

CARRERA DE ESPECIALIZACIÓN EN INTERNET DE LAS COSAS

MEMORIA DEL TRABAJO FINAL

Sistema de monitoreo de servicios de planta

Autor:
Ing. Marcelo Roberto García

Director:
Mg. Ing. Gonzalo Nahuel Vaca (INVAP)

Jurados:
Nombre del jurado 1 (pertenencia)
Nombre del jurado 2 (pertenencia)
Nombre del jurado 3 (pertenencia)

*Este trabajo fue realizado en la ciudad de Buenos Aires,
entre marzo de 2022 y julio de 2023.*

Resumen

La presente memoria describe el desarrollo e implementación de un sistema de recolección de datos de bajo costo enfocado en la optimización del mantenimiento de servicios de planta. El trabajo se realizó para la empresa ROEMMERS S.A.I.C.F en el marco de una propuesta de mejora por parte del departamento de electrónica con la colaboración del departamento de servicios.

En este trabajo se utilizaron los conocimientos obtenidos de la carrera de especialización en IoT referidos a protocolos de comunicación, tecnologías de backend, frontend, bases de datos, sistemas operativos y redes.

Agradecimientos

A mi pareja, por su apoyo incondicional.

A mis padres y hermano.

A los profesores, profesoras, compañeros y compañeras por compartir sus conocimientos y experiencias.

Al Ing. Guillermo Horacio Vidal, Jefe de servicios de laboratorios ROEMMERS, por su confianza.

A Gabriel Méndez, Jefe de mantenimiento electrónico de laboratorios ROEMMERS, por su apoyo.

A todos los que participaron de forma directa e indirecta de este proyecto.

Índice general

Resumen	I
1. Introducción general	1
1.1. Servicios de planta	1
1.2. Motivación	2
1.3. Estado del arte	4
1.3.1. Orígenes	4
1.4. Objetivos y alcances	7
2. Introducción específica	9
2.1. Protocolos de comunicación	9
2.1.1. SPI	9
2.1.2. I2C	10
2.1.3. UART	10
2.1.4. LoRa	11
2.1.5. ModBUS TCP	11
2.1.6. HTTP	12
2.1.7. Wi-Fi	12
2.1.8. MQTT	13
2.2. Tecnologías de backend	14
2.2.1. NodeJS	14
2.2.2. Express	14
2.3. Tecnologías de frontend	15
2.3.1. Ionic	15
2.3.2. Highcharts	15
2.4. Dispositivos <i>baremetal</i>	16
2.4.1. ESP32	16
2.4.2. STM32	17
2.5. Herramientas utilizadas	18
2.5.1. Visual Studio Code	18
2.5.2. STM32 Cube IDE	19
2.5.3. Postman	19
2.5.4. phpMyAdmin	19
2.5.5. Wireshark	19
Bibliografía	21

Índice de figuras

1.1. Línea de bistleado y estuchado. ¹	1
1.2. Planta purificadora de agua de ósmosis inversa.	2
1.3. Distribución de los servicios en planta.	3
1.4. Primera generación de SCADA, estructura monolítica.[1]	4
1.5. Segunda generación de SCADA, estructura distribuida.[1]	5
1.6. Tercera generación de SCADA, estructura en red.[1]	5
1.7. Estructura básica de un SCADA. ²	6
2.1. Conexión entre dispositivos SPI maestro - esclavo.	9
2.2. Bus de conexión de dispositivos I2C.	10
2.3. Conexión entre dispositivos UART[2].	10
2.4. Modulación CSS.	11
2.5. Arquitectura de comunicación ModBUS TCP.	11
2.6. Métodos HTTP de uso frecuente.	12
2.7. Evolución de la tecnología Wi-Fi. ³	13
2.8. Modelo de publicación y suscripción. ⁴	13
2.9. Arquitectura NodeJS.	14
2.10. Modelo de aplicación capacitor.	15
2.11. Gráficos disponibles en librería <i>Highcharts</i> . ⁵	16
2.12. Indicadores disponibles en librería <i>Highcharts</i>	16
2.13. Montaje de módulo ESP32	17
2.14. Módulo STM32	18
2.15. Módulos de comunicación adicionales	18
2.16. Interfaz STLINK V2	19

Índice de tablas

Capítulo 1

Introducción general

En este capítulo se describen las características del mantenimiento de los servicios de planta, los sistemas de control asociados, su estado del arte, y los objetivos y alcances para el desarrollo del siguiente trabajo.

1.1. Servicios de planta

Las plantas industriales son las instalaciones por medio de las que es posible la producción de bienes a gran escala. Casi la totalidad de los elementos que se consumen, utilizan y desechan a diario provienen o han sido procesados en una planta industrial.

Este trabajo se encuentra enfocado en una planta farmacéutica donde se producen medicamentos en diversas presentaciones como: sólidos, polvos, efervescentes, líquidos e inyectables.

A continuación por medio de la figura 1.1 se detalla la configuración de una línea de producción de sólidos. Esta se encuentra compuesta por una serie de máquinas automáticas donde se recibe el medicamento en polvo para ser comprimido, luego blisteado, estuchado, pesado, etiquetado, apilado y finalmente paletizado.

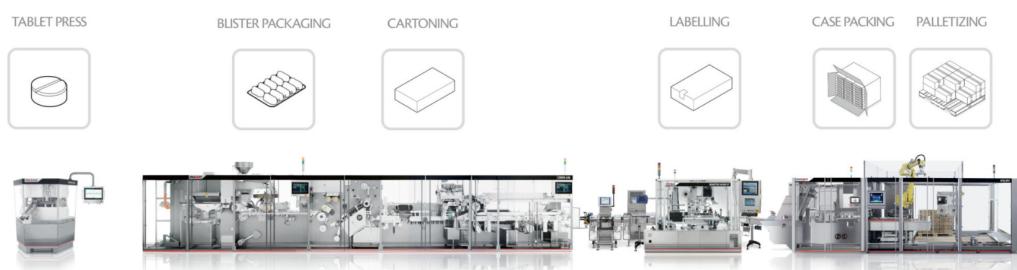


FIGURA 1.1. Línea de blisteado y estuchado.¹.

Las máquinas automáticas cuentan con sistemas de control neumáticos, hidráulicos, térmicos, eléctricos y electrónicos. Estos sistemas requieren servicios esenciales para su funcionamiento, como:

- Electricidad.
- Vapor industrial / sanitario.

¹Imagen tomada de https://ima.it/pharma/wp-content/uploads/sites/2/2022/10/LINEA-BLISTER-con-PROCESSO_ICON_BS-scaled-e1666881021854-2048x726.jpg

- Agua helada / purificada.
- Aire comprimido.

