físicas de los dispositivos.
- Capa 2 o de enlace de datos: proporciona direccionamiento físico y procedimientos de acceso a medios.
- Capa 3 o de red: se encarga del direccionamiento lógico y el dominio del enrutamiento.
- Capa 4 o de transporte: proporciona transporte confiable y control del flujo a través de la red.
- Capa 5 o de sesión: establece, administra y finaliza las conexiones entre las aplicaciones locales y las remotas.
- Capa 6 de presentación: transforma el formato de los datos y proporciona una interfaz estándar para la capa de aplicación.
- Capa 7 o de aplicación: responsable de los servicios de red para las aplicaciones.

2.1.2. Protocolo HTTP

El protocolo de transferencia de hipertexto, conocido como HTTP (en inglés *Hypertext Transfer Protocol*), es el protocolo de comunicación que permite las transferencias de información a través de archivos como XML y HTML entre otros. Está orientado a transacciones y sigue el esquema petición y respuesta entre un cliente y un servidor. El cliente realiza una petición al servidor enviando un mensaje con un formato determinado. El servidor le envía un mensaje de respuesta con la información solicitada o un mensaje de error [9].

Los mensajes HTTP son en texto plano y tienen la siguiente estructura:

- Línea inicial.
- Cabecera.
- Cuerpo.

En la línea inicial se envían las peticiones con el método hacia el servidor o las respuestas con el código devueltas hacia el navegador. Los métodos más comunes, y utilizados en el proyecto, son:

- GET: solicita una representación del recurso especificado.
- POST: envía datos para que sean procesados por el recurso identificado en la URL de la línea de petición.
- PUT: envía datos al servidor. La diferencia con POST es que este está orientado a la creación de nuevos contenidos, mientras que PUT está orientado a la actualización de los mismos.
- DELETE: Borra el recurso especificado.

¹Imagen tomada de: <https://platzi.com/clases/2225-redes/35587-modelo-osi/>

2.1.3. Protocolo MQTT

El protocolo de transporte de telemetría de colas de mensajes, conocido como MQTT (en inglés *Message Queue Telemetry Transport*) es uno de los más utilizados y difundidos en IoT (Internet de las cosas, o en inglés *Internet of Things*). Es un protocolo de capa de aplicación, posee una topología de estrella y sus mensajes se transmiten como colas de publicación/suscripción.

El nodo central de su topología se llama *broker* y es al cual se conectan los clientes remotos. El *broker* se encarga de gestionar la red, recibir todos los mensajes de los clientes y redirigirlos hacia los clientes de destino. Un cliente es cualquier dispositivo que pueda interactuar con el *broker* para enviar y recibir mensajes. Puede ser un sensor de IoT en campo o una aplicación en un centro de datos que procesa datos provenientes de los sensores. Cualquier cliente puede publicar o suscribirse a *topics* (temas) para acceder a la información. Un *topic* se representa mediante una cadena de texto con una estructura jerárquica. Cada jerarquía se separa con el carácter `/`.

La finalidad de MQTT es minimizar el uso recursos en los dispositivos (CPU, RAM, ROM), ser confiable y ofrecer distintos niveles calidad del servicio. Consume un ancho de banda relativamente bajo y mantiene una conexión continua entre el broker y los clientes. Generalmente se suele usar este protocolo como nexo entre los dispositivos de campo con elementos típicos de software como servidores webs, bases de datos o herramientas de análisis. En la figura 2.2 se puede ver una arquitectura básica de conexión MQTT [10].

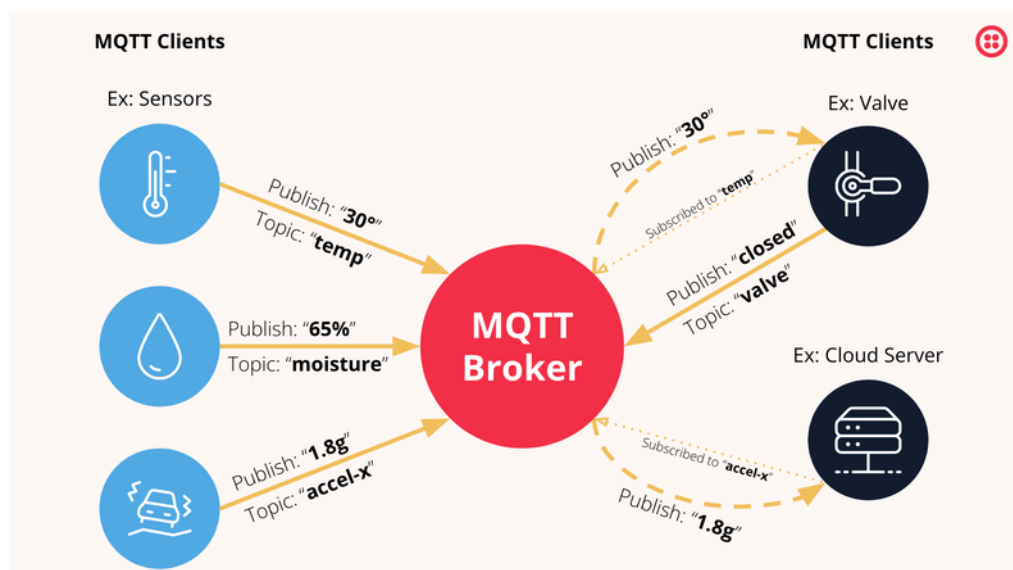


FIGURA 2.2. Arquitectura MQTT. ²

Para reforzar la seguridad en la comunicación mediante MQTT, se pueden proteger los datos mediante el uso de usuario y contraseña, o con cifrado de capa de sockets seguros, conocidos como SSL (en inglés *Secure Sockets Layer*). La capa de transporte segura (en inglés *Transport Layer Security* o TLS), es una versión actualizada y más segura de SSL, y es por esto que en la actualidad se refiere a SSL/TLS como sinónimos.

