

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE CONCEPCIÓN “REY BALDUINO DE BÉLGICA”**

**DISEÑO APLICACIÓN IoT PARA SISTEMA DE
REFRIGERACIÓN INDUSTRIAL**

Trabajo de Titulación para optar al Título
Profesional de Ingeniero en Control e
Instrumentación industrial.

Alumno:
Sr. Manuel Armando Briones Nahuelñir

Profesor Guía:
Sr. Nelson Vásquez Concha

Resumen

En el presente existe una auge en tecnologías que permiten integrar los datos de distintos procesos con aplicaciones que generen valor a estos como Machine Learning, Analítica de datos o Aprendizaje automático.

Unas de las tecnologías que ha tenido mayor auge en los últimos años es la Internet de las cosas o IoT(Internet of the Things) por sus siglas en Inglés.

La internet de las cosas es una red de objetos físicos como sensores, actuadores y controladores, integrados con diferentes herramientas digitales y tecnológicas con el fin de conectarse e intercambiar datos con otros dispositivos a través de Internet.

En este trabajo de título se diseñará una aplicación IoT para un proceso de refrigeración industrial. Llevando los datos de las variables físicas desde la capa de dispositivos, que capturan las variables físicas a través de distintos controladores que poseen sensores que toman las variables físicas del proceso.

Los datos de los dispositivos periféricos son capturados por un PLC CIMON a través de protocolos de comunicación.

Los datos que se registran en el PLC CIMON son capturados por una aplicación desarrollada con el lenguaje Python, la cual captura los datos gracias a la bibliotecas desarrolladas para este propósito y son publicadas a través del protocolo PUB/SUB a un tópico establecido.

Una vez los datos publicados se desarrollará un micro servicio el cual permitirá la captura de los datos publicados, los datos son enviados a una base de datos no relacional de serie de tiempo en InfluxDB.

En Grafana Cloud se integrarán los datos de InfluxDB para su posterior visualización y su eventual gestión.

ÍNDICE

INTRODUCCIÓN	1
CONTEXTO DEL PROBLEMA	2
NATURALEZA DEL PROBLEMA.....	2
PROPUESTA DE SOLUCIÓN.....	3
JUSTIFICACIÓN	3
MOTIVACIÓN.....	4
OBJETIVOS GENERAL Y ESPECÍFICOS	5
<i>Objetivo General</i>	5
<i>Objetivos Específicos</i>	5
HIPÓTESIS.....	5
DESARROLLO	6
1. MARCO TEÓRICO.....	7
1.1 <i>Reseña De Los Sistemas IoT</i>	7
1.2 <i>Sistema De Refrigeración Industrial</i>	7
1.3 <i>Arquitectura IoT</i>	8
1.4 <i>Python</i>	10
1.4 <i>Librerías En Python</i>	10
1.5 <i>Diccionario En Python</i>	11
1.6 <i>JSON (JavaScript Object Notation - Notación de Objetos de JavaScript)</i>	11
1.7 <i>¿Qué es un Framework?</i>	11
1.8 <i>NGROK</i>	12
2. CAPA DE DISPOSITIVOS	13
2.1 <i>Cámara De Refrigeración</i>	13
2.2 <i>Controlador De Temperatura Carel IR33+</i>	13
2.3 <i>Compresor Refrigeración</i>	14
2.4 <i>MYPRO TOUCH</i>	17
2.5 <i>CIMON – PLC S modelo CM3 – SB32MDCF</i>	19
2.6 <i>Raspberry Pi 4 Model B</i>	21
3. CAPA DE COMUNICACIÓN.....	23
3.1 <i>Comunicación En Serie RS-485</i>	23
3.2 <i>Protocolo De Comunicación Modbus</i>	23
3.3 <i>Mensaje De Lectura Modbus RTU</i>	25
3.4 <i>Mensaje De Respuesta Modbus RTU</i>	26
3.5 <i>Protocolo De Comunicación HTTP</i>	26
3.6 <i>Protocolo De Comunicación MQTT</i>	28
3.7 <i>Eclipse Mosquitto</i>	29
3.8 <i>Conceptos de Eclipse Mosquitto</i>	29
4. CAPA DE APLICACIONES	30
4.1 <i>¿Qué es un Base de Datos?</i>	30
4.3 <i>NoSQL</i>	31
4.4 <i>Tipos de NoSQL</i>	32
4.5 <i>Bases de datos relacionales</i>	32
4.6 <i>MySQL</i>	33
4.7 <i>Series De Tiempo</i>	33

4.8	<i>InfluxDB</i>	35
4.8.1	<i>Bucket</i>	36
4.9	<i>¿Qué es un Servicio Web?</i>	36
4.10	<i>API (Aplication Programming Interface)</i>	36
4.11	<i>API REST</i>	36
4.12	<i>Flask</i>	37
5.	CAPA VISUALIZACIÓN	37
5.1	<i>Grafana</i>	37
6.	IMPLEMENTACIÓN	38
6.1	<i>CAPA DE DISPOSITIVOS</i>	38
6.2	<i>CAPA DE ADQUISICIÓN DE DATOS</i>	39
6.3	<i>SOFTWARE DE PROGRAMACIÓN PLC CICON</i>	39
6.4	<i>DIRECCIONES MODBUS DISPOSITIVOS DE CAMPO</i>	40
6.5	<i>PROGRAMACIÓN EN SOFTWARE CICON</i>	41
6.5.1	<i>Configuración protocolo Modbus TCP-IP</i>	42
6.5.2	<i>Configuración programa Modbus RTU Master.</i>	43
7.	ADQUERIR LOS DATOS DEL PLC CON PYTHON	48
7.1	<i>Instalación librería easymodbus</i>	48
7.2	<i>Importación de librería y conexión con PLC esclavo</i>	48
7.3	<i>Lectura de registros PLC esclavo correspondiente a controlador IR33+.</i>	48
7.4	<i>Interpretación de registros leídos</i>	48
7.5	<i>Funciones para escalamiento de datos de lista HoldingRegs.</i>	49
7.6	<i>Generar variables a partir de la lista Coil.</i>	51
7.7	<i>Generar un elemento marca de tiempo</i>	52
7.8	<i>Crear un diccionario en Python con los datos obtenidos</i>	52
8.	PUBLICACIÓN DATOS ADQUIRIDOS A MQTT BROKER MOSQUITTO	53
8.1	<i>Crear una función para conexión de la aplicación con el servidor MQTT</i>	54
9.	DESARROLLO DE LA CAPA NIEBLA	55
9.1	<i>Creación de base de datos MySQL</i>	55
9.2	<i>Conexión a base de datos MySQL.</i>	56
9.3	<i>Insertar registros en base de dato MySQL</i>	57
9.4	<i>Visualización de datos en herramienta MySQL Workbench</i>	57
10.	CAPA APLICACIONES	59
10.1	<i>Configuración InfluxDB</i>	60
10.2	<i>TOKEN</i>	60
10.3	<i>BUCKETS</i>	61
10.4	<i>Escritura de datos en Influx</i>	62
10.5	<i>Visualización de datos en InfluxDB</i>	63
10.6	<i>Transferencia de datos del InfluxDB a Grafana Cloud</i>	65
10.7	<i>Configuración de fuentes de datos</i>	65
10.9	<i>Creación de Dashboard Grafana Cloud</i>	67
10.10	<i>Creación y configuración de panel de serie de tiempo en Grafana</i>	68
10.11	<i>Visualización de datos Grafana Cloud.</i>	70
	CONCLUSIONES	71
	BIBLIOGRAFÍA	72

ÍNDICE DE FIGURAS

Figura 1-1 Sistema de refrigeración industrial	8
Figura 2-1 Controlador de temperatura Carel IR33+	14
Figura 2-2 Interfaz RS-485.....	14
Figura 2-3 Compresor refrigeración Mycom.....	15
Figura 2-4 Mypro Touch Interfaz hombre-maquina.....	18
Figura 2-5 Controlador MYPRO TOUCH	18
Figura 2-6 CIMON-PLC S modelo CM3-SB32MDCF	19
Figura 2-7 Raspberry Pi 4 Model B.....	21
Figura 3-1 Diagrama lectura Modbus RTU.....	25
Figura 3-2 Diagrama respuesta Modbus RTU.....	26
Figura 3-3 Diagrama protocolo HTTP	27
Figura 3-4 Esquema funcionamiento protocolo MQTT	28
Figura 4-1 Clasificación series de tiempo	34
Figura 6-1 Capa de dispositivos; (1)PLC CIMON CM3-SB32MDCF, (2)MYPRO TOUCH, (3)Controlador ir33+,(4)Raspberry Pi 4 model B	38
Figura 6-2 Ventana desplegable parámetros	41
Figura 6-3 Parámetros protocolo Modbus RTU	42
Figura 6-4 Parámetros protocolo Modbus TCP/IP	42
Figura 6-5 Parámetros protocolo Modbus RTU Slave	43
Figura 6-6 Creación programa Modbus RTU Máster	44
Figura 6-7 Ventana configuración lectura de registros	44
Figura 6-8 Configuración de lecturas Holding Register y Coil.....	45
Figura 6-9 Registros a leer controlador IR33+	45
Figura 6-10 Conexión Programa CICON.....	46
Figura 6-11 Descarga programa hacia PLC	47
Figura 6-12 Ventana desplegable para la visualización de datos Monitor de dispositivos ..	47
Figura 9-1 Workbench sistema de gestión de BBDD.....	56
Figura 9-2 Workbench con solicitud a tabla de base de datos.....	58
Figura 10-1 Esquema de arquitectura de referencia	59
Figura 10-2 Fuentes de datos InfluxDB	60

Figura 10-3 Bucket para insertar datos	61
Figura 10-4 Interfaz InfluxDB.....	63
Figura 10-5 Visualización de datos en InfluxDB	64
Figura 10-6 Visualización sonda 1, query solicitud de visualización de datos.	64
Figura 10-7 Interfaz de inicio Grafana Cloud	65
Figura 10-8 Barra de opciones de configuración.....	66
Figura 10-9 Plugins InfluxDB para la integración con Grafana.....	66
Figura 10-10 Configuración de lenguaje de consulta y comunicación.	67
Figura 10-11 Configuración credenciales Influx.....	67
Figura 10-12 Barra de opciones Dashboard.	68
Figura 10-13 Opciones agregar panel.....	68
Figura 10-14 Opciones de paneles a configurar	69
Figura 10-15 Edición de panel para panel de tipo serie de tiempo.....	69
Figura 10-16 Edición de panel para panel de tipo indicador.	70
Figura 10-17 Dashboard con paneles configurados	70

ÍNDICE DE TABLAS

Tabla 2-1 Especificaciones de comunicación.....	19
Tabla 2-2 Especificaciones generales	20
Tabla 2-3 Características Raspberry Pi 4 model B	22
Tabla 6-3 Registros Controlador Mycom	40
Tabla 6-4 Lectura registros controlador IR33+	40

SIGLAS Y SIMBOLOGÍA

SIGLAS

API	: Application Programming Interface (Interfaz de Programación de Aplicaciones).
DBMS	: Database Management System.
ETL	: Extract, Transform and Load.(Extraer, Transformar y Cargar).
HTTP	:Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto).
HVAC	: Heating, Ventilating, Air Conditioned.
IAAS	: Infraestructura como Servicio.
IoT	: Internet of the Things (Internet de las Cosas).
JSON	: JavaScript Object Notation (Notación de Objeto de JavaScript).
MQTT	: Message Queuing Telemetry Transport.
NH3	: Amoniac.
NTC	: Termistores de coeficiente de temperatura negativo.
OOP	: Object-oriented Programming (Programación orientada a objetos).
PAAS	: Plataforma como Servicio.
PLC	: Programmable Logic Controller (Controlador lógico programable).
RTU	: Remote Terminal Unit (Unidad terminal remota).
SAAS	: Software como Servicio.
SBC	: Session border controller (Controlador de borde de sesión).
SCADA	: Supervisión, Control y Adquisición de Datos.
SQL	: Structured Query Language. (Lenguaje de consulta estructurado).
SSH	: Secure Shell.
SSR	: Solid State Relay (Relé de estado sólido).
SVGA	: Super Video Graphics Array.
TTL	: Lógica Transistor a Transistor.
URL	: Uniform Resource Locator (Localizador uniforme de recursos).
USB	: Universal Serial Bus.
UTC	: Coordinated Universal Time (Tiempo universal coordinado).

SIMBOLOGÍA

Kgf/cm^2	: Kilogramo-fuerza por centímetro cuadrado
$^{\circ}\text{C}$	: Grados Celsius
v	: Voltaje
$\text{k}\Omega$	: kilo Ohms

INTRODUCCIÓN

CONTEXTO DEL PROBLEMA

Desde hace algunas décadas, el uso de dispositivos conectados e inteligentes está incidiendo en el desarrollo de diversos segmentos de la industria. En primer lugar, la implantación del IoT en los procesos de refrigeración y sistemas de climatización (HVAC) permite la monitorización remota de los entornos, ya sea para realizar una operación asistida o para garantizar el mantenimiento preventivo.

Estas son algunas de las principales ventajas: Posibilidad de reducir el tiempo de inactividad de las máquinas en caso de avería, aumentar la vida útil de los componentes críticos y reducir el consumo mediante la programación de eventos o el uso de inteligencia artificial.

Las principales tendencias en este ámbito consisten en reducir el consumo de energía y hacer la monitorización preventiva mediante el rastreo del rendimiento de las máquinas e instalaciones de forma permanente. Las tecnologías de comunicación inalámbrica son aliadas de esta tendencia, que en el futuro permitirán difundir estos sistemas y reducir los costos de implementación y funcionamiento.

La posibilidad de visualizar los datos en la nube es también un camino sin retorno, ya que permite gestionar estos activos en cualquier lugar, más aún en situaciones como la que estamos viviendo actualmente, con restricciones a la libre circulación de personas.

NATURALEZA DEL PROBLEMA

Los que conocen el segmento de refrigeración probablemente entienden cuando se dice que esta industria no suele innovar a la hora de probar nuevas tecnologías. Sin embargo, la irrupción de tecnologías de Internet de las cosas permite integrar sistemas de refrigeración industrial, lo que nos permite seguir algunas tendencias probadas en países más desarrollados.

La preocupación por la seguridad de los datos y el funcionamiento de un sistema con acceso remoto y alojado en la nube es una cuestión que plantean habitualmente las partes interesadas que aún no han adoptado este tipo de automatización. Pero durante el proceso de creación del concepto de IoT, la seguridad de la información es una premisa importante.

El cifrado de datos y los protocolos diseñados para el IoT en la comunicación con la nube o los equipos informáticos certifican la seguridad. Los fabricantes de dispositivos son responsables de aplicar correctamente estos protocolos, garantizando la seguridad necesaria. Los instaladores y los clientes finales son responsables de establecer las claves de cifrado en los dispositivos, así como contraseñas seguras para acceder a los datos en la nube.

PROPUESTA DE SOLUCIÓN

Con lo expresado anteriormente se propone el diseño de una solución IoT para la captura de datos, monitoreo, registro de las variables de temperatura capturadas por un controlador y variables de un compresor de refrigeración MYCOM modelo 250-VSD mediante una aplicación IOT.

JUSTIFICACIÓN

En la actualidad la integración de tecnología de Internet de las Cosas en el proceso de refrigeración aumenta la eficiencia de los recursos disponibles para su producción. Siendo esta de gran importancia debido a la actual competencia que existe en el mercado nacional e internacional, así como por la exigencia de calidad de parte de los consumidores.

Chile actualmente sufre un rezago en la implementación de tecnología de tecnificación en procesos de refrigeración, debido mayoritariamente a un tema de desconocimiento. Hoy se trata de una inversión marginal para la importancia de la cadena de frío y los equipos que ella compone, donde generalmente se debe monitorear un túnel de aire forzado para materia prima, un túnel de aire forzado para producto terminado, la temperatura

y humedad relativa de cámaras de mantención y salas de procesos. Además de los equipos como compresores, ventiladores, condensadores y bombas recirculadoras.

El monitoreo de variables de refrigeración como presiones y temperatura resulta ser fundamental en el proceso de refrigeración, por ejemplo, una temperatura baja en la pulpa del arándano tiene una relación directa con su metabolismo así como una presión de descarga alta nos muestra señales de ineficiencia en nuestro proceso de refrigeración.

Otro punto importante es el tiempo para realizar la recolección y procesamiento de variables, actualmente la recolección de datos es de forma manual, teniendo como consecuencia demasiado tiempo para la realización del proceso de refrigeración.

MOTIVACIÓN

El desarrollo de este proyecto responde a la necesidad de la industria de la refrigeración industrial, para optimizar sus procesos productivos y de mantención, para así poder disminuir sus ciclos de refrigeración con lo cual conlleva a aumentar su competitividad y calidad de sus productos. El auge de tecnologías de acceso libre, como las plataformas IoT permite crear este tipo de soluciones que antes estaban disponibles solo para grandes empresas.

OBJETIVOS GENERAL Y ESPECÍFICOS

Objetivo General

Diseñar una solución IoT para la adquirir datos de variables físicas en sistemas de refrigeración industrial con el fin de aumentar la eficiencia en los procesos de frío, para así disminuir tiempos y costos.

Posibilidad de reducir el tiempo de inactividad de las máquinas en caso de avería, aumentar la vida útil de los componentes críticos y reducir el consumo mediante la programación de eventos o el uso de inteligencia artificial.

Objetivos Específicos

- 1.- Realizar un levantamiento de las distintas variables en el proceso de refrigeración.
- 2.- Realizar un repaso por las distintas capas de una Arquitectura IoT simplificada.
- 3.- Diseñar una aplicación IoT utilizando herramientas de captura de datos, protocolos de comunicación, base de datos y plataformas.

HIPÓTESIS

El desarrollo de un aplicación de Internet de las cosas industrial, para el monitoreo y eventual control de variables físicas de un sistema de refrigeración industrial, como la temperatura, presión alarma de eventos y la temperatura de cámaras de refrigeración, permitirá un mayor control, mejorar la gestión de activos e integrar nuevas tecnologías como Machine Learning o Inteligencia Artificial.

