UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Sistema de monitorización en tiempo real de la calidad eléctrica del edificio 8G de la UPV, mediante el medidor WM4096 con comunicación TCP/IP y almacenamiento de datos.

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

AUTOR/A: Tarín Yusà, Carlos David

Tutor/a: Gimeno Sales, Francisco José

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Máster en Ingeniería Mecatrónica 22-23

Sistema de monitorización en tiempo real de la calidad eléctrica del edificio 8G de la UPV, mediante el medidor WM4096 con comunicación TCP/IP y almacenamiento de datos.

- Muestreo de valores de red Eléctrica
- Procesado de datos de interés
- Centralización en base de datos Web/local en InfluxDB
- Representación visual en Grafana e Interpretación Técnica

TUTOR: Gimeno Sales, Francisco José
Depto. Ingeniería Electrónica

ALUMNO: Tarín Yusà, Carlos David
Estudiante Máster en Ingeniería Mecatrónica

Agradecimientos

En primer lugar, a mi madre y a mi abuelo, a toda mi familia, por su apoyo incondicional en mi proyecto de vida y paciencia en toda mi carrera académica, a nivel de afecto, comprensión, apoyo económico y, sobre todo, entereza.

A mi amigo, Roberto García Ávila, por su paciencia a la hora de responder mensajes a horas intempestivas con dudas sobre la depuración de C. Una amistad que ha demostrado estar muy alejada del interés.

A mi colega Nauzet Jiménez Rodríguez por sus críticas constructivas.

Agradecimiento sincero a TODOS mis profesores, desde la Ingeniería Técnica mecánica, la adaptación al grado en la UPV y, especialmente, a los MAESTROS que me he encontrado a lo largo de esta Ilíada, por su consideración y trato personal, no vacuo, cercano y al mismo modo profesional.

Agradezco a la jefatura de estudios su comprensión para con mis dificultades,

A Pedro Yuste Pérez, jefe de Estudios. Director de la ETSID,

Juan Carlos Cano Escribá, Vicerrector de Profesorado y Ordenación Académica

Especialmente con dedicación a mi TUTOR: Don Francisco José Gimeno, porque todavía tengo en cuenta mis traspiés durante este máster; su consideración a nivel humano y competitivo sin parangón me ha alentado a finalizar este ciclo de mi vida de una forma apropiada y digna.

1. Resumen

Proyecto Fin de Máster consistente en el desarrollo de un programa que facilita la recopilación de datos muestreados por el Analizador de calidad de Red WM4096 Carlo Gavazzi.

El proceso del programa clasifica, almacena y permite la representación gráfica de los datos, con el fin de facilitar su interpretación de una forma visual clara.

Además, se pueden enviar a un recurso remoto donde puede operarse con ellas, examinadas, valoradas y facilitan la toma de decisiones.

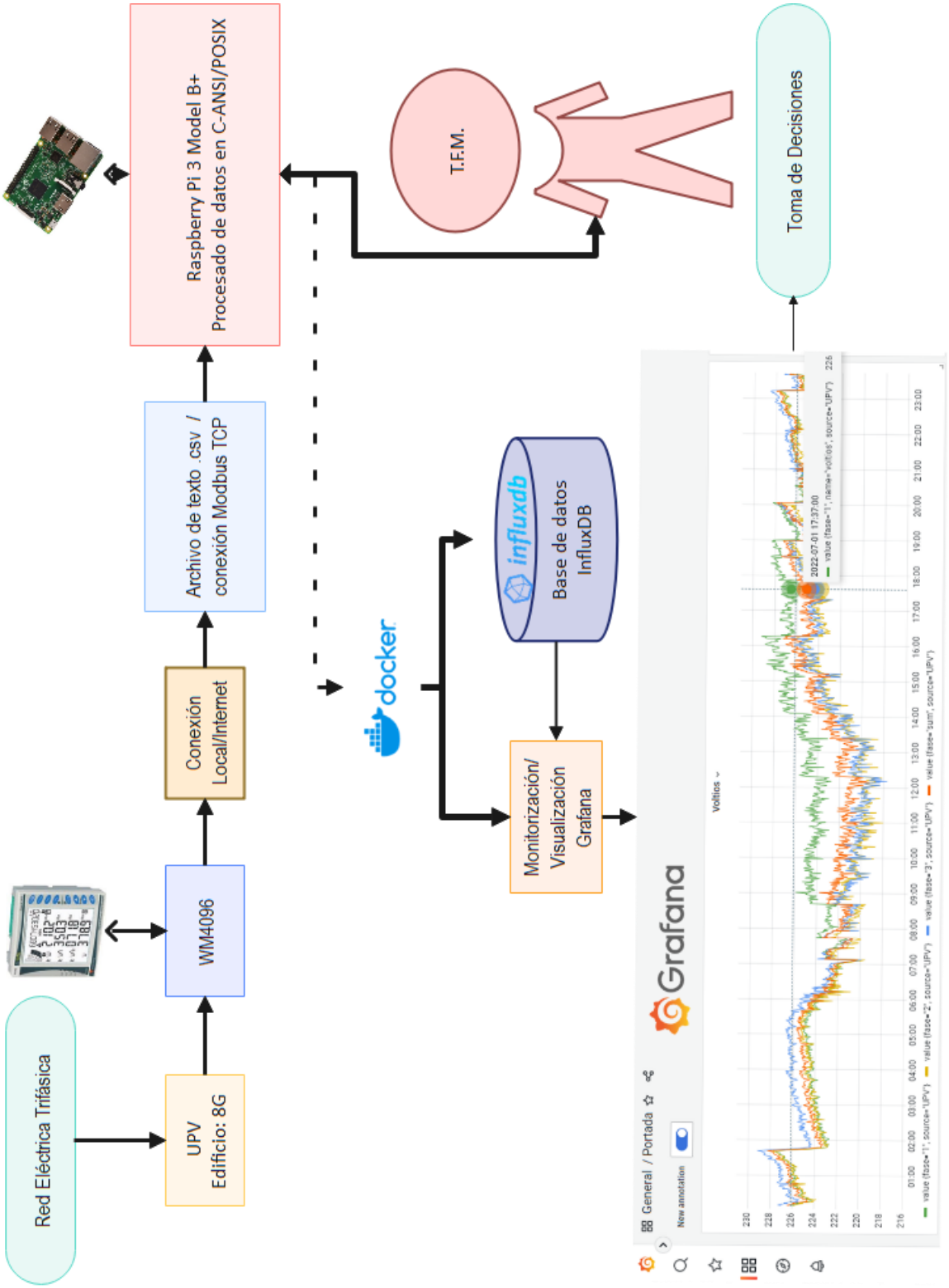
En primer lugar, se ha realizado un estudio de las distintas alternativas en relación a los diferentes lenguajes de programación, y diferentes sistemas de bases de datos, para organizar los valores recopilados, dónde el lenguaje C-POSIX ha resultado ser el seleccionado.

Posteriormente, para trabajar en los distintos protocolos de comunicación y middlewares se han utilizado softwares intermediarios con protocolo MODBUS, como Modbus-POLL o CAS Modbus Scanner, Modbus Master o Simply Modbus TCP Client..

Se utiliza Docker para la virtualización de contenedores; como plataforma es usada una imagen virtual de InfluxDB como gestor de base de datos. A continuación, y como software para monitorizar dichos datos almacenados, es usado Grafana.

La centralización de datos viabiliza el tratamiento, mejora su conectividad, permitiendo procesos posteriores hacia su control.

1.1 Flujograma Básico



INDICE DE CONTENIDOS

Índice

2. Tabla de Ilustraciones

3. Introducción

Desde 2009, surgió el concepto "Internet de las cosas o IoT" (Internet of Things) cuando Kevin Ashton, profesor del MIT, utilizó el término públicamente por primera vez.

El concepto ya fue definido en 1990. El IoT se basa en "una red de objetos físicos que incorporan sensores, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet".

Hoy en día el IoT se aplica en diversos campos como la industria, la domótica, la calidad y estilo de vida, la salud… Es por ello que dichos dispositivos y su entorno de aplicación ha ido desarrollando y generando nuevos retos a los que adaptarse, satisfaciendo necesidades antiguas y nuevas, utilizando nuevas funcionalidades y facilitando su uso de forma cotidiana.

3.1 Contexto Actual

Dentro de las normativas de la EU para el plan energético del 2030; está incluido como objetivo a cumplir la eficiencia energética, para ello deberemos de eliminar la mayor parte de pérdidas innecesarias dentro del sistema eléctrico de distribución.

Si bien el ámbito de aplicación de este proyecto es local, para el caso particular del Edificio 8G de la Universidad Politécnica de Valencia; la gestión y estudio de la calidad de red puede ser extrapolada a todos sus edificios, y puede resultar de interés para industrias de cualquier tamaño.

El almacenamiento, procesado e interpretación de valores relevantes es una tarea en la mayoría de ocasiones compleja, que requiere de varios niveles de desarrollo y conocimiento técnico simplificado para la mera centralización de los datos, para su posterior interpretación por un técnico superior.

Entre las ventajas podemos contemplar que, de una manera accesible, logramos mostrar datos recogidos por sensores in-situ, recibiéndolos de forma remota, desde cualquier sistema informático conectado a internet permitiendo su interpretación por un técnico, hasta que se automatice el proceso (IA).

La gestión de datos de forma ordenada; lógica, ofrece a los técnicos especialistas una fuente muy comprensible para valorar medidas y tomar decisiones, ya sea preventivas, correctivas, de emergencia, o incluso predictivas.

La implantación de la Industria 4.0 ya es una práctica habitual, en nuestra percepción de los sistemas que conocemos, ya sean simples o compuestos, independiente de su nivel de complejidad; es un factor necesario, prácticamente inherente; la integración de lo físico con lo virtual.

Siendo la digitalización de la capa física un ciclo de transformación "retroalimentable", siendo que se tiene que partir de esta capa, con el propio fin de optimizarla.

La virtualización permite abstraer las limitaciones reales de posibilidades a la hora de plantear alternativas; estudiar propuestas de mejora, corregir deficiencias, aumentar la escala de conectividad entre diferentes procesos, exámenes de eficiencia, gestión, rentabilidad…

Siendo más específicos, en este proyecto, además de trabajar desde la perspectiva ya mencionada, pero también explotado, IoT (Internet of Things).

La tecnología de IoT tiene una finalidad clave, evitar la interacción humana en procesos donde no resulte vital, es decir, en tareas con un elevado volumen de carga de trabajo, elevados volúmenes de datos… con el fin de conectar datos con otros sistemas a través de Internet.

Facilitando así que el ser humano se centre en el análisis, interpretación, extracción de conclusiones y toma de decisiones, algo que, por suerte o por desgracia, las computadoras todavía no pueden hacer de una forma comparable al discernimiento humano, dando por evidente que superan la capacidad de cálculo matemático humano.

Sintetizando,

Estamos en transición entre la Cuarta Revolución Industrial (4.0),y la (5.0) cuya línea principal es la **automatización e informatización** experimentada en la Tercera Revolución Industrial, mientras que la (5.0) consistiría entre la colaboración entre la I.A. y el Profesional.

Los procesos actuales están potenciados por el Internet de las Cosas (IoT) y los sistemas físico-virtuales – sistemas autónomos e inteligentes fundamentados en lógica computacional para monitorear y en última instancia interferir de forma eficiente en la maquinaria.

Estas son el medio perfecto para que se interconecten con sistemas *back-end*(1) *tal* como la planificación de recursos.

(1) Los sistemas Back-End son aquellos que no son visibles para el usuario final y que se encargan de realizar tareas y procesos críticos en una aplicación o sitio web. Estos sistemas se encargan de realizar tareas como el almacenamiento de datos, la gestión de bases de datos, el procesamiento de transacciones y la realización de cálculos y procesamientos complejos.

(2) Big-Data se refiere a una gran cantidad de datos que son generados y recopilados a una velocidad tan rápida que resulta difícil de procesar, almacenar y analizar con los sistemas de tecnología de la información tradicionales. Los datos de Big Data pueden provenir de diversas fuentes, como transacciones de comercio electrónico, redes sociales, dispositivos móviles, sensores, entre otros. En resumen, Big Data es una cantidad enorme de información que se genera y se recopila a una velocidad muy rápida, y que es utilizada para obtener conocimientos valiosos y mejorar la toma de decisiones.

Conceptos Clave: *IoT, Big Data, Integración vertical y horizontal, Computación en la nube, Automatización y autonomía, toma de decisiones…*

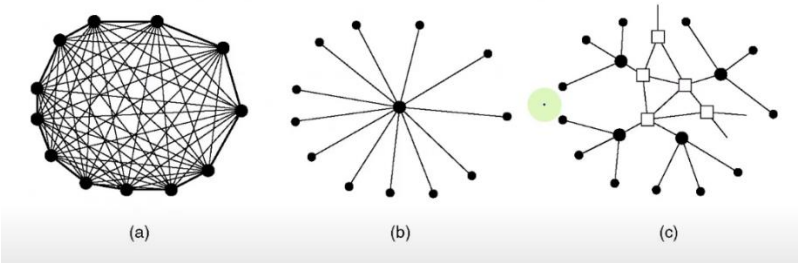


Ilustración a : Diagramas ejemplificantes de tipos de centralización (Distribuida, Centralizada, Cuasi-Descentralizada)

En la línea de concepción de la Industria 4.0 -> 5.0, en nuestros días es una perspectiva ineludible la conectividad e integración de diferentes sistemas.

Más adelante se tratará de explicar y nos adentraremos en los conceptos modernos como la "Industria 4.0" o "Internet of Things".

En particular, es un tema muy actual la gestión masiva de datos, subrayando, muy en especial cuando se trata de temas económicos, como lo es en este proyecto, facilitar el estudio, la viabilidad de toma de decisiones, en relación a la gestión de la red eléctrica y su estado.

Si bien este es un proyecto arquetipo sustancialmente dedicado a recopilar datos cuantitativos abre las puertas a pasos posteriores, hacia proyectos o trabajos que faciliten el cometido de los interesados, el análisis cualitativo de la red eléctrica del consumo y su estado, o sus repercusiones económicas.

El presente proyecto tiene la finalidad de transmitir los datos recogidos por el analizador de calidad de red WM4096 de la marca Carlo Gavazzi, instalado en un edificio de la UPV (8G), en una base de datos llamada InfluxDB®, y su posterior representación visual en Grafana®.

Ilustración b : Analizador de Calidad de red WM-4096

Para ello, la tarea y objeto principal de este trabajo: crearemos un script en C-POSIX que procese todos los datos recogidos por el WM-4096, y realice las tareas de, enviarlos a la base de datos Online de InfluxDB y plasmar los datos volcados.

Posteriormente, mediante peticiones precisas para obtener información, seleccionaremos los datos de interés para crear un Dashboard en Grafana.

Hora punta, Hora valle…

Con los parámetros más relevantes para analizar el estado de la calidad eléctrica,

Tales como:

- Voltaje por fase (R, S, T y N).
- Amperaje por fase.
- Potencia Real por fase.
- Potencia Aparente por fase.
- Potencia Reactiva por fase.
- Factor de Asimetría.
- Coeficientes de Distorsión.
- Porcentaje de Armónicos.
- Factor de Potencia.

4. Objetivos Generales

Obtener la información necesaria de la calidad eléctrica de un edificio determinado con visualización y almacenamiento de dichas ineficiencias. El medio y a la vez fin de este proyecto es en sí mismo realizar el programa en el lenguaje C-POSIX, el desarrollo del programa se ha elegido en este lenguaje dada su *eficiencia y seguridad*, en contraposición a otros más actuales y asequibles en cuanto a comprensión, que ofrecerían un código más sencillo, pero menos robusto, que se compararán más adelante brevemente como alternativas.

Pese a la redundancia conceptual, se indica que el objeto de este TFM es el tratamiento, procesamiento y carga a la base de datos de los mismos recogidos por el analizador de calidad de red, no lo es el mismo análisis eléctrico de la Red, lo que supondría un proyecto demasiado extenso y no es competencia ni el objetivo.

Recapitulando el tema relevante para este estudio, se mostrará en este documento el programa desarrollado, sus respectivos diagramas de flujo (*), hipótesis y conceptos asumidos…

(*) Los flujogramas son orientativos, sintácticos, y el código aquí mostrado es parcial, siendo el depositario del código completo la Universidad Politécnica de Valencia.

Los objetivos desarrollados en este TFM son los siguientes:

- Estudiar las diferentes alternativas respecto a los lenguajes de programación, sus ventajas e inconvenientes. Específicamente se trabajará con C-POSIX. Tratando de garantizar un flujo de datos estable y robusto, probando diferentes tecnologías. Se deberá garantizar la seguridad y confidencialidad de los protocolos de comunicación usando TCP Y HTTP.

- Estudiar las plataformas de gestión de bases de datos Open-Source disponibles, como consecuente selección es InfluxDB una muy potente herramienta para este fin.

- Seleccionar un software de visualización para el tratamiento de dichos datos. En nuestro caso, Grafana suple dichas necesidades con solvencia.

Objetivos secundarios:

- Realizar la captura, procesamiento y almacén de datos con un "microcontrolador" del tipo RaspberryPI 3 B+.

- Familiarizarse con dicho microcontrolador y su entorno de programación, así como la ejecución del programa en diferentes Sistemas Operativos.

- Realizar el primer prototipo funcional e implementar el programa de forma que pueda ser accesible desde cualquier dispositivo conectado en la Red de Internet de la UPV.

Objetivo terciario:

- Diseñar mediante Software CAD una carcasa para la RaspberryPI 4B

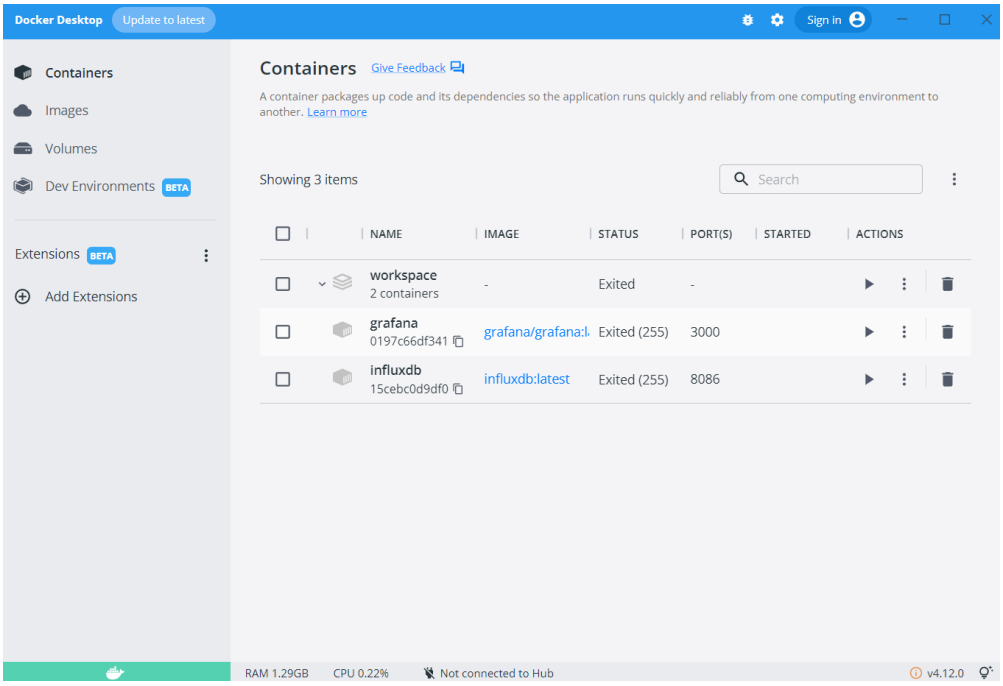


Ilustración c : Preparación de los contenedores virtuales mediante Docker.

Además de lo anterior, se añade la virtualización de las plataformas InfluxDB y Grafana en contenedores de máquinas virtuales, lo que facilita su ejecución y control mediante **Docker**.