1.2. Motivación

El departamento de mantenimiento de planta se encuentra formado por tres sectores: servicios, mecánica y electrónica.

Los servicios de planta son un componente fundamental para su funcionamiento. El departamento de servicios entre otras tareas se encarga de asegurar y mantener su provisión. Algunos de estos se detallan a continuación:

- Potencia eléctrica.
- Gas Natural.
- Vapor industrial / sanitario.
- Agua potable / purificada y agua para la producción de inyectables.
- Aire comprimido.
- Efluentes cloacales / industriales.
- Mantenimiento de edificio, luminarias, etcétera.

La producción de medicamentos en la Argentina es auditada por la A.N.M.A.T [3] y requiere el cumplimiento de las buenas prácticas de manufactura GMP(*Good Manufacturing Practices*). Por este motivo todos los servicios que impactan de manera directa sobre el producto son monitoreados por sistemas de supervisión, control y adquisición de datos denominados SCADA.

Actualmente la planta cuenta con dos sistemas SCADA, uno para el control de HVAC (*Heating, Ventilating, Air Conditioned*) y otro para el control de las plantas de tratamiento de agua purificada. Estos sistemas registran variables críticas de la planta. Cuando estas se encuentran fuera de especificación pueden generar desvíos en la producción y observaciones en los lotes producidos.

La figura 1.2 muestra las instalaciones de una planta purificadora de agua de ósmosis inversa con todos sus servicios.



FIGURA 1.2. Planta purificadora de agua de ósmosis inversa.

Una planta purificadora de agua se alimenta de los servicios de: agua potable para luego ser purificada, electricidad para el funcionamiento del sistema de control, aire comprimido para el accionamiento de válvulas y vapor para el control de temperatura del agua.

Los servicios mencionados se encuentran distribuidos a lo largo y a lo ancho de la planta como se puede apreciar en la figura 1.3. La revisión de su estado se realiza en forma local, lo que implica el control periódico por parte de un técnico de mantenimiento.

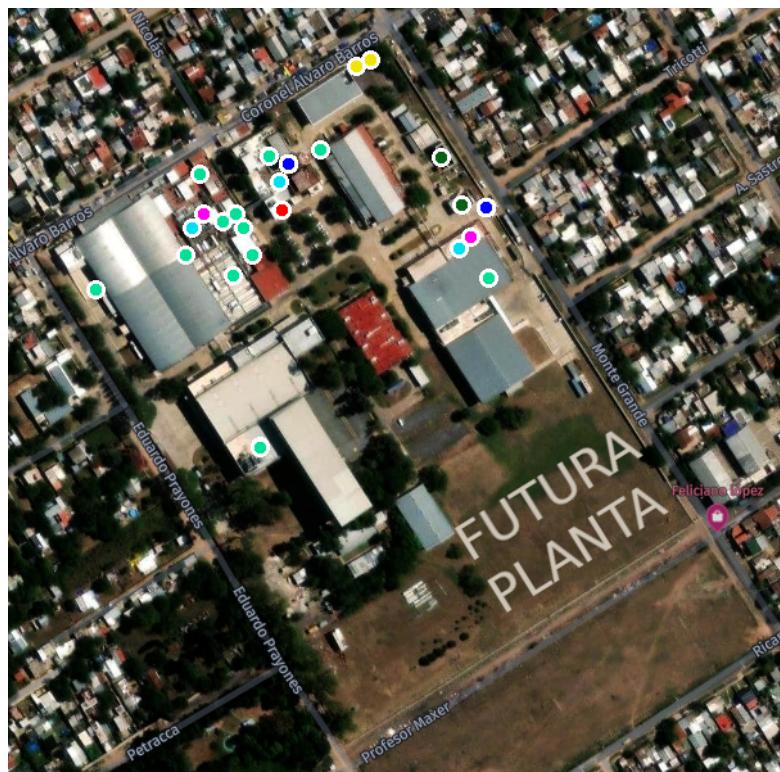


FIGURA 1.3. Distribución de los servicios en planta.

Referencias:

- Amarillo: Gas Natural.
- Azul: Potencia eléctrica.
- Celeste: Agua purificada.
- Rojo: Vapor industrial.
- Violeta: Vapor sanitario
- Verde oscuro: Efluentes.
- Verde claro: Separadores de polvo asociados a HVAC.

La motivación de este proyecto es poder brindarle al departamento de mantenimiento de servicios una herramienta que le permita verificar de manera remota el estado de los servicios de planta, consultar sus valores históricos y facilitar el desarrollo de estrategias para el mantenimiento preventivo y predictivo en base a los datos obtenidos.

1.3. Estado del arte

Los sistemas de supervisión, control y adquisición de datos SCADA son utilizados por organizaciones e industrias del sector público y privado. Los principales objetivos y beneficios de la implementación de estos sistemas son:

- Control y mantenimiento de la eficiencia de los procesos.
- Lectura en tiempo real de indicadores y datos de planta.
- Almacenamiento de registros históricos.
- Información averías para reducir el tiempo de parada.
- Gestión de reportes.
- Centralización de los datos de planta.

1.3.1. Orígenes

Los orígenes de estos sistemas se remontan a la década del 50. En ese entonces el control y operación de los equipamientos de una planta se efectuaba de forma manual mediante el accionamiento de pulsadores, llaves y diales.

A medida que las plantas industriales crecían, se requería más personal para que las operara e incluso debían recorrerse grandes distancias para llegar al punto de operación de cada instalación.

A principios de los años 50 las primeras computadoras fueron desarrolladas con propósitos de control en el ámbito industrial y los sistemas de control se volvieron populares en las industrias que presentaban mayores utilidades como es el caso de las petroleras.

En las décadas del 60 y 70 se incorpora la telemetría para el monitoreo. Esto permitió la transmisión de mediciones provenientes de sitios remotos a un equipo central (*Mainframe*)^[4]. Estas estructuras recibieron el nombre de estructuras monolíticas, cuya estructura se detalla en la figura 1.4.