METODOLOGÍA PARA UTILIZAR

El tipo de metodología empleada en este trabajo de título es la investigación aplicada ya que tiene como objetivo resolver problemas prácticos y trata de unir la investigación con la práctica a través de la aplicación o utilización de los conocimientos adquiridos, a la vez que se adquieren otros, después de implementar y sistematizar la práctica basada en la investigación.

DESARROLLO

1. MARCO TEÓRICO

1.1 Reseña De Los Sistemas IoT

Internet de las cosas o IoT (por su acrónimo en inglés), es un concepto o tendencia tecnológica que se refiere al conjunto de tecnologías que permiten conectar cualquier objeto a internet y procesar los datos generados a partir de esta conexión, con el objetivo de entregar información útil para la toma de decisiones relevantes.

A diferencia de lo que suele creerse el IoT no posee como único objetivo el conectar cualquier objeto a internet, sino que persigue un objetivo mayor; el de entregar valor a través de los datos que se generan a partir de esta conexión de objetos nunca conectados.

Esta tecnología, al contrario de lo que se cree, lleva más de dos décadas de constante evolución, pero han sido los últimos años en que se ha producido una mayor disrupción. Se estima que para el año 2025 hallan conectados más de seis de dispositivos conectados por persona, esto quiere decir que habrá aproximadamente 48 mil millones de dispositivos conectados.

1.2 Sistema De Refrigeración Industrial

Los sistemas de refrigeración industrial se basan en principios termodinámicos y están diseñados para promover el intercambio de calor entre el proceso y el refrigerante y para facilitar la liberación del calor irrecuperable al medio ambiente. Los sistemas de refrigeración industrial pueden clasificarse en función de su diseño y de su principio básico de funcionamiento: agua o aire, o la combinación de ambos.

El intercambio de calorífico entre el medio de proceso y el refrigerante se intensifica a través de intercambiadores, donde el refrigerante (en este caso amoníaco) descarga su calor al ambiente. En los sistemas abiertos, el refrigerante está en contacto con el ambiente, cosa que no ocurre en los sistemas cerrados, donde el refrigerante o el medio de proceso circulan por tubos o serpentines.

El sistema de refrigeración no crea frío (imposible en las leyes de la termodinámica), sino que elimina el calor de los cuerpos y lo expulsa. Para lograrlo se manipula una sustancia refrigerante, que constantemente cambia de estado líquido a gaseoso sin perder sus propiedades y en el proceso, absorbe y evapora el calor.

Todos los procesos de refrigeración cuentan con secciones básicas que trabajan en un ciclo infinito.

Compresión - Condensación- Expansión – Evaporación

El diseño de esta solución se enfoca en dos componentes del sistema de refrigeración: La compresión realizada por un compresor de tipo tornillo y la evaporación producida en una cámara de refrigeración controlada por un dispositivo electrónico (Figura 1-1).

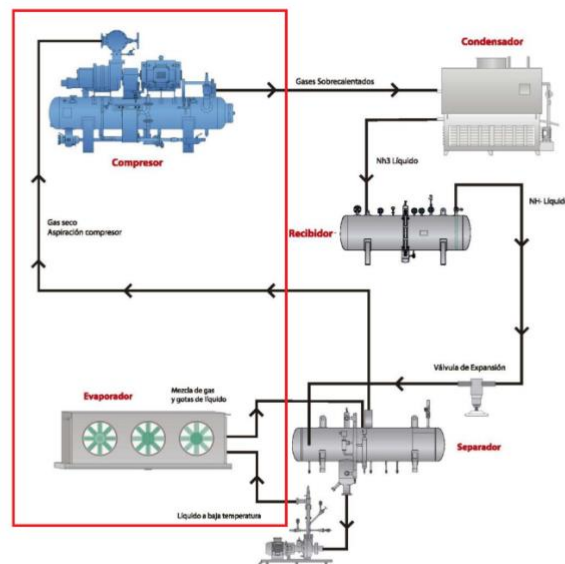


Figura 1-1 Sistema de refrigeración industrial

1.3 Arquitectura IoT

La arquitectura IoT se basa en distintas capas que interactúan entre ellas para cumplir una determinada función, principalmente transportar datos desde los sensores o actuadores desde la capa física, pasando por la capa de comunicación capaz de realizar las conexiones a

través de una red, a centro de gestión de datos corporativos o cloud hasta llegar a la capa de aplicación donde el usuario final los procesará para su análisis y almacenamiento.

En el diseño del presente trabajo se utilizará la arquitectura IoT simplificada

1.3.1 Nivel de dispositivo IoT

Ya que sin sensores precisos, actuadores y dispositivos IoT en general, no hay datos precisos y sin datos precisos no hay Internet de las cosas, y mucho menos proyectos o productos/servicios IoT. Capa donde actuadores y sensores se conectan a dispositivos que transmitirán los datos entre sí y hacia otras capas.

1.3.2 Puerta de enlace de IoT

La puerta de enlace de IoT(Internet de las cosas) es un dispositivo o sistema que actúa como intermediario entre los dispositivos IoT y la red de internet. La función principal de una puerta de enlace de IoT es permitir que los dispositivos IoT se conecten y se comuniquen con otros dispositivos y aplicaciones a través de internet. También pueden proporcionar servicios adicionales, como seguridad, gestión de dispositivos y análisis de datos.

1.3.3 Plataformas IoT

Donde se conectan con las aplicaciones y servicios comerciales y de consumo, así como el desarrollo de estos servicios y la gestión e interconexión con las dos primeras capas.

1.3.4 Capa niebla

Los datos son generalmente transportados a capas superiores a través de Gateway que permiten la comunicación entre aplicaciones y dispositivos. Pero qué ocurre cuando las capas de aplicaciones no conectan con la capa dispositivo, o se requieren respuestas mayores, es aquí donde el ‘Fog’ surge como respuesta a la necesidad de una escalabilidad horizontal en el borde. Proporciona un modelo de servicios similares al “Cloud Computing” (“IAAS”, “SAAS”, “PAAS”) con un enfoque de nodos perimetralmente distribuidos en las proximidades del usuario o de la fuente de datos, pero con una gestión centralizada. Permite

proporcionar elevados tiempos de respuesta, una alta capacidad de computación y un bajo consumo de ancho de banda en una amplia cobertura fuertemente asegurada. En ningún caso se plantea como un sustituto de las infraestructuras “Cloud”, sino como un complemento de estas.

1.4 Python

¿Qué es Python?

Python es un lenguaje de programación informático utilizado para la creación de sitios web, software, automatización de tareas y análisis de datos , creado por Guido Rossum a comienzos de la década del 90. Es un lenguaje de propósito general, por lo cual es ideal para aplicaciones IoT, ya que se puede utilizar en distintas capas.

Es un lenguaje de alto nivel con gramática sencilla, clara y muy legible, que se asemeja a un texto de lenguaje humano.

Características:

- Tipado dinámico y fuerte, no es necesario declarar el tipo de variable al principio del programa.
- Prescinde de punto y coma al final de cada sentencia.
- Lenguaje orientado a objetos (OOP).
- Open Source, su uso y distribución son gratuitos, incluso para fines comerciales.
- Librerías estándar muy amplia versátil
- Multiplataforma

1.4 Librerías En Python

Las librerías en lenguaje de programación son un conjunto de implementaciones funcionales codificadas que ofrecen interfaz bien definida para la funcionalidad que se invoca. Las instrucciones pueden ser utilizadas por diferentes programas

1.5 Diccionario En Python

Un diccionario en Python es una estructura de datos que permite almacenar cualquier tipo de información, desde cadenas de texto o caracteres hasta números enteros, con decimales, listas e incluso otros diccionarios. Al igual que sucede con un diccionario de lengua, los datos se encuentran ordenados utilizando una clave única para cada uno de ellos, lo que permite localizar cada uno de los datos de una forma muy rápida.

Los valores de un diccionario se guardan utilizando un par de valores que siempre van enlazados, característica que comparte con el formato JSON. Una es la denominada como Key o Clave, que es la que nos permite encontrar un dato dentro del diccionario. Cada clave está acompañada por el dato o valor al que representa.

1.6 JSON (JavaScript Object Notation - Notación de Objetos de JavaScript)

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

1.7 ¿Qué es un Framework?

Actualmente en el desarrollo moderno de aplicaciones web se utilizan distintos Frameworks que son herramientas que nos dan un esquema de trabajo y una serie

de utilidades y funciones que nos facilita y nos abstrae de la construcción de páginas web dinámicas.

En general los Frameworks están asociados a lenguajes de programación como Ruby y PHP, en el mundo de Python el más conocido es Django, pero Flask es una opción que quizás no tenga una curva de aprendizaje tan elevada pero nos posibilita la creación de aplicaciones web igual de complejas de las que se pueden crear en Django.

1.8 NGROK

Ngrok es una herramienta que nos permite exponer a internet una URL generada dinámicamente, la cual apunta a un servicio web que se está ejecutando en nuestra máquina local. Por ejemplo: si tenemos un servicio web arrancado en `http://localhost:8080`, NGROK genera dinámicamente una URL del tipo `http://xxxxxx.ngrok.io` visible en internet, y que apunta directamente a nuestro localhost.

Además, es que solo hay que ejecutar un comando desde el terminal. Ngrok también nos puede ayudar en otras situaciones, como, por ejemplo:

- **Testing en dispositivos móviles**

Podremos probar nuestro sitio web en desarrollo desde cualquier dispositivo móvil o tablet de forma rápida y sencilla.

- **Acceso SSH**

Podremos tener una forma pública de acceder a un dispositivo dentro de nuestra red privada de forma segura.

- **Demos con cliente**

Poder mostrar al cliente el estado en el que se encuentra el desarrollo de una aplicación. Está disponible para Linux, Mac y Windows.

2. CAPA DE DISPOSITIVOS

2.1 Cámara De Refrigeración

Una cámara de refrigeración es espacio aislado térmicamente dentro del cual se ingresa algún producto para extraer su energía térmica a través de distintos dispositivos que forman un sistema de refrigeración, por lo general una cámara de refrigeración es gestionada por un controlador de temperatura, el cual es el encargado de accionar ventiladores del evaporador, además del ingreso de líquido refrigerante hacia el evaporador, accionando directamente o indirectamente dispositivos como solenoides o bombas de refrigeración.

2.2 Controlador De Temperatura Carel IR33+

En la figura 2-1 se presenta el dispositivo electrónico encargado de gestionar el proceso de refrigeración en una cámara, además de la sonda NTC de regulación, se pueden configurar otras tres sondas, como una sonda de producto (solo visualización) una sonda de condensación, una sonda anticongelante y una sonda de deshielo. Gracias a las funciones avanzadas de deshielo, se puede retrasar u omitir el deshielo cuando se dan las condiciones oportunas. Con las salidas digitales (relés).

Se puede controlar la electroválvula o el compresor, los ventiladores del evaporador o del condensador, el deshielo, la luz y las alarmas de baja y alta temperatura. Posee protocolo de comunicación Modbus RTU con conexión serial RS-485 con sistemas remotos de supervisión.

2.2.1 Sensores, actuadores y comandos de controlador IR33+.

Comandos on-off y deshielos.

Sonda temperatura NTC rango -50° a 150° C.

Accionamiento relé ventiladores.

Accionamiento relé de deshielo.

Accionamiento relé frio.

Alarmas alta y baja temperatura.



Figura 2-1 Controlador de temperatura Carel IR33+

El controlador electrónico IR33+ no dispone de la electrónica necesaria para conectarlo a una red RS-485, sin embargo, dispone de un conjunto de señales TTL que utilizando el conversor de la figura 2-2, permite la conexión a la red RS-485.



Figura 2-2 Interfaz RS-485

2.3 Compresor Refrigeración

Un compresor tipo tornillo es un equipo que desplaza caudal, en este caso un gas refrigerante de tipo natural (NH₃). Este equipo es el corazón de un sistema de refrigeración ya que tiene la primordial función de desplazar el gas sobrecalentado que viene desde el evaporador instalado en la cámara y se deposita en un estanque separador de líquido y gas.

Este gas sobrecalentado es desplazado hacia el condensador evaporativo para que su temperatura baje abruptamente y mantenga su presión, en la figura 2-3 representa un compresor del tipo tornillo.



Figura 2-3 Compresor refrigeración Mycom

2.3.1 Variables Del Sistema De Refrigeración

Presión y temperatura de succión

El vapor refrigerante que sale del evaporador viaja a través de la línea de succión hacia la entrada del compresor. El compresor toma el gas a baja presión y lo comprime aumentando así su presión y temperatura.

Variable medida por transmisor de presión -1 a 10 kgf/cm²

Variable medida por termocupla -50 a 50°C.

Presión y temperatura de descarga

Cuando el compresor desplaza el caudal del vapor, eleva su presión y temperatura debido a la reducción de volumen a la que ha sido sometida la masa de refrigerante aspirado por el compresor, hasta aproximadamente cinco veces la presión de succión, llegando a una presión nominal de 11 kgf/cm², el gas caliente se desplaza hasta el condensador donde es rápidamente enfriado y cambia de estado gaseoso a líquido a alta presión, donde se acumula en un estanque.

Variable medida por transmisor de presión -1 a 30 kgf/cm²

Variable medida por termocupla 0 a 150°C.

Sobrecalentamiento

El sobrecalentamiento es el número de grados que un vapor está por encima de su temperatura de saturación, o punto de ebullición a una determinada presión. La importancia de conocer el sobrecalentamiento del sistema da una indicación de si la cantidad de refrigerante que fluye hacia el evaporador es adecuada para la carga.

Un sobrecalentamiento alto provoca una deficiente refrigeración y un consumo excesivo de energía. Un sobrecalentamiento bajo, ocasionado por un exceso de refrigerante en el sistema provoca que el refrigerante vaya en estado líquido al compresor ocasionando daños.

Temperatura, presión y presión diferencial de aceite

Todas las etapas del compresor son lubricadas por aceite a través de una bomba, que toma el aceite depositado en el cárter de compresor y lo distribuye a todas las etapas, los transmisores de presión de alimentación de aceite y transmisor de presión después del filtro son los encargados de realizar las mediciones para la correcta lubricación del equipo.

Variable presión de aceite transmisor -1 a 30 kgf/cm²

Variable presión filtro de aceite transmisor -1 a 30 kgf/cm²

Variable temperatura de aceite en colector 0 a 150°C

Variable temperatura de aceite en separador 0 a 150°C

2.4 MYPRO TOUCH

MYPRO TOUCH es un controlador desarrollado para el funcionamiento de las unidades de compresor tornillo producidas por Mayekawa. Figura 2-5.

El controlador cuenta con las siguientes características estándar:

- 16 entradas digitales Sin señales de voltaje de contacto o señales de relé de estado sólido bajo voltaje (SSR).
- 16 entradas analógicas Entradas DC 24 de 4 a 20 mA, entradas DC de 1 a 5 V, o entradas de potenciómetro 1-k Ω

Interfaz usuario-controlador

- LCD color TFT, pantalla de 800X600 puntos (SVGA)
- Sistema de pantalla táctil (sistema de película resistiva analógica), resolución 1024x1024, figura 2-4.

Esta interfaz fácil de usar permite un manejo intuitivo mediante la adopción de la pantalla táctil.

Puertos de comunicación

Los siguientes puertos de comunicación se ofrecen como equipamiento estándar.

- Un Puerto de comunicación Ethernet (IEEE802.3u, 100base-TX) (Conector: RJ11)
- Un Puerto USB 2.0 (Conector Tipo-A))
- Un puerto de comunicación serial semi dúplex de 4 cables conforme RS-485

La función de comunicaciones en serie permite la supervisión remota de las condiciones de operación de compresores, controlar el encendido / parada, y cambiar diversos valores de los parámetros. El protocolo Modbus empleado es el protocolo estándar mundial para comunicaciones seriales en la industria y permite el uso de control de supervisión comercialmente disponible y de adquisición de datos (SCADA) de software, así como la conexión y comunicación con las unidades de comunicaciones PLC Modbus de diversos fabricantes.



Figura 2-4 Mypro Touch Interfaz hombre-maquina



Figura 2-5 Controlador MYPRO TOUCH

2.5 CIMON – PLC S modelo CM3 – SB32MDCF

El controlador CIMON (figura 2-5), nos permitirá la captura de los datos a través de sus puertos seriales, además de agregar un eventual control ante sucesos leídos. En las tablas 2-1 y tablas 2-2 se pueden observar las especificaciones de comunicación y las características generales del dispositivo.



Figura 2-6 CIMON-PLC S modelo CM3-SB32MDCF

Especificaciones de Comunicación Incorporada				
ITEMS		RS232C	RS485	Ethernet
Comm. Protocol	CIMON HMI Protocolo	CIMON HMI Protocol		
	CICON Protocolo	O	O	O
	User Protocolo	O	O	X
	MODBUS/RTU	Maestro/Esclavo	Maestro/Esclavo	Esclavo
	Data Bit	8 Bit		
	Bit Parada	1 o 2 Bit		
	Paridad	Even / Odd / None		
Sincronización		Asincrónico		
Velocidad de transmisión		1200 ~ 38400		
		10/100 Mbps		

Tabla 2-1 Especificaciones de comunicación

Especificaciones Generales			
ITEMS		Especificaciones	Obser.
		CM3-SB32MDCF	
Alimentación		DC 20 -234V	
Corriente consumo		330 mA	
Método de control programa		Ejecución cíclica del programa almacenado, Interrupción impulsada por tiempo	
Método de control I/O		Método indirecto, dirigido por instrucciones del programa	
Lenguaje Programación		LD (Diagrama Ladder), IL (Lista de instrucciones), SFC (Diagrama de funciones secuenciales)	
Método procesamiento datos		32 bit	
Instrucciones	Secuencia	55 Instrucciones	
	Aplicación	389 Instrucciones	
Velocidad Procesamiento (Secuencia)		300 ns/step	
Capacidad programa		10 K Step	
Máximos puntos I/O		DI 16 pts. / DO 16 pts. AI 2 pts. / AO 2 pts.	
Modo de operación		Remote Run, Remote Stop	
Método Back Up		dirección K por (Latch) parámetro	
Total programas		128	
Tipo de Programa	Scan	Scan, Subrutina, Inicialización de arranque en Cold/Hot, Interrupciones periódicas	
	Interrupciones periódicas	máximo 15 programas Scan (Periodo mínimo: 10 ms)	
	Especial	PID, HSC, Posicionamiento	
	Comunicación	Protocolo de usuario (Serial) Comm. Programa, MODBUS RTU Maestro, High-speed PLC Link	
	Etc.	SFC, FBD (Función de diagrama de bloques	
Función autodiagnóstico		Detección de delay del tiempo de escaneo, memoria, I/O, Power Supply	
Re inicio		Reinicio Cold- Hot	
Expansión Máxima		Sin Expansión	
Memoria de dispositivo	X	16 pts. (X00 - X0F)	Bit
	Y	16 pts. (Y10 - X1F)	Bit
	M	8192 pts. (M0000 - M511F)	Bit
	L	4096 pts. (L0000 - L255F)	Bit
	K	4096 pts. (K0000 - K255F)	Bit
	F	2048 pts. (F0000 - F127F)	Bit
	T	512 pts. (T0000 - T0511)	Word
	C	512 pts. (C0000 - C0511)	Word
	S	100 estados x 100 set (00.00 - 99.99)	
	D	10000 words (D0000 - D9999)	Word
	Z	1024 words (Call Stack: Z0000 - Z0063, Z1000 - Z1063)	Word
	Q	8192 pts. (Q0000 - Q511F)	Bit
	R	16 pts. (Index)	
Contador de alta velocidad		20 Kpps (en caso de 2Ph. 2Mult. Max 10Kpps)	

Tabla 2-2 Especificaciones generales

2.6 Raspberry Pi 4 Model B

Es un sistema embebido de bajo costo y formato compacto destinado al desarrollo en Reino Unido, para hacer más accesible la informática a todos los usuarios. La Raspberry Pi (figura 2-6) también se caracteriza por ser muy utilizada para desarrollar pequeños prototipos como servidores, Gateway para la formación sobre informática y electrónica en centro educacionales.