²Imagen tomada de: <https://www.twilio.com/blog/what-is-mqtt>

Calidad de servicio

MQTT está diseñado para ser simple y minimizar el ancho de banda, lo que lo convierte en una buena opción para muchos escenarios, aunque esta simplicidad hace que la interpretación de los mensajes dependa completamente del diseñador del sistema. Para mitigar este tipo de inconvenientes se soportan distintos niveles de calidad de servicio.

Estos niveles determinan cómo se entrega cada mensaje y es necesario especificar un QoS (siglas en inglés para *quality of service*) para cada *topic* MQTT. Es importante elegir el valor de QoS adecuado para cada mensaje que está determinado de la siguiente manera:

- QoS 0 (a lo sumo una vez): los mensajes se mandan sin tener en cuenta si llegan o no. Puede haber pérdida de mensajes. No se hacen retransmisiones.
- QoS 1 (al menos una vez): se asegura que los mensajes lleguen pero se pueden producir duplicados. El receptor recibe el mensaje por lo menos una vez. Si el receptor no confirma la recepción del mensaje o se pierde en el camino se reenvía el mensaje hasta que recibe por lo menos una confirmación.
- QoS 2 (exactamente una vez): se asegura que el mensaje llegue exactamente una vez. Eso incrementa la sobrecarga en la comunicación pero es la mejor opción cuando la duplicación de un mensaje no es aceptable.

Protocolos SSL y TLS

SSL y TLS son protocolos criptográficos, que proporcionan comunicaciones seguras por una red. Se usa criptografía asimétrica para autenticar a la contraparte con quien se está comunicando y para intercambiar una llave simétrica, iniciando una sesión entre las partes intervinientes. Esta sesión es luego usada para cifrar el flujo de datos entre las partes. Esto permite la confidencialidad de la información transmitida y la autenticación de los mensajes.

Antes de que un cliente y el servidor pueden empezar a intercambiar información protegida por TLS, deben intercambiar en forma segura o acordar una clave de cifrado y una clave para usar cuando se cifren los datos. Existen varios métodos utilizados para el intercambio y acuerdo de claves [11].

2.2. Componentes de hardware

2.2.1. Raspberry Pi

Las Raspberry Pi son computadoras de placa simple (en inglés *Single Board Computer* o su sigla SBC). Actualmente en el mercado existen distintas marcas y variantes de este tipo de placas, siendo elegidas tanto como computadoras de escritorio como para algunas aplicaciones específicas. A continuación, se describen las principales características de la familia Raspberry Pi:

- Posee distintos puertos, permitiendo conectarla a periféricos como pueden ser una pantalla, un medio de almacenamiento e incluso cualquier dispositivo que tenga interfaz USB.
- Contiene un procesador gráfico integrado, lo que permite la reproducción de vídeo, incluso en alta definición.

- Permite la conexión a la red a través del puerto de Ethernet y algunos modelos permiten conexión Wi-Fi y Bluetooth.
- Consta de una ranura microSD que permite instalar y ejecutar sistemas operativos a través de una tarjeta de memoria.

El diseño de la Raspberry Pi fue evolucionando con el correr del tiempo. En la actualidad se encuentran los modelos Raspberry Pi 4 y Raspberry Pi 400, siendo los más modernos y robustos lanzados por la marca. La familia de Raspberry Pi 4 cuenta con 4 modelos de placa con distinta capacidad de memoria RAM, yendo desde 1 GB hasta 8 GB. La Raspberry Pi 400 es una variante de la Raspberry Pi 4. También existen otros modelos más pequeños diseñados para otro tipo de aplicaciones.

En la figura 2.3 se pueden ver los modelos Raspberry Pi 4 y Raspberry Pi 400.



FIGURA 2.3. Raspberry Pi 4 y Raspberry Pi 400.³

Algunas de las funciones y aplicaciones más comunes de este tipo de computadoras se describen a continuación:

- Navegar en la red, utilizar aplicaciones de oficina para la edición de documentos y emplearla como si fuese una computadora de escritorio.
- Crear un centro multimedia y ver los archivos guardados en su memoria.
- Se puede utilizar como un servidor privado dentro de una red local.
- Conectar los puertos del microprocesador desde un conector de 40 pines a distintos circuitos y dispositivos.

2.2.2. Sistema en chip

Un sistema en chip o SoC (del inglés *System on a Chip*) es aquel dispositivo que posee integrados todos o gran parte de los módulos que componen un sistema informático o electrónico en un único circuito integrado o chip. El diseño de estos sistemas puede estar basado en circuitos de señal digital, señal analógica y a menudo módulos o sistemas de radiofrecuencia. Un ámbito común de aplicación de la tecnología SoC son los sistemas embebidos.

Un SoC estándar está constituido por [12]:

³Imagen tomada de: <https://all3dp.com/2/raspberry-pi-400-vs-raspberry-pi-4-differences/>

- Un microcontrolador con el núcleo de la CPU. Algunos son construidos con microprocesadores dotados de varios núcleos.
- Módulos de memoria ROM (memoria de sólo lectura), RAM (memoria de acceso aleatorio), EEPROM (memoria de sólo lectura programable y borrrable electrónicamente) y Flash (memorias de acceso muy rápido).
- Generadores de frecuencia fija.
- Componentes periféricos como contadores, temporizadores y relojes en tiempo real o RTC (en inglés *Real Time Clock*).
- Controladores de comunicación con interfaces externas normalmente estándar como USB, Ethernet, UART, o SPI.
- Controladores de interfaces analógicas, incluyendo conversores analógico a digital (ADC) y digital a analógico (DAC).
- Reguladores de voltaje y circuitos de gestión eficaz de la energía.

Familia ESP32

ESP32 es la denominación de una familia de chips SoC de bajo costo y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada. Emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros y módulos de administración de energía. El ESP32 fue creado y desarrollado por Espressif Systems.

En la figura 2.4 se pueden ver algunos de los módulos de desarrollo que contienen ESP32.