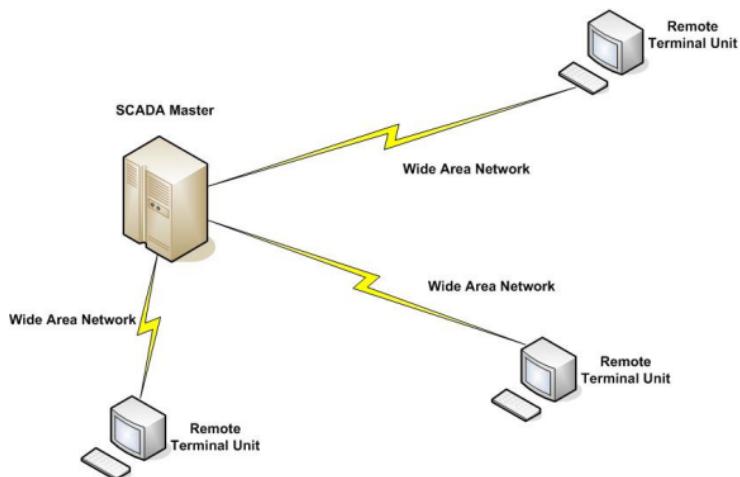


FIGURA 1.4. Primera generación de SCADA, estructura monolítica.
ca.[1]

En las décadas del 80 y 90 los sistemas evolucionaron con el advenimiento de la tecnología LAN (*Local Area Networking*) y el desarrollo de computadoras más pequeñas. Cada sistema contaba con protocolos LAN de tipo propietario, por lo tanto la comunicación entre dispositivos de otros sistemas no era posible. Estos sistemas recibieron el nombre de "sistemas distribuidos[4]". La figura 1.5 representa un sistema SCADA distribuido.

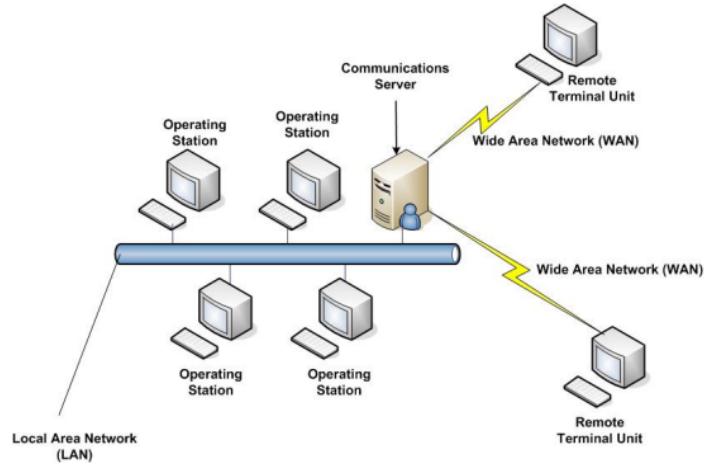


FIGURA 1.5. Segunda generación de SCADA, estructura distribuida.[1]

A mediados de los años 90 y principios del año 2000 el crecimiento industrial y la aparición de nuevos fabricantes de equipamiento llevó a los sistemas SCADA a un modelo de arquitectura abierta. La comunicación se basó en protocolos no propietarios, lo que permitió que la funcionalidad adquirida por los SCADA pueda distribuirse en redes de área extensa WAN. La utilización del protocolo IP y nuevos estándares permitieron su desarrollo de manera más eficiente y efectiva en el tiempo[4]. En la figura 1.6 puede observarse la implementación de una estructura en red.

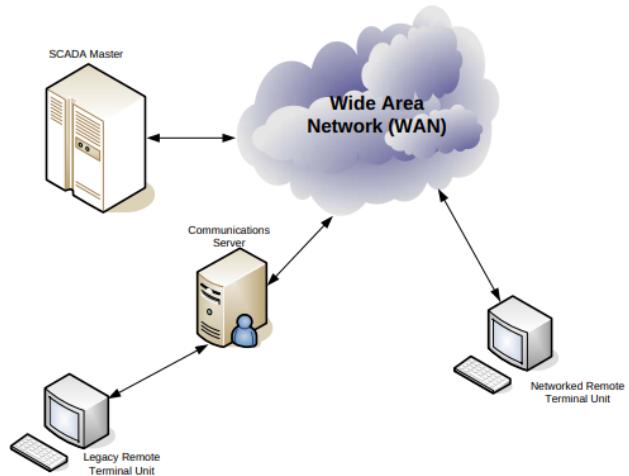


FIGURA 1.6. Tercera generación de SCADA, estructura en red.[1]

A medida que los sistemas SCADA incorporaron protocolos de comunicación abiertos, la interoperabilidad y compatibilidad entre sistemas y hardware mejoró, al punto de que un SCADA WinCC diseñado por Siemens puede encontrarse vinculado a controladores de otros fabricantes como Allen Bradley, Omron y otros.

Si bien esta integración es perfectamente realizable, cabe aclarar que desde el punto de vista de la practicidad y velocidad de implementación, esta será más eficiente utilizando componentes de hardware y software del mismo fabricante, además presenta ventajas económicas debido a que los programas necesarios para el diseño y puesta en marcha requieren licencias pagas.

Este tipo de estructura puede visualizarse en la figura 1.7.

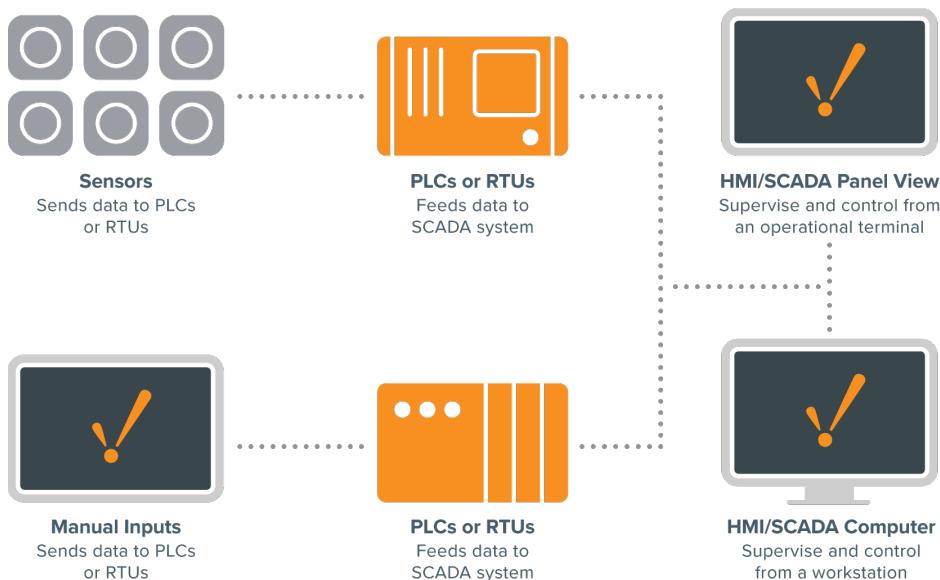


FIGURA 1.7. Estructura básica de un SCADA.²

Las empresas líderes en sistemas SCADA utilizan en la actualidad el concepto de SaaS (*Software as a Service*)^[5]. La idea de este concepto es proveer un servicio de software en la nube donde el usuario pueda contratar servicios adicionales en tanto los requiera por medio de una suscripción. Entre las ventajas de estos sistemas se pueden destacar las siguientes:

- Gestión y mantenimiento en la nube.
- Disponibilidad para agregar o quitar servicios.
- Interacción con sistemas de producción de planta.
- Acceso al sistema por medio de dispositivos móviles.
- Sistema escalable y con mayor potencia de cálculo para el uso de herramientas de predicción.