Todos los diseños de Raspberry Pi se basan en el hardware libre, habitualmente se utilizan también sistemas operativos libres basados en Linux. Para este sistema embebido se ha desarrollado Raspberry Pi OS (antes conocido como Raspbian) que es una versión personalizada de Debian. En la tabla 2-3 se señalan las características principales de este sistema embebido.



Figura 2-7 Raspberry Pi 4 Model B

RASPBERRY PI 4 MODEL B	
PROCESADOR	Broadcom BCM2711B0, quad-core Cortex-A72
FRECUENCIA DE RELOJ	1,5 GHz
GPU	VideoCore VI 500 MHz
MEMORIA	1/2/4 GB LPDDR4-3200
CONECTIVIDAD INALÁMBRICA	Wi-Fi 2,4GHz / 5GHz
	IEEE 802.11.b/g/n/ac
	Bluetooth 5.0, BLE
CONECTIVIDAD DE RED	Gigabit Ethernet
PUERTOS	GPIO 40 pines
	2 x Micro HDMI
	2 x USB 2.0
	2 x USB 3.0
	CSI (cámara Raspberry Pi)
	DSI (pantalla tácil)
	Toma auriculares / vídeo compuesto
	Micro SD
	USB-C (alimentación)
	Power-over-Ethernet (PoE)
FECHA DE LANZAMIENTO	24-06-2019

Tabla 2-3 Características Raspberry Pi 4 model B

3. CAPA DE COMUNICACIÓN

3.1 Comunicación En Serie RS-485

Es una de las especificaciones de la norma RS-449 publicada como estándar en 1983 en donde se especifica las características eléctricas de un circuito de comunicación digital de salida ya sea el estado “0” o “1” lógico, mediante una señal diferencial más una señal de alta impedancia o “tercer estado” con lo cual es posible conectar más de un emisor en la red, motivo por el cual, es muy útil en redes locales industriales a dos hilos.

Permite la conexión de hasta 32 emisores con 32 receptores en transmisión doble simultánea full dúplex capaz de enlazar procesadores de comunicación principal (maestro) con procesadores subordinado (esclavo) cuyo funcionamiento está definido por los mismos arreglos topológicos de las redes de datos.

En una comunicación de la norma RS-485 se tiene que el emisor opera el “1” lógico a un voltaje de -1.5 a -5 volts, el “0” lógico a la entrada del receptor en el rango de 0.2 a 12 volts y la máxima tensión aplicada a la línea de salida es de -7 a 12 volts.

3.2 Protocolo De Comunicación Modbus

Modbus surge en 1979 ante la necesidad de comunicar los equipos Modicon presentes cada vez más en la industria. Modbus pasa a ser un protocolo abierto, soportado por Modbus Organization. Es un protocolo de comunicación ubicado en la capa 1,2 y 7 del modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor para protocolo TCP/IP. Convertido en un protocolo de comunicación estándar de facto en la industria, es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales.

Existen varias versiones de Modbus para el puerto serie y los más comunes son Modbus RTU, Modbus ASCII, Modbus TCP y Modbus Plus.

Modbus se comunica a través de varios tipos de medios físicos tales como RS-232, RS-485, RS-422 y sobre Ethernet.

La interfaz Modbus original funcionaba con comunicación serial RS-232, pero la mayoría de la implementación posterior de Modbus utilizan RS-485, porque permitió distancias más largas, velocidades más altas y la posibilidad de múltiples dispositivos en una sola red multipunto. Comunicación Modbus maestro-esclavo a través de medio físico serie RS-458 mostrando conexiones de transmisión y recepción de dos hilos.

En interfaz simples como RS-485 o RS-232, los mensajes Modbus se envían de forma simple a través de la red y la red se dedicará sola a la comunicación Modbus. Sin embargo, si la red requiere múltiples dispositivos heterogéneos usando un sistema de red más versátil como TCP/IP sobre Ethernet, los mensajes Modbus están incrustados en paquetes Ethernet con el formato prescrito para esta interfaz física.

Entonces, en este caso Modbus y otros dispositivos mixtos pueden coexistir en la misma interfaz física al mismo tiempo.

La estructura principal del mensaje Modbus es de igual a igual. Modbus puede funcionar tanto en redes punto a punto como en redes multipunto.

Los dispositivos Modbus se comunican utilizando una técnica maestro-esclavo (cliente- servidor para Ethernet) en el que solo un dispositivo puede iniciar transacciones (llamadas consultas).

Los otros dispositivos responden suministrando los datos requeridos al maestro, o tomando la acción solicitada en la consulta.

Un esclavo es cualquier dispositivo periférico, como un transductor de E/S, válvula, unidad de red u otros tipos de dispositivos de medición que procesan información y envía su mensaje de respuesta al maestro usando Modbus.

3.3 Mensaje De Lectura Modbus RTU

Los maestros pueden dirigirse a los esclavos individualmente o iniciar un mensaje de difusión a todos los esclavos, y estos devuelven una respuesta a todas las consultas de mensajes dirigidos individualmente a ellos, pero no responden a los mensajes de difusión.

Los esclavos no inician mensajes por sí solos y solo responden a las consultas de mensajes transmitidos desde el maestro.

La consulta del maestro consistirá en una dirección esclava (o dirección de difusión), un código de función con un comando de lectura o escritura de datos al esclavo, junto con los datos del comando de escritura si el maestro inició un comando de escritura, y un campo de comprobación de errores.

La comprobación de errores es un valor que el maestro o esclavo crea al comienzo de la transmisión o respuesta y luego verificado cuando se recibe el mensaje para verificar que el contenido es correcto.

Byte 01: Identificador del dispositivo esclavo Modbus con el que desea comunicar

Byte 02: Comando de lectura que tiene que ejecutar el dispositivo esclavo

Byte 03 y 04: Dirección donde comienza la lectura.

Byte 05 y 06 Número de bytes a leer.

Byte 07 y 08: Código de redundancia cíclica.

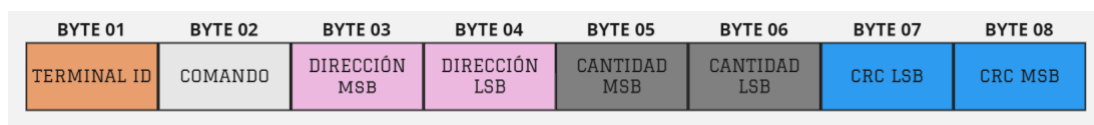


Figura 3-1 Diagrama lectura Modbus RTU

3.4 Mensaje De Respuesta Modbus RTU

La respuesta de un esclavo consta de campos de confirmación que recibió la solicitud, los datos que se devolverán y un error al verificar los datos. Si no se produce error, la respuesta del esclavo contiene los datos solicitados.

Si se produce un error en la consulta de mensaje recibido por el esclavo, o si el esclavo no puede realizar la acción solicitada, el esclavo devolverá un mensaje de excepción como respuesta. El campo de verificación de error del marco de mensajes del esclavo permite al maestro confirmar que el contenido del mensaje es válido.

Byte 01: Identificador del dispositivo esclavo Modbus.

Byte 02: Comando que ha ejecutado el dispositivo esclavo.

Byte 03: Cantidad de bytes de datos.

Byte 04 a n: Datos enviados

Byte 07 y 08: Código de redundancia cíclica.

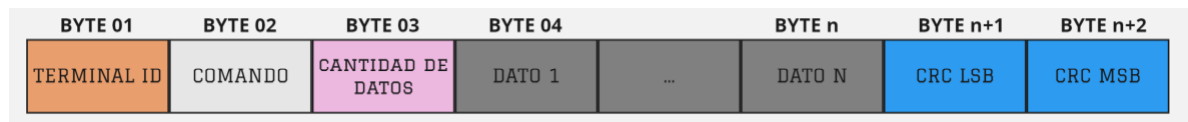


Figura 3-2 Diagrama respuesta Modbus RTU

3.5 Protocolo De Comunicación HTTP

El protocolo de comunicación HTTP (HyperText Transfer Protocol), es un conjunto de reglas creado en 1999 por el World Wide Web Consortium en conjunto con la colaboración con la Internet Engineering Task Force.

Está basado en un esquema petición- respuesta (request-response) entre un cliente y un servidor.

Es un protocolo sin estado, por lo que no guarda ninguna información sobre conexiones anteriores. Los mensajes HTTP son en texto plano, lo que lo hace más legible y fácil de depurar. Esto tiene el inconveniente de hacer los mensajes más largos.

Métodos para mensajes del tipo petición (request):

- GET: solicita una representación del recurso especificado. Las solicitudes que usan GET solo deben recuperar datos y no deben tener ningún otro efecto.
- POST: Envía datos para que sean procesados por el recurso identificado en la URL de la línea de petición.

Códigos para mensajes del tipo respuesta(responses):

- 1xx: Respuesta informativas. Indica que la petición ha sido recibida y se está procesando.
- 2xx: Respuesta correcta. Indica que la petición ha sido procesada correctamente.
- 3xx: Respuesta de redirección. Indica que el cliente necesita realizar más acciones para finalizar la petición.
- 4xx: Errores causados por el cliente. Indica que ha habido un error en el procesamiento de la petición a causa de que el cliente ha hecho algo mal.
- 5xx: Errores causados por el servidor. Indica que ha habido un error el procesamiento de la petición a causa de un fallo en el servidor.

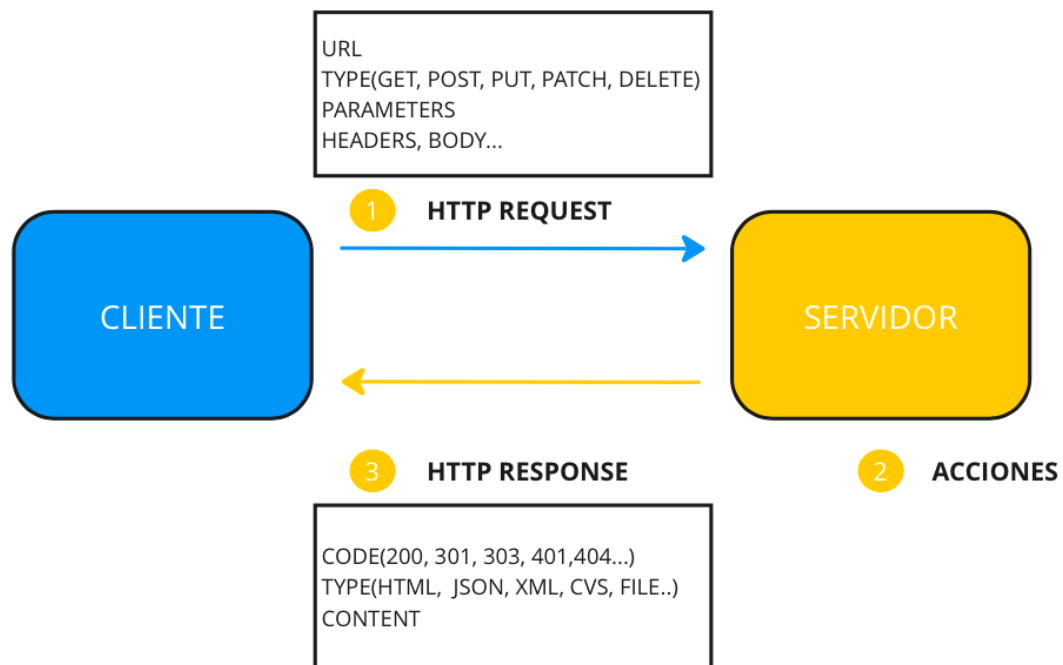


Figura 3-3 Diagrama protocolo HTTP

3.6 Protocolo De Comunicación MQTT

El protocolo MQTT (Message Queue Telemetry Trasport) se inventó en 1999 para su uso en la industria del petróleo y gas. Este protocolo está basado en un esquema de publicación y suscripción de los clientes a una cola de mensajes.

Esta solución consiste en un servidor central encargado de recibir los mensajes de todos los dispositivos emisores y distribuirlos a los receptores. De forma general a este servidor se le llama Router o Broker.

El servidor mantiene un registro de los dispositivos conectados, recibe los mensajes y los distribuye al resto de los dispositivos, filtrando los destinatarios según algún criterio como el tema o el contenido del mensaje. Los dispositivos en ningún momento ven o depende del resto de dispositivos. Por lo cual, esta infraestructura nos proporciona escalabilidad.

La metodología Publisher/Subscribers es un patrón de mensajería donde un agente, el Subscribers, informa al Broker que quiere un tipo de mensaje. El otro agente, el Publisher puede publicar el mensaje. El router distribuye los mensajes a los Subscribers, inmediatamente los envia a los clientes conectados. La figura 3-4 representa un esquema de funcionamiento del protocolo MQTT.

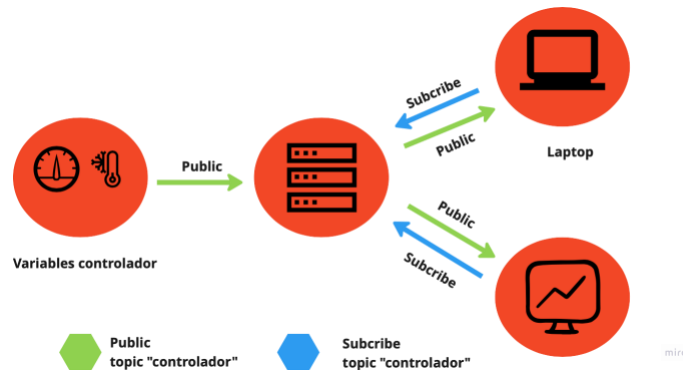


Figura 3-4 Esquema funcionamiento protocolo MQTT

3.7 Eclipse Mosquitto

Es un intermediario(broker) de mensajes de código abierto.

Implementa el protocolo MQTT en las versiones:

- 5.0
- 3.1.1
- 3.1

Mosquitto es liviano y es adecuado para su uso en todos los dispositivos, desde computadores de placa única (SBC) de bajo consumo hasta servidores completos.

3.8 Conceptos de Eclipse Mosquitto

Publicar/suscribir: en un sistema de publicación y suscripción, un dispositivo puede publicar un mensaje sobre un tema o puede suscribirse a un tema en particular para recibir mensajes.

- Mensajes: son la información que desea intercambiar entre sus dispositivos. Esto puede ser un comando o datos.
- Temas: son la forma en que se registra el interés por los mensajes entrantes o dónde se desea publicar el mensaje.
- Intermediario: es el principal responsable de recibir todos los mensajes, filtrarlos, decidir quién está interesado en ellos y luego publicar el mensaje para todos los clientes suscritos.

4. CAPA DE APLICACIONES

4.1 ¿Qué es un Base de Datos?

Según Oracle:

“Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS).

En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.”

4.2 Base de datos relacionales

Conjunto de información relacionada que se encuentra agrupada de forma sistemática para su posterior uso.

Corresponden al modelo más utilizado en la actualidad para dar solución a problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de relaciones que permiten la interacción entre un conjunto de registros (tuplas, listas, diccionarios), organizados como filas de una tabla y compuestos por una serie de campos o columnas. En este modelo, el lugar y la forma en que se almacenan los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. El lenguaje más

habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language (Lenguaje Estructurado de Consultas) un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante su diseño, una base de datos relacional pasa por un proceso denominado normalización de base de datos.

Bases de datos dinámicas, son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como inserción, eliminación y edición de datos; además de las operaciones fundamentales de consulta. Un ejemplo puede ser la base de datos de clientes de una entidad bancaria.

Tipos de dato en una base de datos SQL

En el proceso de modelado de datos una de las etapas importantes dentro de la definición de la información a almacenar en las tablas, es definir a cuál tipo de datos corresponde el valor que se ingresará posteriormente. Por ejemplo, si en una tabla que contendrá datos de personas se decide almacenar ID, nombre, apellido y fecha de nacimiento, la columna ID podría ser un número, nombre y apellido debiese ser de tipo texto (string) y la fecha de nacimiento debiese ser de tipo fecha. Todo eso para evitar problemas posteriores en el uso de la información almacenada. Según la documentación oficial, podríamos clasificar los tipos de datos en tres:

- Numéricos
- Fecha
- String

4.3 NoSQL

No SQL, not only SQL. Hace referencia a lo No relacional o No estructurado.

Según AWS:

“Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos

NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala.”

4.4 Tipos de NoSQL

- Clave-valor: altamente divisibles y permiten escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar. Los casos de uso como juegos, tecnología publicitaria e IoT se prestan particularmente bien con este modelo.
- Documentos: los datos se representan a menudo como un objeto o un documento de tipo JSON porque es un modelo de datos eficiente e intuitivo para los desarrolladores.
- Gráficos: facilitan la creación y la ejecución de aplicaciones que funcionan con conjuntos de datos altamente conectados. Los casos de uso típicos para una base de datos de gráficos incluyen redes sociales, motores de recomendaciones, detección de fraude y gráficos de conocimiento.