(Selección que será justificada posteriormente)

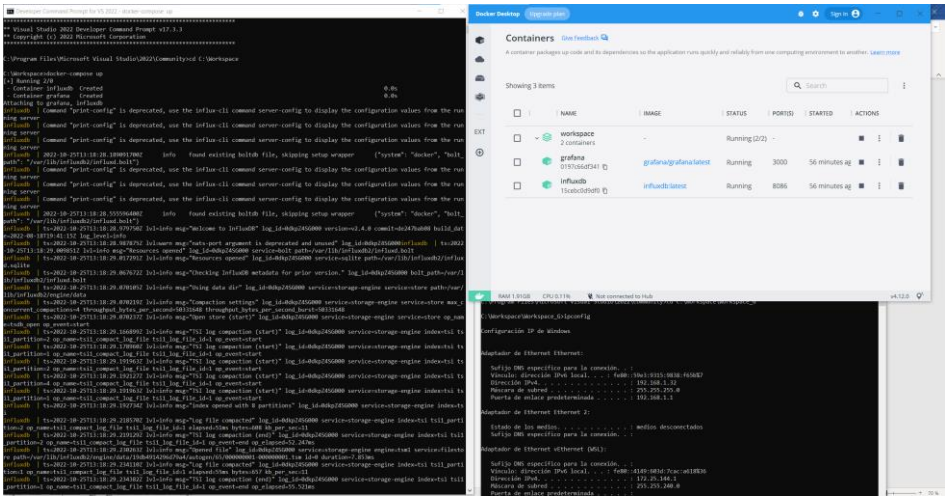


Ilustración d : Puesta en marcha de InfluxDB y Grafana mediante un Script C.

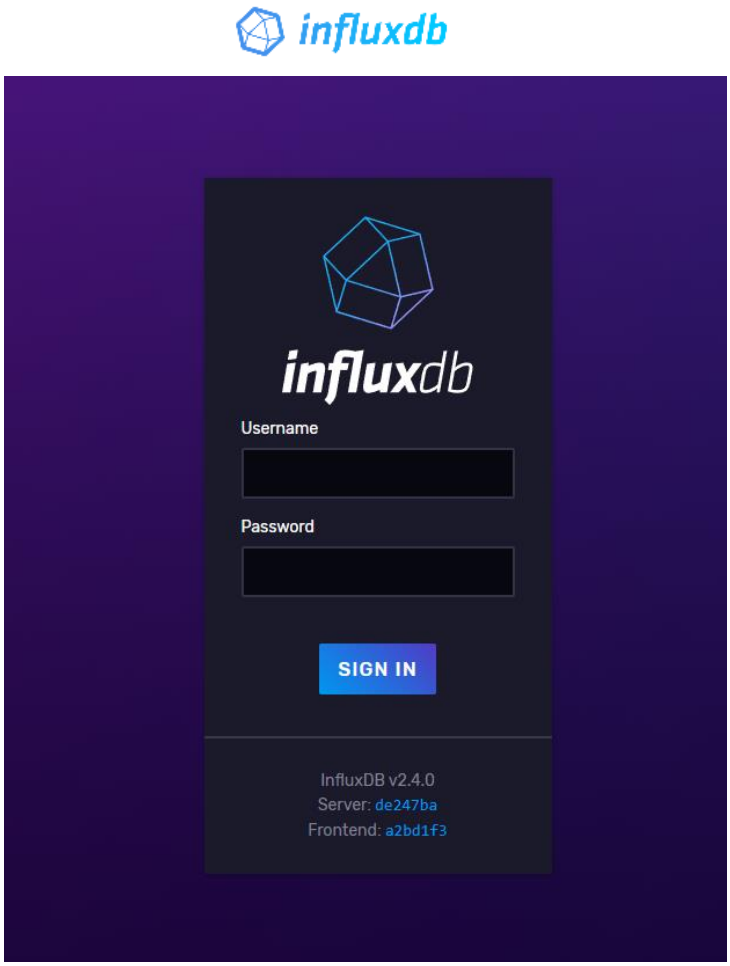


Ilustración e : Acceso Local a InfluxDB.

InfluxDB será la herramienta utilizada como base de datos para tomar las muestras.

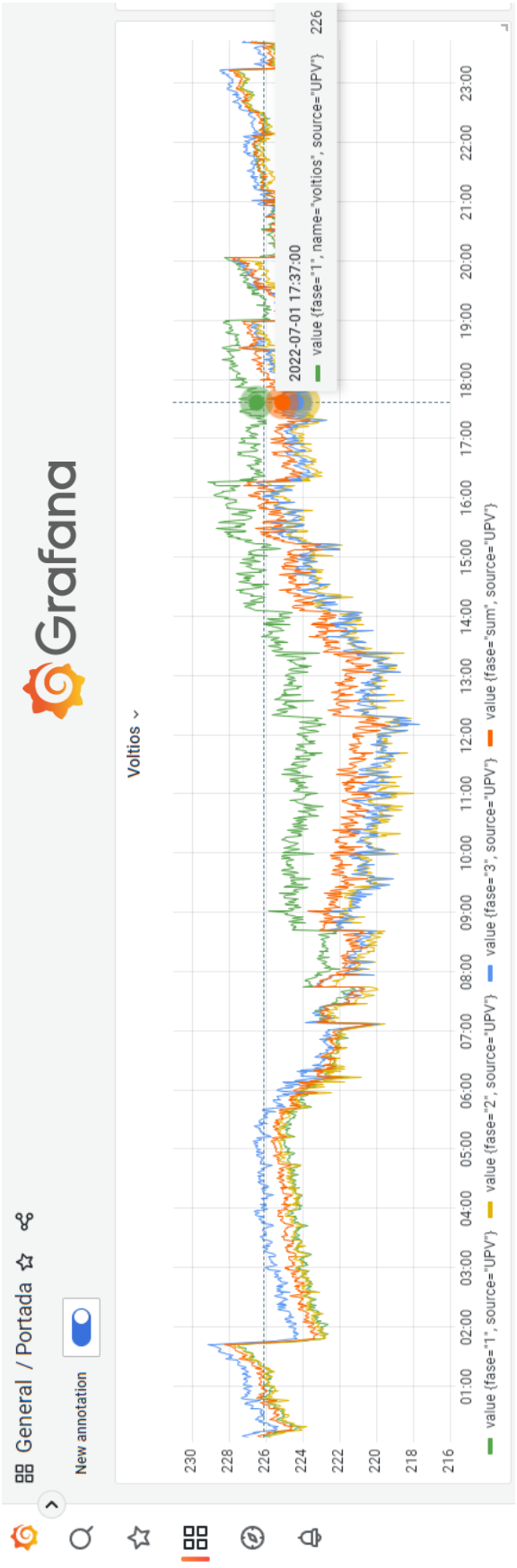


Ilustración f : Representación Gráfica en base de datos centralizada en protocolo local/Web

4.1 Objetivos particulares

- Para obtener la información necesaria de la calidad eléctrica de un edificio determinado con visualización y almacenamiento de dichas ineficiencias desglosamos el proceso desarrollado:

Para ello la solución propuesta, a partir de un analizador de calidad eléctrica y mediante un Software desarrollado en lenguaje C. Se comunicará con dicho analizador y le enviará los parámetros adecuados y os almacenará en una base de datos para luego visualizarlos en Grafana y facilitar la toma de decisiones y análisis.

- Generar y Tratar los datos del archivo .csv devuelto por el analizador de Calidad de Red.
- Procesar dichos datos por el programa en lenguaje C-POSIX.
- Transmisión a través de un protocolo Local/Internet.
- Almacenamiento correcto en la base de datos InfluxDB.
- Sincronizar dichos datos entre la base de datos, y el visualizador Grafana
- Crear un Dashboard accesible y visual de representación de valores de relevancia.
- Comprobar la eficiencia del código y su respuesta temporal, Refactorizarlo.

No Incluye:

- Variación del sistema de adquisición de datos

- Análisis específico del estado de la Red Eléctrica

Los beneficios obtenidos de este sistema ofrecido en este TFM son muy prácticos:

El analizador inteligente entrega una cantidad ingente de mediciones diarias de un solo sistema al que está conectado, si, extrapolamos a varios edificios, con varios WM40-96, y resulta de interés estudiar dichos datos por un técnico eléctrico, el volumen de datos haría esta tarea prácticamente intratable.

Al manejar los datos con el programa aquí desarrollado, la carga en una base de datos y la representación en un Dashboard, la conectividad de dichos datos permite su acceso, interpretación y valoración de una forma humanamente comprensible.

Pudiendo ser una propuesta de mejora no realizada en este TFM, pero planteada, crear una IA capaz de injerir en los procesos de estudio de la red.

5. InfluxDB y Grafana

InfluxDB es un servidor de base de datos *opensource* [3] de series temporales, (series de tiempo), perfecto para almacenamiento de datos o logs para ser graficadas posteriormente, idealmente en vivo. Es decir, permite que dichos *datos se recuperen*.

Además de la supervisión, se puede utilizar para el Internet de las Cosas, IoT.

Su programación permite la interacción vía API HTTP, y los datos se gestionan en un lenguaje similar a *SQL*.

- Almacén de datos personalizado de alto rendimiento escrito específicamente para datos de series temporales.
- El motor TSM permite una alta velocidad de ingesta y compresión de datos
- 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.

Grafana, en el otro extremo, es un software igualmente open source, que permite crear cuadros de mando y gráficos mediante obtención de datos métricos, ofrece un sistema de visualización personalizado de datos de serie temporales, incluso usar un sistema de alarmas, donde todo ello es configurable desde la interfaz gráfica del usuario.

Permite, mediante el uso de plugin, conectar las fuentes de datos a través de las API y presentan los datos en tiempo real sin necesidad de migrar los datos.

[3]Open source es un modelo de desarrollo y distribución de software que permite a los usuarios acceder al código fuente del software, modificarlo y redistribuirlo libremente. En otras palabras, el código fuente de un software de código abierto está disponible para ser estudiado, modificado y compartido por cualquier persona interesada. Es un modelo de desarrollo de software que permite a los usuarios acceder y modificar el código fuente, con el objetivo de promover la colaboración y el intercambio de conocimientos.

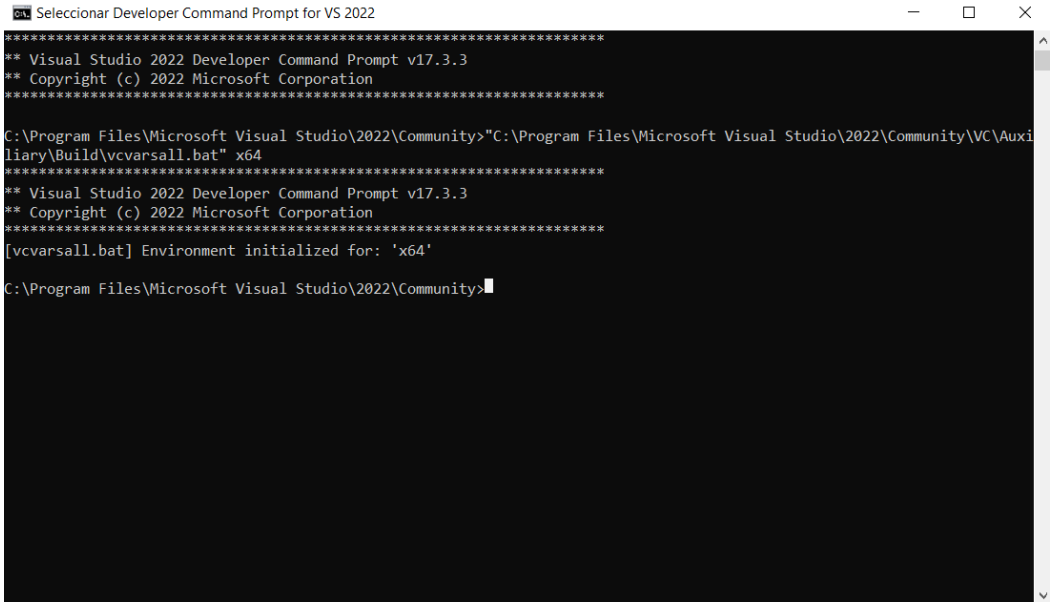


Ilustración g : Consola preparando la ejecución del .bat "Docker-compose up"

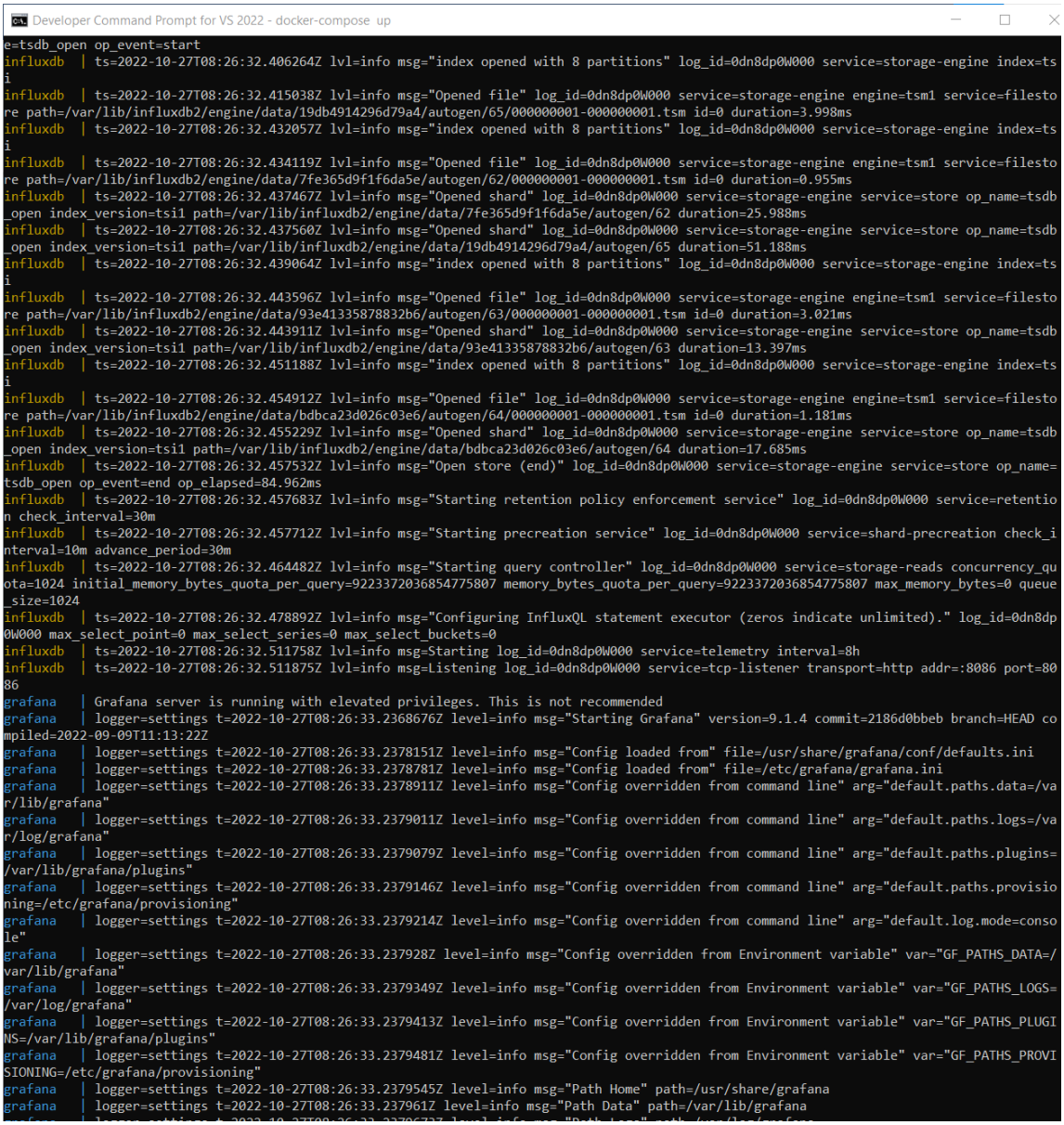


Ilustración h :.Consola de log del Docker-Compose up.

6. Selección Final

Docker es un proyecto de código abierto para automatizar la implementación de aplicaciones como contenedores portátiles y autosuficientes que se pueden ejecutar en la nube o localmente.

Permite empaquetar una aplicación y sus dependencias en un contenedor de manera aislada

Proporcionando una capa adicional de abstracción y virtualización de aplicaciones.

InfluxDB ha sido diseñado para **guardar** datos que almacenan series temporales. Estas bases de datos se usan entre otras aplicaciones, para almacenar y evaluar datos de sensores, protocolos, etc. con marcas temporales.

Es un sistema de base de datos de serie de tiempo diseñado para almacenar y consultar grandes cantidades de datos. Es especialmente útil para aplicaciones de monitoreo, análisis de datos en tiempo real y visualización de datos en tiempo real. InfluxDB es una solución de código abierto y ofrece características como alta disponibilidad, escalabilidad y rendimiento.

Grafana es una plataforma de visualización y análisis de datos de código abierto. Se integra con varias fuentes de datos, incluyendo bases de datos de tiempo de serie como InfluxDB, y permite crear paneles y personalizados para visualizar y analizar datos. Grafana es ampliamente utilizado en aplicaciones de monitoreo y supervisión, y es conocido por su facilidad de uso y personalización. Además, Grafana tiene una amplia comunidad de desarrolladores y ofrece una amplia gama de plugin y para expandir sus capacidades.

6.1 Hipótesis asumidas

Teniendo en cuenta las siguientes consideraciones;

1. El WM-4096, siempre devolverá un archivo .csv con el mismo formato, es decir, mismas columnas y filas, sin espacios vacíos.

2. Se propone un sistema de alertas real cuando se alcancen valores indeseados como puede ser de distorsión armónica elevada, o factores de potencia excesivos.

3. Dado que el programa ya supone un reto por sí mismo, especialmente al realizarlo en lenguaje C-ANSI, no es objetivo el análisis y conclusiones sobre el estado de la RED ELÉCTRICA, por lo que muchos valores recogidos no resultan de interés en este instante. Además, pueden verse afectadas las características de este proyecto.

...

C: > Workspace > docker-compose.yml

```yaml
version: "3"
services:
  influxdb:
    image: influxdb:latest
    container_name: influxdb
    ports:
      - "8086:8086"
    volumes:
      # Data persistency
      # sudo mkdir -p /srv/docker/influxdb/data
      - C:\Workspace\Worspace_C\influx:/var/lib/influxdb
  #####
  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3000:3000"
    user: "0"
    links:
      - influxdb
    volumes:
      # Data persistency
      # sudo mkdir -p /srv/docker/grafana/data; chown 472:472 /srv/docker/grafana/data
      - C:\Workspace\Worspace_C\grafana:/var/lib/grafana

```

Ilustración i : Código parcial que levanta las imágenes virtuales de InfluxDB y Grafana de forma Local

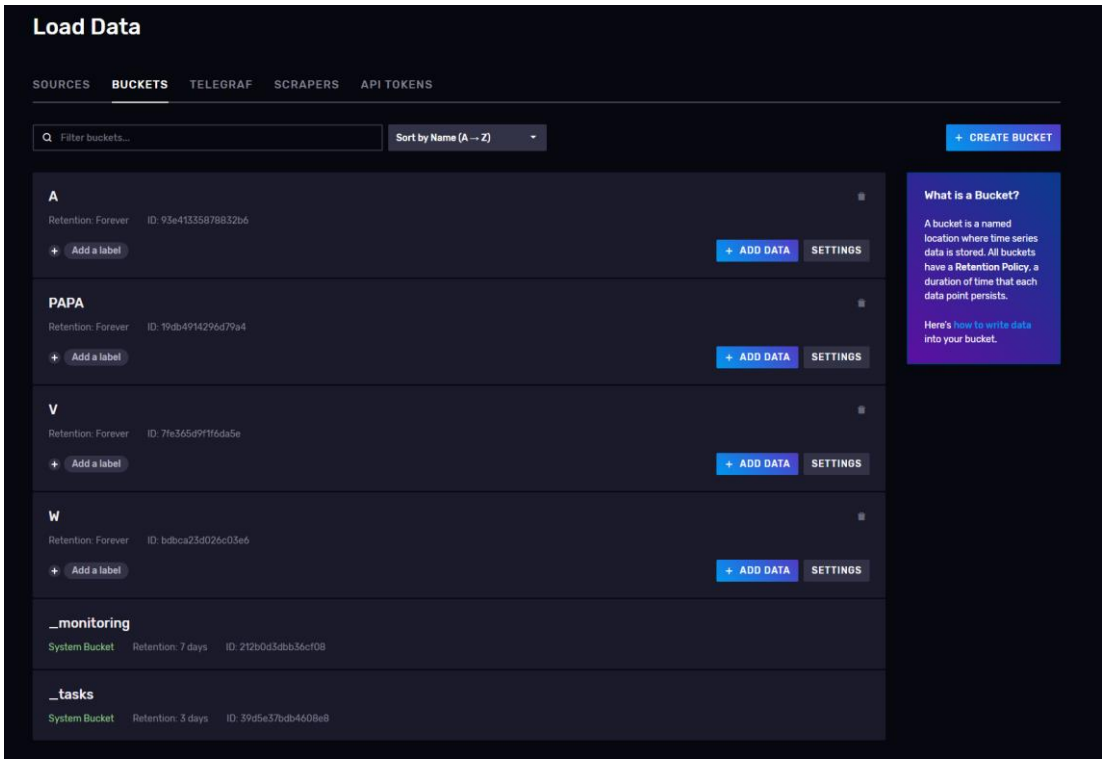


Ilustración j : Propuesta actual de Buckets.