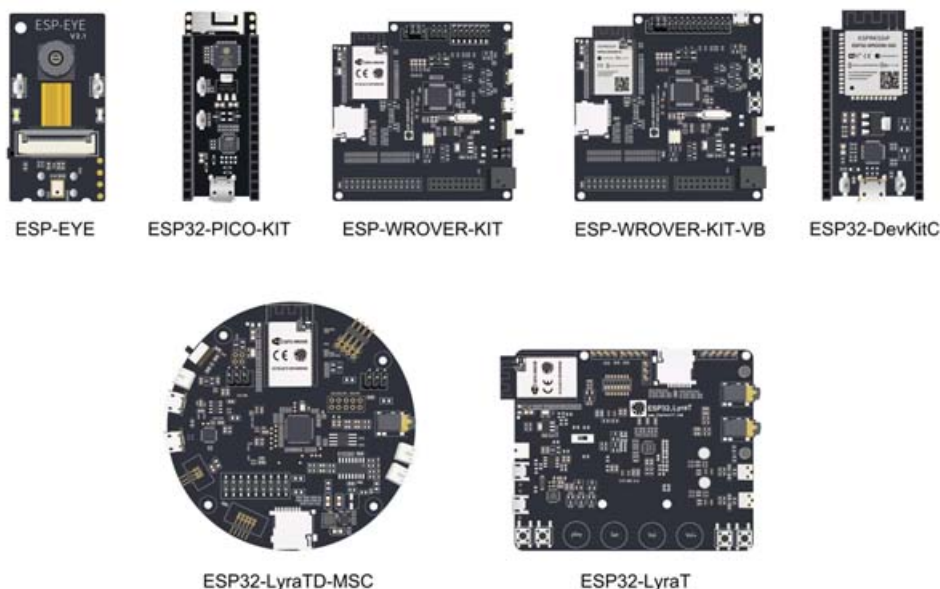


FIGURA 2.4. Módulos de la familia ESP32.⁴

⁴Imagen tomada de: <https://www.electrodaddy.com/esp32/>

2.3. Herramientas de software

2.3.1. Visual studio code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para múltiples sistemas operativos. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Es personalizable y tiene la posibilidad de instalar extensiones para agregar lenguajes, depuradores y herramientas para el desarrollo de código. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.

Algunos de los lenguajes de programación que soporta son: C, C++, Dockerfile, Git-commit, HTML, JSON, Java, JavaScript, PHP, Python, Ruby, Rust, SQL, Shell script, TypeScript y Visual Basic entre otros.

2.3.2. Marco de desarrollo ESP

El marco de desarrollo de ESP, o ESP-IDF (en inglés *ESP IoT Development Framework*) es un entorno completo de programación para desarrollar sistemas embebidos para dispositivos ESP. Es desarrollado por Espressif y se puede descargar como una extensión de Visual studio code. El lenguaje de programación es C e incluye herramientas para cargar el código desarrollado al chip y depurar el programa en tiempo real.

El lenguaje C es un lenguaje de programación de propósito general de tipos de datos estáticos, débilmente tipado. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a bajo nivel. Uno de los objetivos de diseño del lenguaje C es que solo sean necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución [13].

2.3.3. Lenguajes JavaScript y TypeScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque también se utiliza del lado del servidor. Todos los navegadores modernos interpretan el código en este lenguaje integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del modelo de objeto de documento, o DOM (en inglés *Document Object Model*). Es el único lenguaje de programación que entienden de forma nativa los navegadores [14].

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, su antecesor, que esencialmente añade tipos estáticos y objetos basados en clases. Es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas. Extiende la sintaxis de su antecesor, por tanto cualquier código en el lenguaje original existente debería funcionar sin

problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript [15].

2.3.4. Angular y Ionic

Angular es un marco de desarrollo (framework en inglés) de ingeniería de software de código abierto mantenido por Google, que sirve para desarrollar aplicaciones web de estilo aplicación de una sola página (en inglés *single page application* o SPA) y aplicación web progresiva (en inglés *Progressive Web App* o PWA). Sirve tanto para versiones móviles como de escritorio. Ofrece soluciones robustas, escalables y optimizadas para lograr un estilo de codificación homogéneo y de gran modularidad. Su desarrollo se realiza por medio de TypeScript o JavaScript. En este último se ofrecen diversas herramientas adicionales al lenguaje como tipado estático o decoradores [16].

El marco de desarrollo Ionic es un kit de desarrollo de software (en inglés *Software Development Kit* o SDK) de frontend de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Es decir, un framework que nos permite desarrollar aplicaciones multiplataforma desde una única base de código. Posee la capacidad de integrarse con otros marcos populares como Angular, React y Vue. Su principal característica es que permite desarrollar y desplegar aplicaciones híbridas, que funcionan en múltiples plataformas, como iOS nativo, Android, escritorio y la web (como una aplicación web progresiva) con una única base de código [17].

2.3.5. Bases de datos relacionales MySQL y MariaDB

Una base de datos relacional es un conjunto de una o más tablas estructuradas en registros (filas) y campos (columnas), que se vinculan entre sí por un campo en común. MySQL es un sistema de administración de bases de datos relacionales de código abierto desarrollado por Oracle. Se considera como la base de datos de código abierto más utilizada en el mundo. Posee cuatro funciones básicas que se conocen con la sigla CRUD: *create* (crear), *read* (leer), *update* (actualizar) y *delete* (borrar). Estas funciones son las que se aplican a los nuevos registros que se quieran crear y a los ya existentes que se deseen leer, actualizar y borrar. Este tipo de bases de datos posee una arquitectura cliente-servidor, siendo el cliente el que hace las solicitudes de datos y el servidor el que posee dichos datos y responde a dicha solicitud.

MariaDB es una versión modificada de MySQL. Fue creada por el equipo de desarrollo original de MySQL debido a problemas de licencia y distribución después de que Oracle Corporation adquiriera MySQL. MariaDB adopta los archivos de definición de tablas y datos de MySQL y también usa protocolos de cliente, API de cliente, puertos y sockets idénticos. Con ello se pretende que los usuarios de MySQL puedan cambiar a MariaDB sin problemas [18].