4.5 Bases de datos relacionales

Las bases de datos relacionales son un tipo de base de datos que almacena y organiza puntos de datos con relaciones definidas para un acceso rápido. Con una base de datos relacional, los datos se organizan en tablas que contienen información sobre cada entidad y representan categorías predefinidas a través de filas y columnas. La estructuración de los datos de esta manera hace que el acceso sea eficaz y flexible, motivo por el que las bases de datos relacionales son más comunes. Las bases de datos relacionales también se crean para comprender lenguaje de consulta estructurado (SQL), un lenguaje de programación estandarizado que se usa para almacenar, manipular y recuperar datos. En SQL, hay un lenguaje integrado para crear tablas denominado Lenguaje de definición de datos (DDL) y un lenguaje para manipular datos denominado Lenguaje de manipulación de datos (DML).

¿Qué significa relacional? Relacional significa indicar o analizar la relación. En el contexto de las bases de datos, la forma en que definimos relacional se aplica principalmente a los propios datos. Los conjuntos de datos relacionales tienen relaciones predefinidas entre ellos. Por ejemplo, una base de datos que incluye información de cliente de una empresa también puede incluir datos de transacción individuales adjuntos a cada cuenta. Las bases de datos relacionales centran la atención en la relación entre los elementos de datos almacenados.

4.6 MySQL

MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basado en código abierto. Desarrollado originalmente por MySQL AB, fue adquirida por Sun Microsystems en 2008 y esta su vez comprada por Oracle Corporation en 2010, la cual ya era dueña de un motor propio InnoDB para MySQL.

MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle.

4.7 Series De Tiempo

Los datos de series de tiempo son una colección de observaciones (comportamiento) para un solo sujeto (entidad) en diferentes intervalos de tiempo. Por ejemplo: temperatura máxima, humedad y viento (los tres comportamientos) en una estación meteorológica (entidad única) recopilados el primer día de cada mes (múltiples intervalos de tiempo)

En la figura 4-1 se representa los dos tipos de datos de series de tiempo datos que se pueden clasificar en tipos:

- Mediciones recopiladas a intervalos de tiempo regulares (métricas)
- Mediciones recopiladas en intervalos de tiempo irregulares (eventos)

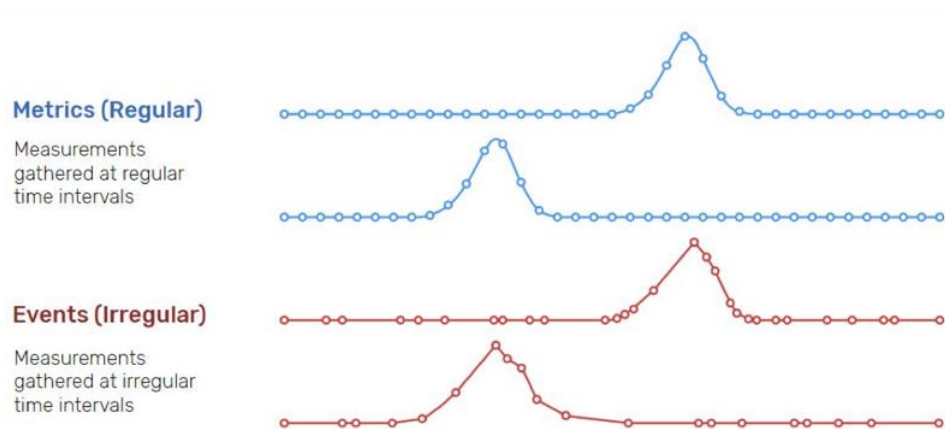


Figura 4-1 Clasificación series de tiempo

Un punto de medida se denomina Data Point y está compuesto por:

- Time: almacena marcas de tiempo. Muestra la fecha y la hora, en formato RFC3339 UTC, asociada con datos particulares.
- Los Fields se componen de Field Key y Field Value.
- Field Key: String que indica a que corresponde cada valor obtenido.
- Field Value: son los datos; pueden ser string, float, enteros o booleanos. Un field value siempre está asociado con una marca de tiempo.
- Los Tags se componen de Tag Key y Tag Value.
- Los tags son opcionales. No se necesita tener tags en la estructura de datos, pero generalmente es una buena idea usarlas porque, a diferencia de los fields, los tags están indexadas.

4.8 InfluxDB

InfluxDB es una plataforma de series de tiempo de código abierto. Esto incluye API para almacenar y consultar datos, procesarlos en segundo plano para ETL o para fines de monitoreo y alerta, paneles de usuario, y visualización y exploración de los datos.

Características:

- Almacén de datos personalizado de alto rendimiento escrito específicamente para datos de series temporales.
- API HTTP de escritura y consulta simples y de alto rendimiento.
- Los complementos son compatibles con otros protocolos de ingesta de datos como Graphite, Collectd y OpenTSDB.
- Lenguaje de consulta expresivo similar a SQL adaptado para consultar fácilmente datos agregados.
- Las etiquetas permiten indexar series para consultas rápidas y eficientes.
- Las políticas de retención caducan automáticamente de manera eficiente los datos obsoletos.
- Las consultas continuas calculan automáticamente los datos agregados para que las consultas frecuentes sean más eficientes.

Recolección de datos y visualización

InfluxDB es un sistema basado en push. Requiere una aplicación para insertar activamente los datos en InfluxDB.

InfluxDB cuenta con visualizaciones y paneles básicos. Para una mayor eficiencia deben implementarse junto con soluciones dedicadas de visualización y tablero, como Grafana y Chronograf.

4.8.1 Bucket

Un Bucket es una ubicación con nombre donde se almacenan los datos de series temporales. Todos los Buckets tienen una política de retención, una duración del tiempo que cada punto de datos persiste.

4.9 ¿Qué es un Servicio Web?

Según IBM:

“Son aplicaciones modulares autocontenidas que pueden describir, publicar, localizar e invocar a través de una red.

...ayudan a mejorar la flexibilidad de los procesos de negocio mediante la integración con aplicaciones que, de lo contrario, no se comunican.”

4.10 API (Application Programming Interface)

- Interfaz de Programación de Aplicaciones.
- Mecanismo que permite a dos componentes de software comunicarse entre sí.
- Se utiliza un conjunto de definiciones y protocolos comunes.
- La arquitectura de las API suele explicarse en términos de cliente y servidor.

4.11 API REST

REST significa Transferencia de Estado Representacional.

- REST define un conjunto de funciones para acceder a los datos del servidor:
GET, POST, PUT, DELETE.
- Clientes y los servidores intercambian datos mediante HTTP distintos formatos: HTML, XML, texto sin formato y JSON.
- Principal característica es la ausencia de estado, los servidores no guardan datos del cliente entre las solicitudes.

4.12 Flask

Flask es un framework para Python basado en Werkzeug que permite desarrollar aplicaciones web básicas, con Flask tenemos las herramientas necesarias para crear una aplicación web funcional, pero si se necesita en algún momento una nueva funcionalidad hay un conjunto de extensiones (plugins) que se pueden instalar con Flask y le van dotando de funcionalidad.

El patrón MVC es una manera o una forma de trabajar que permite diferenciar y separar lo que es el modelo de datos (los datos que van a tener la App que normalmente están guardados en BD), la vista (página HTML) y el controlador (donde se gestiona las peticiones de la app web).

5. CAPA VISUALIZACIÓN

5.1 Grafana

Grafana es una solución de código abierto para ejecutar análisis de datos, sacando métricas que tienen sentido a la enorme cantidad de datos y para monitorear nuestras aplicaciones con la ayuda de paneles personalizables. También nos permite escribir plugins desde cero para integrarnos con varias fuentes de datos diferentes.

Grafana se conecta con todas las fuentes de datos posibles, comúnmente conocidas como bases de datos como Graphite, Prometheus, InfluxDB, Elasticsearch, MySQL, PostgreSQL.

La herramienta permite estudiar, analizar y monitorear datos durante un período de tiempo, técnicamente llamado análisis de series temporales.

Permite rastrear el comportamiento del usuario, el comportamiento de la aplicación, la frecuencia de errores que aparecen en la producción o en un entorno pre-productivo, el tipo de errores que aparecen y los escenarios contextuales al proporcionar datos relativos.

Una gran ventaja del proyecto es que puede ser implementado in situ por organizaciones que no quieren que sus datos se transmitan a una nube de proveedores por seguridad y otras razones.

6. IMPLEMENTACIÓN

La implementación se lleva a cabo utilizando la arquitectura simplificada, para el presente diseño serán descritas desde la capa inferior hasta la superior.

6.1 CAPA DE DISPOSITIVOS

En la figura 6-1 se puede observar una descripción de la conexión física de los dispositivos tanto como el controlador IR33+ y MYPRO TOUCH con el PLC CIMON, además de la conexión entre el PLC y la Raspberry Pi.

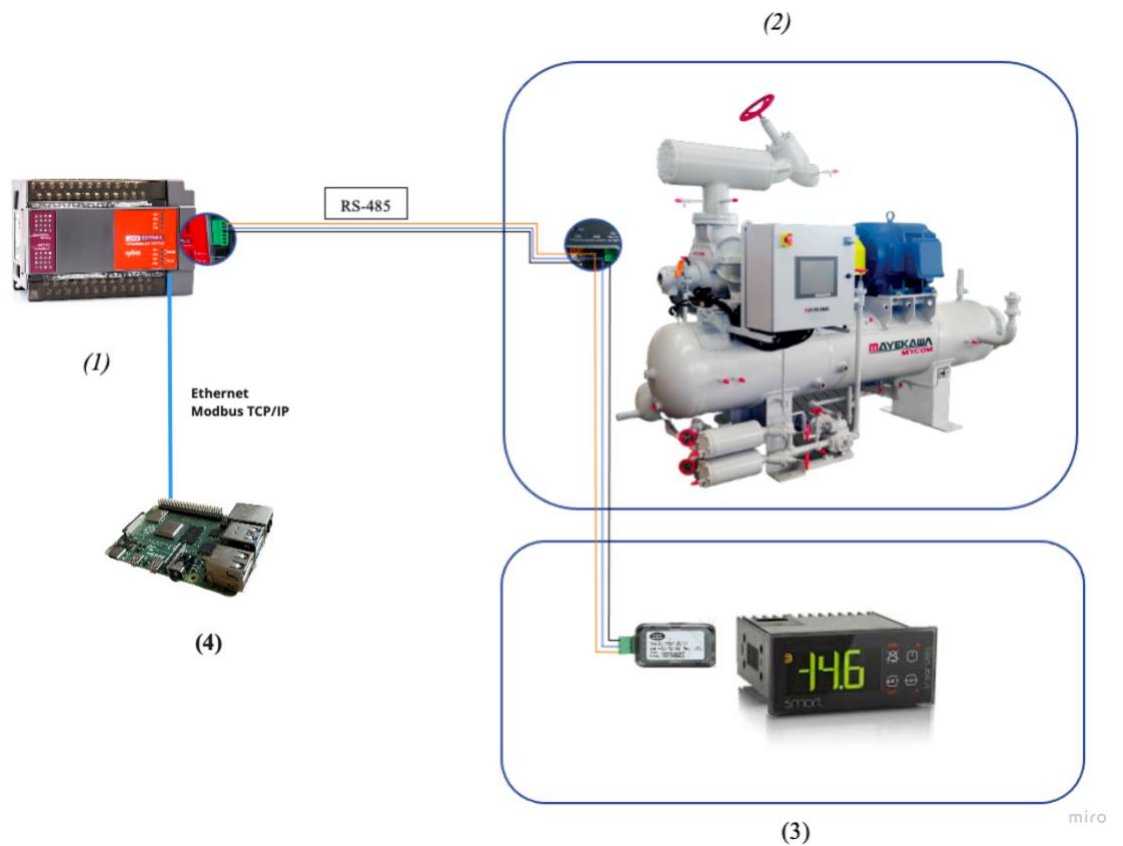


Figura 6-1 Capa de dispositivos; (1)PLC CIMON CM3-SB32MDCF, (2)MYPRO TOUCH, (3)Controlador ir33+,(4)Raspberry Pi 4 model B

6.2 CAPA DE ADQUISICIÓN DE DATOS

La captura de datos de los dispositivos de campos se realiza con el controlador programable de la marca CIMON modelo CM3-SB32MDCF a través de sus puertos seriales RS-485. Utilizando el software de programación Cimon, previa instalación de todos los drivers requeridos por sistema.

Conexión física de los dispositivos de campo

La conexión física de los dispositivos de campo se realiza utilizando el protocolo Modbus, utilizando como transporte el protocolo RS-485, lo que permite la escalabilidad del diseño.

6.3 SOFTWARE DE PROGRAMACIÓN PLC CICON

Un software de programación que permite a los usuarios crear programas de PLC. Funciona con todas las series de PLC CIMON para comodidad del usuario. Un simulador incorporado ayuda a probar proyectos sin conectarse a un PLC físico.

Requerimiento de sistema

Sistema operativo: Windows 8/7/XP/2000/98SE

Procesador (CPU): Pentium 133 MHz o superior

RAM: 64 MB de RAM mínimo

Espacio libre en disco duro: 13 MB mínimo

Puerto de comunicación: se recomienda puerto USB, puerto Ethernet o puerto serie.

6.4 DIRECCIONES MODBUS DISPOSITIVOS DE CAMPO

Compresor MYCOM			
Atributo	Dirección	Descripción	nota
read/write	000513	Comunicación con compresor Entrada de partida	
read/write	000514	Comunicación con compresor entrada de parada	
read/write	000576	Comunicación entrada para limpiar alarma	
Read Only	400321	Presión de Succión	SP
Read Only	400322	Presión de Descarga	DP
Read Only	400323	Presión de suministro aceite	OP
Read Only	400324	Presión después del filtro	AFP
Read Only	400325	Temperatura de succión	ST
Read Only	400326	Temperatura de descarga	DT
Read Only	400327	Temperatura suministro de aceite	OT
Read Only	400394	Capacidad	CAP
Read Only	400385	diferencial de aceite (DP-OP)	dOP
Read Only	400332	Corriente motor eléctrico de compresor	MA
Read Only	400386	Presión diferencia de filtro	dFP

Tabla 6-4 Registros Controlador Mycom

Controlador IR33+			
Atributo	Dirección	Descripción	nota
Read/Write	000059	Comando ON/OFF controlador	on_off
Read Only	000001	Estado de relé compresor	rele_frio
Read Only	000002	Estado de relé deshielo	rele_deshielo
Read Only	000003	Estado de relé ventiladores	rele_ventiladores
Read Only	400003	Sonda 1	S1
Read Only	400004	Sonda 2	S2
Read/Write	000034	Comando solicitud de deshielo	on_off_deshielo
Read Only	00010/11	Alarma sonda 1/2/ averiada	alar_s1 alar_s2
Read/Write	400016	Punto de ajuste	set_point

Tabla 6-5 Lectura registros controlador IR33+

6.5 PROGRAMACIÓN EN SOFTWARE CICON

Para la programación del PLC se utiliza el software CICON, el cual permite la programación de los distintos programas como: la captura de datos, programación de entradas y salidas físicas del PLC, programación de parámetros de comunicación, configuración de las entradas y salidas analógicas del PLC, etc.

Programación para la captura de datos de dispositivos periféricos.

Para la comunicación del PLC con el software, se utiliza comunicación serial USB. Los parámetros de comunicación definidos para el protocolo Modbus RTU se pueden observar en la figura 6-3 la configuración del Channel 1, con los parámetros de comunicación.

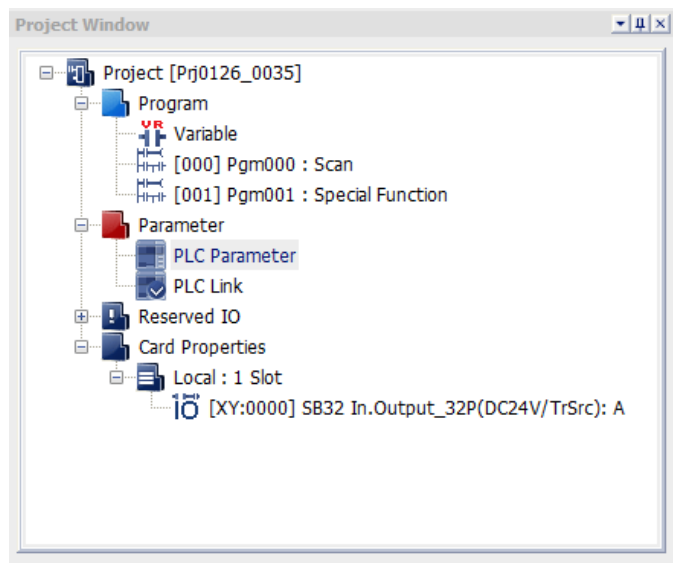


Figura 6-2 Ventana desplegable parámetros

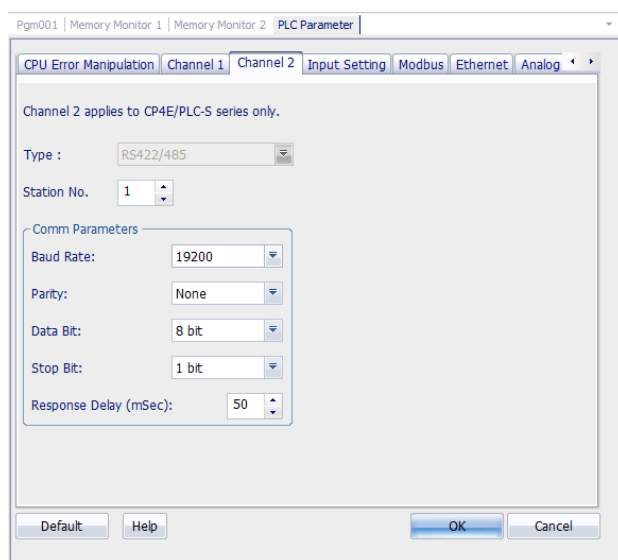


Figura 6-3 Parámetros protocolo Modbus RTU

6.5.1 Configuración protocolo Modbus TCP-IP

En la figura 6-4 se observan los parámetros de comunicación Modbus TCP-IP con direcciones IP privadas.