²Imagen tomada de <https://inductiveautomation.com/blog/sites/default/files/inline-images/BasicSCADADiagram>

1.4. Objetivos y alcances

Los principales objetivos de este trabajo son:

- Facilitar la implementación de la lectura y registro de variables de planta mediante el uso de herramientas de hardware y software de bajo costo.
- Reducir la frecuencia de chequeo *in situ* de las instalaciones.
- Aportar nuevos datos al departamento de mantenimiento de servicios para poder desarrollar estrategias de mantenimiento preventivo y predictivo en base a su análisis.
- Permitir la adaptación del sistema de monitoreo de acuerdo a las necesidades requeridas del sector.
- Implementar un sistema que sea la base para la creación de nuevas funcionalidades que facilite la gestión y obtención de los datos de planta.

De acuerdo a los objetivos planteados, se definen los alcances que permitirán lograr los objetivos propuestos, que se detallan a continuación:

- Instalación de un servidor en sala de control DDC1.
- Diseño e instalación de las bases de datos.
- Desarrollo de backend y frontend de la solución.
- Desarrollo del hardware y firmware de una interfaz de conexión y una interfaz de adquisición capaces de comunicarse con el backend utilizando los protocolos Modbus TCP, MQTT TLS y HTTPS.

No se encuentran contemplados dentro del alcance de este trabajo los siguientes objetivos:

- Vinculación del sistema a servicios en la nube.
- Desarrollo de aplicación para dispositivos móviles.

Capítulo 2

Introducción específica

En el siguiente capítulo se realiza una introducción a las tecnológicas utilizadas en el desarrollo de este trabajo. Estas se aplican a las distintas capas del modelo de arquitectura de IoT *Internet of Things*.

2.1. Protocolos de comunicación

Los protocolos de comunicación son estándares que se utilizan para definir la manera en la que se vinculan uno o mas dispositivos. Existe un gran número de protocolos diseñados para resolver distintas problemáticas, a continuación se detallan los utilizados en el desarrollo de este trabajo.

2.1.1. SPI

EL protocolo de comunicación SPI *Serial Peripheral Interface* es un protocolo de comunicación serie, sincrónico y *full duplex*. Los dispositivos sincrónicos cuentan con una señal de *clock* para la sincronización de los datos enviados y recibidos. A su vez, las señales MOSI *Master Output Slave Input* y MISO *Master Input Slave Output* permiten la transferencia simultánea de datos entre los dispositivos, esta funcionalidad recibe el nombre de *full duplex*[6].

En la figura 2.1 se observa la conexión entre dispositivos SPI.

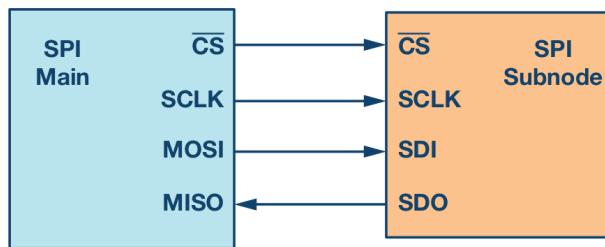


FIGURA 2.1. Conexión entre dispositivos SPI maestro - esclavo.

Los dispositivos SPI pueden ser direccionables mediante las señales de CS *chip select* y alcanzar velocidades de reloj de hasta 50MHz. Esto permite una gran capacidad de transferencia de datos, motivo por el cual son utilizados en dispositivos como, pantallas LCD, módulos ethernet y memorias entre otros.

2.1.2. I2C

I2C *Inter Integrated Circuit* es un protocolo de comunicación serie, sincrónico y bidireccional con un número reducido de hilos para su conexión. Se caracteriza por ser muy versátil y económico, el protocolo I2C se ha implementado en más de 1000 circuitos integrados que han sido fabricados por más de 50 compañías. Además, se utiliza en arquitecturas de control como SMBus *System Management Bus*, BPM *Bus Power Management Bus*, IPMI *Intelligent Platform Management Interface*, DDC *Display Data Channel* and ATCA *Advanced Telecom Computing Architecture*[7].

Cada dispositivo cuenta con una dirección única e inalterable para su direccionamiento. De acuerdo al tipo, pueden incorporar una o más entradas CS *chip select* para comunicarse con dispositivos idénticos sobre el mismo bus.

En la figura 2.2 se observa la conexión de dispositivos en un bus I2C.

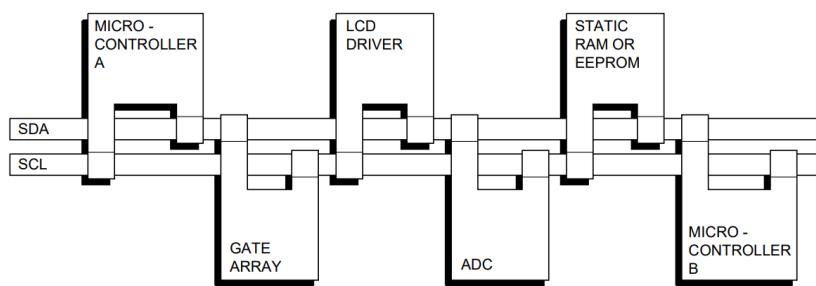


FIGURA 2.2. Bus de conexión de dispositivos I2C.

Las velocidades de comunicación varían desde los 100 kHz a los 5 MHz, en la actualidad, cuentan con dispositivos para aplicaciones, militares, medicinales e industriales. Entre los más utilizados se pueden encontrar, memorias, conversores ADC, DAC, sensores de temperatura y humedad, RTC *Real Time Clock*, giróscopos y otros.

2.1.3. UART

El protocolo de comunicación UART *Universal Asynchronous Receiver Transmitter* es uno de los más antiguos y utilizados para comunicaciones serie, este protocolo cuenta con la posibilidad de manejar 3 modos de funcionamiento:

- *simplex* : El dispositivo solo envía o recibe datos.
- *half duplex*: El dispositivo envía y luego recibe datos.
- *full duplex*: El dispositivo envía y recibe datos simultáneamente.

Las conexiones entre dispositivos se realizan entre 1 transmisor y un 1 receptor, no pueden existir más de dos dispositivos conectados, dado que no tiene capacidad de direccionamiento. En la figura 2.3 se observa la conexión entre dispositivos USART.

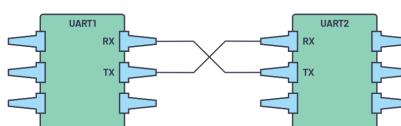


FIGURA 2.3. Conexión entre dispositivos UART[2].