2.3.6. Contenedores con Docker

Los contenedores son una forma de virtualización del sistema operativo. Un solo contenedor se puede usar para ejecutar cualquier aplicación, desde un microservicio o un proceso de software a una aplicación de mayor tamaño. Dentro de un contenedor se encuentran todos los ejecutables, el código binario, las bibliotecas

y los archivos de configuración necesarios. Sin embargo, en comparación con los métodos de virtualización de máquinas o servidores, los contenedores no contienen imágenes del sistema operativo. Esto los hace más ligeros y portátiles, con una sobrecarga significativamente menor. En implementaciones de aplicaciones de mayor tamaño, se pueden poner en marcha varios contenedores como uno o varios clústeres de contenedores. Estos clústeres se pueden gestionar mediante un orquestador de contenedores, como Kubernetes o Docker Compose [19].

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Usar Docker para crear y gestionar contenedores puede simplificar la creación de sistemas altamente distribuidos, permitiendo que múltiples aplicaciones, las tareas de los trabajadores y otros procesos funcionen de forma autónoma en una única máquina física o en varias máquinas virtuales [20]. En la figura 3.1 se puede ver de forma gráfica el funcionamiento de un contenedor Docker.

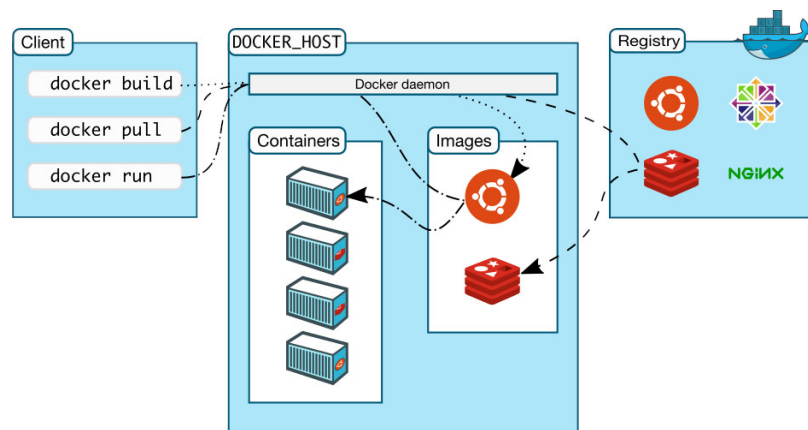


FIGURA 2.5. Contenedor Docker.⁵

Docker Compose es una herramienta para definir y ejecutar aplicaciones Docker de varios contenedores. Se utiliza un archivo YAML para configurar los servicios de su aplicación, y luego, con un solo comando, se crean e inician todos los servicios desde su configuración. Tiene comandos para gestionar todo el ciclo de vida de una aplicación con los que se pueden[21]:

- Iniciar, detener y reconstruir servicios.
- Ver el estado de los servicios en ejecución.
- Transmitir la salida del registro de los servicios en ejecución.
- Ejecutar un comando único en un servicio.

⁵Imagen tomada de: <https://algodaily.com/lessons/what-is-a-container-a-docker-tutorial>

Capítulo 3

Diseño e implementación

En el presente capítulo se presentan los detalles del diseño y los criterios adoptados para el desarrollo del trabajo junto con los pasos seguidos para su implementación.

3.1. Arquitectura del sistema

El sistema tiene una configuración de arquitectura del tipo cliente-servidor. Está constituido por dos nodos y un servidor, los cuales están conectados a la red local y se comunican a través del protocolo MQTT. El servidor recibe los parámetros actuales de estado y cambios desde cada dispositivo, los procesa y almacena en la base de datos. También envía mensajes hacia los dispositivos para cambiar el estado de las salidas o el parámetro que se desee cambiar. Los usuarios pueden consultar y modificar el estado de los dispositivos desde un navegador web móvil o desde una computadora.

En la figura 3.1 se puede observar la arquitectura cliente-servidor del sistema implementado en el trabajo.

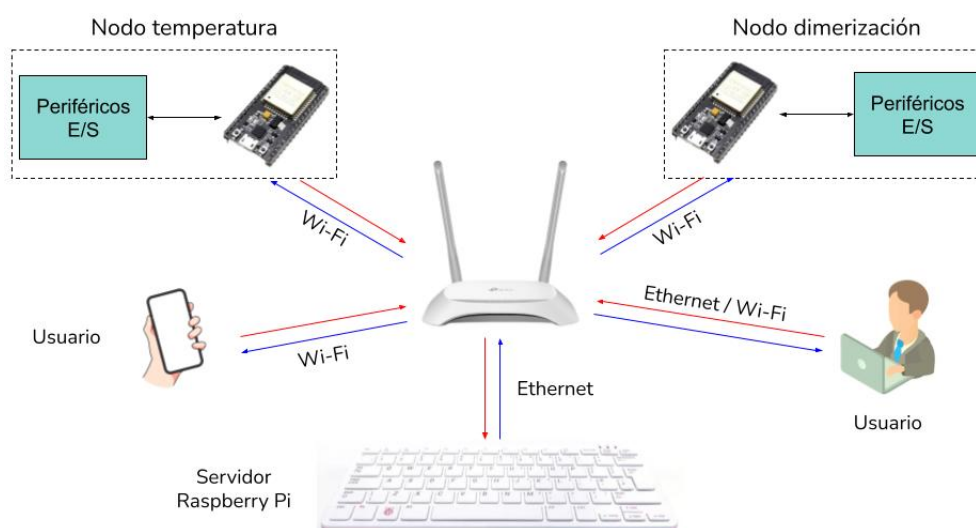


FIGURA 3.1. Arquitectura cliente-servidor.¹

¹Imagen tomada de: <https://www.bujarra.com/raspberry-pi-servidor-vpn-con-pptp/>

Uno de los nodos tiene como función sensar y controlar de temperatura de un recinto, y el otro controlar la iluminación. Originalmente el proyecto estaba pensado para que un solo nodo implemente estas dos funciones, pero durante el desarrollo se optó por la implementación separada. Este cambio se basó en la idea de modularizar los nodos y que sus funciones sean específicas. De esta forma es más amigable para el usuario visualizar y modificar el estado en pantalla e implementar el alta de nuevos dispositivos en el sistema.