Figura 6-4 Parámetros protocolo Modbus TCP/IP

En la figura 6-5 se observa la configuración de Modbus Esclavo, donde se configura el tipo de dato para cada tipo de registro.

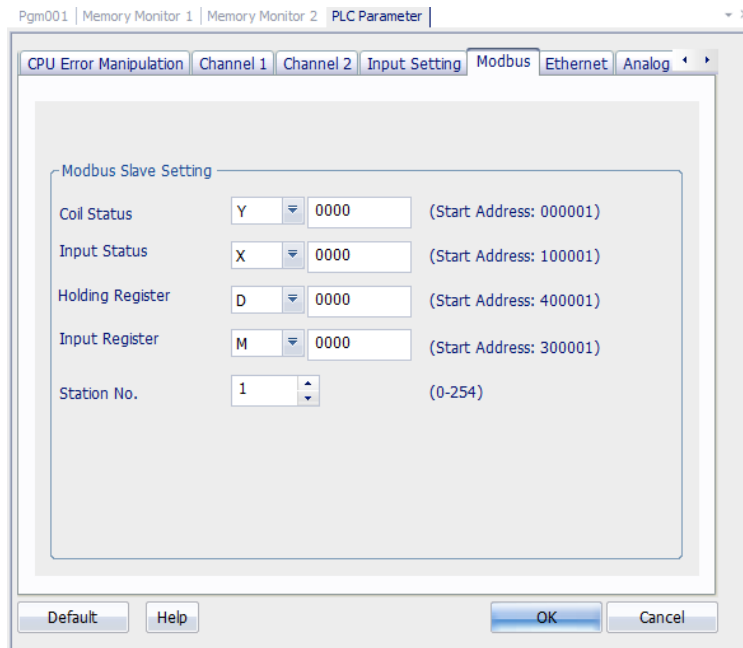


Figura 6-5 Parámetros protocolo Modbus RTU Slave

6.5.2 Configuración programa Modbus RTU Master.

Al seleccionar la opción New Program se despliega una ventana emergente, representada en la figura 6-6, en Configuración de comunicación se selecciona la opción MODBUS/RTU Master.

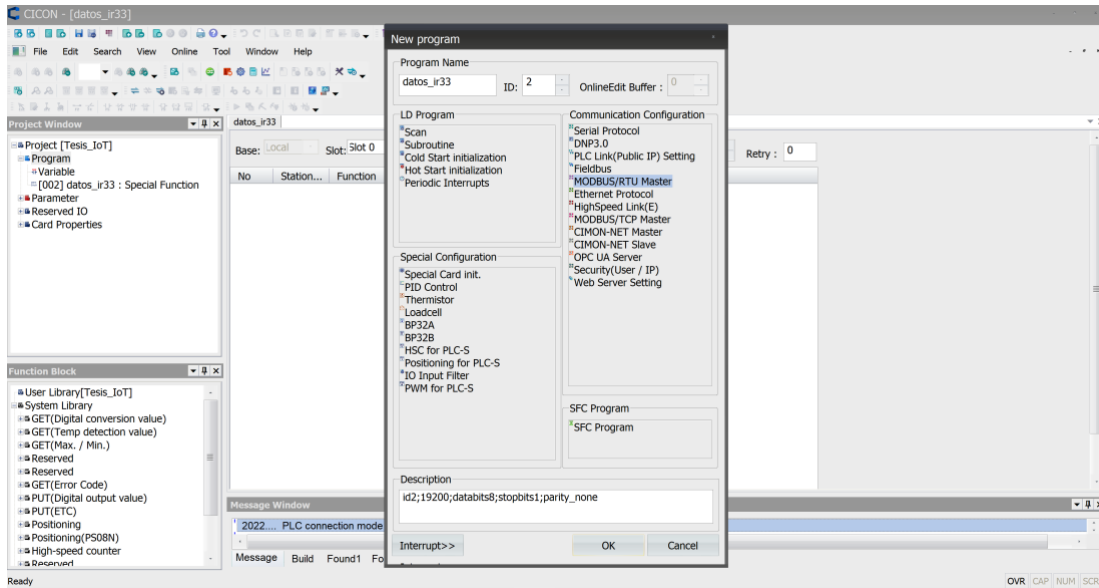


Figura 6-6 Creación programa Modbus RTU Máster

Al seleccionar la configuración de comunicación Modbus RTU Master se despliega la ventana de la figura 6-7, en la cual se selecciona el botón Add para agregar un nuevo bloque.

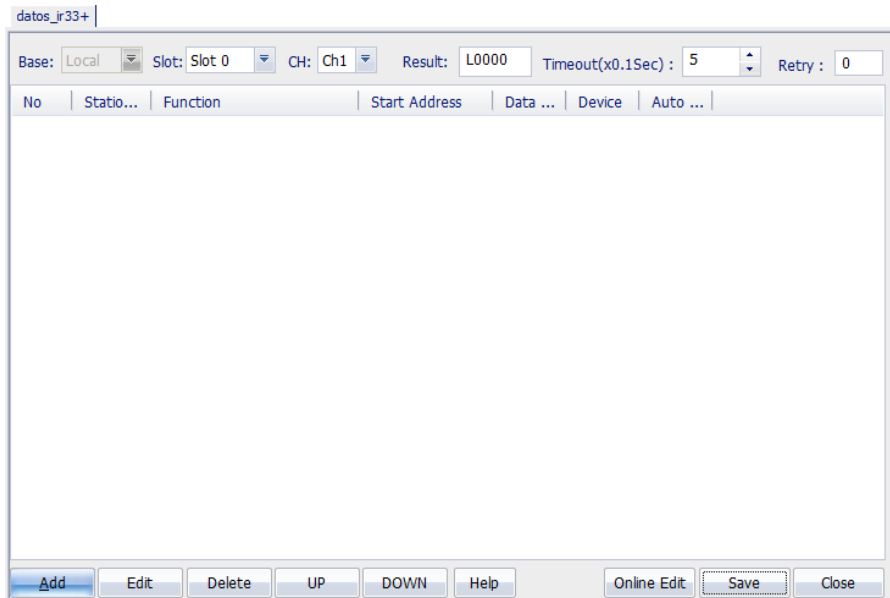


Figura 6-7 Ventana configuración lectura de registros

Se desplegará un bloque donde se configurará la estación de destino, el tipo de función, en la figura 6-8 se puede observar la configuración para leer un estado de la bobina, comenzando en la dirección 00001, la cantidad de datos a leer y el registro donde quedará en el dispositivo, en este caso Y0001.

En la figura 6-8 se puede observar también la configuración para la lectura de Holding Register, la cual comenzará en la dirección 40003, el tipo de dato será INT16, lectura de un dato que se almacenará en el registro D04 del dispositivo.

Figura 6-8 Configuración de lecturas Holding Register y Coil.

En la figura 6-9 se observa el listado de registros configurado perteneciente al controlador IR33+

Pgm001

Base: Local Slot: Slot 0 Ch: Ch2 Result: L0000 Timeout(x0.1Sec): 5 Retry: 0

No	Stato...	Function	Start Address	Data ...	Device	Auto ...
0	2 (H02)	01 Read Coil Status (0x)	1 (H0001)	1	Y0001	Yes
1	2 (H02)	01 Read Coil Status (0x)	3 (H0003)	1	Y0002	Yes
2	2 (H02)	01 Read Coil Status (0x)	11 (H000B)	1	Y0003	Yes
3	2 (H02)	01 Read Coil Status (0x)	59 (H003B)	1	Y0004	Yes
4	2 (H02)	01 Read Coil Status (0x)	2 (H0002)	1	Y0005	Yes
5	2 (H02)	01 Read Coil Status (0x)	10 (H000A)	1	Y0006	Yes
6	2 (H02)	01 Read Coil Status (0x)	16 (H0010)	1	Y0007	Yes
7	2 (H02)	01 Read Coil Status (0x)	34 (H0022)	1	Y0006	Yes
8	2 (H02)	03 Read Holding Registers ...	3 (H0003)	1	D00004	Yes
9	2 (H02)	03 Read Holding Registers ...	16 (H0010)	1	D00001	Yes
10	2 (H02)	03 Read Holding Registers ...	5 (H0005)	1	D00003	Yes
11	2 (H02)	03 Read Holding Registers ...	4 (H0004)	1	D00002	Yes

Add Edit Delete UP DOWN Help Online Edit Save Close

Figura 6-9 Registros a leer controlador IR33+

En la pestaña Online (figura 6-10) se selecciona la opción Connect para establecer comunicación con el PLC. Al establecer comunicación se Descarga el programa desde el PC hacia el PLC (figura 6-11).

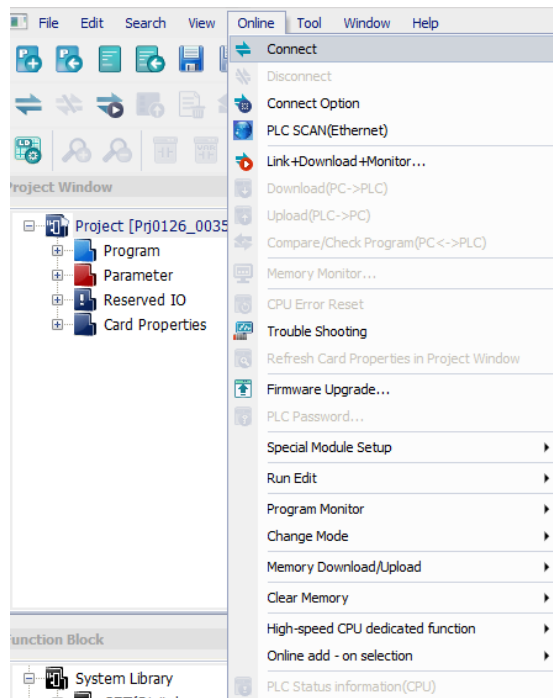


Figura 6-10 Conexión Programa CICON

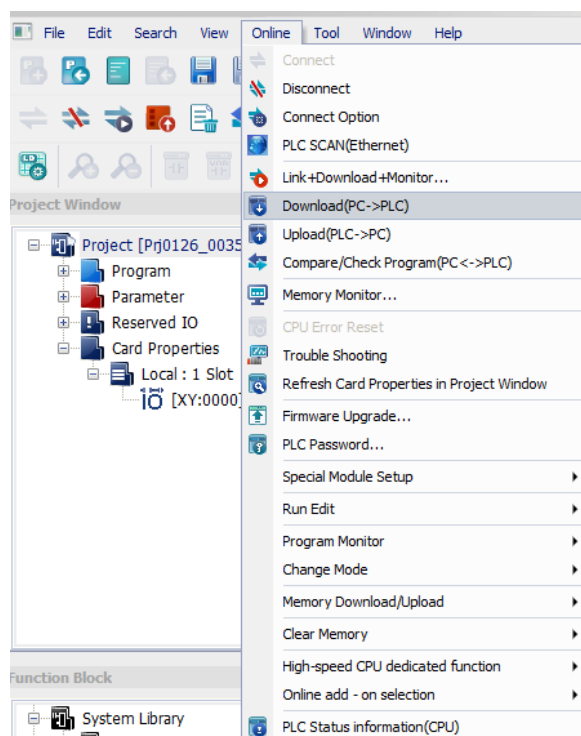


Figura 6-11 Descarga programa hacia PLC

Una vez descargado el programa en PLC, se despliega la pestaña View y se selecciona la opción Device Monitoring. En Device Monitor (figura 6-12) se pueden observar las variables del controlador IR33+ solicitadas por el PLC.

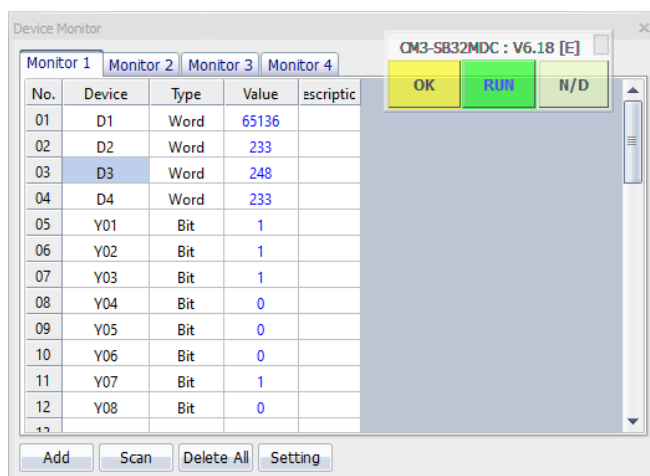


Figura 6-12 Ventana desplegable para la visualización de datos Monitor de dispositivos

7. ADQUERIR LOS DATOS DEL PLC CON PYTHON

7.1 Instalación librería easymodbus

Para la adquisición de datos desde el PLC se utiliza la librería easymodbus la cual se instala por consola, utilizando los comandos `pip install EasyModbus`

7.2 Importación de librería y conexión con PLC esclavo

```
import easymodbus.modbusClient
modbusclient=easymodbus.modbusClient.ModbusClient ('192.168.1.10',502)
modbusclient.connect()
```

7.3 Lectura de registros PLC esclavo correspondiente a controlador IR33+.

Se genera un bucle con una pausa de diez segundos, mientras la comunicación sea verdadera se leerán cuatro holding register desde la dirección 40001 hasta la dirección 40004, que se guardaran en una lista llamada `HoldingRegs`. Los registros Coil se leerán desde la dirección 00001 hasta la dirección 00008 y se guardarán en una lista llamada `Coil`.

```
while True:
    HoldingRegs=modbusclient.read_holding_registers(1,4)
    Coil=modbusclient.read_coils(1,6)
```

7.4 Interpretación de registros leídos.

Al ejecutar las funciones `print(HoldingRegs)` y `print(Coil)`.

Imprime las siguientes listas :

```
[65526, 255, 248, 255]
```

```
[True, True, True, False, False, False, True, False]
```

El dato de la posición 0 de la lista de HoldingRegs corresponde al punto de ajuste del controlador el cual tiene el ajuste de -1.0° Celsius, el valor que podemos observar corresponde a un unsigned int 16 bit (65536) que entrega valores desde 0 a 65536.

7.5 Funciones para escalamiento de datos de lista HoldingRegs.

Escalamiento de dato en posición [0] de la lista HoldingRegs.

Para determinar la función de escalamiento se determina el rango de ajuste que en este caso corresponde desde -50° C. hasta 60° C.

Se declara una variable llamada set_point la que es de tipo float.

```
set_point=float
```

Se genera una condición if, si el registro de la posición [0] es mayor 65036 que corresponde a -50 , el cual es el límite inferior de punto de ajuste, se sobrescribe la variable set_point aplicando la función correspondiente.

Si el registro de la posición [0] de la lista está entre los rangos > 500 y < 65036 se ejecuta la función asociada a la sentencia elif.

Si el registro [0] de la lista es mayor a 0 se ejecuta la sentencia else.

```
if HoldingRegs[0]>=65036:
    set_point=float((HoldingRegs[0]-65536)/10)
elif HoldingRegs[0]>500 and HoldingRegs[0]<65136:
    print("Punto de ajuste fuera de rango")
else:
    set_point=float((HoldingRegs[0]/10))
```

Escalamiento de datos en posición [1] de la lista HoldingRegs.

Para determinar la función de escalamiento se determina el rango de medición del sensor que en este caso corresponde desde -40° C. hasta 150° C.

Se declara una variable llamada sonda_1 la que es de tipo float.

```
sonda_1=float
```

Se genera una condición if, si el registro de la lista HoldingRegs [1] es mayor 65136 que corresponde a -40, el cual es el límite inferior de lectura, se sobrescribe la variable sonda_1 aplicando la función correspondiente.

Si el registro de la posición[1] de la lista está entre los rangos > 1500 y < 65136 se ejecuta la función asociada a la sentencia elif.

Si el registro [1] de la lista es mayor a 0 se ejecuta la sentencia else.

```
if HoldingRegs[1]>=65136:
    sonda_1=float((HoldingRegs[1]-65536)/10)
elif HoldingRegs[1]>1500 and HoldingRegs[1]<65136:
    print("Medición fuera de rango")
else:
    sonda_1=float((HoldingRegs[1]/10))
```

Escalamiento de dato en posición [2] de la lista HoldingRegs.

Para determinar la función de escalamiento se determina el rango de medición del sensor que en este caso corresponde desde -40° C. hasta 150° C.

Se declara una variable llamada sonda_2 la que es de tipo float.

```
sonda_2=float
```

Se genera una condición if, si el registro de la posición [2] es mayor 65136 que corresponde a -40, el cual es el límite inferior de lectura, se sobrescribe la variable sonda_2 aplicando la función correspondiente.

Si el registro de la posición[2] de la lista esta entre los rangos > 1500 y < 65136 se ejecuta la función asociada a la sentencia elif.

Si el registro [2] de la lista es mayor a 0 se ejecuta la sentencia else.

```

if HoldingRegs[2]>=65036:
    sonda_2=float((HoldingRegs[2]-65536)/10)
elif HoldingRegs[2]>1500 and HoldingRegs[2]<65036:
    print("Medición sonda 2 fuera de rango")
else:
    sonda_2=float((HoldingRegs[2]/10))

```

Escalamiento de dato en posición [3] de la lista HoldingRegs.

Para determinar la función de escalamiento se determina el rango de medición del sensor que en este caso corresponde desde -40° C. hasta 150° C.

Se declara una variable llamada sonda_virtual la que es de tipo float.

```
sonda_virtual=float
```

Se genera una condición if, si el registro de la posición [3] es mayor 65036 que corresponde a -40, el cual es el límite inferior de lectura, se sobrescribe la variable sonda_virtual aplicando la función correspondiente.

Si el registro de la posición[3] de la lista está entre los rangos > 1500 y < 65136 se ejecuta la función asociada a la sentencia elif.