El protocolo UART es utilizado en periféricos de computadora, microcontroladores, automóviles, *smart cards*, SIM telefónica y otros.

2.1.4. LoRa

LoRa *Long Range* es una tecnología de comunicación inalámbrica que utiliza la modulación *CSSChip Spread Spectrum* desarrollada por la empresa Semtech. Este tipo de modulación posee grandes ventajas a nivel de alcance, inmunidad al ruido, seguridad y consumo de energía[8].

En la figura 2.4 se observa la forma de onda modulada de un dispositivo LoRa.

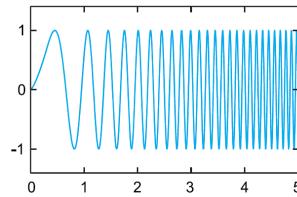


FIGURA 2.4. Modulación CSS.

Los dispositivos LoRa utilizan el espectro de frecuencias no licenciado ISM *Industrial, Scientific and Medical* de 915 MHz en la Argentina, esto representa una ventaja frente a otras tecnologías que utilizan frecuencias licenciadas como es el caso de SigFox, NB-IoT y LTE-M.

En la actualidad los dispositivos LoRa sin utilizados en aplicaciones de agricultura, ciudades inteligentes, cuidado de la salud, hogar, control industrial y cadena de suministros entre otros.

2.1.5. ModBUS TCP

Modbus es un protocolo de aplicación abierto maestro-esclavo que puede ser implementado sobre distintas capas físicas. Modbus TCP es la implementación del protocolo Modbus sobre Ethernet TCP/IP, un protocolo orientado a la conexión con el que se busca asegurar la entrega de datos.

En la figura 2.5 puede observarse una arquitectura de comunicación ModBUS TCP en la que se combina ModBUS TCP y ModBUS Serial a través de un *gateway*[9].

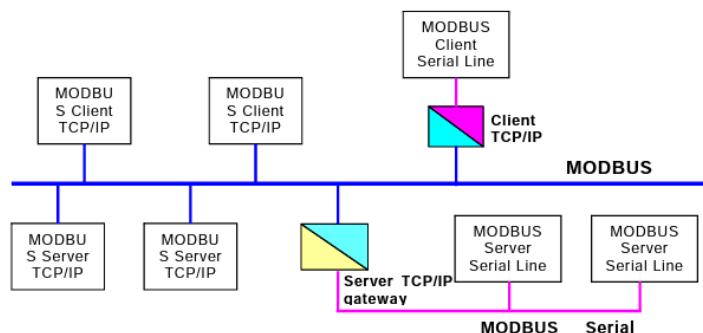


FIGURA 2.5. Arquitectura de comunicación ModBUS TCP.

Desde su implementación, este protocolo ha sido adoptado en la industria por una gran cantidad de fabricantes. Entre los más utilizados se pueden encontrar, controladores lógicos programables, interfaces hombre máquina, sensores, actuadores, variadores de velocidad y dispositivos de campo.

ModBUS es un protocolo sencillo, económico y de rápida implementación que al día de hoy se continúa utilizando en la industria.

2.1.6. HTTP

HTTP *Hypertext Transfer Protocol* es un protocolo de la capa de aplicación utilizado para la transmisión de documentos de hipertexto como HTML *HyperText Markup Language*[10].

Es un protocolo del tipo cliente - servidor, en el que un cliente establece una conexión con el servidor, realiza una petición y espera la respuesta del servidor. A este mecanismo se lo denomina *request/response*.

HTTP define un conjunto de métodos de petición en el que indica la acción que se desea realizar para un recurso determinado. Entre los más utilizados se pueden encontrar, los métodos PUT, GET, POST Y DELETE[11].

En la figura 2.6 se detallé el método y la acción que se ejecuta.

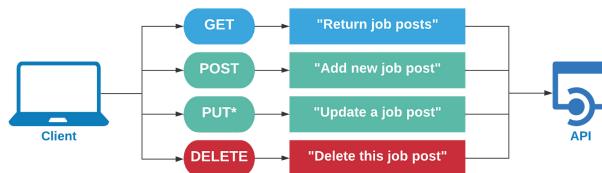


FIGURA 2.6. Métodos HTTP de uso frecuente.

En materia de seguridad, el protocolo HTTP puede encriptarse sobre TLS *Transport Layer Security* o bien sobre SSL *Secure Sockets Layer*.

2.1.7. Wi-Fi

Wi-Fi es un protocolo de red inalámbrico basado en las normas IEEE 802.11.

Las normas IEEE 802.11 especifican los protocolos de capa física y control de acceso al medio para la implementación de una red de área local que permita la comunicación entre dispositivos. Dentro de los dispositivos que conforman una red Wi-Fi se pueden encontrar los *access point* y los *routers*.

Un *access point* permite que los dispositivos inalámbricos se conecten a la red inalámbrica. El *access point* toma el ancho de banda proveniente de un router y lo extiende para que muchos dispositivos puedan conectarse a la red desde distancias más lejanas.

Un *router* inalámbrico, también llamado router Wi-Fi, combina las funciones de red de un router y un punto de acceso inalámbrico. Un router conecta las redes locales a otras redes locales o a Internet[12].

El uso de los recursos de internet se encuentra en constante crecimiento, los servicios de *streaming*, aplicaciones móviles y redes sociales generan mas contenidos

y de mayor calidad. Estos servicios demandan un mayor ancho de banda, lo que representa un desafío para las tecnologías disponibles adaptarse a los nuevos requerimientos.

En la figura 2.7 se observa la evolución de la tecnología Wi-Fi.

Wi-Fi generations					
	Wi-Fi 4	Wi-Fi 5	Wi-Fi 6	Wi-Fi 6E	Wi-Fi 7 (expected)
Launch date	2007	2013	2019	2021	2024
IEEE standard	802.11n	802.11ac	802.11ax	802.11be	
Max data rate	1.2 Gbps	3.5 Gbps	9.6 Gbps	46 Gbps	
Bands	2.4 GHz and 5 GHz	5 GHz	2.4 GHz and 5 GHz	6 GHz	1–7.25 GHz (including 2.4 GHz, 5 GHz, 6 GHz bands)

FIGURA 2.7. Evolución de la tecnología Wi-Fi.¹

En la actualidad, la tecnología Wi-Fi se encuentra disponible en un gran número de dispositivos hogareños, industriales y *weareables*. Estos dispositivos son utilizados diariamente, convirtiéndose en un recurso esencial para el desarrollo de las actividades cotidianas.