El servidor está montado sobre una Raspberry Pi 400 con un sistema operativo Raspbian con interfaz gráfica. Este sistema operativo es la versión oficial ofrecida por la fundación Raspberry Pi y está basado en Debian versión 11 (*bullseye*). En esta etapa de desarrollo se optó por una Raspberry Pi 400 por una cuestión de costos y practicidad a la hora de desarrollar y hacer las pruebas, aunque tiene un mayor volumen que los otros modelos de la familia. Al momento de ofrecer una solución definitiva está pensado que sea implementado en una placa con el formato más pequeño como cualquiera de las Raspberry Pi 4. La conexión del servidor a la red local es por cable Ethernet.

3.1.1. Especificaciones técnicas del servidor

El sistema operativo del servidor está instalado y se ejecuta desde un disco de estado sólido por USB. Este tipo de discos tienen una mayor capacidad de escrituras y lecturas que una memoria microSD, lo que resulta favorable al momento de hacer modificaciones y pruebas de ejecución de software. En la versión final del sistema todo el software estará instalado en una tarjeta microSD para que todo el conjunto sea lo más pequeño posible.

En la tabla 3.1 pueden verse las especificaciones técnicas de hardware más importantes del modelo Raspberry Pi 400.

TABLA 3.1. Especificaciones de Raspberry Pi 400.

Procesador	Broadcom BCM2711 quad-core Cortex-A72(ARM v8) 64-bit SoC @ 1.8GHz
Memoria	4GB LPDDR4-3200
Conectividad	2.4GHz and 5.0GHz 802.11b/g/n/ac wireless LAN Bluetooth 5.0, BLE Gigabit Ethernet
Alimentación	5V DC vía USB-C

3.2. Modelo de datos

En esta sección se describen las diferentes tablas dentro de la base de datos de tipo relacional MariaDB. Con el objetivo de mostrar una representación visual fácil de comprender se muestran las imágenes representadas en la página phpMyadmin. Las tablas que forman dicha base son:

- Dispositivos.
- Usuarios.
- Mediciones.

Debe tenerse en cuenta que todo el contenido que se ejecuta del lado del servidor se encuentra dentro de un contenedor Docker. Dicho contenido corresponde a las imágenes de los servicios de Ionic, MariaDB, phpMyAdmin, backend con Node y mosquito. En la figura 3.2 se observa el contenido del archivo *docker-compose.yml* pertinente a la configuración de los servicios de MariaDB y phpMyAdmin.

```

mariadb:
  image: tobi312/rpi-mariadb:10.6-alpine
  hostname: mariadb
  environment:
    MARIADB_ROOT_PASSWORD: userpass
    MARIADB_DATABASE: Domotica
    MARIADB_USER: mysql
    MARIADB_PASSWORD: userpass2
  container_name: mariadb
  restart: unless-stopped
  volumes:
    - ./db/dumps:/docker-entrypoint-initdb.d
    - ./db/data:/var/lib/mysql
  networks:
    - app-fullstack-net
  ports:
    - "3306:3306"

phpmyadmin:
  image: phpmyadmin
  environment:
    PMA_HOST: mariadb
    PMA_PORT: 3306
    MARIADB_ROOT_PASSWORD: userpass
  container_name: phpmyadmin
  networks:
    - app-fullstack-net
  depends_on:
    - mariadb
  ports:
    - "8001:80"

```

FIGURA 3.2. Configuración de Docker de la base de datos. ²

3.2.1. Tabla Dispositivos

Esta tabla contiene los datos de los dispositivos dados de alta en el sistema. Dichos datos son el ID del dispositivo, el nombre, la ubicación, la dirección MAC, el tipo, el valor de la alarma y el estado de la alarma (0 para desactivada y 1 para activada).

En la figura 3.3 puede verse la tabla cargada con 2 dispositivos funcionando.

dispositivoid	nombre	ubicacion	mac	tipo	alarma	act_al
0028192332001	ESP32+DHT22	Habitación	94:B5:55:2B:FF:64	Temperatura	26	0
0128192332001	ESP32	Sala	B0:A7:32:DD:18:0C	Luz dimmer	0	0

FIGURA 3.3. Tabla de dispositivos. ³

El ID del dispositivo es un número brindado por el fabricante del dispositivo que consta de 13 números y está formado por: los primeros 2 dígitos que corresponden al tipo de dispositivo (el sistema está pensado para cubrir hasta un total de 100 tipos de dispositivos, aunque en esta etapa de prototipo sólo se hayan desarrollado 2); 4 dígitos de seguridad fijos que en este caso son "2819z" sirven para corroboración y como seguridad; y los últimos 5 dígitos corresponden al número de serie del equipo (2 para el año, 2 para la semana y 3 para el número de fabricación del equipo en esa semana, en ese orden). Como puede observarse, el sistema está pensado para que en un futuro sea parte de una producción en serie.

Tanto el nombre como la ubicación son campos alfanuméricos elegidos por el usuario para describir al dispositivo. La dirección MAC es enviada por el dispositivo la primera vez que se conecta al sistema y no se completa por el usuario. El tipo corresponde al tipo de sensor y para esta etapa del desarrollo puede ser "Temperatura." o "Luz dimmer".

El valor de la alarma es un campo numérico y sólo tiene efecto para dispositivos del tipo de temperatura. Se enviará una notificación por mail cada vez que el valor enviado por el dispositivo sea mayor o igual al seteado en este campo, siempre que dicha alarma esté activada en el campo.

3.2.2. Tabla Usuarios

La tabla de usuarios contiene todos los datos relacionados a aquellas personas que vayan a utilizar el sistema. En esta etapa está implementado con 3 usuarios, ya que al ser de tipo hogareño resultaría difícil que más de 3 personas dentro de una misma casa se registren. Pero con pocos cambios el sistema podría ser escalable a la cantidad de usuarios que se desee, generando una página de alta de usuarios.

Los valores almacenados en esta tabla son el ID de usuario (de 1 a 3), el nombre de usuario, la clave, el nombre de la persona y su apellido, su e-mail y un campo de tipo booleano que se modificará cuando se haya actualizado por primera vez. Cabe aclarar que cada vez que se ingrese al sistema con un usuario que no haya actualizado sus datos, se mostrará en pantalla un aviso para que este actualice sus datos.