Si el registro [3] de la lista es mayor a 0 se ejecuta la sentencia else.

```

if HoldingRegs[3]>=65036:
    sonda_virtual=float((HoldingRegs[3]-65536)/10)
elif HoldingRegs[3]>1500 and HoldingRegs[3]<65036:
    print("Medición sonda virtual fuera de rango")
else:
    sonda_virtual=float((HoldingRegs[3]/10))

```

7.6 Generar variables a partir de la lista Coil.

A partir de la lista generada por la lectura de las bobinas se desglosan los elementos en las siguientes variables:

```

rele_compr=Coil[0]
rele_vent=Coil[1]

```

```

alarm_sonda2=Coil[2]
comando_on_off= Coil[3]
deshielo=Coil[4]
comando_deshielo=Coil[5]
alarma_Hi=Coil[6]
alarma_sonda1=[7]

```

Se crea una lista con todas las variables generadas en el escalamiento de la lista HoldingRegs y la lista Coil.

```

datos_controlador_ir33=[set_point,   sonda_1,   sonda_2,   sonda_virtual,
rele_compr,   rele_vent,   alarm_sonda2,   comando_on_off,   deshielo,
comando_deshielo]

```

7.7 Generar un elemento marca de tiempo

Para generar una marca de tiempo se debe importar la librería datetime, la cual devuelve una marca de fecha y hora. El formato se determina con la función `.strftime('%Y-%m-%d %H:%M:%S')`

```

from datetime import datetime
current_date = datetime.now()
formatted_date = current_date.strftime('%Y-%m-%d %H:%M:%S')

```

Se agrega el elemento `formatted_date` al final de la lista `datos_controlador_ir33`, utilizando la función `.append`

```

datos_controlador_ir33.append(formatted_date)

```

Se ejecuta la función `print(datos_controlador_ir33)`

Imprime una lista con los datos obtenidos

```

[-1.0, 25.5, 24.8, 25.5, True, True, True, False, False, False, True,
False, '2022-11-25 23:45:38']

```

7.8 Crear un diccionario en Python con los datos obtenidos

```

diccionario_ir33={}
diccionario_ir33['setpoint']=controlador_ir33[0]
diccionario_ir33['sonda_1']=controlador_ir33[1]
diccionario_ir33['sonda_2']=controlador_ir33[2]
diccionario_ir33['sonda_virtual']=controlador_ir33[3]
diccionario_ir33['rele_compr']=controlador_ir33[4]
diccionario_ir33['rele_vent']=controlador_ir33[5]
diccionario_ir33['alarma_s2']=controlador_ir33[6]
diccionario_ir33['comando_off_on']=controlador_ir33[7]
diccionario_ir33['deshielo']=controlador_ir33[8]
diccionario_ir33['comando_deshielo']=controlador_ir33[9]
diccionario_ir33['alarma_HI']=controlador_ir33[10]
diccionario_ir33['alarma_sonda1']=controlador_ir33[11]
diccionario_ir33['timestam']=controlador_ir33[12]

```

8. PUBLICACIÓN DATOS ADQUIRIDOS A MQTT BROKER MOSQUITTO

Paho MQTT librería requerida para la conexión a BROKER Eclipse Mosquitto. Librería JSON principalmente usada para el análisis de sintaxis de archivos o strings en formato JSON. También analiza el JSON en un diccionario o en una lista en Python y viceversa, es decir, convierte los diccionarios y las listas de Python en cadenas JSON.

```

from paho.mqtt import client as mqtt_client
import json

```

Se define y se declara el BROKER y el puerto en el cual se trabaja, además se define el tópico en cual se realizará la publicación.

```

broker = 'test.mosquitto.org'
port = 1883
topic = "sadema/camara_1/controlador_1"

```

8.1 Crear una función para conexión de la aplicación con el servidor MQTT.

```
def connect_mqtt():
    def on_connect(client, userdata, flags, return_code):
        if return_code == 0:
            print("Conectado a MQTT broker! :)")
        else:
            print("Error, codigo %d\n", return_code)
    client = mqtt_client.Client()
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

Definición de función que realiza la publicación al MQTT Broker, la cual mientras la conexión al Broker sea verdadera el cliente publicará los datos.

El método `json.dumps()` permite serializar los objetos de Python como un string. Lo que se debe considerar es que las claves en las pares clave/valor en JSON son siempre del tipo string. Cuando un diccionario se convierte en JSON, todas las claves del diccionario se convierten en cadenas.

Al publicar el broker devuelve una lista con el resultado de la publicación, si el resultado de la publicación es igual a cero, el mensaje fue enviado con éxito de lo contrario hubo un error al enviar.

```
def publish(client):
    while True:
        time.sleep(10)
        current_date = datetime.now()
        formatted_date = current_date.strftime('%Y-%m-%d %H:%M:%S')

        controlador_ir33.append(formatted_date)
        diccionario_ir33['setpoint'] = controlador_ir33[0]
        diccionario_ir33['sonda_1'] = controlador_ir33[1]
```

```

diccionario_ir33['sonda_2'] = controlador_ir33[2]
diccionario_ir33['sonda_virtual'] = controlador_ir33[3]
diccionario_ir33['rele_compr'] = controlador_ir33[4]
diccionario_ir33['rele_vent'] = controlador_ir33[5]
diccionario_ir33['alarm_s2'] = controlador_ir33[6]
diccionario_ir33['comando_off_on'] = controlador_ir33[7]
diccionario_ir33['deshielo'] = controlador_ir33[8]
diccionario_ir33['comando_deshielo'] = controlador_ir33[9]
diccionario_ir33['alarma_HI'] = controlador_ir33[10]
diccionario_ir33['alarma_sonda1'] = controlador_ir33[11]
diccionario_ir33['timestam'] = controlador_ir33[12]
msg = json.dumps(diccionario_ir33 )
result = client.publish(topic, msg)
status = result[0]
if status == 0:
    print('mensaje enviado')
else:
    print('error al enviar')

```

9. DESARROLLO DE LA CAPA NIEBLA

9.1 Creación de base de datos MySQL

El diseño de la base de datos en la capa niebla se utiliza usando la herramienta visual de diseño MySQL Workbench .

Se crea una base de datos llamada Sistema de refrigeración, en la cual se crea una tabla llamada variables_ir33_1 con las columnas señaladas en la figura 9-1.

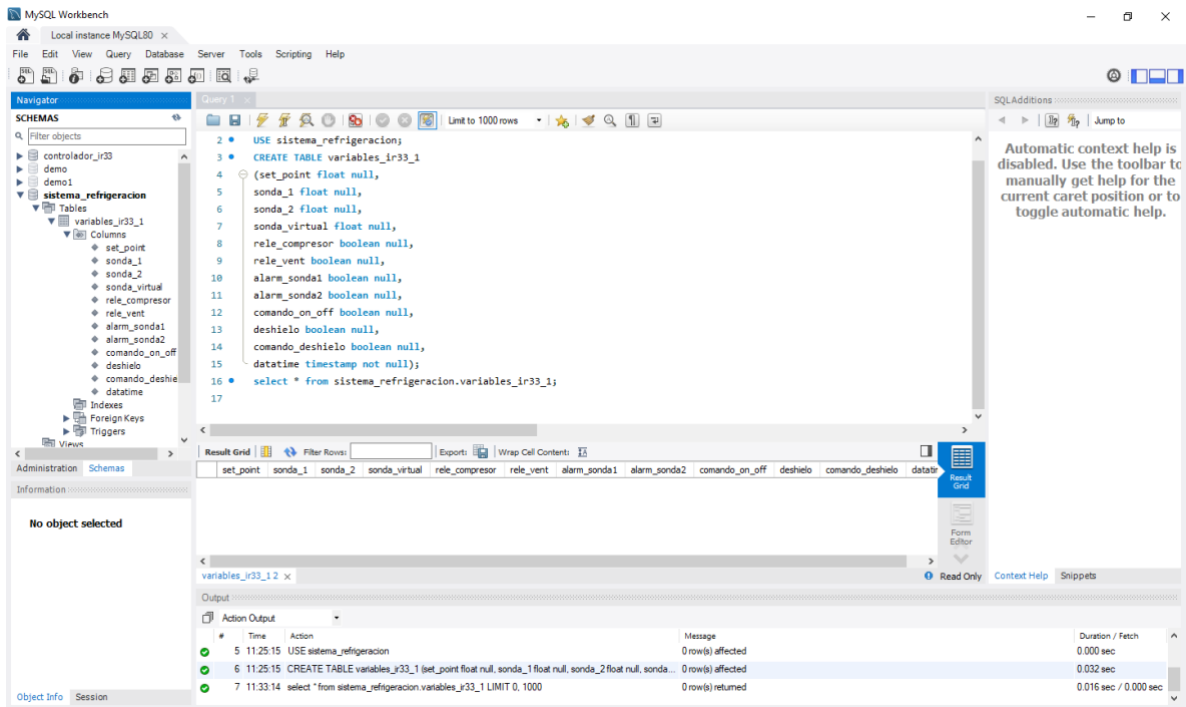


Figura 9-1 Workbench sistema de gestión de BBDD.

9.2 Conexión a base de datos MySQL.

Para conexión del programa con el servidor de MySQL se debe importar la librería `mysql.connector`, para luego crear un elemento `mysql_db` con la función `connect()`

```
import mysql.connector
```

Se define el elemento `mysql_db` al cual se agrega el módulo `connect` y como argumento se ingresa los parámetros para conectar al servidor MySQL.

```
mysql_db = mysql.connector.connect(
    host='localhost',
    user='root',
    password='*****',
    database="sistema_refrigeracion")
```

9.3 Insertar registros en base de dato MySQL

Para insertar datos desde la captura de datos a la base de datos se define una función llamada `InsertQuery`.

Se crea el elemento `mysql_cursor` el cual está compuesto de los elementos de conexión más el módulo `cursor` de la librería `mysql.connector`.

```
Def InsertQuery(datos_controlador_ir33)
mysql_cursor = mysql_db.cursor()
```

Se crea un elemento con la sentencia a ejecutar en el servidor de MySQL

```
insert_sql = "INSERT INTO variables_ir33_1 (set_point, sonda_1, sonda_2,
sonda_virtual, rele_compresor, rele_vent, alarm_sonda2, comando_on_off,
deshielo, comando_deshielo, alarma_HI, alarm_sonda1, datetime) VALUES (%s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
```

Ejecuta la función del cursor con el módulo `.execute` con la sentencia MySQL y `datos_controlador` como argumentos.

```
mysql_cursor.execute(insert_sql, datos_controlador_ir33)
mysql_db.commit()
```

9.4 Visualización de datos en herramienta MySQL Workbench

En la herramienta MySQL Workbench se ejecuta la sentencia `SELECT * FROM` para seleccionar la tabla `variables_ir33` de la base de datos `sistema_refrigeracion`. En la figura 9-2 se pueden observar los datos ingresados a través de la función `InsertQuery(datos_controlador_ir33)`.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'controlador_ir33', 'demo', 'demo1', and 'sistema_refrigeracion'. The 'sistema_refrigeracion' schema is expanded, showing tables like 'variables_ir33_1'. The main editor window contains a SQL query: `SELECT * FROM sistema_refrigeracion.variables_ir33_1;`. The 'Result Grid' pane shows the query results, which are 10 rows of data. The columns are: `la_2`, `sonda_virtual`, `rele_compresor`, `rele_vent`, `alarm_sonda2`, `comando_on_off`, `deshielo`, `comando_deshielo`, `alarma_HI`, `alarm_sonda1`, and `datetime`. The 'Action Output' pane at the bottom shows the execution details: '9 12:44:55 SELECT * FROM sistema_refrigeracion.variables_ir33_1 LIMIT 0, 1000' and '10 row(s) returned'.

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

la_2	sonda_virtual	rele_compresor	rele_vent	alarm_sonda2	comando_on_off	deshielo	comando_deshielo	alarma_HI	alarm_sonda1	datetime
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:38:55
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:04
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:09
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:14
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:19
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:24
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:29
20.8	1	1	1	0	0	0	1	0	0	2022-11-28 12:39:35
21	1	1	1	0	0	0	1	0	0	2022-11-28 12:42:54
21.2	1	1	1	0	0	0	1	0	0	2022-11-28 12:50:00

Table: variables_ir33_1

Columns:

- set_point float
- sonda_1 float
- sonda_2 float
- sonda_virtual float
- rele_compresor tinyint(1)
- rele_vent tinyint(1)
- alarm_sonda2 tinyint(1)
- comando_on_off tinyint(1)
- deshielo tinyint(1)
- comando_deshielo tinyint(1)
- alarma_HI tinyint(1)

Action Output

Time Action Message Duration / Fetch

9 12:44:55 SELECT * FROM sistema_refrigeracion.variables_ir33_1 LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec

Figura 9-2 Workbench con solicitud a tabla de base de datos.

10.CAPA APLICACIONES

En Python se crea una aplicación tipo microservicio ETL que extraiga los mensajes desde el servicio MQTT, transforme los mensajes recibidos y los almacene como datos en InfluxDB. Crear función la captura de datos del BROKER MQTT. En la figura 10-1 se señala un esquema de la arquitectura de referencia.

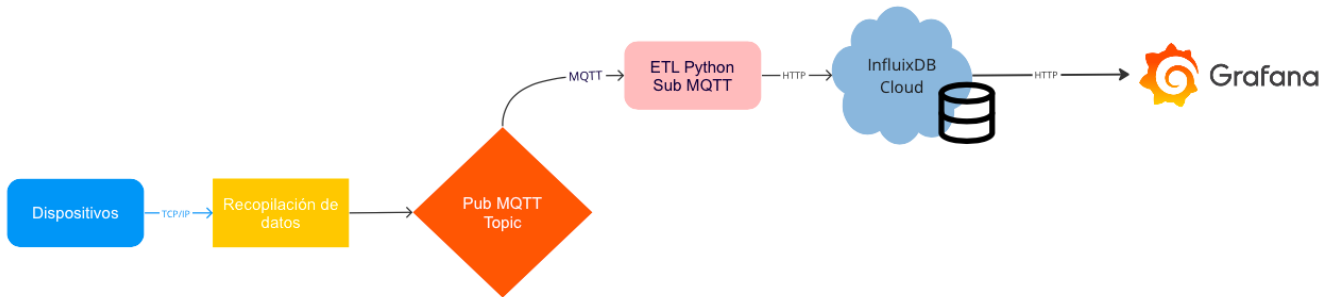


Figura 10-1 Esquema de arquitectura de referencia

Se define la función para la suscripción al servidor MQTT, las cuales capturan el mensaje en formato JSON publicado con el tópico "sadema/camara_1/controlador_1" y lo guarda en `data`.

```

def subscribe(client):
    def on_message(client, userdata, msg):
        data = msg.payload
        print(json.loads(data))

    client.subscribe(topic)
    client.on_message = on_message
  
```

10.1 Configuración InfluxDB.

Una de las formas de configurar y utilizar InfluxDB es a través de InfluxDB Cloud, para lo cual se dirige al siguiente link <https://www.influxdata.com/> con las credenciales de Google se genera cuenta.

Para escribir y consultar los datos usando el lenguaje de programación a elección, en este caso Python, se realiza la configuración correspondiente a la fuente de datos. Figura 10-2

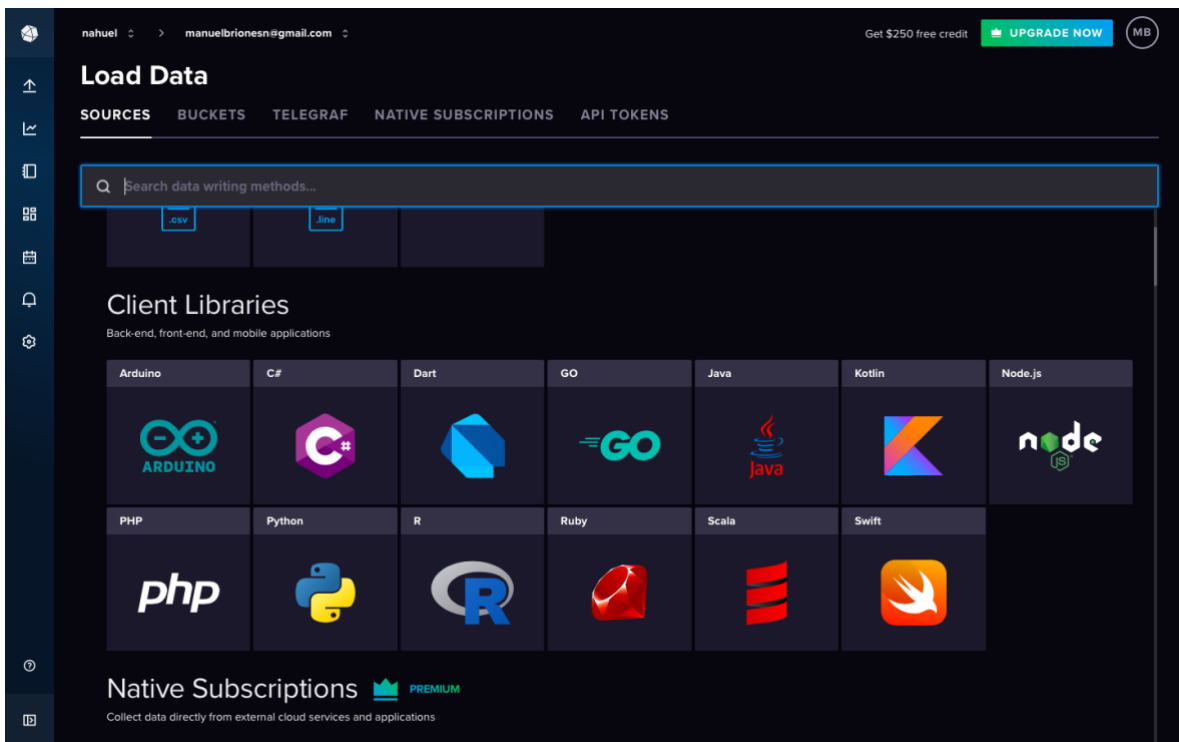


Figura 10-2 Fuentes de datos InfluxDB

10.2 TOKEN

InfluxDB Cloud utiliza tokens para autenticar el acceso a la API. Al configurar se crea un token de acceso total para este proceso de configuración.

export

```
INFLUXDB_TOKEN=X57EeJkYc19Sx8izur8GzvE8ICf2X_CbTPJEaat3Ig4RW1ptzh11CICjt1
G6lJ05F1Cx16vCB4ISfvS7bWNvjw==.
```

Inicializar el token, la información de la organización y la URL del servidor que se necesitan para configurar la conexión inicial a InfluxDB. La conexión del cliente se establece con la inicialización de `InfluxDBClient`.