Un ***Bucket*** es un depósito con nombre donde se almacenan datos de series temporales. Todos los Buckets tienen un período de retención, una duración de tiempo que persiste para cada punto de datos. InfluxDB elimina todos los puntos con marcas de tiempo anteriores al período de retención del Bucket. Un Bucket pertenece a una organización.

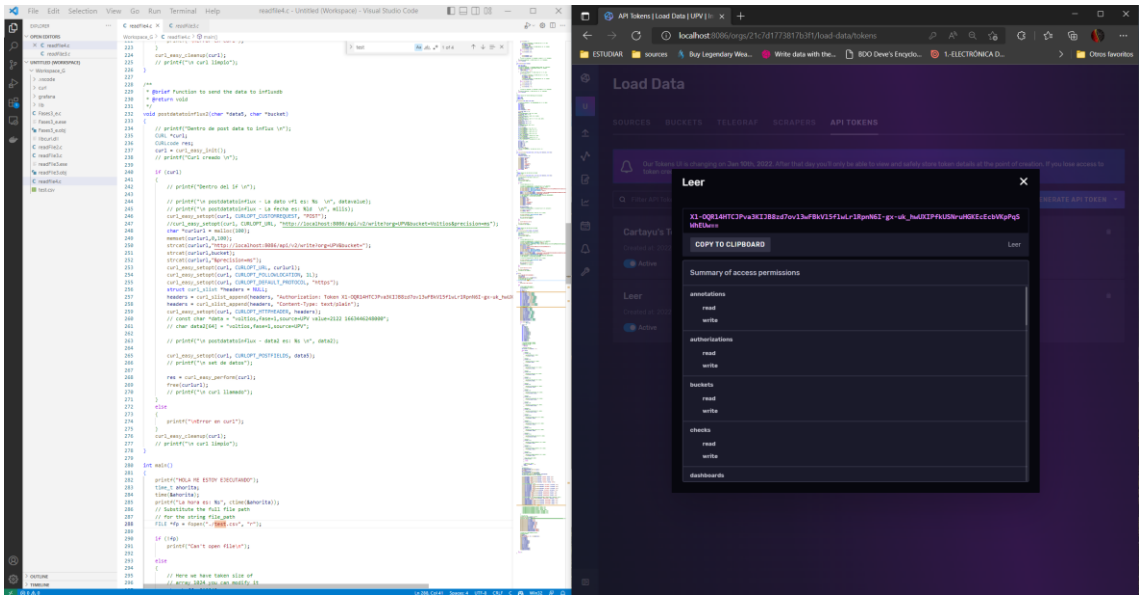


Ilustración k : Relevancia del API-Token, en términos de comunicación y seguridad.

Los tokens de la API de InfluxDB garantizan una interacción segura entre los usuarios y los datos. Un token pertenece a una organización e identifica los permisos de InfluxDB dentro de la organización.

Ilustración l : Representación de un Bucket de este TFM en INFLUXDB, la base de datos.

6.2 Análisis de Alternativas e investigación

El presente proyecto tiene un carácter iterativo o recursivo, es decir; su ejecución requiere de un trabajo simultáneo a entre lo desarrollado y lo que queda por desarrollar, lo que es natural a la hora de depurar un programa de estas características.

Se han refrescado conceptos tanto de programación, siendo el grueso el código en C, como de sistemas de red eléctrica de baja tensión, no siendo este último tema principal de este TFM, igualmente necesario para manejar conceptos sobre los datos.

La diferencia entre Python y C es que Python es un lenguaje de paradigma múltiple y C es un lenguaje de programación estructurado. Python es un lenguaje de uso general que se utiliza para el aprendizaje automático, el procesamiento de lenguaje natural, el desarrollo web…

En C ++ se compila en un código objeto que luego se puede ejecutar para producir una salida.

Java es un lenguaje compilado e interpretado. La salida compilada de un código fuente de Java es un código de bytes que es independiente de la plataforma.

IEEE 754

El estándar IEEE 754 es un estándar internacional para la representación de números reales en la computadora. Este estándar especifica cómo se almacenan los números con punto flotante en la memoria de una computadora, lo que garantiza la precisión y compatibilidad entre diferentes sistemas. La implementación de este estándar permite una representación uniforme de los números reales, lo que facilita la interoperabilidad de software y hardware en diferentes plataformas.

El estándar IEEE 754 define dos formatos de representación de números con punto flotante: el formato simple y el formato doble precisión. El formato simple utiliza 32 bits para almacenar un número real, mientras que el formato doble precisión utiliza 64 bits. Ambos formatos tienen una estructura similar, con una parte de la memoria reservada para el signo, una parte para la representación de la magnitud y una parte para la representación de la precisión.

El estándar define:

- Formatos aritméticos: conjuntos de datos de punto flotante binarios y decimales, que consisten en números finitos, incluidos los ceros con signo y los números desnormalizados, infinitos y valores especiales "no numéricos" (NaN).
- Formatos de intercambio: codificaciones (cadenas de bits) que se pueden utilizar para intercambiar datos de punto flotante de forma eficiente y compacta.
- Reglas de redondeo: propiedades que deben satisfacerse al redondear los números durante las operaciones aritméticas y las conversiones.
- Operaciones: operaciones aritméticas y otras (como funciones trigonométricas) en formatos aritméticos.
- Manejo de excepciones: indicaciones de condiciones excepcionales, tales como división por cero, desbordamiento, etc.

La norma también es conocida como **IEC 60559:1989,** *Binary floating-point arithmetic for microprocessor systems* (originalmente el número de referencia era IEC 559:1989).

En resumen, el estándar IEEE 754 es una norma esencial para el desarrollo de software y hardware que manejan números reales en la computadora, ya que permite la representación uniforme y precisa de los números y garantiza la compatibilidad entre diferentes sistemas.

ENDIANNESS

El término inglés ***endianness*** ("extremidad") designa el formato en el que se almacenan los datos **de más de un byte** en un ordenador. El problema es similar a los idiomas en los que se escriben de derecha a izquierda, como el árabe, o el hebreo, frente a los que se escriben de izquierda a derecha, pero trasladado de la escritura al almacenamiento en memoria de los bytes.

Endianess es un término utilizado en informática que describe el orden en que se almacenan los bytes de un número en la memoria de una computadora. Hay dos formas de almacenar los bytes: en orden "big-endian" o "little-endian".

En big-endian, el byte más significativo se almacena en la dirección de memoria más baja, mientras que en little-endian, el byte menos significativo se almacena en la dirección de memoria más baja.

De forma simple, endianess describe la forma en que se guardan los bytes de un número en la memoria de la computadora.

6.3 Dispositivos recomendados

El dispositivo seleccionado debe tener unas características determinadas orientadas a la realización de proyectos IoT. Una de estas características esenciales es la capacidad de comunicación de forma inalámbrica, preferiblemente por Wifi.

En añadidura, otras características que influirán en su elección son:

Capacidad de procesamiento del microcontrolador, precio, capacidad de conectividad USB, velocidad del reloj, memoria…

Se pretende que el dispositivo escogido tenga un microprocesador que le permita tratar los datos a gran velocidad y que sea capaz de almacenar dichos datos en su memoria hasta que pueda enviarlos al sistema de base de datos de forma inalámbrica.

El dispositivo preseleccionado con el que se ha desarrollado este TFM es una Raspberry PI 3 model B+.

Ilustración m : RaspberryPI 3

Característica	Raspberry Pi 400	ASUS Tinker Board 2s	LattePanda 3
Procesador	Broadcom BCM2711B0, quad-core Cortex-A72 64-bit SoC @ 1,8 GHz	Dual-core Arm Cortex-A72 a 2 GHz Quad-core Arm Cortex-A53 a 1,5 GHz	Intel Celeron N4100 @ 1.1GHz - 2.4GHz (Quad-Core Gemini Lake)
GPU	VideoCore VI 500 MHz	Arm Mali-T860 MP4 a 800 MHz	Intel UHD Graphics 600
RAM	1GB, 2GB, 4GB 8GB	2GB Dual-Channel DDR3	8GB LPDDR4
Almacenamiento	MicroSD (no incluido)	MicroSD (no incluido)	64GB eMMC
Conectividad	Ethernet, Wi-Fi, Bluetooth	Gigabit Ethernet, Wi-Fi, Bluetooth 4.0	Gigabit Ethernet, Wi-Fi 5, Bluetooth 4.2
Puertos	GPIO 40 pines 2 x Micro HDMI 1 x USB 2.0 2 x USB 3.0 Micro SD USB-C (alimentación)	HDMI, 3.5mm audio jack, 2 USB, Gigabit Ethernet, 40-pin GPIO header	HDMI, 3.5mm audio jack, 3 USB, Gigabit Ethernet, 40-pin GPIO header
Sistema operativo	Raspbian, Ubuntu, NOOBS, entre otros	Ubuntu, Android, entre otros	Windows 10
Precio (aproximado)	Desde 100€	Desde 120€	Desde 280€

Los tres dispositivos son populares en el mundo de la computación embebida y tienen diferentes fortalezas y debilidades. Raspberry Pi es conocido por su bajo costo y amplia compatibilidad con diferentes sistemas operativos, mientras que Asus Tinker Board y LattePanda 3 tienen un rendimiento más potente y conectividad mejorada. La decisión final depende de las necesidades específicas de cada proyecto.

Raspberry Pi 3 Model B+ es un ordenador de placa única de bajo coste, desarrollada con el objetivo de estimular la enseñanza de informática y programación. Posee la capacidad de funcionar como una computadora personal completa y usarla para navegar por internet, redactar documentos, programar y jugar.

Además, posee la opción de trabajar de forma similar al Arduino: permitiendo programar el uso de sus pines GPIO, que incluye también comunicación serial SPI e I2C. Estas funciones hacen que sea un dispositivo ideal para ser empleado en proyectos de electrónica y robótica interactuando con sensores y actuadores, especialmente útil en aplicaciones con procesamiento de imágenes/video, y cálculos matemáticos complejos.

La tarjeta tiene como componente principal al System on Chip (SoC) Broadcom BCM2837 compuesto por un procesador ARMv8 de cuatro núcleos de 64 bits a una velocidad de 1.4 Ghz, GPU Broadcom VideoCore IV y memoria Ram de 1GB.

Además, incluye cuatro puertos USB 2.0, conexión HDMI, Wifi Doble banda, Bluetooth.

Soportando el rango de distribuciones ARM GNU/Linux, es Raspbian x64 la usada en este TFM.

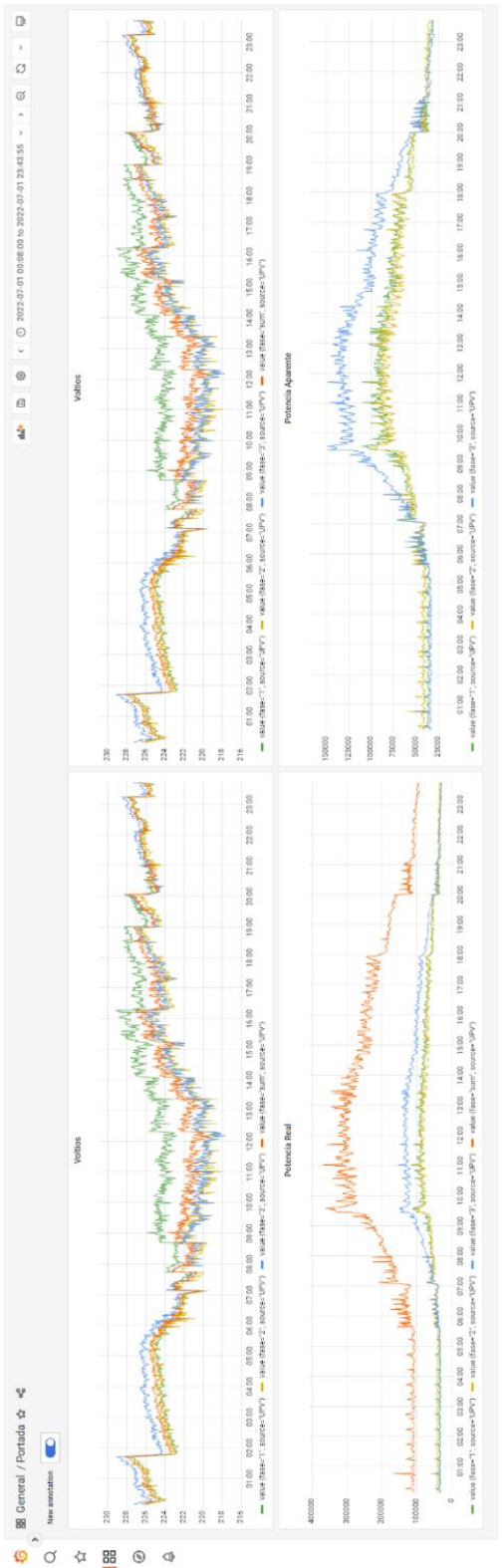


Ilustración n : Dashboard de Grafana con los datos reales proporcionados.

7. Desarrollo del Trabajo

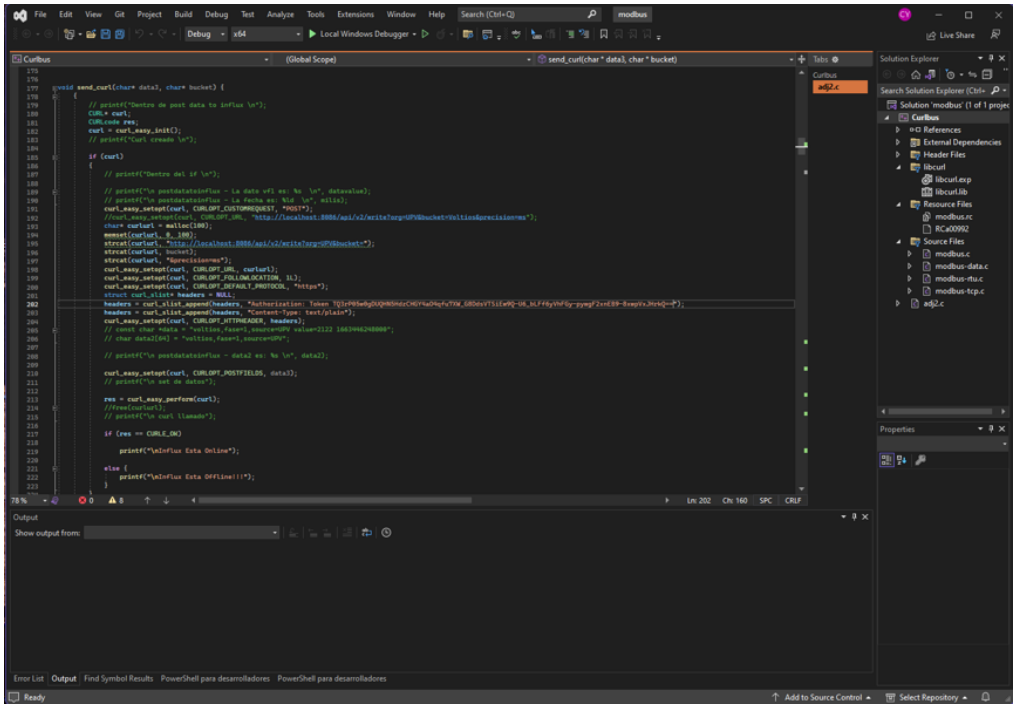
Hay tres códigos desarrollados en este TFM:

6.1. Código de Ensamblado del .csv

6.2. Código de Lectura del .csv

6.3. Código de Lectura (Modbus) en tiempo Real

Estos tres códigos son el núcleo del desarrollo de este TFM, siendo el método seleccionado para alcanzar los objetivos especificados. *Los fragmentos de códigos aquí mostrados, son un desarrollo intermedio, siendo la UPV los depositarios del código final y funcional de este trabajo.*



7.1 Código de Ensamblado del .csv

Este código ejemplificado en un desarrollo por el Tutor, Francisco José Jimeno Sales, en lenguaje JavaScript como medio, con el fin de guiar al Tutelado y servir de Ejemplo.

Además, este código de ensamblado del archivo de texto .csv, sirvió de base y facilitó el fin del resto de códigos.

Este código genera un archivo de texto plano, donde los datos serán tratados posteriormente en los siguientes códigos, en el mismo programa, para su lectura (8.2) o tratamiento en tiempo real (8.3).