2.1.8. MQTT

El protocolo MQTT *Message Queuing Telemetry Transport* tiene sus orígenes en el año 1999 en la industria del petróleo y gas. El monitoreo de los oleoductos se realizaba vía satélite, esto requería de un protocolo con ancho de banda bajo y consumo de baterías mínimo.

El protocolo MQTT se basa en los principios del modelo de publicación y subscripción. Un cliente publica datos sobre un tópico, el broker los recibe y los envía a los clientes que estén subscriptos a ese tópico. Los tópicos son la forma en que los mensajes se filtran y organizan jerárquicamente.

En la figura 2.8 se detalla el modelo de publicación y subscripción.

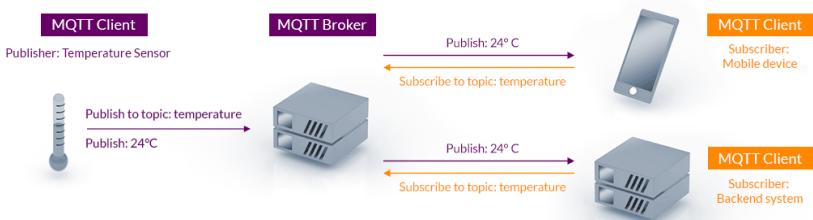


FIGURA 2.8. Modelo de publicación y subscripción.²

A nivel de seguridad, utiliza el protocolo SSL para proteger los datos, implementar identidad y autenticación entre clientes y brokers.

¹Imagen tomada de <https://techunwrapped.com/know-the-characteristics-and-speed-of-the-new-wifi-7-or-802-11be>

²Imagen tomada de <https://mqtt.org/assets/img/mqtt-publish-subscribe.png>

2.2. Tecnologías de backend

El *backend* es el desarrollo de la lógica y comunicación de servicios de una página web que permite disponer de los datos a ser visualizados por el usuario. Las instrucciones y consultas que el usuario realiza son procesadas por el *backend* y luego visualizadas en el *frontend*.

2.2.1. NodeJS

NodeJS es una plataforma de código abierto que ejecuta el motor de JavaScript V8 de Chrome. Las aplicaciones en NodeJS se ejecutan en un único proceso que corre en un solo hilo del procesador.

Se utiliza el concepto de programación orientada a eventos para poder dar soporte a la concurrencia de tareas, el *loop* principal *Event Loop* escucha los eventos y dispara una función *callback* cuando uno de estos eventos es detectado[13].

En la figura 2.8 puede observarse la arquitectura de la plataforma NodeJS.

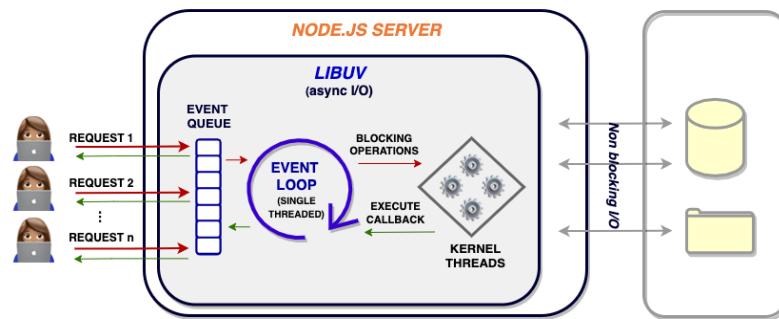


FIGURA 2.9. Arquitectura NodeJS.

A diferencia de otras plataformas, NodeJS consume menos recursos, todo la carga de trabajo de I/O se realiza de manera asíncrona para evitar la demora entre tareas.

2.2.2. Express

Express es un *framework* que otorga funcionalidad sobre el servidor web de NodeJS, es uno de los *frameworks* más utilizados. Entre sus principales características se destacan las siguientes[14]:

- Desarrollo de aplicaciones web de forma rápida y sencilla.
- Facilidad de configuración y uso.
- Definición de rutas de aplicaciones utilizando métodos HTTP y URL.
- Fácil integración con motores de plantillas.
- Especificación de *middleware* para manejo de errores.

2.3. Tecnologías de frontend

El *frontend* es la parte del desarrollo web que se encarga de la interacción, visualización y experiencia del usuario con la aplicación web.

2.3.1. Ionic

Ionic Framework es un kit de desarrollo de software *frontend* de código abierto para aplicaciones híbridas basado en tecnologías web HTML, CSS y JavaScript[15].

Ionic permite la creación de aplicaciones para iOS, Android y la web desde un único código. Esto es posible gracias a Capacitor, un *framework* para el desarrollo de aplicaciones multiplataforma que implementa las aplicaciones como una página web local.

Capacitor permite convertir cualquier proyecto web en una aplicación nativa de iOS o Android, cuenta con una librería de *plugins* nativos que habilitan el acceso a distintos dispositivos y características del sistema operativo[16].

En la figura 2.10 se visualiza el modelo de aplicación capacitor.

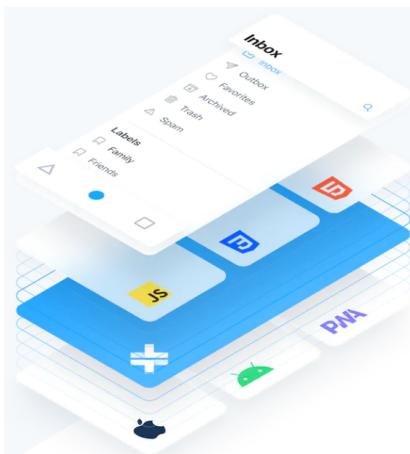


FIGURA 2.10. Modelo de aplicación capacitor.

Ionic cuenta también con una serie de herramientas para el desarrollo de la interfaz de usuario UI, entre las que se pueden encontrar, tarjetas, botones, formularios, animaciones y otros. Estos componentes trabajan con los *frameworks* React, Angular y Vue.

2.3.2. Highcharts

Highcharts es una librería de gráficos desarrollada en JavaScript, se caracteriza por su facilidad de integración en la página web, sus opciones de configuración y tipo de gráficos.

En la figura 2.11 puede observarse una vista reducida de los gráficos disponibles.