```
import influxdb_client, os, time
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

token = os.environ.get("INFLUXDB_TOKEN")
org = "manuelbrionesn@gmail.com"
url = "https://us-east-1-1.aws.cloud2.influxdata.com"

client = influxdb_client.InfluxDBClient(url=url, token=token, org=org)
write_api = client.write_api(write_options=SYNCHRONOUS)
```

10.3 BUCKETS

Para comenzar a escribir datos, se necesita un lugar para almacenar los datos de series temporales, este contenedor se denomina `bucket="data_controlador_ir33"`.

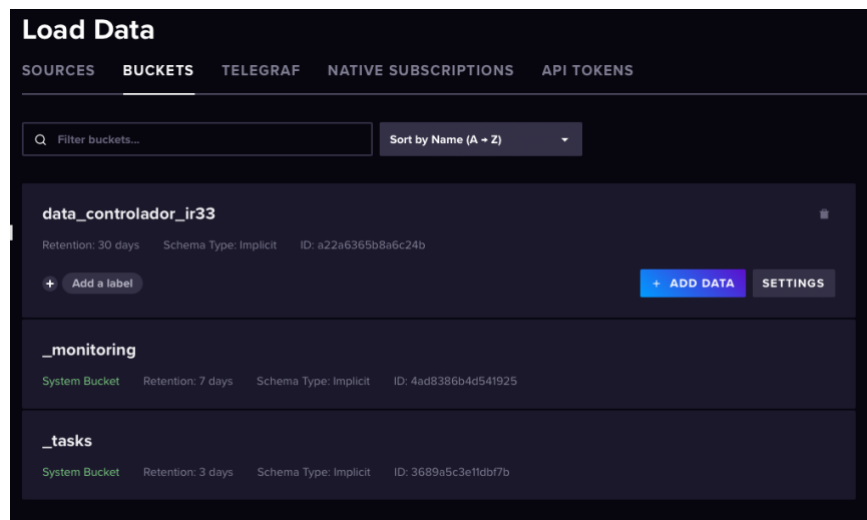


Figura 10-3 Bucket para insertar datos

10.4 Escritura de datos en Influx

Desde la suscripción al tópico se define una aplicación con la función de conexión al BROKER MQTT, se define la función para determinar el mensaje que se recibe del tópico, el cual es un mensaje en formato JSON, se estructura el mensaje JSON a diccionario con la función `.loads`. Se define el punto con los campos los cuales toman el valor de las claves.

```
def subscribe(client):
    def on_message(client, userdata, msg):
        data = msg.payload
        data_dicc=json.loads(data)
        point_test = Point("controlador_1")\
            .field("setpoint",data_dicc["setpoint"])\
            .field("sonda_1",data_dicc["sonda_1"])\
            .field("sonda_2",data_dicc["sonda_2"])\
            .field("sonda_virtual", data_dicc["sonda_virtual"]) \
            .field("rele_compr", data_dicc["rele_compr"])\
            .field("rele_vent",data_dicc["rele_vent"])\
            .field("alarma_s2",data_dicc["alarm_s2"])\
            .field("comando_off_on",data_dicc["comando_off_on"])\
            .field("deshielo",data_dicc["deshielo"])\
            .field("comando_deshielo", data_dicc["comando_deshielo"])\
            .field("alarma_HI",data_dicc["alarma_HI"])\
            .field("alarma_sonda1",data_dicc["alarma_sonda1"])
        write_api.write(bucket=bucket, org= org, record= point_test)