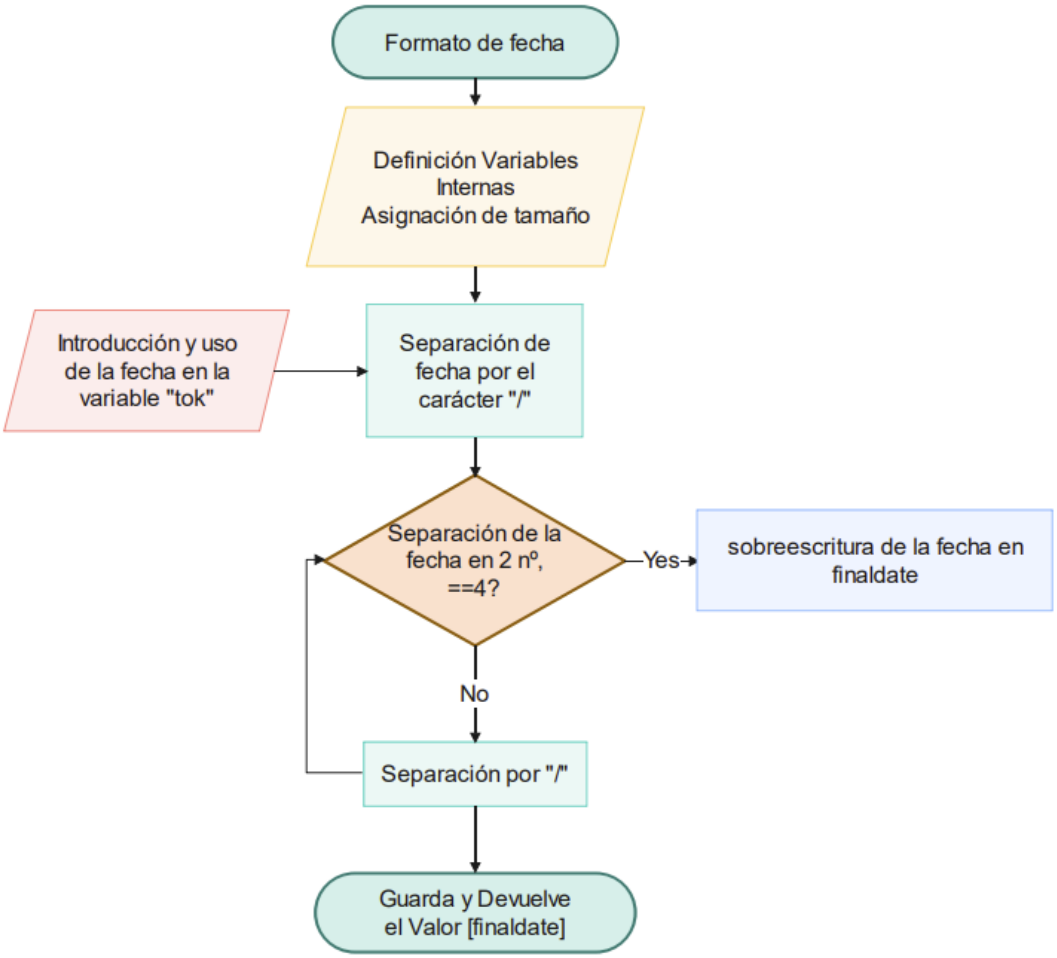
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	fecha	hora	Voltaje_L1_N	Voltaje_L2_N	Voltaje_L3_N	Voltaje_L_N_	Corriente_L1	Corriente_L2	Corriente_L3	Corriente_N	Pt_Real_L1	Pt_Real_L2	Pt_Real_L3	Pt_Real_SUM	Pt_Aparente	Pt_Aparente_	Pt_Aparente_	Pt_Aparente_	Pt_Reactiva_	Pt_Reactiva_	Pt_Reactiva_	Pt_Reactiva_	PF
2	01/07/2022	0:01:23	225.92	225.59	226.88	226.16	167.79	191.16	156.25	42.06	37178.94	42970.42	35195.75	115071.88	37621.06	43249.55	35400.71	116357.30	-6773.44	-5168.48	-3814.13	-15752.84	-0.
3	01/07/2022	0:02:23	225.96	225.75	227.06	226.33	166.16	188.29	155.72	42.24	37073.28	42004.03	35109.89	114104.41	37712.26	42347.34	35198.80	114977.90	-6820.75	-5413.35	-3747.54	-15930.15	-0.
4	01/07/2022	0:03:23	225.80	225.57	226.75	225.99	168.95	186.91	156.49	40.78	37332.08	41963.89	35277.51	114730.62	37973.39	42162.16	35590.69	115981.48	-6912.25	-5496.44	-3623.00	-16017.04	-0.
5	01/07/2022	0:04:23	225.97	225.99	226.97	226.21	161.58	178.75	149.48	39.28	35681.55	39943.55	33645.27	109259.52	36501.98	40189.54	33767.49	110505.27	-7479.12	-6204.66	-4475.00	-18153.62	-0.
6	01/07/2022	0:05:23	226.01	225.79	227.00	226.23	159.81	186.55	148.80	41.80	35300.61	41195.14	33402.32	109579.63	35980.20	40692.66	33814.17	110298.07	-7586.21	-5505.60	-4474.58	-17463.21	-0.
7	01/07/2022	0:06:23	225.69	225.74	226.74	226.07	161.22	172.18	146.41	38.58	35731.01	39716.19	32829.20	107172.88	36469.57	38963.98	33325.27	108850.41	-7384.21	-5463.70	-4683.37	-17695.11	-0.
8	01/07/2022	0:07:23	225.74	225.52	226.89	226.01	156.05	176.88	146.92	38.51	34452.21	39649.35	32845.16	106761.00	35055.13	39896.58	33200.69	108061.80	-7212.78	-5539.46	-4823.40	-17543.36	-0.
9	01/07/2022	0:08:23	225.77	225.74	226.96	226.23	152.24	179.10	145.45	40.51	33643.96	40074.21	32846.73	106793.84	34574.08	40516.73	33168.71	107996.31	-7305.12	-5425.86	-4721.84	-17481.48	-0.
10	01/07/2022	0:09:23	226.07	225.82	227.28	226.43	155.00	179.92	145.75	39.39	34383.76	40602.99	32895.37	107037.46	34489.69	40590.11	33200.38	108425.98	-7283.93	-5314.13	-4716.12	-17307.67	-0.
11	01/07/2022	0:10:23	225.84	225.56	226.98	226.08	154.13	182.44	148.55	39.60	34102.12	40689.04	33318.06	108291.02	34603.00	40952.97	33733.02	109551.98	-7157.33	-5469.45	-4444.07	-17135.81	-0.
12	01/07/2022	0:11:23	225.52	225.22	226.67	225.82	156.01	192.61	148.96	46.11	34366.29	43091.79	33439.46	110832.91	35040.64	43479.59	33753.65	111943.16	-7100.63	-5636.63	-4469.83	-17093.22	-0.
13	01/07/2022	0:12:23	225.32	224.96	226.37	225.69	153.85	185.00	148.25	40.91	33635.73	41353.63	33196.02	108337.57	34399.89	41658.36	33815.86	110060.60	-7454.46	-5530.08	-4489.38	-17507.43	-0.
14	01/07/2022	0:13:23	225.60	225.34	226.67	225.89	153.96	179.67	151.19	37.32	33814.52	40100.59	33927.04	108559.19	35163.54	40441.57	34277.26	109539.00	-7272.59	-5416.36	-4488.96	-17117.45	-0.
15	01/07/2022	0:14:23	225.15	224.86	226.19	225.38	156.43	181.85	152.27	39.40	34531.63	40696.47	34046.33	108960.02	35078.16	40847.71	34376.95	110174.81	-7212.36	-5354.77	-4429.83	-17008.34	-0.
16	01/07/2022	0:15:23	224.43	224.03	225.48	224.58	186.06	215.06	182.98	41.14	41772.30	48217.85	41236.04	131391.13	42250.45	48250.83	41305.82	131949.02	-5023.41	-3809.41	-3136.80	-11983.36	-0.
17	01/07/2022	0:16:23	224.44	224.15	225.43	224.67	182.75	206.03	173.28	41.46	40584.88	46008.57	39043.08	125687.82	40982.81	46045.57	38943.07	125915.45	-5228.88	-3895.17	-3015.80	-12147.61	-0.
18	01/07/2022	0:17:23	224.23	224.04	225.60	224.61	170.00	207.72	158.54	49.74	37001.67	46314.45	35743.28	119432.95	37481.22	46399.71	35934.37	119777.15	-6299.43	-4755.45	-3647.98	-14632.39	-0.
19	01/07/2022	0:18:23	224.08	223.80	225.34	224.56	164.81	194.40	155.87	42.66	36488.91	43320.63	35024.15	114743.31	37054.63	43591.56	35145.84	115399.39	-6632.31	-4660.93	-3746.60	-14953.43	-0.
20	01/07/2022	0:19:23	224.22	224.00	225.39	224.61	166.18	196.14	155.51	45.56	36618.71	43700.82	35081.23	115232.59	37267.02	43825.93	35306.22	116487.71	-6593.26	-4899.41	-3762.67	-15242.79	-0.
21	01/07/2022	0:20:23	224.68	224.54	225.82	224.99	166.02	190.75	155.69	42.16	36539.33	42453.22	34847.48	114011.99	37197.39	42757.80	35205.80	115718.78	-6578.09	-5159.70	-3759.34	-15600.35	-0.
22	01/07/2022	0:21:23	224.56	224.44	225.77	224.96	169.18	190.90	156.31	42.17	37414.29	42608.99	35008.81	115091.54	37979.55	43029.40	35401.80	116256.02	-6382.23	-5155.17	-3730.81	-15331.59	-0.
23	01/07/2022	0:22:23	224.73	224.59	225.97	225.21	169.24	190.32	155.54	41.71	37460.95	42464.88	34895.35	114780.44	38198.04	42819.37	35043.38	116499.25	-6313.49	-5015.18	-3695.84	-15114.57	-0.
24	01/07/2022	0:23:23	225.09	224.99	226.37	225.39	159.77	186.09	148.84	43.37	35157.23	41648.72	33180.20	109866.67	35895.86	41762.92	33465.30	110932.45	-7066.99	-5436.18	-4463.27	-16885.43	-0.
25	01/07/2022	0:24:23	225.10	224.91	226.17	225.40	162.73	178.64	147.59	38.34	34902.90	39882.81	33038.32	107702.58	35492.43	40004.85	33481.70	109069.89	-7174.90	-5116.17	-4542.07	-16856.29	-0.
26	01/07/2022	0:25:23	225.12	224.97	226.21	225.40	159.70	181.14	147.08	40.69	35167.79	40317.48	33069.49	108614.72	35819.39	40628.00	33231.11	109448.49	-7163.34	-5079.75	-4579.44	-16896.30	-0.
27	01/07/2022	0:26:23	225.41	225.11	226.55	225.76	160.61	181.66	146.54	41.28	35396.77	40601.48	32792.41	108621.23	36000.40	40813.10	33081.00	109709.25	-7177.06	-5029.68	-4629.52	-16729.34	-0.
28	01/07/2022	0:27:23	225.30	225.11	226.46	225.60	159.69	183.24	146.40	42.23	35103.98	41111.55	32871.09	109021.38	35947.10	41318.57	33120.90	110596.45	-7302.08	-5033.57	-4465.61	-16838.62	-0.
29	01/07/2022	0:28:23	225.00	224.82	226.40	225.51	163.58	184.82	146.72	42.56	36031.36	41175.18	32840.79	109680.93	35881.29	41667.13	33265.00	110774.38	-7443.82	-5392.77	-4536.16	-17404.22	-0.
30	01/07/2022	0:29:23	225.05	224.77	226.12	225.31	160.14	181.88	148.99	39.93	35224.36	40573.53	33400.09	109164.73	35949.95	41114.34	33656.16	110673.19	-7358.17	-5443.88	-4593.24	-17373.76	-0.
31	01/07/2022	0:30:23	224.94	224.75	226.22	225.41	165.63	188.66	154.02	39.74	36732.58	42089.28	34432.47	113514.20	37412.30	42463.08	34864.00	114561.63	-6715.38	-4826.10	-3819.71	-15333.97	-0.
32	01/07/2022	0:31:23	225.01	224.92	226.08	225.26	162.59	182.33	147.45	40.81	35922.62	40660.10	32985.20	109538.80	36630.93	40916.44	33200.36	111861.45	-7438.88	-5508.71	-4487.83	-17408.95	-0.
33	01/07/2022	0:32:23	225.11	225.10	226.33	225.59	163.79	179.29	146.53	42.16	36091.68	40180.85	32649.18	108876.21	36829.15	40396.02	33153.39	111690.12	-7109.31	-5899.29	-4755.18	-17777.09	-0.
34	01/07/2022	0:33:23	225.00	225.02	226.24	225.45	165.00	180.78	146.41	43.35	35437.66	40215.55	32903.88	108564.98	36228.09	40605.57	33225.88	110885.42	-7124.33	-5825.92	-4607.14	-17578.07	-0.
35	01/07/2022	0:34:23	225.23	225.09	226.33	225.56	161.27	188.23	145.10	45.06	35430.70	41967.00	32559.48	109926.90	36072.49	42152.05	33065.08	112561.83	-7292.84	-5794.94	-4590.45	-17723.46	-0.
36	01/07/2022	0:35:23	225.07	224.92	226.18	225.29	170.01	191.09	150.11	45.71	36451.17	42574.69	34210.79	113972.92	37127.43	43180.24	34539.97	116180.46	-6440.60	-5253.03	-3875.72	-15613.09	-0.
37	01/07/2022	0:36:23	225.12	225.06	226.37	225.56	165.68	192.28	153.12	44.67	36670.53	42964.28	34483.06	114632.23	37302.56	43531.89	34887.09	116693.49	-6444.61	-5217.40	-3881.72	-15544.23	-0.
38	01/07/2022	0:37:23	225.30	225.19	226.81	225.79	166.07	200.13	152.13	47.60	36679.27	44466.27	34249.53	115712.30	37444.13	44661.44	34617.57	116560.53	-6494.46	-5093.56	-3810.34	-15434.29	-0.
39	01/07/2022	0:38:23	225.59	225.47	226.75	225.88	162.45	188.28	154.57	41.98	36066.85	42243.80	34863.23	113355.97	36770.79	42534.71	35128.69	113887.12	-6134.94	-5390.06	-3893.87	-15399.04	-0.
40	01/07/2022	0:39:23	225.46	225.20	226.68	225.80	160.78	191.81	154.00	44.34	35744.54	42757.34	34653.41	113919.81	36793.04	43366.41	34856.64	114643.98	-6221.38	-5242.42	-3947.56	-15354.03	-0.
41	01/07/2022	0:40:23	225.49	225.24	226.72	225.86	161.90	195.69	153.97	46.22	36014.06	43860.65	34821.88	115638.46	36708.32	44102.26	34869.05	115621.75	-6265.60	-5257.97	-3928.55	-15432.61	-0.
42	01/07/2022	0:41:23	225.77	225.62	227.11	226.19	163.19	189.52	154.10	41.30	36333.54	42397.80	34871.17	113844.22	36946.93	42688.75	35066.68	114674.46	-6175.36	-4854.74	-3869.34	-14948.74	-0.

Ilustración o : Archivo propuesto 01/07/2022, 24h, muestreo/min, en formato texto .csv

7.2 Código de lectura de .csv

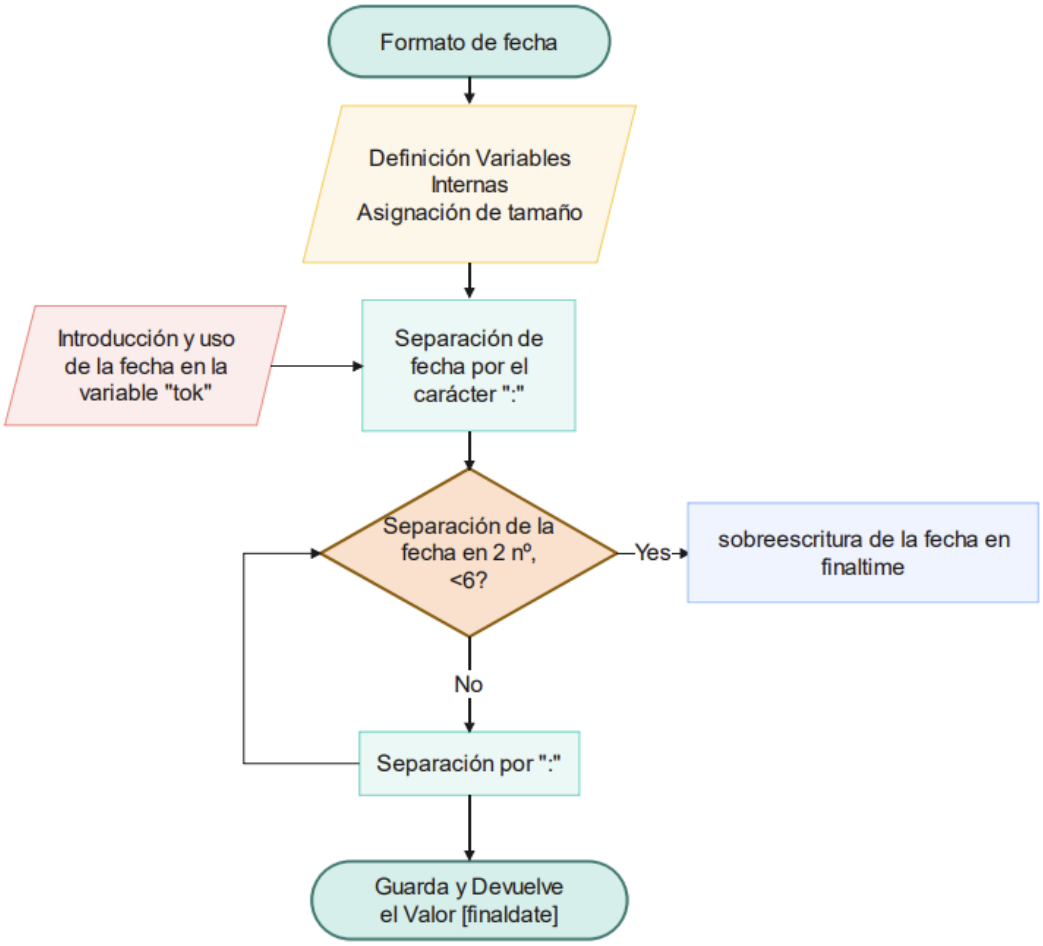
Composición de la marca de tiempo

```
char *dateformat(char *date)
{
    const char *tok;
    char finaldate[11];
    // printf("dateformat - la finaldate empieza siendo: %s", finaldate);
    for (tok = strtok(date, "/");
         tok && *tok;
         tok = strtok(NULL, "/\n"))
    {
        // printf("\n dateformat - El token actual es: %s  \n", tok);
        if (strlen(tok) < 2)
        {
            // printf("\n dateformat - El token malo es: %s  \n", tok);
            char *newtok = malloc(20);
            strcpy(newtok, "0");
            strcat(newtok, tok);
            // printf("\n dateformat - El token bueno es: %s  \n", newtok);
            strcat(finaldate, "/");
            strcat(finaldate, newtok);
        }
        else if (strlen(tok) == 4)
        {
            strcat(finaldate, tok);
        }
        else
        {
            strcat(finaldate, "/");
            strcat(finaldate, tok);
        }
        // printf("\n dateformat - El finaldate va siendo: %s  \n", finaldate);
    }
    // printf("\n dateformat - La fecha final es: %s  \n", finaldate);
    return finaldate;
}

```


Formato de fecha
Definición Variables Internas
Asignación de tamaño
Introducción y uso de la fecha en la variable "tok"
Separación de fecha por el carácter "/"
Separación de la fecha en 2 nº, ==4?
Yes
sobreescritura de la fecha en finaldate
No
Separación por "/"
Guarda y Devuelve el Valor [finaldate]

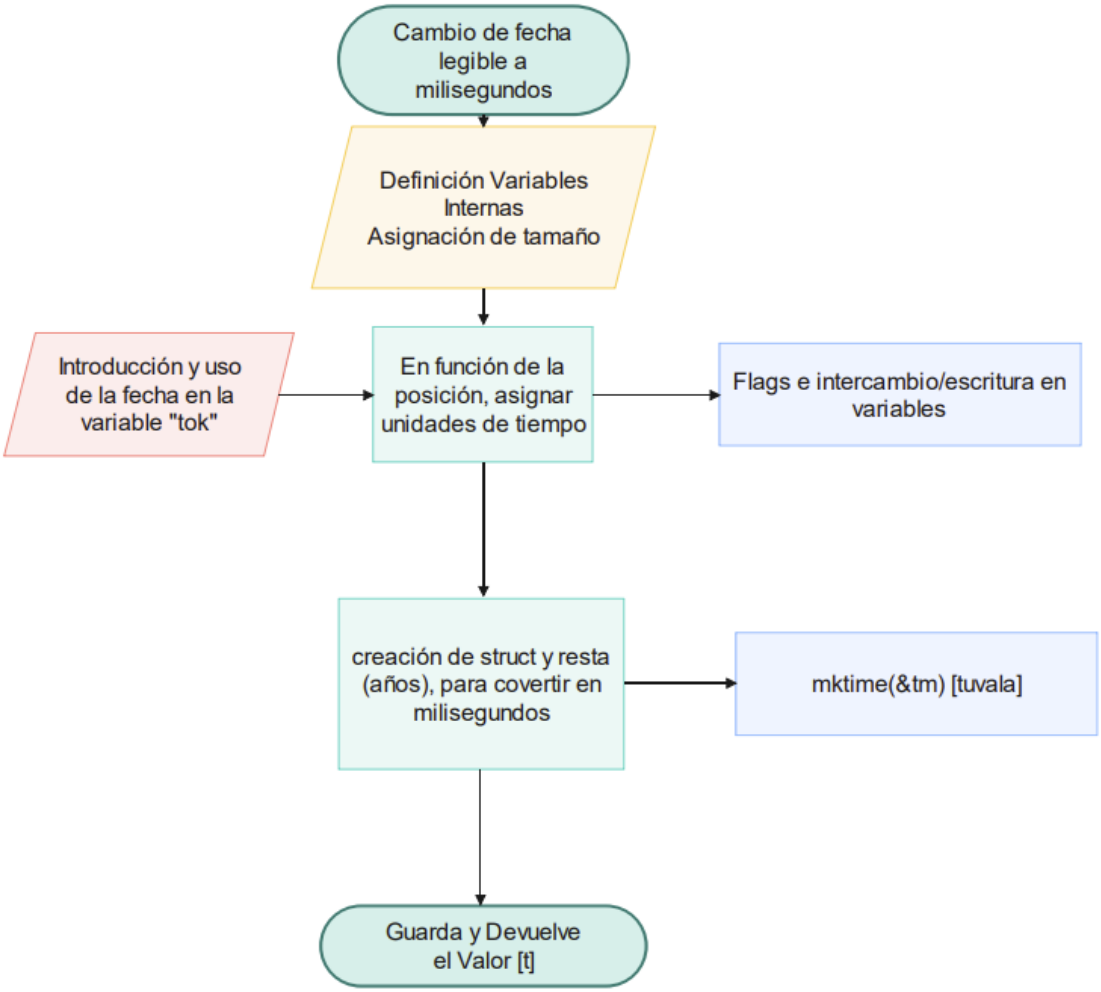
Marca de tiempo

```
44  char *timeformat(char *oldtime)
45  {
46      const char *tok;
47      char finaltime[11];
48      // printf("\n timeformat - la finaldate empieza siendo: %s", finaltime);
49      // printf("\n timeformat - la time empieza siendo: %s", oldtime);
50      for (tok = strtok(oldtime, ":");
51           tok && *tok;
52           tok = strtok(NULL, ":\n"))
53      {
54          // printf("\n timeformat - El token actual es: %s  \n", tok);
55          if (strlen(tok) < 2)
56          {
57              // printf("\n timeformat - El token malo es: %s  \n", tok);
58              char *newtok = malloc(20);
59              strcpy(newtok, "0");
60              strcat(newtok, tok);
61              // printf("\n timeformat - El token bueno es: %s  \n", newtok);
62              strcat(finaltime, newtok);
63          }
64          else if (strlen(tok) == 2)
65          {
66              strcat(finaltime, tok);
67          }
68          if (strlen(finaltime) < 6)
69          {
70              strcat(finaltime, ":");
71          }
72
73          // printf("\n timeformat - El finaltime va siendo: %s  \n", finaltime);
74      }
75      // printf("\n timeformat - La hora final es: %s  \n", finaltime);
76      return finaltime;
77  }
```