En la figura 3.4 puede verse la tabla cargada con los 3 usuarios de los cuales hay 2 actualizados y el tercero tiene los valores por defecto.

userid	user	password	nombre	apellido	email	updated
1	javier	celot	Javier	Fanelli	javifanelli@gmail.com	1
2	romina	user	Romina	Carrizo	rominamarcarrizo@gmail.com	1
3	user3	user	Nombre	Apellido	user3@example.com	0

FIGURA 3.4. Tabla de usuarios. ⁴

3.2.3. Tabla Mediciones

La tabla de mediciones contiene todos los datos referentes a las mediciones realizadas por los dispositivos y los datos de modo seleccionado. Los dispositivos reportan cada 5 minutos estos datos y se almacenan en esta tabla.

Los datos almacenados en esta tabla son: el ID de la medición (un valor auto-incremental), el ID del dispositivo, el tipo de dispositivo, la fecha y hora de la medición, el valor de la medición, el modo de funcionamiento (manual o automático), el valor de la salida (si es de tipo temperatura es 0 o 100 y corresponde a encendido o apagado, y si es de tipo luz dimmer va de 0 a 100 con saltos de 10), y la hora y minutos de encendido y apagado para el modo automático.

En la figura 3.5 pueden verse algunas de las mediciones dentro de la correspondiente tabla, ordenadas por ID. Cabe aclarar que sólo se ven algunas ya que los dispositivos se encuentran reportando y llenando de datos la base.

medicionId	dispositivoId	tipo	fecha	valor	set_point	modo	salida	hon	mon	hoff	moff
1	0128192332001	Luz dimmer	2023-10-01 21:32:50	0	50	Manual	0	20	0	8	0
2	0028192332001	Temperatura	2023-10-01 21:32:56	21	20	Manual	0	20	0	8	0
3	0128192332001	Luz dimmer	2023-10-01 21:37:53	0	50	Manual	0	20	0	8	0
4	0028192332001	Temperatura	2023-10-01 21:38:02	21	20	Manual	0	20	0	8	0
5	0128192332001	Luz dimmer	2023-10-01 21:42:57	0	50	Manual	0	20	0	8	0
6	0028192332001	Temperatura	2023-10-01 21:43:07	21	20	Manual	0	20	0	8	0
7	0128192332001	Luz dimmer	2023-10-01 21:48:00	0	50	Manual	0	20	0	8	0
8	0028192332001	Temperatura	2023-10-01 21:48:13	21	20	Manual	0	20	0	8	0
9	0128192332001	Luz dimmer	2023-10-01 21:53:03	0	50	Manual	0	20	0	8	0
10	0028192332001	Temperatura	2023-10-01 21:53:18	21	20	Manual	0	20	0	8	0
11	0128192332001	Luz dimmer	2023-10-01 21:58:06	0	50	Manual	0	20	0	8	0
12	0028192332001	Temperatura	2023-10-01 21:58:24	21	20	Manual	0	20	0	8	0
13	0128192332001	Luz dimmer	2023-10-01 22:03:09	0	50	Manual	0	20	0	8	0
14	0028192332001	Temperatura	2023-10-01 22:03:30	21	20	Manual	0	20	0	8	0
15	0128192332001	Luz dimmer	2023-10-01 22:08:13	0	50	Manual	0	20	0	8	0
16	0028192332001	Temperatura	2023-10-01 22:08:35	21	20	Manual	0	20	0	8	0
17	0128192332001	Luz dimmer	2023-10-01 22:13:16	0	50	Manual	0	20	0	8	0
18	0028192332001	Temperatura	2023-10-01 22:13:41	21	20	Manual	0	20	0	8	0
19	0128192332001	Luz dimmer	2023-10-01 22:18:19	0	50	Manual	0	20	0	8	0
20	0028192332001	Temperatura	2023-10-01 22:18:46	21	20	Manual	0	20	0	8	0

FIGURA 3.5. Tabla de mediciones. ⁵

3.3. Desarrollo del frontend

El frontend del presente trabajo fue desarrollado en el lenguaje TypeScript con Angular como *framework* integrado con Ionic. El prototipo de aplicación está diseñado para acceder desde un navegador web tanto desde una computadora como un móvil, pero se optó por usar Ionic para un posterior desarrollo de una aplicación para sistemas operativos móviles. Es por esto que en esta instancia puede referirse a la aplicación web como una SPA y no como una PWA.

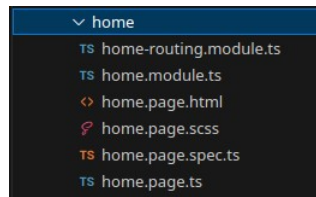
En la figura 3.6 se puede observar el fragmento de código correspondiente a la configuración de Ionic dentro del archivo *docker-compose.yml*.

```
ionic-ui:
  build:
    context: ./src/frontend/dam
    dockerfile: Dockerfile
  ports:
    - "8100:8100"
  container_name: ionic-ui
  volumes:
    - ./src/frontend/dam:/src/frontend/dam
    - /src/frontend/dam/node_modules
  command: ionic serve --external
```

FIGURA 3.6. Configuración de Docker de Ionic. ⁶

La estructura de archivos de cada página esta diseñada de la misma forma como se muestra en la imagen 3.7. Allí se pueden ver como ejemplo los archivos que componen la página *home*.

Además se utilizaron interfaces para definir las estructuras de datos de los dispositivos, las mediciones y los usuarios, y servicios para el proceso de autenticación y de consultas HTTP al backend, que serán explicadas posteriormente.

FIGURA 3.7. Estructura de archivos de página.⁷

El servicio de autenticación consta de el ingreso de un usuario y contraseña en la página de *login* que se comparan con los que están almacenados en la base de datos. Si dichos valores ingresados corresponden con alguno de los existentes, se genera un *token* desde el backend que se almacena en el dispositivo que está haciendo la consulta. Esto permite que se puedan acceder a las demás páginas de la aplicación.