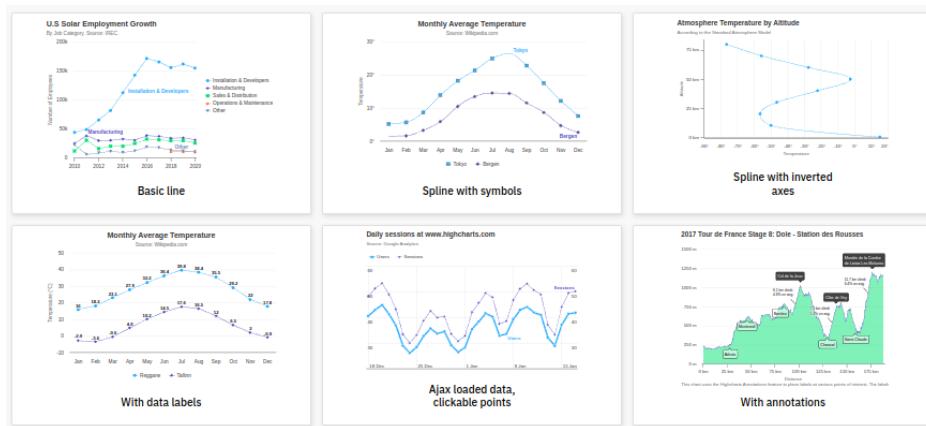


FIGURA 2.11. Gráficos disponibles en librería *Highcharts*.³

Otra herramienta muy útil para la representación de variables son los indicadores, estos son muy utilizados para la visualización de métricas en páginas relacionadas con servicios IoT.

En la figura 2.12 puede observarse una vista de los distintos tipos de indicadores.

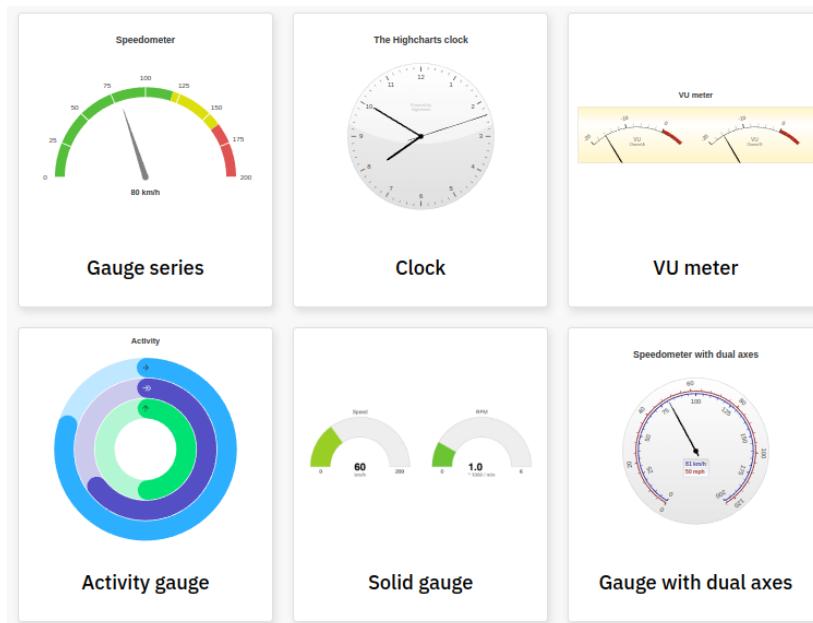


FIGURA 2.12. Indicadores disponibles en librería *Highcharts*

2.4. Dispositivos *baremetal*

Los dispositivos *baremetal* son dispositivos de baja capacidad de procesamiento, son utilizados en la capa de percepción del modelo IoT para la implementación de sensores y recolección de datos.

2.4.1. ESP32

ESP32 es un SoC *System On Chip* desarrollado por la compañía Espressif, el concepto de SoC es la integración de distintos componentes en el mismo circuito integrado, con esto se busca reducir la cantidad de componentes externos y mejorar

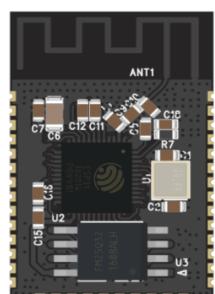
el funcionamiento del sistema.

Los componentes que integran en SoC son los siguientes[17]:

- Microprocesador doble núcleo de 32 bits.
 - Módulo WiFi + Bluetooth de 2,4 GHz.
 - Módulo Bluetooth LE.
 - Selectores de antena.
 - RF Balun.
 - Amplificador de potencia.
 - Amplificador de recepción de bajo ruido.
 - Filtros.
 - Módulos de manejo de potencia.

Espressif facilitó la implementación de sus SoCs mediante la creación de módulos, estos módulos cuentan con el SoC, la memoria de programa, los círculos y la antena montados en un circuito impreso listo para su uso.

En la figura 2.13a puede observarse el módulo ESP-WROOM32 con el SoC y sus componentes, por otro lado en la figura 2.13b se observa el módulo montado en una placa que permite el acceso a sus periféricos y cuenta con una interfaz USB-SERIE para su programación.



(A) Módulo ESP-WROOM32



(B) Módulo ESP-WROOM32
montado en placa desarrollo

FIGURA 2.13. Montaje de módulo ESP32

2.4.2. STM32

STM32 es una familia de microcontroladores desarrollados por la compañía ST que utiliza la arquitectura ARM-Cortex. A diferencia del ESP32, este microprocesador tiene la memoria integrada en el mismo chip[18].

Existen placas de desarrollo que cuentan con este tipo de microcontrolador, en estas se incorpora la conexión de sus entradas y salidas, fuente de alimentación, cristal de reloj, cristal de tiempo real, conector USB y pines para su programación y verificación.

En la figura 2.14 se visualiza una placa de desarrollo que utiliza este microcontrolador, a esta placa se la conoce con el nombre de *blackpill*.

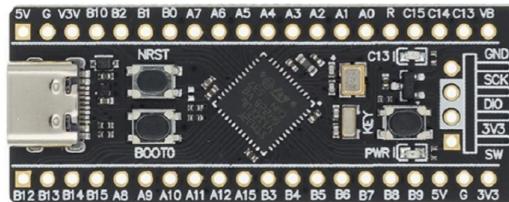


FIGURA 2.14. Módulo STM32

El módulo no tiene una estructura de SoC de comunicación, por lo que estas funciones se realizan con módulos adicionales. En la figura 2.15 se observan módulos de comunicación Wi-Fi, Ethernet y LoRa.

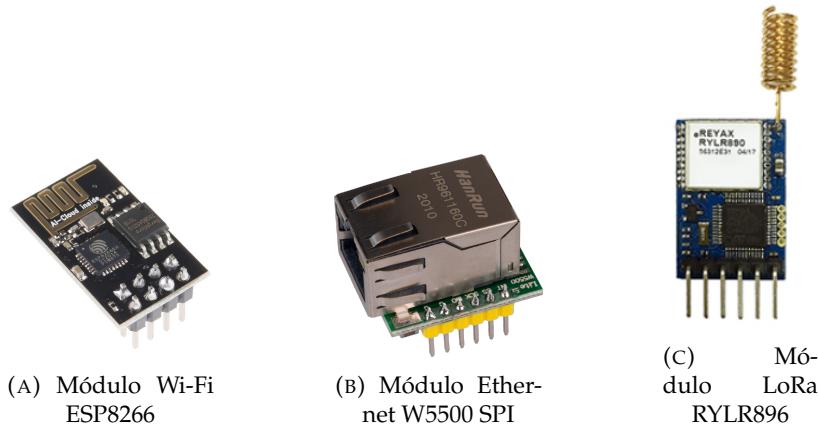


FIGURA 2.15. Módulos de comunicación adicionales

2.5. Herramientas utilizadas

En la desarrollo de este trabajo se utilizaron herramientas de software que permitieron la programación del *backend*, *frontend*, base de datos y los dispositivos *baremetal*. Además se utilizó software para el testeo del *backend*, simulación de datos y escucha de tráfico de red para la verificación de los mensajes enviados.