Formato de fecha
Definición Variables Internas
Asignación de tamaño
Introducción y uso de la fecha en la variable "tok"
Separación de fecha por el carácter ":"
Separación de la fecha en 2 nº, <6?
Yes
sobreescritura de la fecha en finaltime
No
Separación por ":"
Guarda y Devuelve el Valor [finaldate]

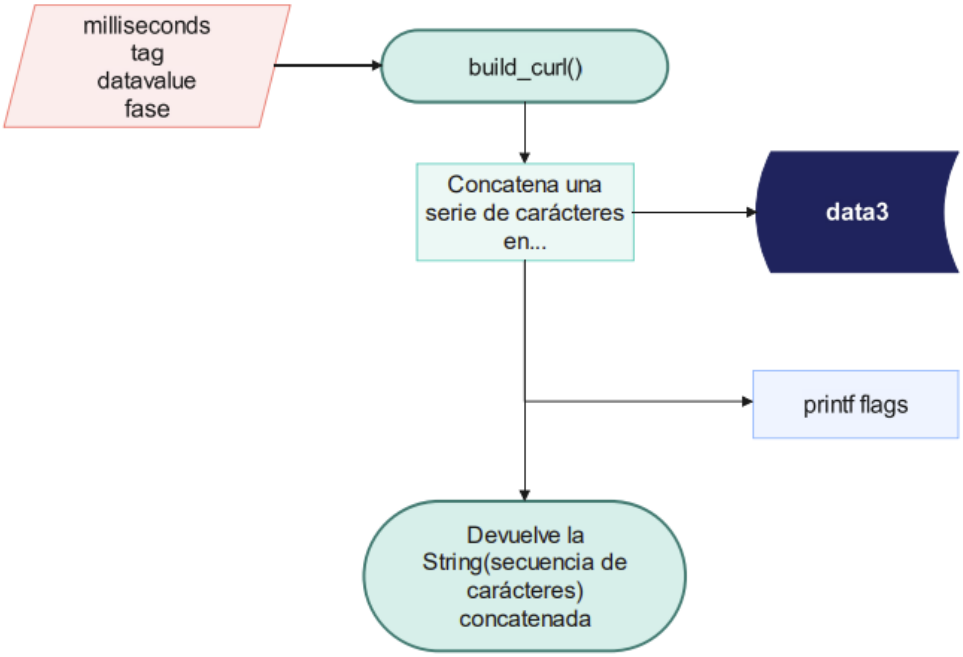
Conversión de la marca de tiempo a ms

```
int datetomilis(char *date, char *time)
{
    // printf("\n datetomilis - La fechaasdadasdas es: %s  \n", date);
    int milis[13];
    char day[3];
    char month[3];
    char year[5];
    char hour[3];
    char minute[3];
    char second[3];
    char milisecond[4] = "000";

    strncpy(day, date + 8, 2);
    day[2] = 0;
    // printf("datetomilis - day: %s \n", day);
    strncpy(month, date + 5, 2);
    month[2] = 0;
    // printf("datetomilis - month: %s \n", month);
    strncpy(year, date, 4);
    year[4] = 0;
    // printf("datetomilis - year: %s --- %s \n", year, date);
    strncpy(hour, time, 2);
    hour[2] = 0;
    strncpy(minute, time + 3, 2);
    minute[2] = 0;
    strncpy(second, time + 6, 2);
    second[2] = 0;
    // strncpy(milisecond, time + 9, 3);
    milisecond[3] = 0;

    int d = atoi(day);
    // printf("datetomilis - day: %d \n", d);
    int m = atoi(month);
    // printf("datetomilis - month: %d \n", m);
    int y = atoi(year);
    // printf("datetomilis - year: %d \n", y);
    int h = atoi(hour);
    // printf("datetomilis - hour: %d \n", h);
    int min = atoi(minute);
    // printf("datetomilis - minute: %d \n", min);
    int s = atoi(second);
    // printf("datetomilis - second: %d \n", s);
    int ms = atoi(milisecond);
    // printf("datetomilis - milisecond: %d \n", ms);

    struct tm tm;
    tm.tm_year = y - 1900;
    tm.tm_mon = m - 1;
    tm.tm_mday = d;
    tm.tm_hour = h;
    tm.tm_min = min;
    tm.tm_sec = s;
    tm.tm_isdst = -1;
    time_t t = mktime(&tm);
    // printf("\n datetomilis - los milis son: %ld \n", t);
    return t;
}
```


Cambio de fecha legible a milisegundos
Definición Variables Internas
Asignación de tamaño
Introducción y uso de la fecha en la variable "tok"
En función de la posición, asignar unidades de tiempo
Flags e intercambio/escritura en variables
creación de struct y resta (años), para covertir en milisegundos
mktime(&tm) [tuvala]
Guarda y Devuelve el Valor [t]

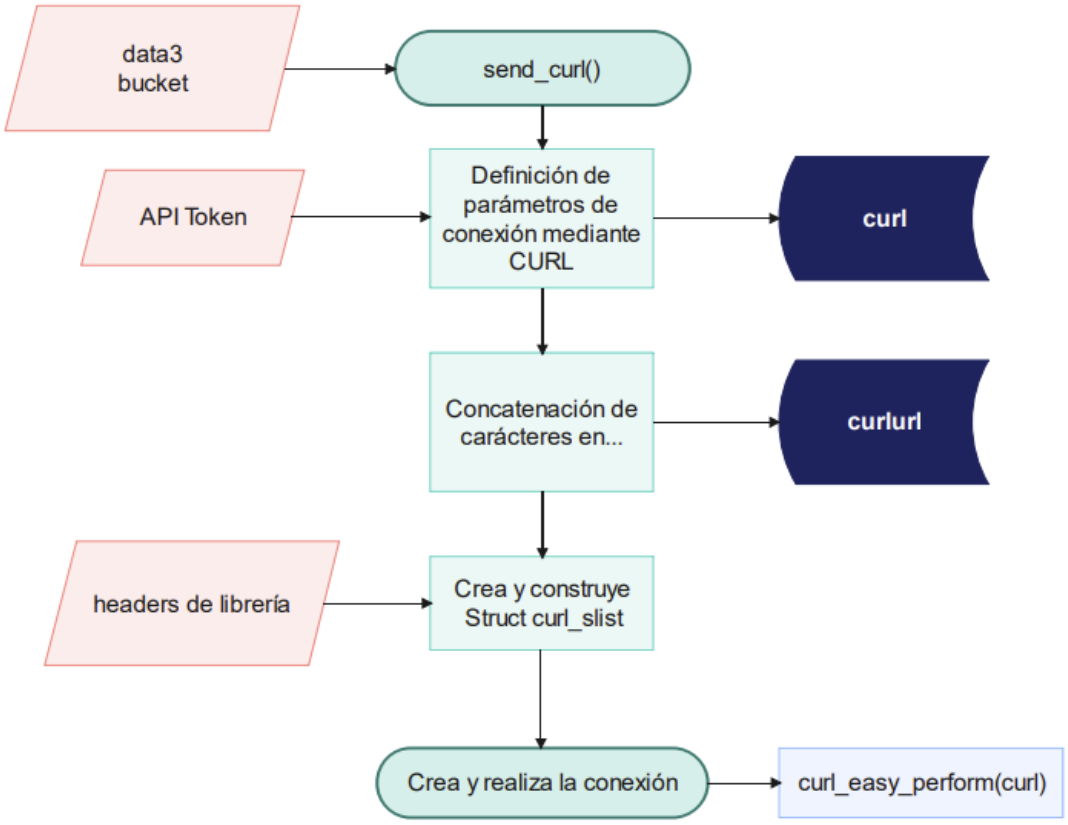
Creación de la cadena de datos para enviar al cURL

```
144 char *buildinfluxdata(long double milis, char *tag, char *datavalue, char *fase)
145 {
146     char data3[1024];
147     char millisstr[13];
148     ltoa(milis, millisstr, 10);
149 
150     strcpy(data3, millisstr);
151     strcat(data3, "000");
152     strcpy(data3, tag);
153     strcat(data3, ",");
154     strcat(data3, "fase=");
155     strcat(data3, fase);
156     strcat(data3, ",");
157     strcat(data3, "source=UPV");
158     strcat(data3, " value=");
159     strcat(data3, datavalue);
160     strcat(data3, " ");
161     strcat(data3, millisstr);
162     strcat(data3, "000");
163     return data3;
164 }
```


milliseconds
tag
datavalue
fase
build_curl()
Concatena una serie de carácteres en...
data3
printf flags
Devuelve la String(secuencia de carácteres) concatenada

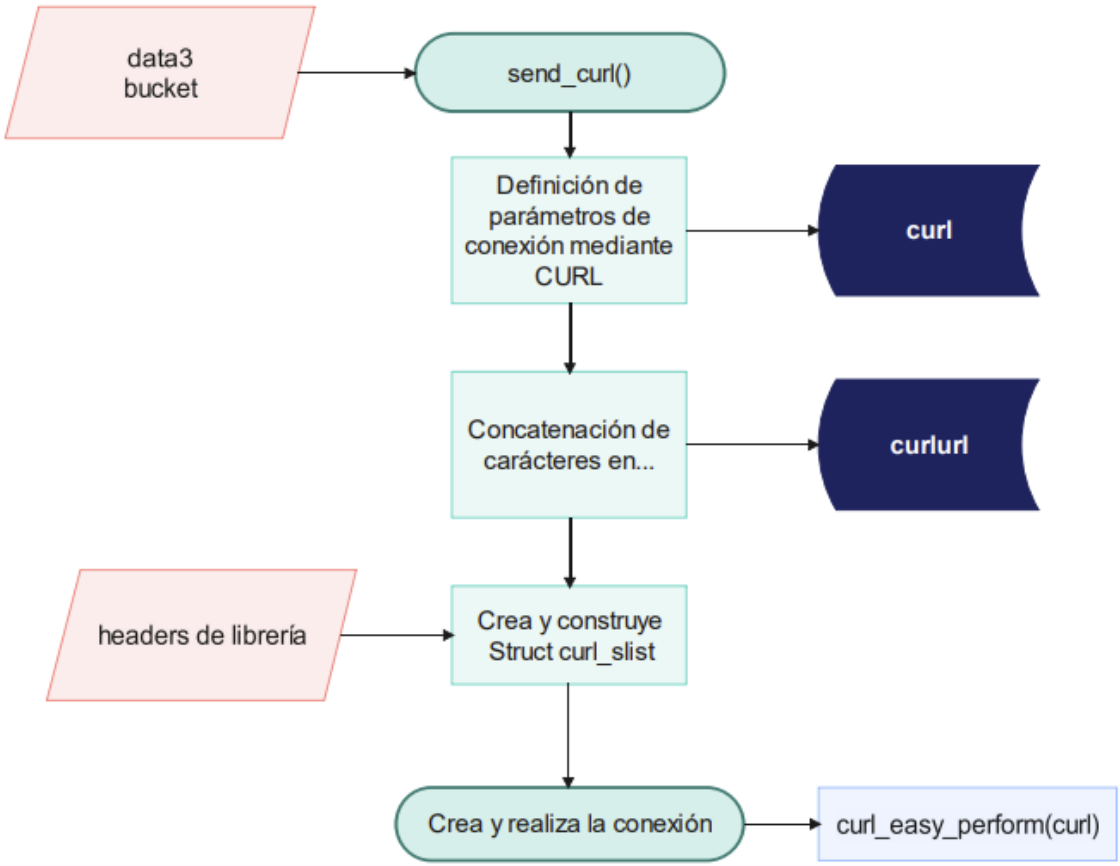
Construcción del CUrl

```
*/
void postdatatoinflux(long double milis, char *tag, char *datavalue, char *fase)
{
    // printf("Dentro de post data to influx \n");
    CURL *curl;
    CURLcode res;
    curl = curl_easy_init();
    // printf("Curl creado \n");

    if (curl)
    {
        // printf("Dentro del if \n");

        // printf("\n postdatatoinflux - La dato vf1 es: %s  \n", datavalue);
        // printf("\n postdatatoinflux - La fecha es: %ld  \n", milis);
        curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "POST");
        curl_easy_setopt(curl, CURLOPT_URL, "http://localhost:8086/api/v2/write?org=UPV&bucket=Voltios&precision=ms");
        curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
        curl_easy_setopt(curl, CURLOPT_DEFAULT_PROTOCOL, "https");
        struct curl_slist *headers = NULL;
        headers = curl_slist_append(headers, "Authorization: Token X1-OQR14HTCJPva3KIJB8zd7ov13wFBkV15flwLr1RpnN6I-gx-uk_hwUXIPfkUSNruHGKEcEcbVKpPqS
        headers = curl_slist_append(headers, "Content-Type: text/plain");
        curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
        // const char *data = "voltios,fase=1,source=UPV value=2122 1663446248000";
        // char data2[64] = "voltios,fase=1,source=UPV";
        char data2[64];
        strcpy(data2, tag);
        strcat(data2, ",");
        strcat(data2, "fase=");
        strcat(data2, fase);
        strcat(data2, ",");
        strcat(data2, "source=UPV");
        strcat(data2, " value=");
        strcat(data2, datavalue);
        strcat(data2, " ");

        char millisstr[13];
        ltoa(milis, millisstr, 10);
        // printf("\n postdatatoinflux - los millisstr son: %s \n", millisstr);

        strcat(data2, millisstr);
        strcat(data2, "000");

        // printf("\n postdatatoinflux - data2 es: %s \n", data2);

        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data2);
        // printf("\n set de datos");

        res = curl_easy_perform(curl);
        // printf("\n curl llamado");
    }
    else
    {
        printf("\nError en curl");
    }
    curl_easy_cleanup(curl);
    // printf("\n curl limpio");
}
```


data3
bucket
send_curl()
API Token
Definición de parámetros de conexión mediante CURL
curl
Concatenación de carácteres en...
curlurl
headers de librería
Crea y construye Struct curl_slist
Crea y realiza la conexión
curl_easy_perform(curl)

Envío del cURL

```
232 void postdatatoinflux2(char *data5, char *bucket)
233 {
234     // printf("Dentro de post data to influx \n");
235     CURL *curl;
236     CURLcode res;
237     curl = curl_easy_init();
238     // printf("Curl creado \n");
239 
240     if (curl)
241     {
242         // printf("Dentro del if \n");
243 
244         // printf("\n postdatatoinflux - La dato vf1 es: %s  \n", datavalue);
245         // printf("\n postdatatoinflux - La fecha es: %ld  \n", milis);
246         curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "POST");
247         //curl_easy_setopt(curl, CURLOPT_URL, "http://localhost:8086/api/v2/write?org=UPV&bucket=Voltios&precision=ms");
248         char *curlurl = malloc(100);
249         memset(curlurl,0,100);
250         strcat(curlurl,"http://localhost:8086/api/v2/write?org=UPV&bucket=");
251         strcat(curlurl,bucket);
252         strcat(curlurl,"&precision=ms");
253         curl_easy_setopt(curl, CURLOPT_URL, curlurl);
254         curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
255         curl_easy_setopt(curl, CURLOPT_DEFAULT_PROTOCOL, "https");
256         struct curl_slist *headers = NULL;
257         headers = curl_slist_append(headers, "Authorization: Token X1-OQR14HTCJPva3KIJ88zd7ov13wFBkV15flwLr1RpnN6I-gx-uk_hwUXIPfkUSNruHGKEcEcbVKpPqSWhEUw==");
258         headers = curl_slist_append(headers, "Content-Type: text/plain");
259         curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
260         // const char *data = "voltios,fase=1,source=UPV value=2122 1663446248000";
261         // char data2[64] = "voltios,fase=1,source=UPV";
262 
263         // printf("\n postdatatoinflux - data2 es: %s \n", data2);
264 
265         curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data5);
266         // printf("\n set de datos");
267 
268         res = curl_easy_perform(curl);
269         free(curlurl);
270         // printf("\n curl llamado");
271     }
272     else
273     {
274         printf("\nError en curl");
275     }
276     curl_easy_cleanup(curl);
277     // printf("\n curl limpio");
278 }
```

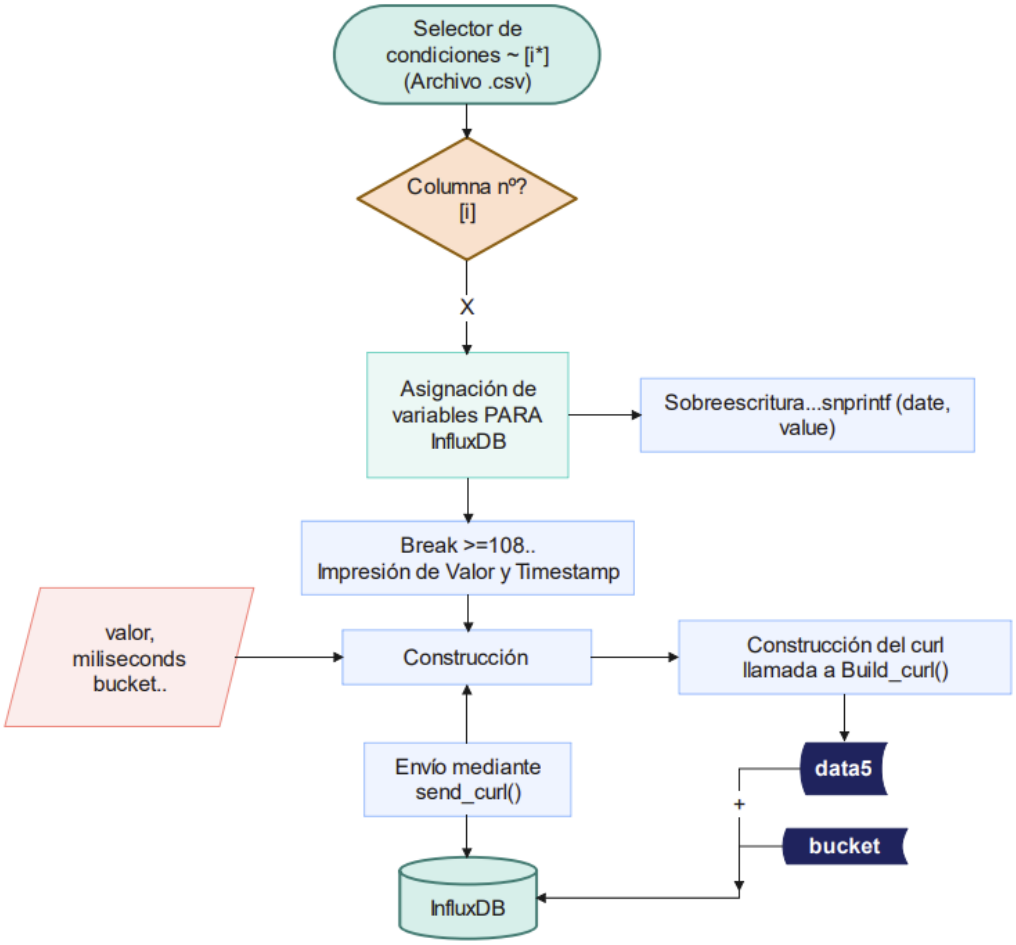

data3
bucket
send_curl()
Definición de parámetros de conexión mediante CURL
curl
Concatenación de carácteres en...
curlurl
headers de librería
Crea y construye Struct curl_slist
Crea y realiza la conexión
curl_easy_perform(curl)