En el código 3.1 puede verse el fragmento de la autenticación.

```

1
2 export class AuthGuard {
3
4   constructor(private _loginService: LoginService, private
      _router: Router) {}
5
6   canActivate(
7     route: ActivatedRouteSnapshot,
8     state: RouterStateSnapshot): Observable<boolean | UrlTree> |
      Promise<boolean | UrlTree> | boolean | UrlTree {
9
10    if (!this._loginService.logIn) {
11      this._router.navigate(['/login'])
12      return false
13    }
14
15    return true;
16  }
17 }

```

CÓDIGO 3.1. Autenticación de rutas

3.3.1. Rutas y páginas más importantes

Las rutas disponibles en la aplicación se describen en la tabla 3.2:

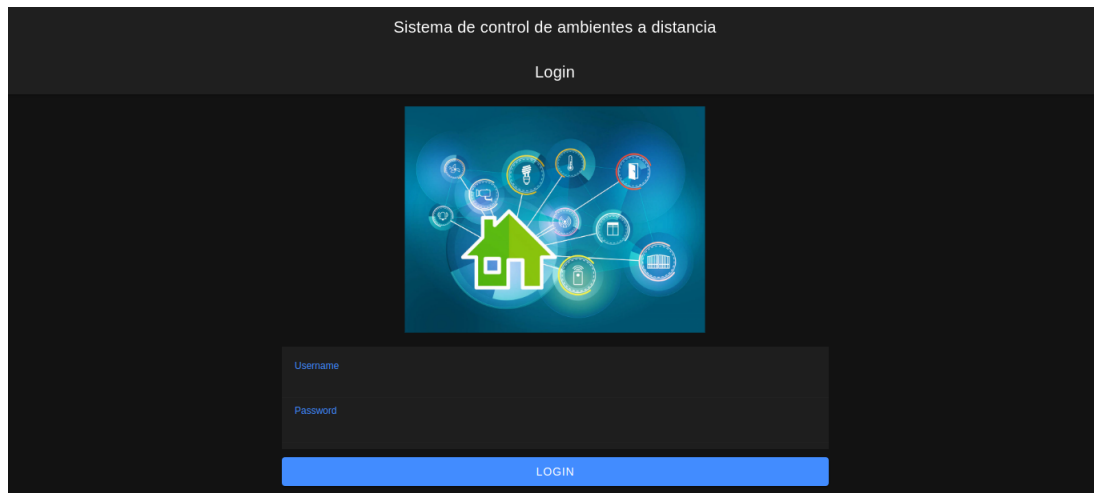
Pantalla de *login*

En la figura 3.8 se muestra la página de inicio de sesión en la aplicación.

El sistema trae la posibilidad de configurar 3 usuarios distintos para que utilicen, visualicen y configuren la aplicación. Todos poseen el mismo rol y pueden configurar cualquier campo que corresponda a dicho usuario ingresando en la opción Usuario dentro de la pantalla principal.

TABLA 3.2. Rutas de la aplicación

Ruta	Descripción
/login	Pantalla de inicio de sesión
/home	Pantalla principal con funciones y listado de dispositivos
/usuario/:id	Pantalla de edición de datos de usuario
/ayuda	Pantalla de manual de usuario
/dispositivos/:id	Pantalla de visualización de estado y última medición del dispositivo
/medicion/:id	Pantalla de visualización de las mediciones del dispositivo
/grafico/:id	Pantalla de visualización del gráfico de las mediciones del día
/config/:id	Pantalla de configuración del dispositivo
/modificar/:id	Pantalla de edición de datos de un dispositivo existente
/agregar	Pantalla para agregar un dispositivo nuevo

FIGURA 3.8. Pantalla de *login*.⁸

Pantalla principal *home*

Esta es la pantalla principal de la aplicación. Aquí se pueden encontrar todas las opciones más relevantes de configuración y visualización del sistema.

3.4. Desarrollo del backend

3.5. Nodos, sensores y actuadores

Los nodos están compuestos por el módulo central y los periféricos de entrada y salida. En el desarrollo del prototipo se utilizaron 2 versiones de módulo central, una que cuenta con un microcontrolador ESP32 para el nodo que controla temperatura y otra que cuenta con un ESP32 para el nodo que controla la iluminación dimerizable. En la tabla 3.3 pueden verse las características más importantes de ambos modelos de microcontroladores.

TABLA 3.3. Especificaciones técnicas de los módulos ESP32.

Característica	ESP32	ESP32C3
Núcleo	Xtensa® dual-core 32-bit LX6 @240 MHz	32-bit single-core RISC-V @160 MHz
Flash	0 MB, 2 MB o 4MB (dependiendo la versión)	0 MB o 4 MB (dependiendo la versión)
Protocolo Wi-Fi	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz

A continuación se listan periféricos utilizados como sensores y actuadores con sus principales características:

- DHT22: sensor de temperatura y humedad relativa ambiente.
 - Rango de temperatura: -40 a 80 grados Celsius.
 - Resolución: 0,1 grado Celsius.
 - Comunicación: serie, bus de 1 hilo, 40 bits por trama.
- KY-040: encoder rotativo con interruptor cuya función es cambiar los parámetros desde el nodo. 20 pulsos por vuelta.
- SSD1306: display OLED 1,2 pulgadas.
 - Resolución: 128x64 píxeles.
 - Interfaz: I2C.
- Control de potencia para 220 V: módulo con aislación y salida de triac para control de la calefacción. Se puede utilizar también para cualquier tipo de carga de 220 V 50 Hz.
 - Tipo: encendido - apagado (on - off).
 - Carga máxima: 8 A.
 - Tipo de aislación: optoacoplador.
 - Tensión máxima de aislación: 7500 V AC pico, 1 segundo de duración.
- Control de intensidad de luz: módulo de control para iluminación LED de corriente continua implementado con modulación por ancho de pulso (PWM). También se puede utilizar como módulo de encendido y apagado para cualquier tipo de cargas de corriente continua.
 - Tipo: PWM y encendido - apagado (on - off).
 - Carga máxima: 800 mA CC.
 - Tensión de alimentación: 5 a 24 V CC.

En la imagen 2.4 puede observarse el diagrama en bloques del nodo de temperatura y control de calefacción. Los periféricos de entrada son el encoder y el sensor de temperatura y los de salida son el display que muestra la información y la salida de potencia.