2.5.1. Visual Studio Code

Visual Studio Code es un editor de código liviano compatible con Windows, iOS y Linux. Este editor tiene la posibilidad de agregar extensiones específicas para distintos lenguajes y aplicaciones[19].

En el desarrollo del presente trabajo se utilizan las siguientes extensiones:

- C/C++ Extensions: Utilizada para la escritura de programa en lenguaje C.
- Espressif IDF: Herramienta para la programación de dispositivos ESP32.
- Docker: Herramienta para facilitar la creación y manejo de contenedores.
- Angular Essentials : Utilizado para la programación del frontend.
- Ionic: Herramienta para el desarrollo en Ionic y Capacitor

2.5.2. STM32 Cube IDE

STM32 Cube IDE es un entorno de desarrollo de C/C++ basado en Eclipse®/CDT™ para la programación, configuración de periféricos, generación, compilación y depuración de código de microcontroladores STM32[20].

La conexión para programar y depurar el programa en el microcontrolador se realiza a través de una interfaz ST-LINKV2 USB.

En la figura 2.16 se observa la interfaz de programación STLINK V2.



FIGURA 2.16. Interfaz STLINK V2

2.5.3. Postman

Postman es una interfaz de programación de aplicaciones que contiene herramientas para acelerar el diseño, prueba y verificación de las aplicaciones que se encuentran en desarrollo[21].

Postman se utilizó para la prueba y verificación de los *endpoints* del *backend*.

2.5.4. phpMyAdmin

phpMyAdmin es un software gratuito desarrollado en php que se utiliza para la administración de bases de datos MySQL y MariaDB. Soporta un amplio rango de operaciones de uso frecuente como la administración de la bases de datos, tablas, columnas, relaciones, índices, usuarios y permisos entre otros[22].

Una herramienta muy utilizada en el desarrollo de este trabajo fue la representación gráfica de las consultas SQL realizadas al *backend*.

2.5.5. Wireshark

Wireshark es un software de análisis de protocolos de red, soporta todo tipo de protocolos y permite la selección de las interfaces de red sobre las cuales se pretende analizar el tráfico[23].

Wireshark fue utilizado como herramienta de verificación para el estudio de los protocolos HTTPS y MQTTS.

Bibliografía

- [1] McClanahan. «The Benefits of Networked SCADA Systems Utilizing IPEnabled Networks, Rural Electric Power Conference». En: IEEE, 2002, págs. C5 -C5-7.
- [2] Mary Grace Legaspi Eric Peña. «UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter». En: *Analog Dialogue* (2020).
- [3] ANMAT. *Normativa de medicamentos*. Visitado el 2023-09-19. 2023. URL: http://www.anmat.gob.ar/webanmat/normativas_medicamentos_cuerpo.asp.
- [4] Alexandru UJVAROSI. «EVOLUTION OF SCADA SYSTEMS». En: *Bulletin of the Transilvania University of Brașov* (2016).
- [5] Jeremy Wilbert. «Rethinking HMI/SCADA for a digitally connected workforce». En: *AVEVA WHITEPAPER* (2023).
- [6] Piyu Dhaker. «Introduction to SPI Interface». En: *Analog Dialogue* (2018).
- [7] *I2C-bus specification and user manual*. UM10204. Rev. 7. NXP. 2021.
- [8] Lora Alliance. «A technical overview of LoRa® and LoRaWAN™». En: *Technical Marketing Workgroup 1.0* (2015).
- [9] MODBUS. *Messaging on TCP/IP Implementation Guide*. Rev. 1.0b. Modbus Organization. 2006.
- [10] Mozilla. *HTTP*. <https://developer.mozilla.org/es/docs/Web/HTTP>. Dic. de 2023. (Visitado 02-09-2023).
- [11] Mozilla. *Métodos de petición HTTP*. <https://developer.mozilla.org/es/docs/Web/HTTP/Methods>. Dic. de 2023. (Visitado 02-09-2023).
- [12] Cisco. ¿Qué es Wi-Fi? https://www.cisco.com/c/es_mx/products/wireless/what-is-wifi.html~preguntas-y-respuestas. Sep. de 2023. (Visitado 03-09-2023).
- [13] OpenJS Foundation. *Introduction to Node.js*. <https://nodejs.dev/en/learn/an-example-nodejs-application>. Sep. de 2023. (Visitado 04-09-2023).
- [14] Mozilla. *Express/Node introduction*. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. Sep. de 2023. (Visitado 05-09-2023).
- [15] ionic DOCS. *Introduction to Ionic*. <https://ionicframework.com/docs>. Sep. de 2023. (Visitado 06-09-2023).
- [16] ionic DOCS. *The Ionic Platform*. <https://ionic.io/docs/platform>. Sep. de 2023. (Visitado 06-09-2023).
- [17] ESPRESSIF. *ESP32*. <https://www.espressif.com/en/products/socs/esp32>. Sep. de 2023. (Visitado 10-09-2023).
- [18] STMicroelectronics. *STM32 32-bit Arm Cortex MCUs*. <https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html>. Sep. de 2023. (Visitado 11-09-2023).

- [19] Visual Studio Code. *Getting Started*. <https://code.visualstudio.com/docs>. Sep. de 2023. (Visitado 12-09-2023).
- [20] STMicroelectronics. *Integrated Development Environment for STM32*. <https://www.st.com/en/development-tools/stm32cubeide.html>. Sep. de 2023. (Visitado 12-09-2023).
- [21] Inc. Postman. *What is Postman?* <https://www.postman.com/>. Sep. de 2023. (Visitado 12-09-2023).
- [22] phpMyAdmin. *Bringing MySQL to the web*. <https://www.phpmyadmin.net/>. Sep. de 2023. (Visitado 12-09-2023).
- [23] Ulf Lampert Richard Sharpe Ed Warnicke. *Preface*. https://www.wireshark.org/docs/wsug_html_chunked/Preface.html. Sep. de 2023. (Visitado 12-09-2023).