Alojamiento de memoria por variables

```
280 int main()
281 {
282     printf("HOLA ME ESTOY EJECUTANDO");
283     time_t ahorita;
284     time(&ahorita);
285     printf("La hora es: %s", ctime(&ahorita));
286     // Substitute the full file path
287     // for the string file_path
288     FILE *fp = fopen("./test.csv", "r");
289 
290     if (!fp)
291         printf("Can't open file\n");
292 
293     else
294     {
295         // Here we have taken size of
296         // array 1024 you can modify it
297         char buffer[1024];
298 
299         int row = 0;
300         int column = 0;
301         char testdata2[20000];
302         char *voltf1data = malloc(80000);
303         char *voltf2data = malloc(80000);
304         char *voltf3data = malloc(80000);
305         char *voltsumdata = malloc(80000);
306         char *corrientef1data = malloc(80000);
307         char *corrientef2data = malloc(80000);
308         char *corrientef3data = malloc(80000);
309         char *corrientendata = malloc(80000);
310         char *ptrealf1data = malloc(100000);
311         char *ptrealf2data = malloc(100000);
312         char *ptrealf3data = malloc(100000);
313         char *ptrealsdata = malloc(100000);
314         char *ptaparentef1data = malloc(120000);
315         char *ptaparentef2data = malloc(120000);
316         char *ptaparentef3data = malloc(120000);
317         //strcpy(voltf1data, "");
318         strcpy(testdata2, "");
319         memset(voltf1data,0, 80000);
320         memset(voltf2data,0, 80000);
321         memset(voltf3data,0, 80000);
322         memset(voltsumdata,0, 80000);
323         memset(corrientef1data,0, 80000);
324         memset(corrientef2data,0, 80000);
325         memset(corrientef3data,0, 80000);
326         memset(corrientendata,0, 80000);
327         memset(ptrealf1data,0, 100000);
328         memset(ptrealf2data,0, 100000);
329         memset(ptrealf3data,0, 100000);
330         memset(ptrealsdata,0, 100000);
331         memset(ptaparentef1data,0, 120000);
332         memset(ptaparentef2data,0, 120000);
333         memset(ptaparentef3data,0, 120000);
```

Árbol de Condiciones

```
364         // Splitting the data
365         char *value = strtok(buffer, ";");
366 
367         while (value)
368         {
369             // Column 1
370             if (column == 0)
371             {
372                 //printf("Fecha: %s", value);
373                 strcpy(date, value);
374             }
375 
376             // Column 2
377             if (column == 1)
378             {
379                 //printf("\t Hora: %s", value);
380                 strcpy(time, value);
381             }
382 
383             // Column 3
384             if (column == 2)
385             {
386                 //printf("\t Voltios F1: %s", value);
387                 strcpy(voltf1, value);
388             }
389 
390             // Column 4
391             if (column == 3)
392             {
393                 //printf("\t Voltios F2: %s", value);
394                 strcpy(voltf2, value);
395             }
396 
397             // Column 5
398             if (column == 4)
399             {
400                 //printf("\t Voltios F3: %s", value);
401                 strcpy(voltf3, value);
402             }
403 
404             // Column 6
405             if (column == 5)
406             {
407                 //printf("\t Voltios SUM: %s", value);
408                 strcpy(voltsum, value);
409             }
410 
411             // Column 7
412             if (column == 6)
413             {
414                 //printf("\t Corriente F1: %s", value);
415                 strcpy(corriente1, value);
416             }
417 
418             // Column 8
419             if (column == 7)
420             {
421                 //printf("\t Corriente F2: %s", value);
422                 strcpy(corriente2, value);
423             }
424 
```


Selector de condiciones ~ [i*] (Archivo .csv)
Columna nº? [i]
X
Asignación de variables PARA InfluxDB
Sobreescritura...snprintf (date, value)
Break >=108.. Impresión de Valor y Timestamp
valor, miliseconds bucket..
Construcción
Construcción del curl llamada a Build_curl()
Envío mediante send_curl()
data5
+
bucket
InfluxDB

7.3 Código de ejecución en tiempo Real

Librerías necesarias, ejecución en Linux

C: > Users > Kdt_T > Desktop > WiP > C Gold_1.c > hora_local()

```c
#include <stdint.h>
#include <stdio.h>
#include <modbus.h>
#include <errno.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <stdlib.h>
#include "curl/curl.h"
```

STDINT.h

Es un archivo de encabezado en la biblioteca estándar de C introducido en la sección 7.18 de la biblioteca estándar de C99 para permitir a los programadores escribir código más portátil al proporcionar un conjunto de typedefs que especifican tipos enteros de ancho exacto, junto con los valores permisibles mínimos y máximos definidos para cada tipo, utilizando macros.

STDIO.h

La palabra clave stdio.h en C es un archivo de encabezado. La razón por la que usamos stdio.h en C es que este archivo de encabezado importa diferentes variables, macros y funciones para realizar operaciones de entrada y salida. Para realizar operaciones de entrada y salida en nuestro programa C, necesitamos importar el archivo de encabezado stdio.h a nuestro programa.

MODBUS.h

Esta biblioteca permite que el microcontrolador (Raspberry PI) se comunique a través del protocolo Modbus. El Modbus es un protocolo maestro-esclavo utilizado en la automatización industrial y se puede utilizar en otras áreas, como la domótica. El Modbus generalmente usa serial RS-232 o RS-485 como capa física (entonces llamado Modbus Serial) y TCP/IP vía Ethernet o WiFi (Modbus IP). En la versión actual, la biblioteca permite que la Raspberry funcione como un esclavo, compatible con Modbus Serial y Modbus sobre IP.

ERRNO.h

errno.h es un archivo de encabezado en la biblioteca estándar del lenguaje de programación C. Define macros para informar y recuperar condiciones de error utilizando el símbolo errno (abreviatura de "número de error").

errno actúa como una variable entera. Ciertas funciones de biblioteca almacenan un valor (el número de error) en errno cuando detectan errores. Al inicio del programa, el valor almacenado es cero. Las funciones de biblioteca almacenan solo valores mayores que cero. Cualquier función de biblioteca puede alterar el valor almacenado antes de la devolución, ya sea que detecten errores o no. La mayoría de las funciones indican que detectaron un error al devolver un valor especial, normalmente NULL para funciones que devuelven punteros y -1 para funciones que devuelven números enteros. Algunas funciones requieren que la persona que llama establezca errno en cero y luego lo pruebe para ver si se detectó un error.

La macro errno se expande a un lvalue con tipo int, a veces con los especificadores de tipo extern y/o volatile dependiendo de la plataforma.Originalmente, esta era una ubicación de memoria estática, pero las macros casi siempre se usan hoy en día para permitir subprocesos múltiples, de modo que cada subproceso vea su propio número de error local de subproceso.

El archivo de encabezado también define macros que se expanden a constantes enteras que representan los códigos de error

STRING.h

string.h es un archivo de la Biblioteca estándar del lenguaje de programación C que contiene la definición de macros, constantes, funciones y tipos y algunas operaciones de manipulación de memoria.

TIME.h

time.h relacionado con formato de hora y fecha es un archivo de cabecera de la biblioteca estándar del lenguaje de programación C que contiene funciones para manipular y formatear la fecha y hora del sistema.

UNISTD.h

En los lenguajes de programación C y C++, unistd.h es el nombre del archivo de encabezado que brinda acceso a la API del sistema operativo POSIX. Está definido por el estándar POSIX.1, la base de la especificación única de Unix y, por lo tanto, debería estar disponible en cualquier sistema operativo y compilador compatible con POSIX. Por ejemplo, esto incluye sistemas operativos Unix y similares a Unix, como variantes de GNU, distribuciones de Linux y BSD y macOS, y compiladores como GCC y LLVM.

STDLIB.h

stdlib.h es el encabezado de la biblioteca estándar de propósito general del lenguaje de programación C que incluye funciones relacionadas con la asignación de memoria, control de procesos, conversiones y otras. Es compatible con C++ y se conoce como cstdlib en C++. El nombre "stdlib" significa "biblioteca estándar".

CURL.h

cURL, que significa URL del cliente, es una herramienta de línea de comandos que los desarrolladores usan para transferir datos hacia y desde un servidor. Básicamente, cURL te permite hablar con un servidor especificando la ubicación (en forma de URL) y los datos que deseas enviar. cURL admite varios protocolos diferentes, incluidos HTTP y HTTPS, y se ejecuta en casi todas las plataformas. Esto hace que cURL sea ideal para probar la comunicación desde casi cualquier dispositivo (siempre que tenga una línea de comandos y conectividad de red) desde un servidor local hasta la mayoría de los dispositivos perimetrales.

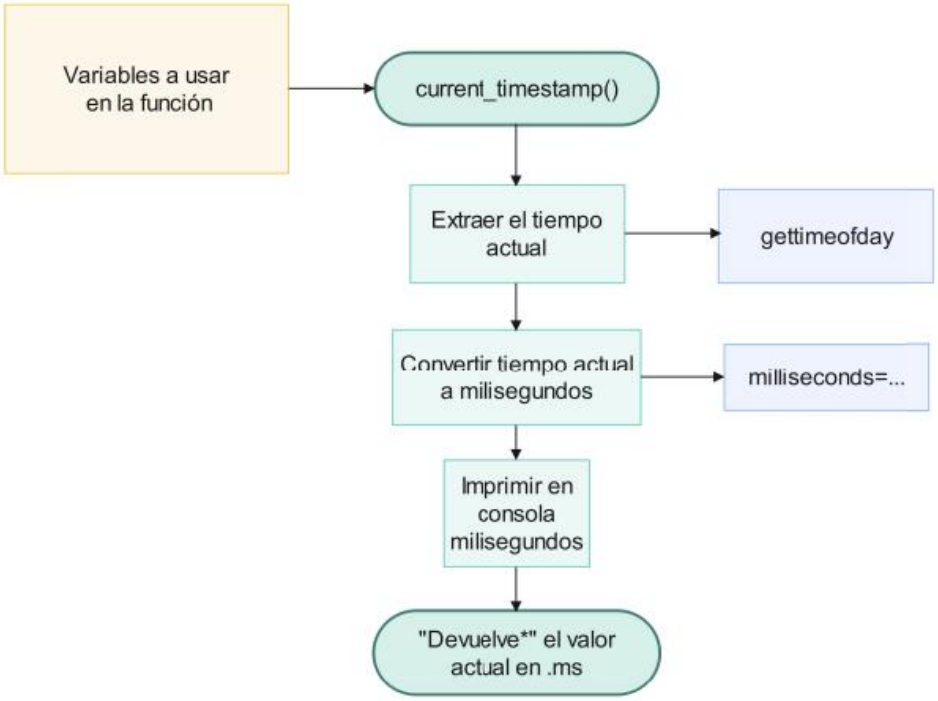
El comando más básico en curl es curl http://example.com. El comando curl va seguido de la URL, de la que nos gustaría recuperar algún tipo de datos. En este caso, devolvería la fuente html por ejemplo.com.

Detrás del comando curl se encuentra la biblioteca de desarrollo libcurl

Devolución de la marca de tiempo en ms

```
11
12  #define ltoa(long,str,len) snprintf(str,len,"%l",long)
13
14      char data3[2048];
15  char millis_str[256];
16        char envio;
17  float valor;
18  long long milliseconds;
19
20
21  long long current_timestamp() {
22      struct timeval te;
23      gettimeofday(&te, NULL); // get current time
24      long long milliseconds = te.tv_sec*1000LL + te.tv_usec/1000; // calculate milliseconds
25      printf("milliseconds: %lld\n", milliseconds);
26      return milliseconds;
27  }
28
```

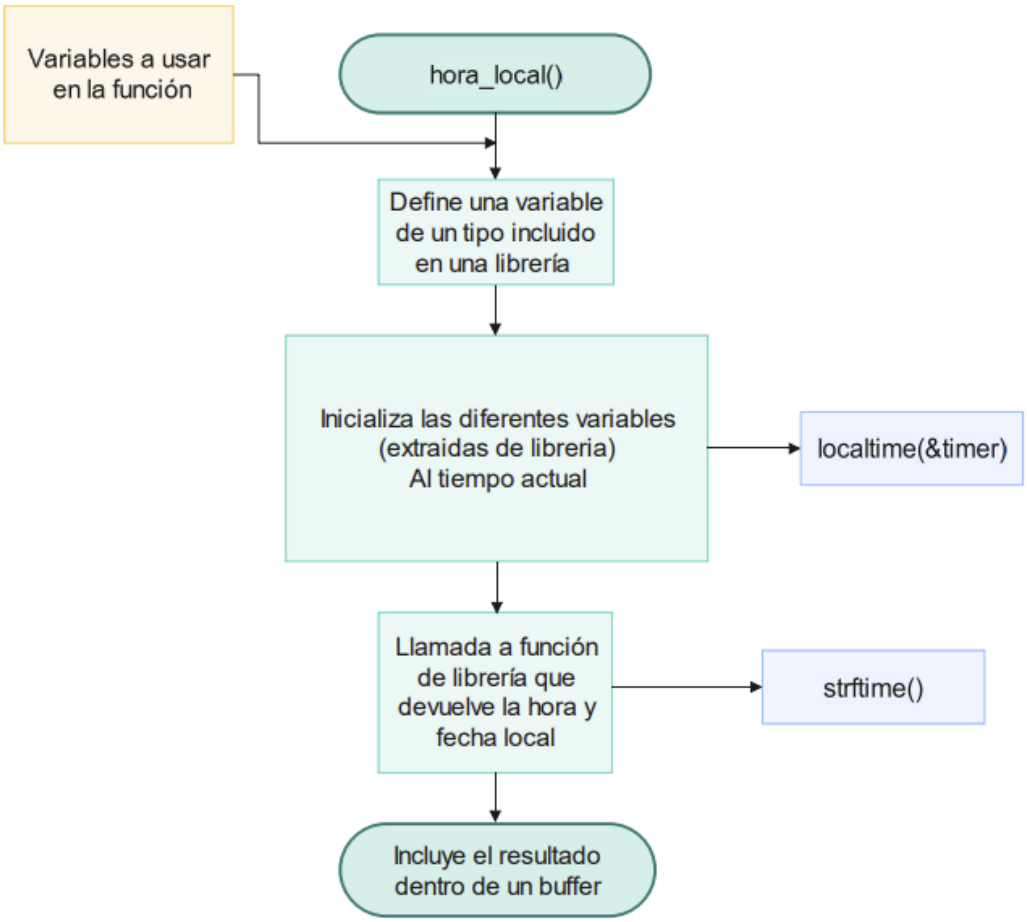
Este fragmento de código consulta el tiempo presente, y realiza la transformación necesaria para devolver la marca de tiempo en milisegundos, formato ineludible para enviar los datos en InfluxDB con marca de tiempo, de forma que cada valor enviado a la base de datos corresponda a una marca temporal de la serie, de otro modo, los datos serían sobrescritos constantemente.


Variables a usar en la función
current_timestamp()
Extraer el tiempo actual
gettimeofday
Convertir tiempo actual a milisegundos
milliseconds=...
Imprimir en consola milisegundos
"Devuelve*" el valor actual en .ms

Hora local en formato legible

```
29  void hora_local(){
30    time_t timer;
31      char buffer[26];
32      struct tm* tm_info;
33
34      timer = time(NULL);
35      tm_info = localtime(&timer);
36
37      strftime(buffer, 26, "%Y-%m-%d %H:%M:%S", tm_info);
38      puts(buffer);
39
40  //modbus_close(ctx);
41  //modbus_free(ctx);
42
43  }
```

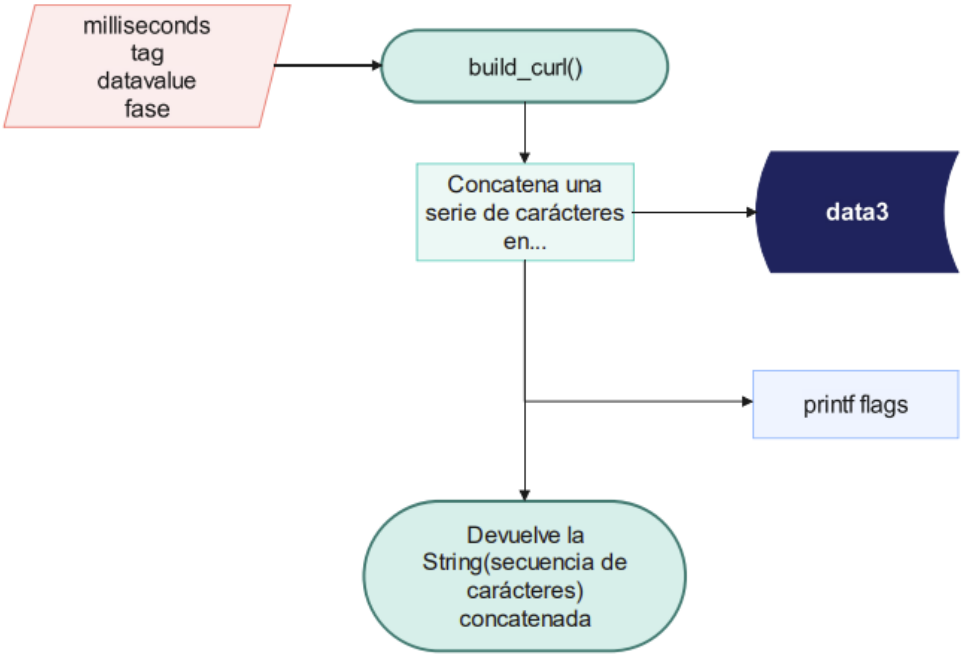
Este script devuelve la hora local del terminal en formato legible, este timer no se utiliza internamente en otros scripts, tan solo es para reflejar mediante printf el valor impreso a la hora indicada.