    client.subscribe(topic)
    client.on_message = on_message
```

Al consultar los datos que se insertan en la base de datos. Se utiliza el lenguaje de secuencias de comandos Flux para consultar datos.

10.5 Visualización de datos en InfluxDB

InfluxDB cuenta con su propia interfaz gráfica para el monitoreo de las variables. Además permite la gestión de la base de datos, realización de “queries”, que permiten la integración con Grafana. En la figura 10-4 se observa la interfaz de InfluxDB con los puntos y los campos de datos. En la figura 10-5 se observa un gráfico con los datos guardados en un intervalo de un minuto. En la figura 10-6 se observa un gráfico con los datos capturados durante una hora, además del “query” para la consulta de los datos.

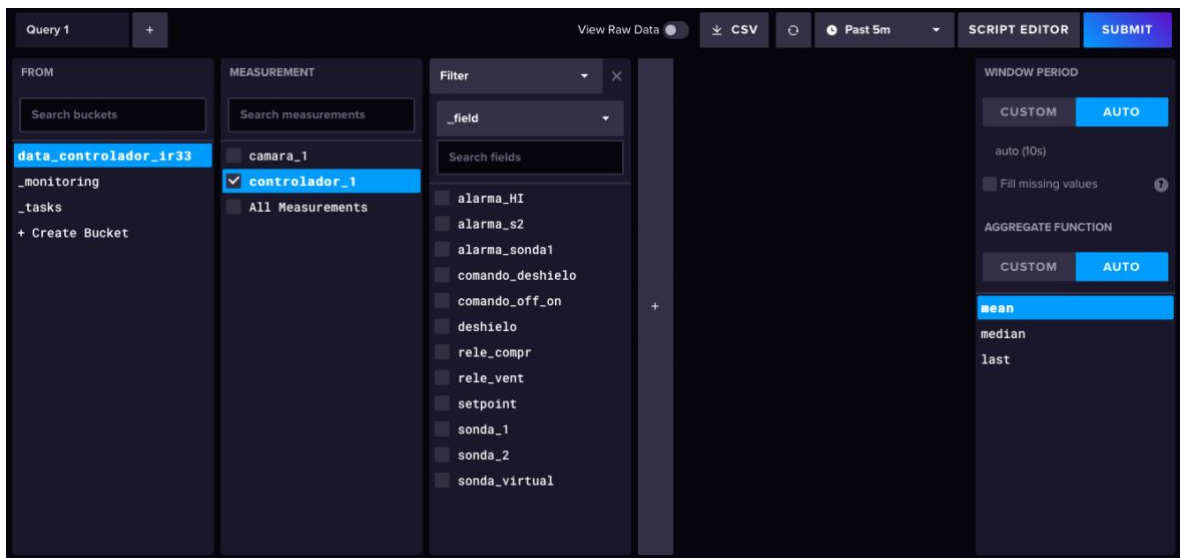


Figura 10-4 Interfaz InfluxDB

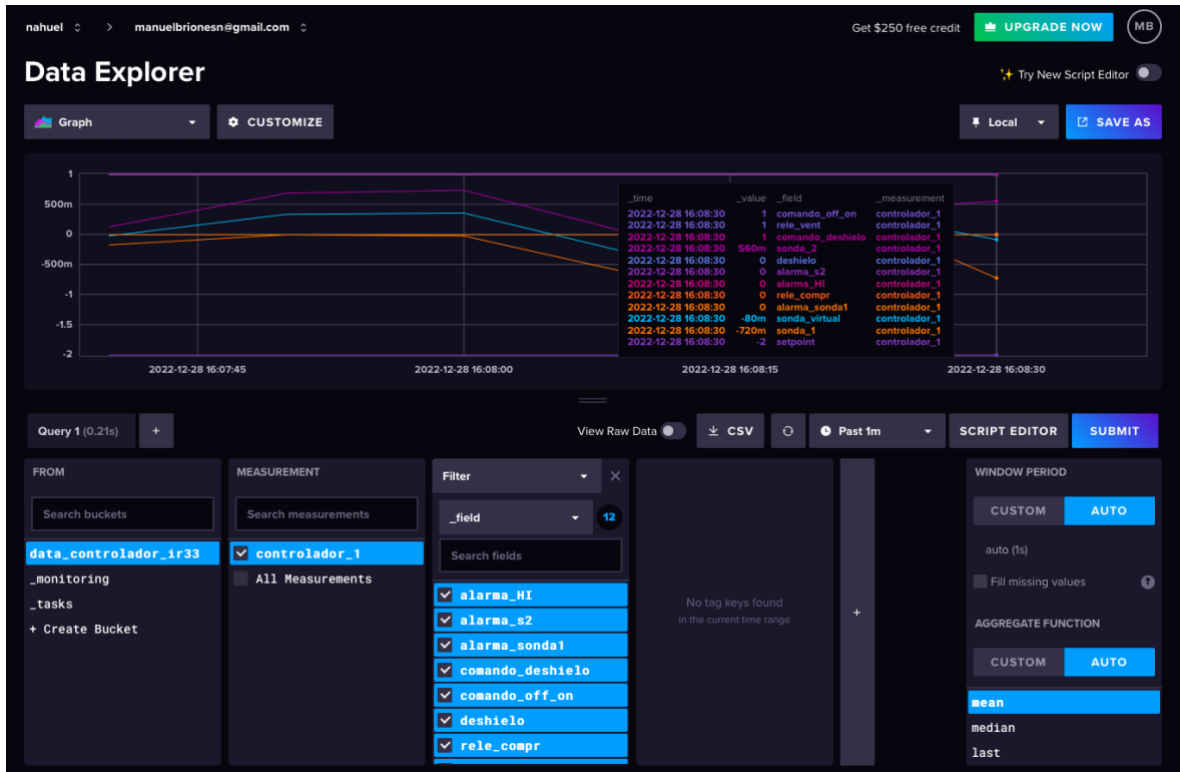


Figura 10-5 Visualización de datos en InfluxDB

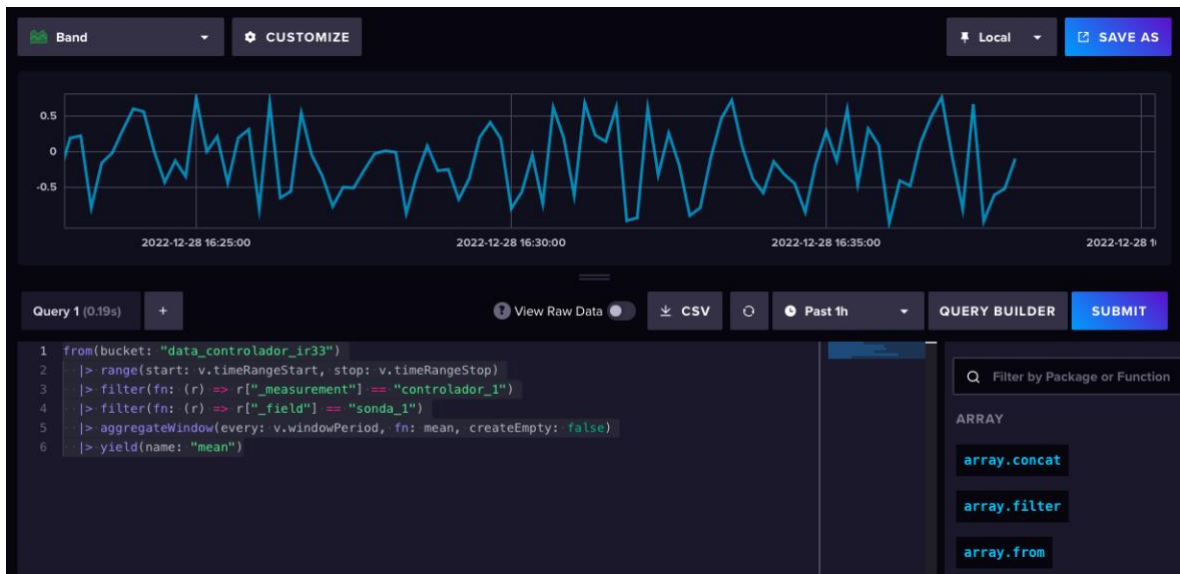


Figura 10-6 Visualización sonda 1, query solicitud de visualización de datos.

10.6 Transferencia de datos del InfluxDB a Grafana Cloud

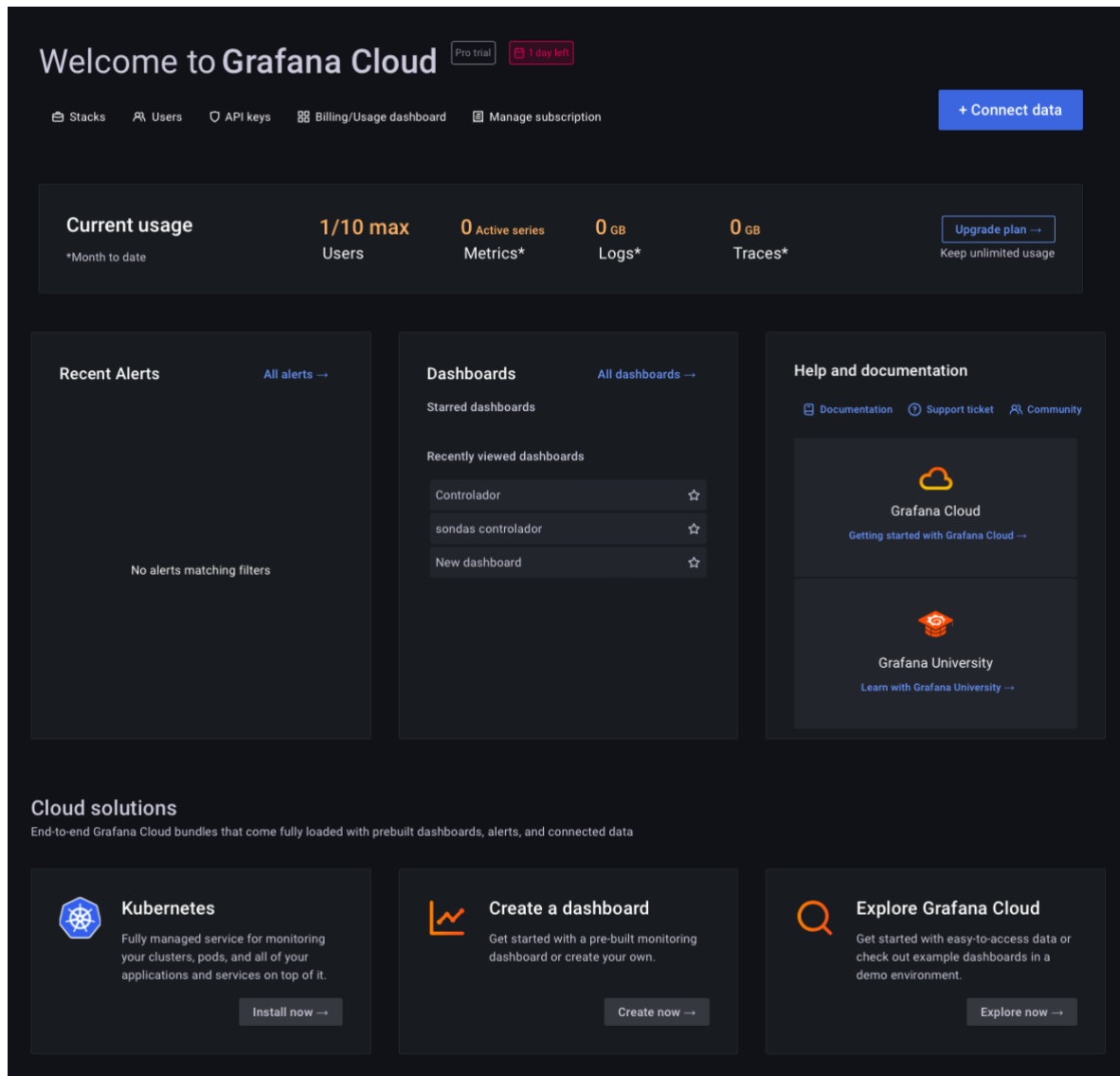


Figura 10-7 Interfaz de inicio Grafana Cloud

10.7 Configuración de fuentes de datos

Se despliega en la barra de opciones la opción Configuration (figura 10-8), para determinar la fuente de datos que se integrara con Grafana, se selecciona la opción de InfluxDB (figura 10-9).

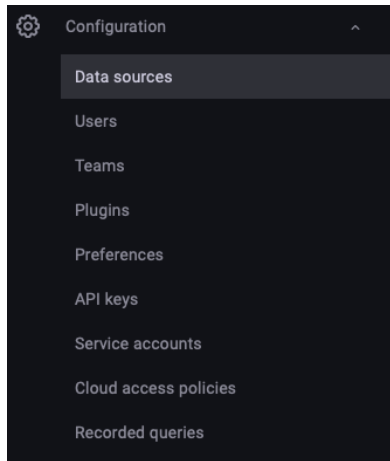


Figura 10-8 Barra de opciones de configuración.

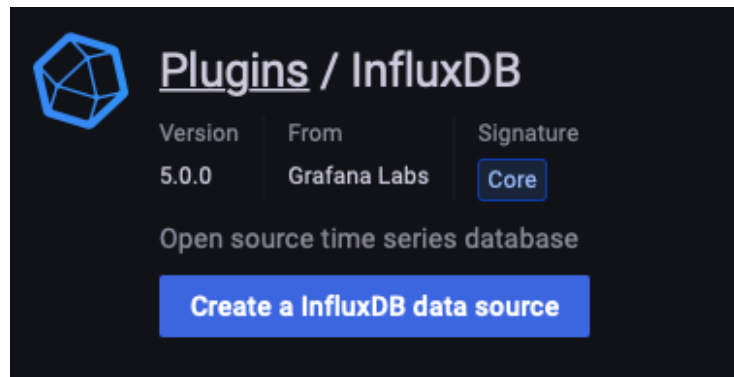


Figura 10-9 Plugins InfluxDB para la integración con Grafana.

10.8 Configuración fuentes de datos InfluxDB

Se determina el nombre de la fuente de datos, el lenguaje de consulta, para este caso lenguaje Flux (El soporte para consulta Flux en Grafana está actualmente en versión beta).

Se configura el Protocolo de comunicación HTTP con la URL con las credenciales para ingresar a InfluxDB(figura 10-11). Además se configura las credenciales de la base de datos como la Organización, Token y Bucket por defecto, mismas credenciales que se utilizaron para la conexión con el microservicio de Python, también se configura el tiempo mínimo de intervalo y la cantidad máxima de series (figura 10-10).

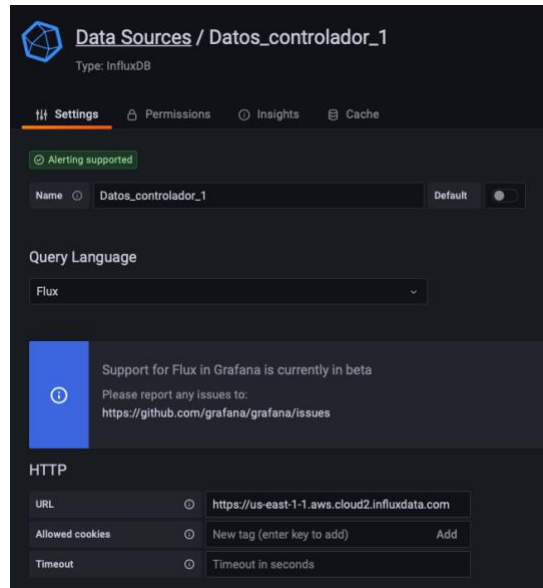


Figura 10-10 Configuración de lenguaje de consulta y comunicación.

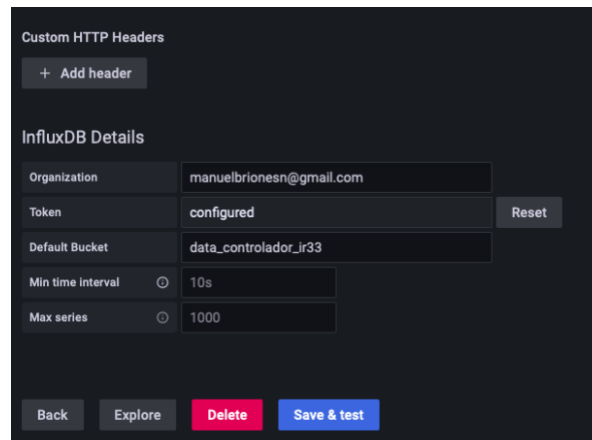


Figura 10-11 Configuración credenciales Influx.

10.9 Creación de Dashboard Grafana Cloud

En la barra de opción lateral se despliegan las opciones de Dashboard, las cuales van desde la creación de nuevos Dashboard, importación, creación de carpetas, búsquedas de Dashboard creados, se selecciona New Dashboard (figura 10-12). Se despliega una ventana con las opciones de agregar nuevo panel, agregar una nueva fila y agregar paneles desde librerías (figura 10-13).

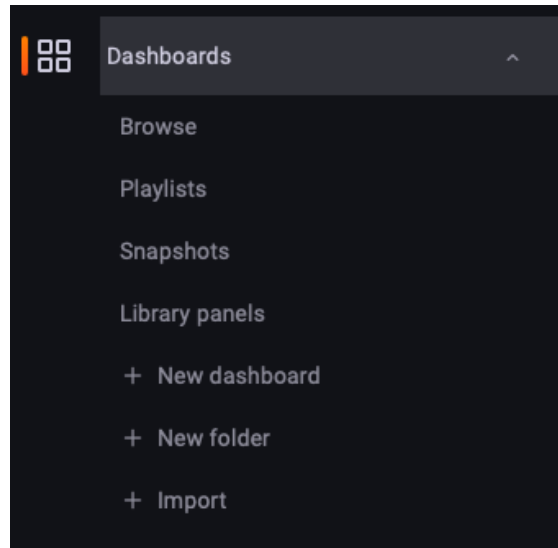


Figura 10-12 Barra de opciones Dashboard.

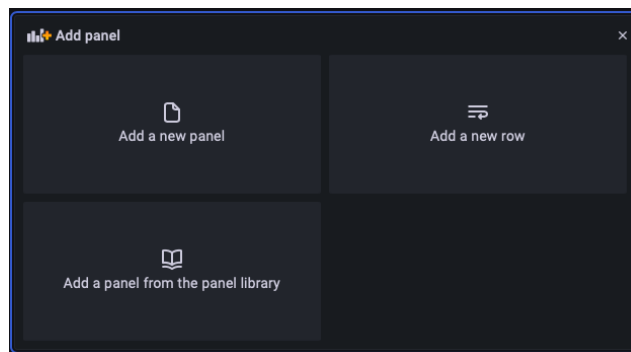


Figura 10-13 Opciones agregar panel.

10.10 Creación y configuración de panel de serie de tiempo en Grafana

Grafana entrega una variada opción de creación de paneles (figura 10-14) desde series de tiempo, indicadores, tablas, gráficos de torta, histogramas, historial de estados, etc.

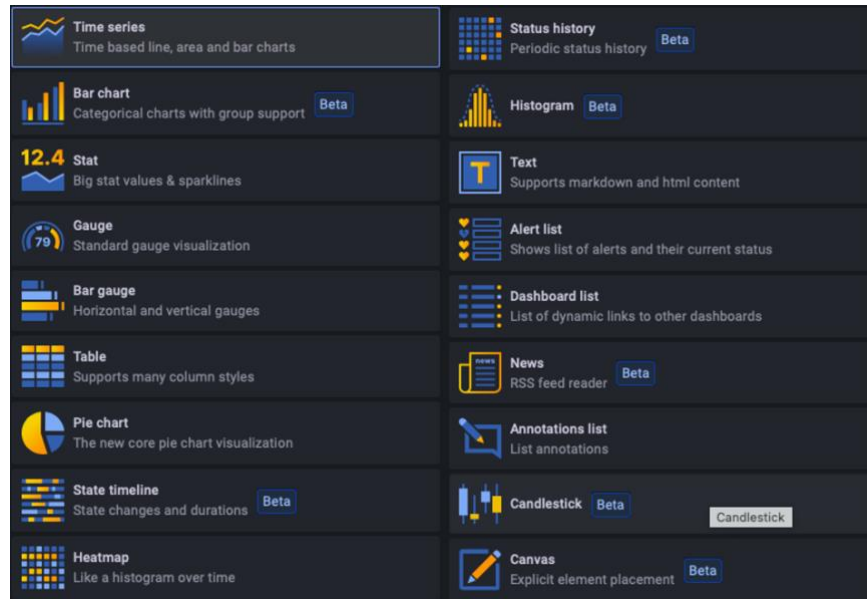


Figura 10-14 Opciones de paneles a configurar

En la configuración de un panel de serie de tiempo se define desde el nombre del panel pasando por nombre de ejes, estilos de gráficos, tamaños de las líneas, tipo de unidad, colores del esquema, la fuente de datos, la consulta de los datos (figura 10-15).

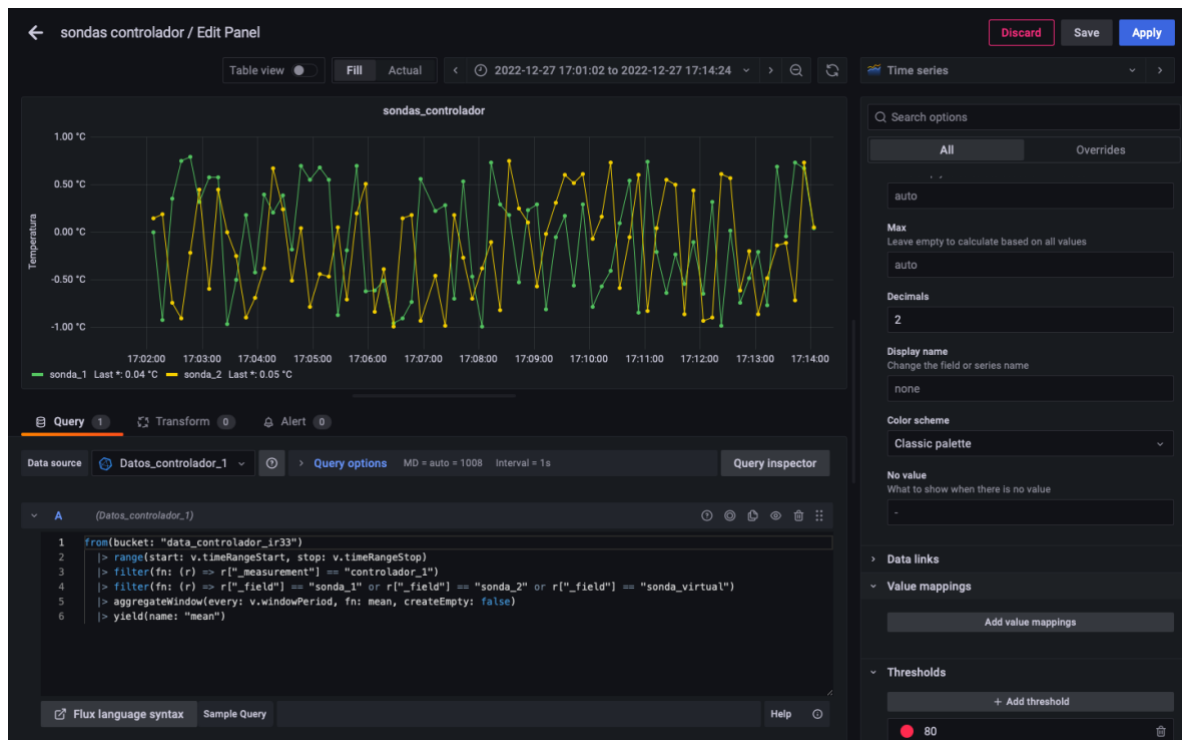


Figura 10-15 Edición de panel para panel de tipo serie de tiempo

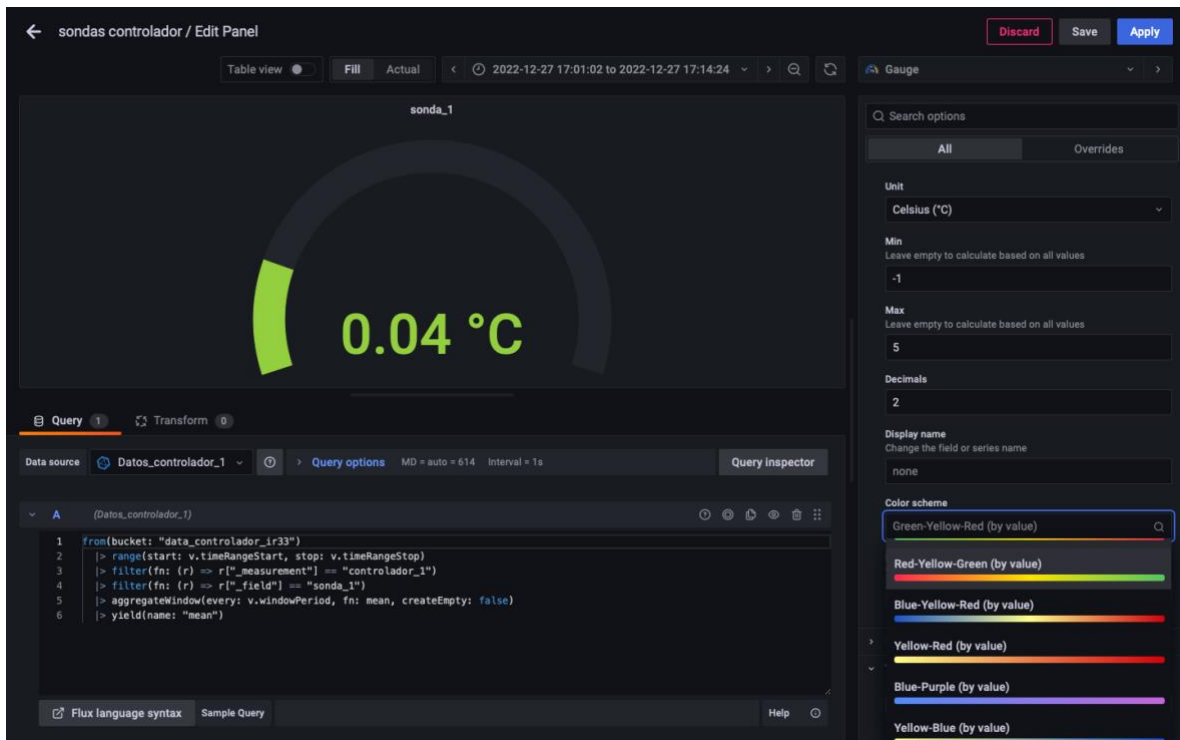


Figura 10-16 Edición de panel para panel de tipo indicador.

10.11 Visualización de datos Grafana Cloud.

Creado los paneles, se despliegan en el Dashboard.

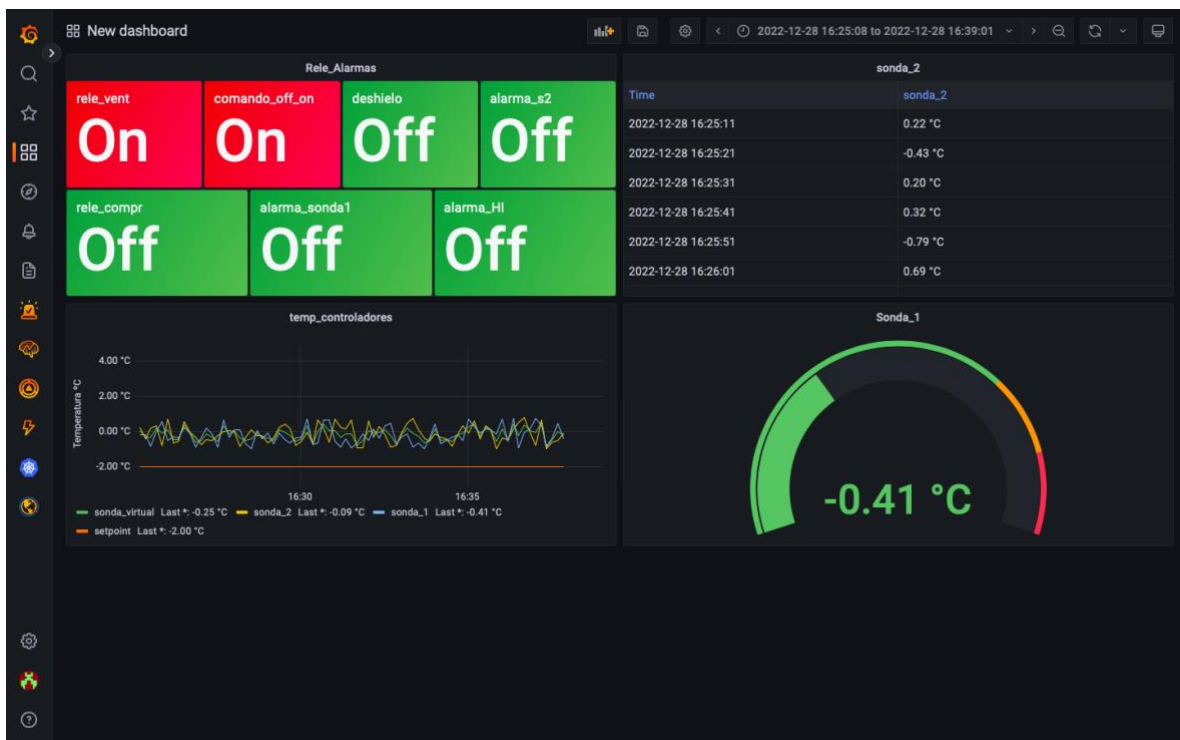


Figura 10-17 Dashboard con paneles configurados

CONCLUSIONES

Se logra realizar un repaso por todas las capas de un sistema IoT desde la capa dispositivos, la capa de comunicación a través de sus distintos protocolos de comunicación permite integrar los datos con distintas aplicaciones.

Una vez los datos publicados logran ser distribuidos a la capa de aplicaciones, la cual permite realizar las gestiones que se determinen necesaria, siempre considerando agregar valor a los datos.

En este trabajo de título se logra diseñar una aplicación IoT para un proceso de refrigeración llevando los datos de las variables físicas desde la capa de dispositivos, que capturan las variables físicas a través de distintos sensores.

Los datos de los dispositivos periféricos son capturados por el PLC CIMON a través de su puerto RS-485 con el protocolo Modbus RTU.

Los datos que se registran en el PLC CIMON son capturados por una aplicación desarrollada con el lenguaje Python, la cual captura los datos gracias a la biblioteca easymodbus y son publicadas a un BROKER MQTT con un topic establecido.

Una aplicación se conecta al BROKER MQTT y se suscribe al topic para la captura de los datos, los datos son enviados a una base de datos no relacional de serie de tiempo InfluxDB.

En Grafana Cloud se integran los datos de InfluxDB para su posterior visualización y su eventual gestión.

BIBLIOGRAFÍA

AWS. ¿Qué es NoSQL?. [en línea]. <<https://aws.amazon.com/es/nosql/>>. [consultado 5 de noviembre de 2022].

CIMON. CIMON Resources. [en línea]. <<https://www.cimon.com/archives>>. [consulta 23 de noviembre 2022].

GRAFANA. Grafana Cloud. [en línea]. <<https://grafana.com/docs/grafana/latest/introduction/grafana-cloud/>>. [consulta 26 de diciembre 2022].

IBM. Servicios Web. [en línea]. <<https://www.ibm.com/docs/es/was-nd/9.0.5?topic=services-web>>. [consultado 5 de noviembre de 2022].

INFLUXDB. Influxdata documentation. [en línea]. <<https://docs.influxdata.com/influxdb/cloud/>>. [consulta 25 de diciembre 2022].

JSON ORG. Introducción a JSON. [en línea]. <<https://json.org/json-es.html>>. [consulta 05 de diciembre].

LUTZ, Mark. Learn Python. 4º edition. O'Reilly Media. 2009. Sebastopol. Introducing Python Object Types p 75. ISBN: 978-0-596-15806-4.

MICROSOFT. Puertas de enlace de IoT Edge de campo y de nube. [en línea]. <<https://learn.microsoft.com/es-es/azure/architecture/example-scenario/iot/field-cloud-edge-gateways>>. [consulta 05 de diciembre 2022].

MYSQL. MySQL Workbench. [en línea]. <<https://dev.mysql.com/doc/workbench/en/>>. [consulta 11 de noviembre 2022].

ORACLE. ¿Qué es una base de datos?. [en línea]. <<https://www.oracle.com/mx/database/what-is-database/>>. [consultado 5 de noviembre de 2022].

PYPI, org. EasyModbus 2.0. [en línea]. <<https://pypi.org/project/EasyModbus/>>. [consulta 5 de enero 2023].

ROTH, Roberta. IoT en refrigeración: Tendencias en automatización [en línea]. <<https://www.novus.com.br/blog/iot-en-refrigeracion-tendencias-enautomatizacion/?lang=es>>. [23 de febrero de 2022].