En la figura 2.5 puede verse el diagrama en bloques del nodo de dimerización de la luminaria de corriente continua. En este caso, a diferencia del modelo descripto

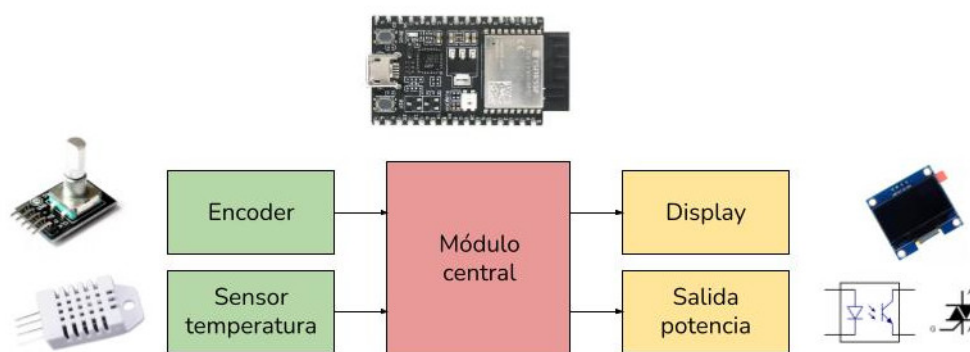


FIGURA 3.9. Diagrama en bloques del nodo de temperatura.

antes, posee un solo periférico de entrada y el de salida controla una salida de corriente continua de 5 a 24 V. En el caso del proyecto se utilizó una luminaria de 5 V y un consumo de 200 mA.

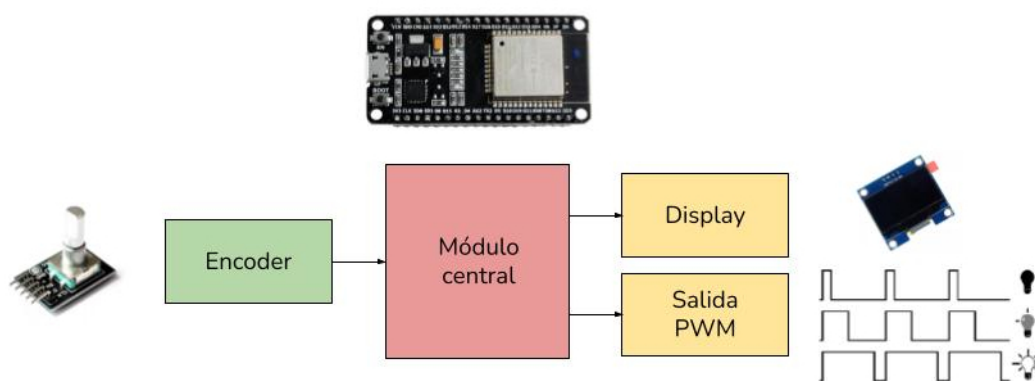


FIGURA 3.10. Diagrama en bloques del nodo de dimerización.

3.6. Comunicación del sistema

Capítulo 4

Ensayos y resultados

Capítulo 5

Conclusiones

Bibliografía

- [1] *Domótica ¿Qué es la domótica? ¿Cómo funciona?* 2023. URL: <https://e-ficiencia.com/domotica-que-es-y-como-funciona/>.
- [2] *Domótica - Wikipedia*. 2023. URL: <https://es.wikipedia.org/wiki/Domotica>.
- [3] *Matter*. 2023. URL: <https://csa-iot.org/all-solutions/matter/>.
- [4] *Home assistant*. 2023. URL: <https://www.home-assistant.io/>.
- [5] *Domotic*. 2018. URL: <https://www.sistemasdomotic.com.ar/>.
- [6] *Commax*. 2023. URL: <http://domotica.com.ar/>.
- [7] *Reactor*. 2023. URL: <https://www.reactor.com.ar/>.
- [8] *Modelo OSI, Wikipedia*. 2023. URL: https://es.wikipedia.org/wiki/Modelo_OSI.
- [9] *Protocolo HTTP, Wikipedia*. 2023. URL: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto.
- [10] *Protocolo MQTT, Goto IoT*. 2021. URL: https://www.gotoiot.com/pages/articles/mqtt_intro/index.html.
- [11] *Seguridad de la capa de transporte, Wikipedia*. 2023. URL: https://es.wikipedia.org/wiki/Seguridad_de_la_capa_de_transporte.
- [12] *Sistema en un chip, Wikipedia*. 2023. URL: https://es.wikipedia.org/wiki/Sistema_en_un_chip.
- [13] *Lenguaje de programación C, Wikipedia*. 2023. URL: [https://es.wikipedia.org/wiki/C_\(lenguaje_de_programaci3n\)](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci3n)).
- [14] *Lenguaje de programación JavaScript, Wikipedia*. 2023. URL: <https://es.wikipedia.org/wiki/JavaScript>.
- [15] *Lenguaje de programación TypeScript, Wikipedia*. 2023. URL: <https://es.wikipedia.org/wiki/TypeScript>.
- [16] *¿Qué es Angular?, Hubspot*. 2022. URL: <https://blog.hubspot.es/website/que-es-angular>.
- [17] *Qué es Ionic: ventajas y desventajas de usarlo para desarrollar apps móviles híbridas, profile*. 2021. URL: <https://profile.es/blog/que-es-ionic/>.
- [18] *¿Cuál es la diferencia entre MariaDB y MySQL?, AWS*. 2023. URL: <https://aws.amazon.com/es/compare/the-difference-between-mariadb-vs-mysql/#:~:text=MariaDB%20is%20more%20scalable%20and,multiple%20engines%20in%20one%20table..>
- [19] *¿Qué son los contenedores?, NetApp*. 2021. URL: <https://www.netapp.com/es/devops-solutions/what-are-containers/>.
- [20] *Docker (software), Wikipedia*. 2023. URL: [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software)).
- [21] *Descripción general de Docker Compose, Docker*. 2023. URL: <https://docs.docker.com/compose/>.