Variables a usar en la función
hora_local()
Define una variable de un tipo incluido en una librería
Inicializa las diferentes variables (extraidas de libreria) Al tiempo actual
localtime(&timer)
Llamada a función de librería que devuelve la hora y fecha local
strftime()
Incluye el resultado dentro de un buffer

Construcción del String del cURL

```
44
45  char *build_curl(long long milliseconds, char *tag, char *datavalue, char *fase){
46
47  //curl -i -H "Authorization: Token 7sys4OFsxhyes-desFydT0G2hgSbm_nZ3v3nkxA3rq1PmlOlmV2nD0noOdF7GjNxJh1
48
49
50  sprintf(millis_str, "%lld", milliseconds);
51  printf("%s\n", millis_str);
52
53  printf("0.0\n");
54      strcpy(data3, tag);
55  printf("1.0\n");
56      strcat(data3, ",");
57  printf("1.2\n");
58      strcat(data3, "fase=");
59      strcat(data3, fase);
60      strcat(data3, ",");
61      strcat(data3, "source=UPV");
62      strcat(data3, " value=");
63      strcat(data3, datavalue);
64      strcat(data3, " ");
65      strcat(data3, millis_str);
66  printf("data3: %s\n",data3);
67      return data3;
68  }
```

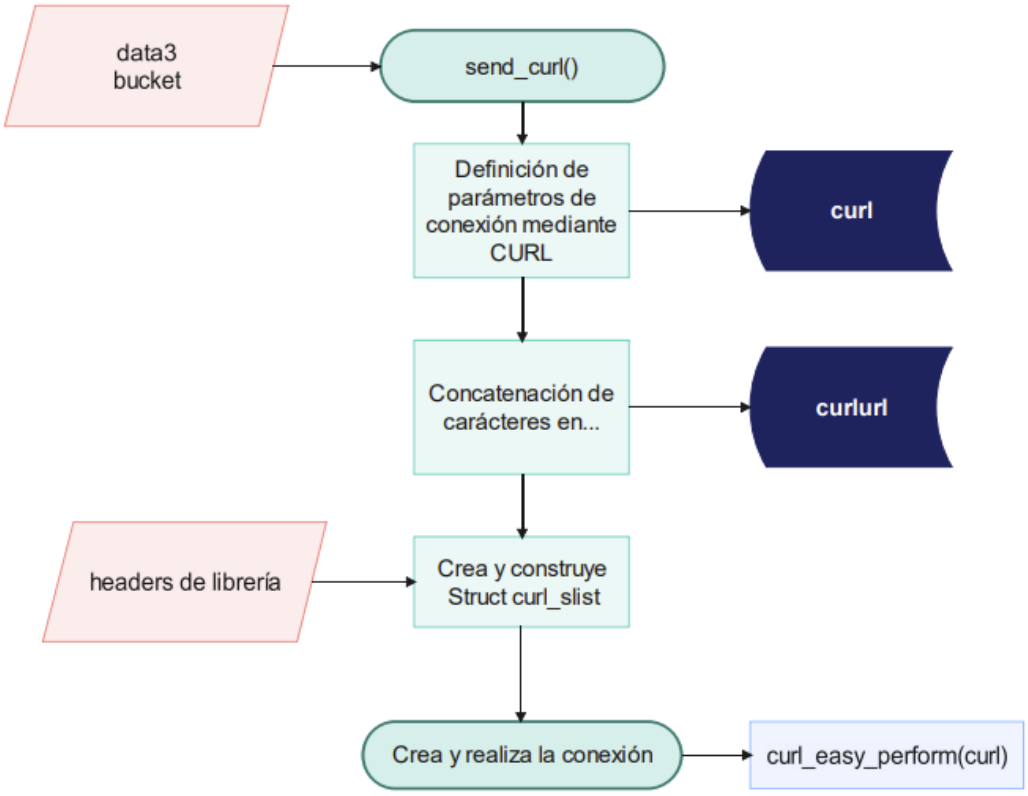
Este fragmento de código realiza una concatenación necesaria para construir el cURL de forma que el mismo funcione debidamente, con sus espacios, comas, y palabras clave que determinan cómo se construye la llamada a InfluxDB.


milliseconds
tag
datavalue
fase
build_curl()
Concatena una serie de carácteres en...
data3
printf flags
Devuelve la String(secuencia de carácteres) concatenada

Fragmento del Envio del cURL

```
69  void send_curl(char *data3, char *bucket){
70  {
71      // printf("Dentro de post data to influx \n");
72      CURL *curl;
73      CURLcode res;
74      curl = curl_easy_init();
75      // printf("Curl creado \n");
76
77      if (curl)
78      {
79          // printf("Dentro del if \n");
80
81          // printf("\n postdatatoinflux - La dato vf1 es: %s  \n", datavalue);
82          // printf("\n postdatatoinflux - La fecha es: %ld  \n", milis);
83          curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "POST");
84          //curl_easy_setopt(curl, CURLOPT_URL, "http://localhost:8086/api/v2/write?org=UPV&bucket=Volti
85          char *curlurl = malloc(100);
86          memset(curlurl,0,100);
87          strcat(curlurl,"http://localhost:8086/api/v2/write?org=UPV&bucket=");
88          strcat(curlurl,bucket);
89          strcat(curlurl,"&precision=ms");
90          curl_easy_setopt(curl, CURLOPT_URL, curlurl);
91          curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
92          curl_easy_setopt(curl, CURLOPT_DEFAULT_PROTOCOL, "https");
93          struct curl_slist *headers = NULL;
94          headers = curl_slist_append(headers, "Authorization: Token 7sys4OFsxhyes-desFydT0G2hgSbm_nZ3v
95          headers = curl_slist_append(headers, "Content-Type: text/plain");
96          curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
97          // const char *data = "voltios,fase=1,source=UPV value=2122 1663446248000";
98          // char data2[64] = "voltios,fase=1,source=UPV";
99
100         // printf("\n postdatatoinflux - data2 es: %s \n", data2);
101
102         curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data3);
103         // printf("\n set de datos");
104
105         res = curl_easy_perform(curl);
106         //free(curlurl);
107         // printf("\n curl llamado");
108     }
109     else
110     {
111         printf("\nError en curl");
112     }
113     curl_easy_cleanup(curl);
114     // printf("\n curl limpio");
115 }
116 }
```

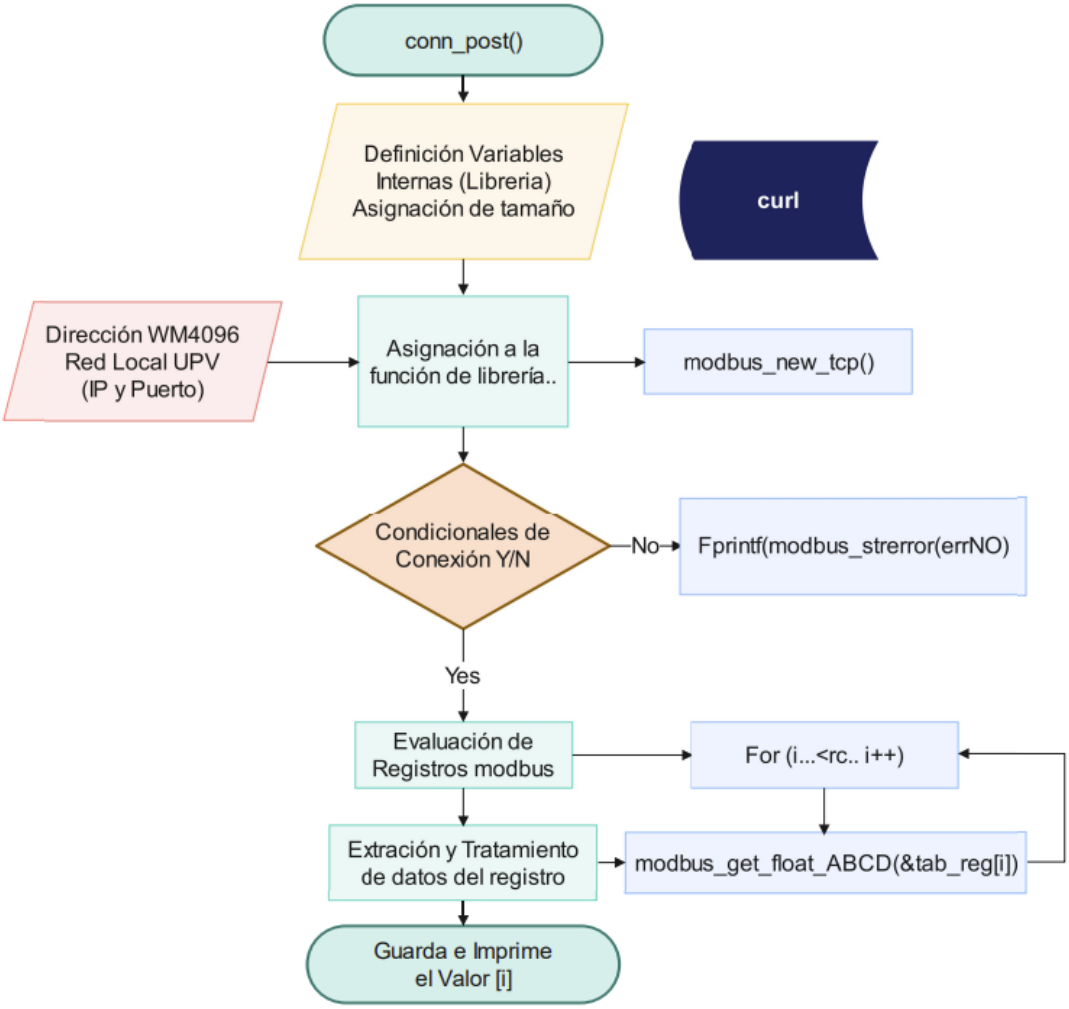
Una vez se ha construido el cURL completamente, en este fragmento de código, mediante el uso del API Token privado creado en InfluxDB, se realizan los envíos de datos a cada Bucket, que posteriormente se selecciona en el árbol de condiciones.


data3
bucket
send_curl()
Definición de parámetros de conexión mediante CURL
curl
Concatenación de carácteres en...
curlurl
headers de librería
Crea y construye Struct curl_slist
Crea y realiza la conexión
curl_easy_perform(curl)

Conexión mediante Modbus (Librería)

```
118
119 int conn_post (void){
120
121   modbus_t *ctx;
122   uint16_t tab_reg[2048];
123   int rc;
124   int i;
125   char src[20];
126   char src1[20];
127
128
129
130   ctx = modbus_new_tcp("158.42.160.141", 502);
131 if (modbus_connect(ctx) == -1) {
132     fprintf(stderr, "Connection failed: %s\n", modbus_strerror(errno));
133 //    modbus_free(ctx);
134     return -1;
135   }
136
137   rc = modbus_read_registers(ctx, 80, 70, tab_reg);
138 if (rc == -1) {
139     fprintf(stderr, "%s\n", modbus_strerror(errno));
140     return -1;
141   }
142
143 else {
144
145         printf("Ha conectado bien\n");
146   }
147
148 for (i=1; i < rc; i++) {
149     //printf("reg[%d]=%d (0x%X)\n", i, tab_reg[i], tab_reg[i]);
150
151  // printf("%hu\n", (unsigned int) tab_reg[i]);
152
153   float valor = modbus_get_float_abcd(&tab_reg[i]);
154
155
156   printf("valor[%d] = %f\n ",tab_reg[i], valor);
157   modbus_close(ctx);
158   //modbus_free(ctx);
159
160
161
```

Aquí se evalúa cada uno de los registros pautados mediante la librería MODBUS.h. Es notable que fue necesario una comunicación previa mediante otros software intermedios para dilucidar el tipo de formato de datos que el WM4096 registraba. Formato que podía encontrarse en el manual de uso de la casa distribuidora. Dicho formato es el Little endian byte swap, definido por el tipo de función modbus_get_float_abcd, Que facilita de forma automática la recomposición de los registros, no siendo tan sencillo el que dichos registros que contienen los datos sean los impares.


conn_post()
Definición Variables Internas (Libreria) Asignación de tamaño
curl
Dirección WM4096 Red Local UPV (IP y Puerto)
Asignación a la función de librería..
modbus_new_tcp()
Condicionales de Conexión Y/N
No
Fprintf(modbus_strerror(errNO)
Yes
Evaluación de Registros modbus
For (i...<rc.. i++)
Extración y Tratamiento de datos del registro
modbus_get_float_ABCD(&tab_reg[i])
Guarda e Imprime el Valor [i]

Alojamiento de memoria por variables

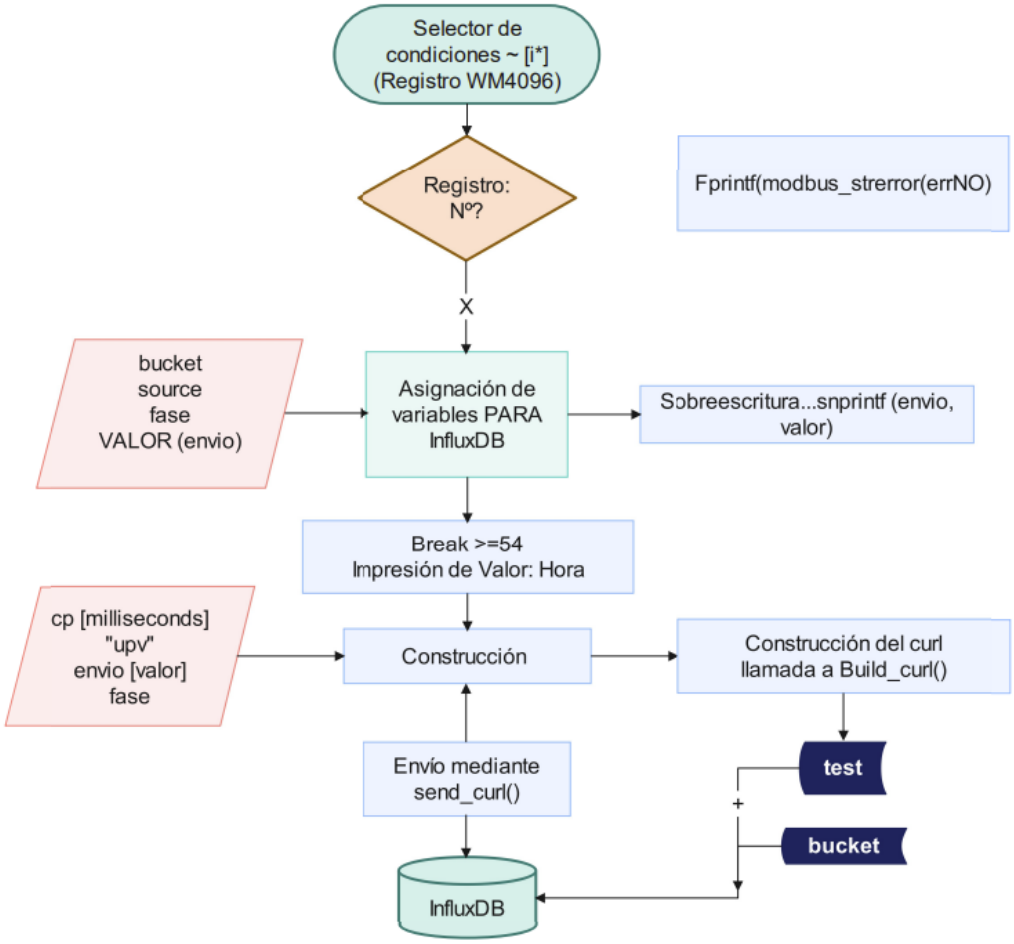
```
160
161
162   char *test = malloc(80000);
163   char *voltf1 = malloc(80000);
164   long long cp = current_timestamp();
165
166   char *bucket = malloc(80000);
167   char *source = malloc(80000);
168   char *fase;
169   char *neutro = malloc(80000);
170   char *envio = malloc(80000);
171   char *nuevas = malloc(80000);
172   char *FP = malloc(80000);
173   char N;
174   char *PReac = malloc(80000);
175   char *PApar = malloc(80000);
176   char *VAR = malloc(80000);
177   char *VA = malloc(80000);
178   char *Watt = malloc(80000);
179   char *PReal = malloc(80000);
180   char *amperios = malloc(80000);
181   char *Amp = malloc(80000);
182   char *voltios = malloc(80000);
183   char *Volt = malloc(80000);
184   char *coef = malloc(80000);
185   char *new = malloc(80000);
186
187   printf("0.0\n");
188   strcpy(test, "");
189   memset(test,0,80000);
190   memset(envio,0,80000);
191
192
```

A fin de evitar constantes problemas de pérdidas/fugas de memoria, impresiones y sobreescritura de caracteres no deseados ni reconocidos, se estableció un espacio de memoria reservado para cada variable.

Árbol selector de registro-datos

```
192
193  printf(" Entrando: Arbol de If\n ");
194
195
196
197                  if (i == 1)
198              {
199                      bucket = "Volt";
200  printf(" 1.0\n");
201                      source = voltios;
202  printf(" 1.1\n");
203                      fase = "1";
204  printf(" 1.2\n");
205                      snprintf(envio, sizeof(envio), "%f", valor);
206  printf(" 1.3 valor33=%f\n",valor);
207  printf(" 1.3 envio33=%f\n",envio);
208
209              }
210
211                  // Column 2
212                  if (i == 3)
213                  {
214  printf(" segundopost\n");
215                      bucket ="Volt";
216                      source = voltios;
217                      fase = "2";
218                      snprintf(envio, sizeof(envio), "%f", valor);
219
220                  }
221
222                  // Column 3
223                  if (i == 5)
224                  {
225                      bucket ="Volt";
226                      source = voltios;
227                      fase = "3";
228                      snprintf(envio, sizeof(envio), "%f", valor);
229
230                  }
231
232                  // Column 4
233                  if (i == 7)
234                  {
235                      bucket ="Volt";
236                      source = voltios;
237                      fase = "N";
```

En este árbol selector mediante condiciones, de 54 condiciones en total, se define el Bucket al que debe ir el valor, se define el origen de los datos, se define la fase, y finalmente se sobrescribe el valor sobre la variable envío, que será usada posteriormente.


Selector de condiciones ~ [i*] (Registro WM4096)
Registro: Nº?
Fprintf(modbus_strerror(errNO)
X
bucket
source
fase
VALOR (envio)
Asignación de variables PARA InfluxDB
Sobreescritura...snprintf (envio, valor)
Break >=54
Impresión de Valor: Hora
cp [milliseconds]
"upv"
envio [valor]
fase
Construcción
Construcción del curl llamada a Build_curl()
test
+
bucket
Envío mediante send_curl()
InfluxDB

Fin del árbol selector, funciones principales

```
476
477                 if (i >= 54)
478                 {
479                     break;
480                 }
481
482
483
484     printf("i:%i envio = %s, hora: %lld\n",i,envio, cp);
485   test = build_curl(cp,"UPV",envio,fase);
486
487
488   printf("test: %s\n",test);
489
490         printf("\n vamos a postear los Valores\n");
491   send_curl(test,bucket);
492     printf("\n #####\n");
493       }
494
495
496   }
497
498   int main()
499   {
500   int a = 0;
501
502           while( a < 10) {
503
504               current_timestamp();
505               conn_post();
506   sleep(10);
507   a++;
508
509   }
510   return 0;
511   }
512
```

Además de mostrar la rotura del árbol de condiciones, se imprimen los resultados en pantalla y se postean los valores realizando las pertinentes llamadas a las funciones anteriormente descritas.

```
1050    Programa principal que hace todas las llamadas a las funciones anteriores
1051    en el órden lógico de ejecución.
1052
1053    1º Imprime las cabeceras en el .csv
1054    2º Imprime la fecha actual
1055    3º Se conecta al 2M4096 y lee los valores
1056    4º imprime los valores en el .csv
1057    5º Imprime el timestamp en .ms
1058    6º Ejecuta cada 60 sec el ciclo del programa completo.
1059
1060    */
1061
1062
1063    int main() {
1064
1065        //float result;
1066
1067        headers();
1068
1069        for (int a = 0; a < 1500 ; a++) {
1070
1071            strclean();
1072
1073        //while (c <= y){
1074
1075           fecha();
1076          // result = random();
1077
1078             conn_post();
1079             //  printf("valor envio1= %f\n", enviog);
1080
1081               strbuild();
1082               printf("valor str=  %s\n", str);
1083                printf("valor valor strTOTAL= %s\n", strtotal);
1084
1085             post_datoscsv(*datoscsv);
1086
1087
1088           current_timestamp();
1089
1090
1091          // mail();
1092           loading();
1093
1094
1095        }
1096        fclose(fp);
1097        printf("Archivo (fechaydatos.csv)creado correctamente\n"); //cambiar nombredoc con fecha
1098
1099        mail();
1100        return 0;
1101    }
1102
```

El script MAIN es muy breve, teniendo apenas un pequeño bucle a definición de cuantos ciclos realizará, dos llamadas a las funciones principales, y un tiempo de descanso de 10 segundos, todo esto es modificable según sea necesario.

8. Análisis de Resultados

Si bien el programa pudiera estar mejor optimizado, desarrollado y estructurado bajo un estándar de código de programación entre los existentes; El programa desarrollado por el autor es totalmente funcional, tanto en Linux como en el dispositivo Raspberry PI, que a su vez usa raspbian (Disribución de Linux) como Sistema operativo.

Los resultados han resultado ser óptimos en cuanto a tiempo de ejecución, ya que el presente código es capaz de cargar en la base de datos InfluxDB con una precisión de .ms, de forma cíclica, sin interrupciones, todos los valores relevantes para su estudio posterior… mencionando que la RaspberryPI usada, cuenta con 1GB de RAM, más que suficiente para crear las imágenes virtuales y permitir su acceso a ellas desde cualquier dispositivo que comparta la red de la UPV.

Además de levantar dichos contenedores, con la virtualización y nivel de abstracción necesarios, cargar los datos en InfluxDB, comunicarse con Grafana, como ya hemos mencionado, desde un portátil, móvil, o cualquier otro dispositivo conectado a la red de la UPV, con la dirección IP de la RaspberryPI, (192.168.1.148 en mi caso), cualquier usuario sería capaz de consultar los datos e incluso visualizarlos de forma remota; lo que, de forma reiterada, demuestra las virtudes del concepto IoT.

El "Internet de las Cosas" usado en este proyecto ha sido validado y resulta muy práctico, un usuario inexperto como el autor, con un debido tiempo de investigación, puede acceder a multitud de servicios en red para consultar los datos de la red eléctrica del edificio 8G de la UPV, almacenando todos los datos en servidores virtualizados y permitiendo su visualización adaptada en Dashboard personalizables en Grafana.

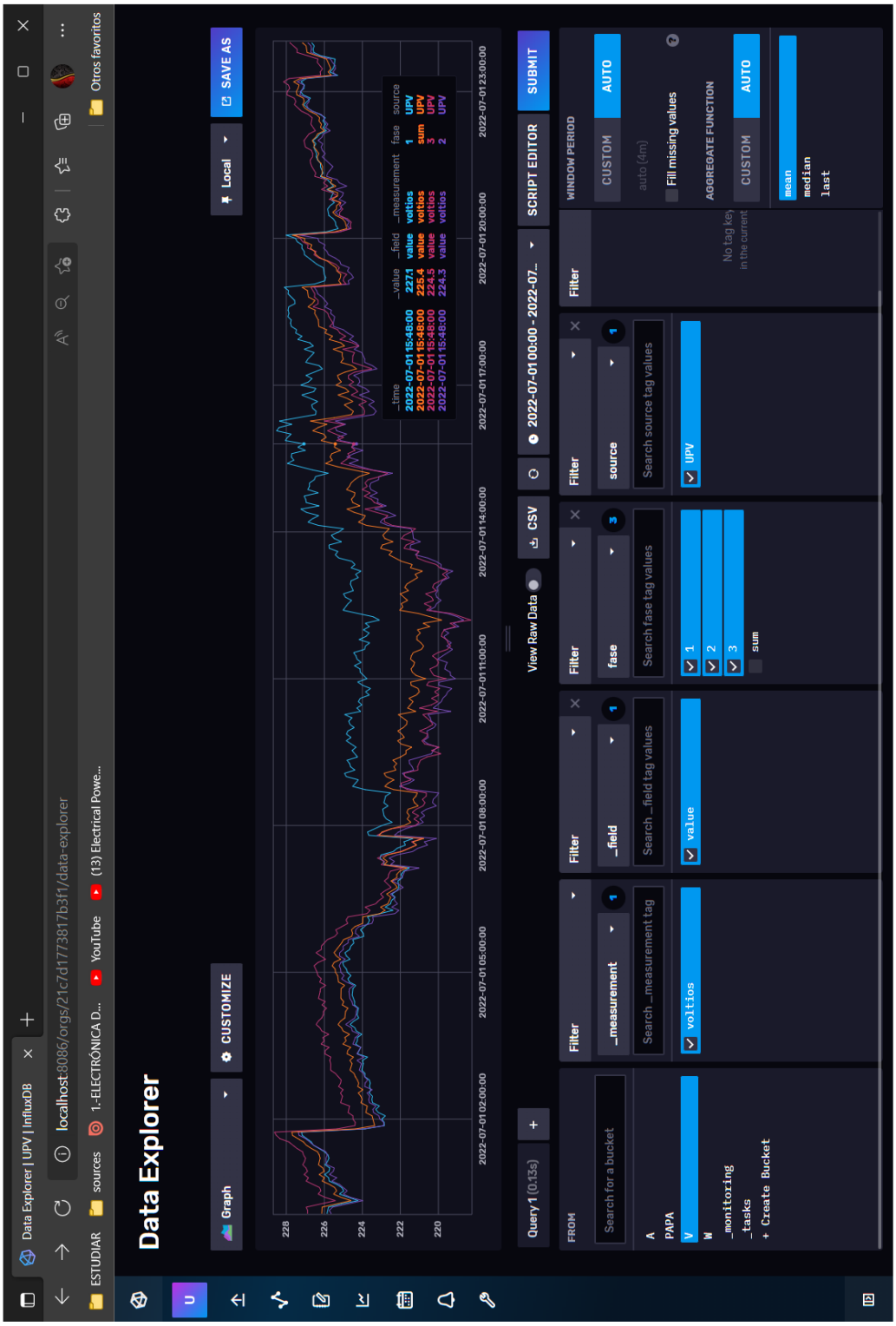


Ilustración p : InfluxDB

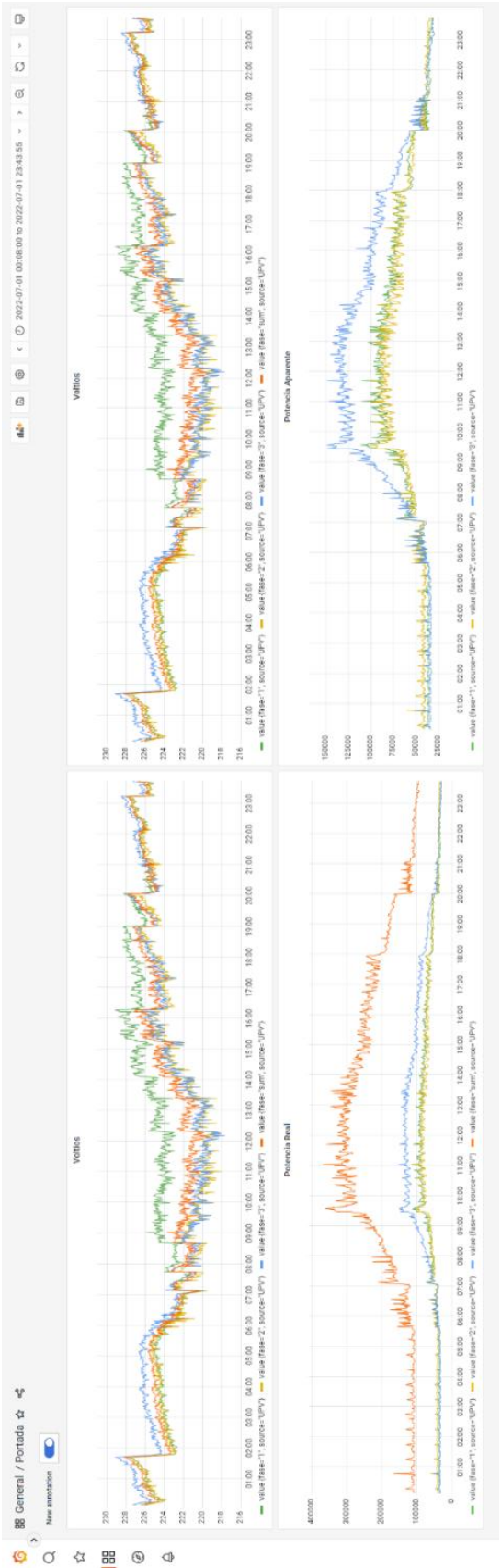


Ilustración q : Ilustraciones funcionales, aunque básicas, del Dashboard de Grafana

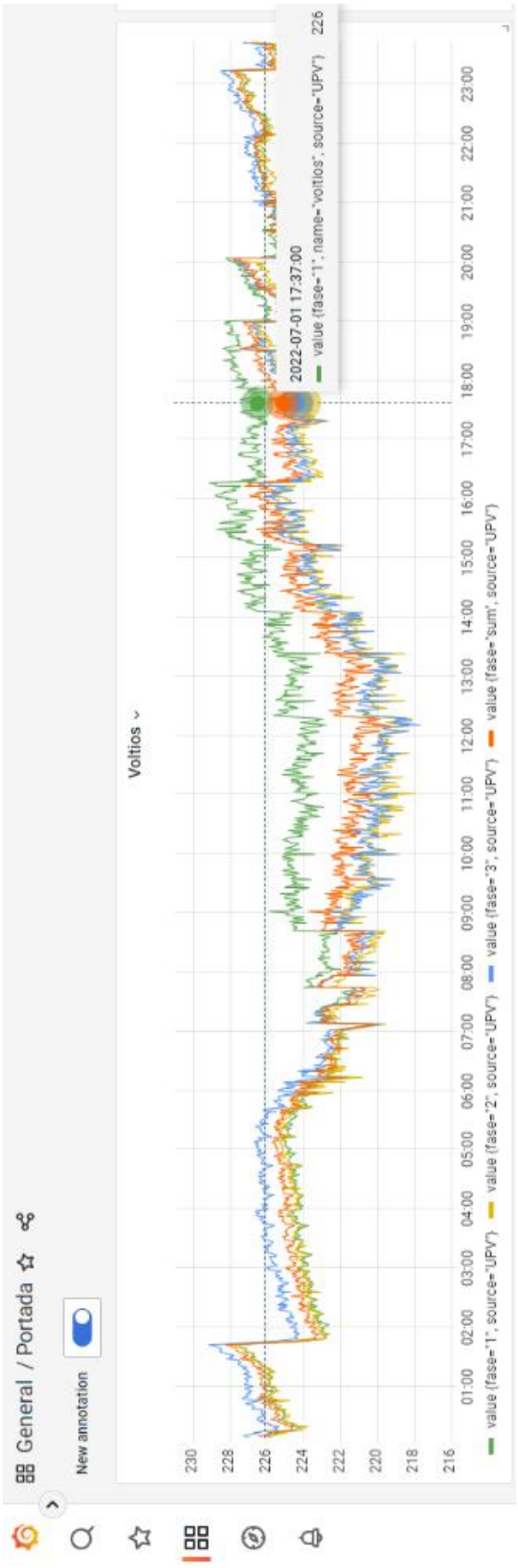
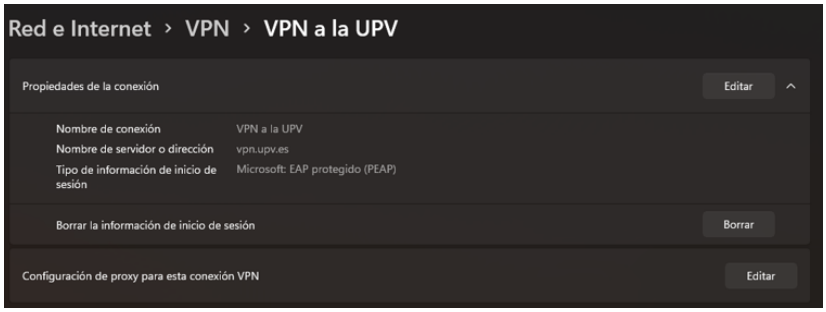


Ilustración r : Muestra experimental real que ilustra valores registrados en serie de tiempo

8.1 Manual de uso del proyecto

Raspbian OS x64

Es imprescindible, para asegurar la funcionalidad del proyecto, que el sistema sea, Raspbian x64 o cualquier distribución de Linux x64.



ps -ef|grep ssh

ip addr

ssh usuario@ip

Esos son los comandos necesarios para conectar, a través de ssh, si no se desea usar VNC, al terminal de la Raspberry.

sudo sh get-docker.sh

sudo usermod -aG Docker pi

sudo apt install libtool autoconf automake cmake cmake-curses-gui

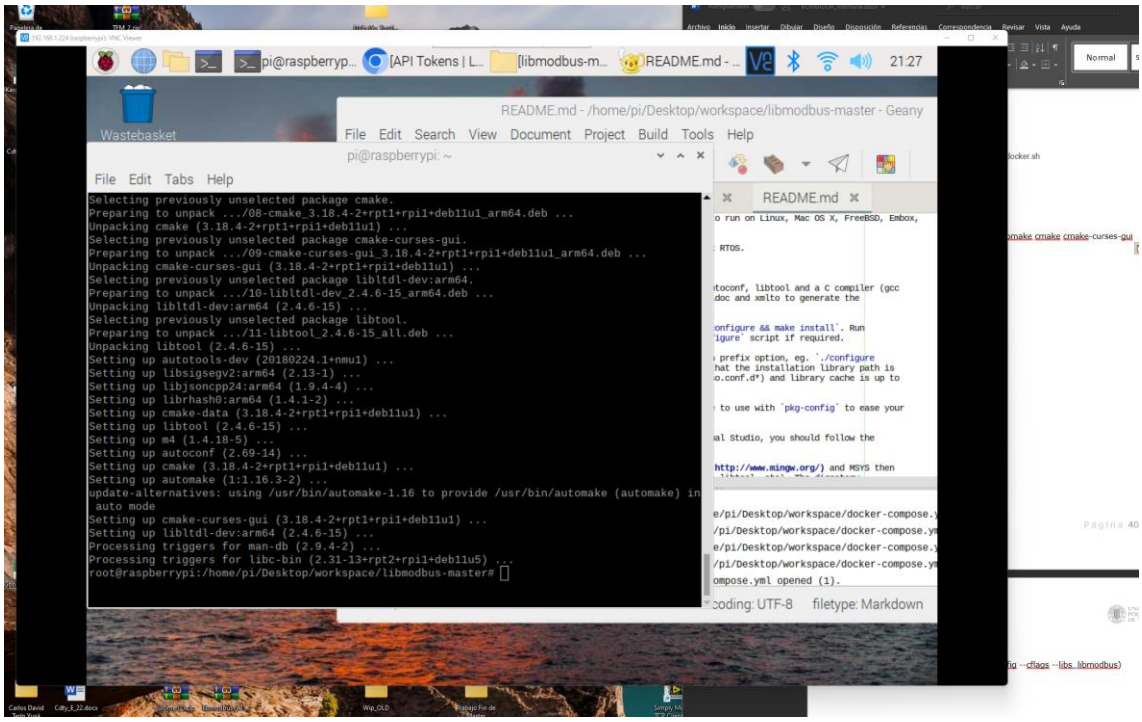
En modo super-user, se instalan los módulos necesarios y librerías para compilar y ejecutar el código

./autogen.sh

./configure.sh

./configure && make install

En la carpeta del workspace, donde esté alojado libmodbus, se procede a compilar y vincular debidamente la librería libmodbus.h a nuestro proyecto



sudo apt update

Con este comando nos aseguramos de que las librerías y aplicaciones asi como sus repositorios estén actualizadas e instaladas

sudo apt-get install libcurl4-openssl-dev

Opcionalmente, es recomendable instalar la última versión para desarrolladores de cURl y su libreria

sudo ldconfig

Con este comando linkeamos dichas librerias

gcc Gold_1.c -o Y1 -lcurl $(pkg-config --cflags --libs libmodbus)

Ya se puede proceder a compilar el código Gold_1.c, con la salida de nombre Y1.out, es imprescindible linkear debidamente las librerías usadas en este proyecto.

gcc Gold_1.c -o $1 -lcurl $(pkg-config --cflags --libs libmodbus)

```
Espera UN minuto , gracias \
Borrando temp.csv
Timestamp: 05/04/2023 15:06
Timestamp en .ms: 1680700018153
Error?
Intenta conectar
Ha conectado bien
valor[17252] = 228.791748
  Entrando: Arbol de If
 Usando strcat() en strotal[1][108]: 228.791;
valor VALOR= 228.791748
valor envioGLOBAL= 228.791000
TEST 208.703;66754.797;108296.000;256854.000;-6862.800;7116.610;-5336.400;-5059.300;-0.998;0.995;-1.000;-1.000;49.903;1.433;0.801;378.526;378.526;
i:1 envio = 228.791, hora: 1680700018153
1680700018153
data3: UPV,fase=1,source=UPV value=228.791 1680700018153
test: UPV,fase=1,source=UPV value=228.791 1680700018153

 vamos a postear los Valores 228.791000, 1

Influx Esta Online
 #####
valor[51888] = -5775792.000000
  Entrando: Arbol de If
 TEST 208.703;66754.797;108296.000;256854.000;-6862.800;7116.610;-5336.400;-5059.300;-0.998;0.995;-1.000;-1.000;49.903;1.433;0.801;378.526;378.526;
i:2 envio = 228.791, hora: 1680700018153
1680700018153
data3: UPV,fase=1,source=UPV value=228.791 1680700018153
test: UPV,fase=1,source=UPV value=228.791 1680700018153

 vamos a postear los Valores 228.791000, 2

Influx Esta Online
 #####
valor[17248] = 224.712250
  Entrando: Arbol de If
 Usando strcat() en strotal[1][108]: 228.791;224.712;
TEST 208.703;66754.797;108296.000;256854.000;-6862.800;7116.610;-5336.400;-5059.300;-0.998;0.995;-1.000;-1.000;49.903;1.433;0.801;378.526;378.526;
i:3 envio = 224.712, hora: 1680700018153
1680700018153
data3: UPV,fase=2,source=UPV value=224.712 1680700018153
test: UPV,fase=2,source=UPV value=224.712 1680700018153

 vamos a postear los Valores 224.712006, 3

Influx Esta Online
 #####
valor[46678] = -0.000003
  Entrando: Arbol de If
 TEST 208.703;66754.797;108296.000;256854.000;-6862.800;7116.610;-5336.400;-5059.300;-0.998;0.995;-1.000;-1.000;49.903;1.433;0.801;378.526;378.526;
i:4 envio = 224.712, hora: 1680700018153
1680700018153
data3: UPV,fase=2,source=UPV value=224.712 1680700018153
test: UPV,fase=2,source=UPV value=224.712 1680700018153
```

El programa en ejecución, se ve de esta manera en el terminal de consola.

9. Presupuestos

Herramienta	Utilidad	Precio Estimado	Precio con I.V.A.
Raspberry Pi 3 B+	Mini-PC, Microcontrolador para acceder al dispositivo, a la red de internet, y ejecutar el programa.	43,50 €	52,64 €
Raspberry Pi 4 B 8GB	Modelo opcional, modernizado, con la misma funcionalidad, aunque mejor eficacia	78,99 €	95,58 €
Computadora con Sistema Operativo	Ordenador completo con al menos terminal de ejecución de comandos. (Linux, Raspbian, Debian, Noobs, Windows…)	~	~
Conexión a Internet	Imprescindible que el dispositivo tenga acceso a Internet.	~	~
Software Open Source	Docker, Grafana, InfluxDB	OpenSource	OpenSource
CARLO GAVAZZI (WM40AV63H)	Analizadores de tensión WM40-96 BASE. Las distintas funcionalidades de los módulos agregados, incrementan el valor del dispositivo.	Aprox. ~ 922,31 €	Aprox. ~1.167,48 €
Mano de obra	(300h. aprox)	12,5€/h	3.750€ brutos
		Subtotal Aproximado	5065,7 €

1. https://es.rs-online.com/

2. https://www.mouser.es/c/test-measurement/analyzers/power-analyzers/?m=Carlo%20Gavazzi&series=WM40

10. Conclusiones

✓ -Tratar los datos del archivo .csv devuelto por el analizador de Calidad de Red.

El programa desarrollado es capaz de gestionar los datos, clasificarlos en función del nº de registro en el que se encuentren y crear un archivo de texto plano y enviarlo por correo electrónico.

✓ -Procesar dichos datos por el programa en lenguaje C-POSIX.

En este lenguaje, robusto y fiable, comprensible y estructurado, comentado, es una tarea lograda el identificar cada valor del registro

Lo que facilita operaciones matemáticas que pudieran interesar para el Estudio de la Red Eléctrica, medias, medianas, factores de emergencia…

✓ -Transmisión a través de un protocolo Local/Internet.

En este caso, se usó la VPN de la UPV, una vez conectado a la Intranet, es sencillo acceder al dispositivo, y a su vez trabajar en Internet, es evidente que en las dos computadoras, y la RaspberryPI, con las credenciales adecuadas, el API-TOKEN de Influx y Grafana, o simplemente accesos ofrecidos, podría acceder e interactuar cualquier usuario

✓ -Almacenamiento correcto en la base de datos InfluxDB.

Los datos registrados, es decir, leídos desde el dispositivo, y guardados en la base de datos, son coherentes.

- ✓ -Sincronizar dichos datos entre la base de datos, y el visualizador Grafana

La sincronización entre InfluxDB y Grafana es una Integración que ofrecen ambas plataformas, en desarrollo, si bien no son la misma compañía, trabajan juntas de forma adecuada.

- ✓ -Crear un Dashboard accesible y visual de representación de valores de relevancia.

Es una tarea que debería ser especificada por las necesidades del técnico especialista en Análisis de Calidad de Red, debe ser una tarea compleja, pues ya entraríamos en valoraciones económicas, factores de potencia indeseados, formas de onda perjudiciales… armónicos…

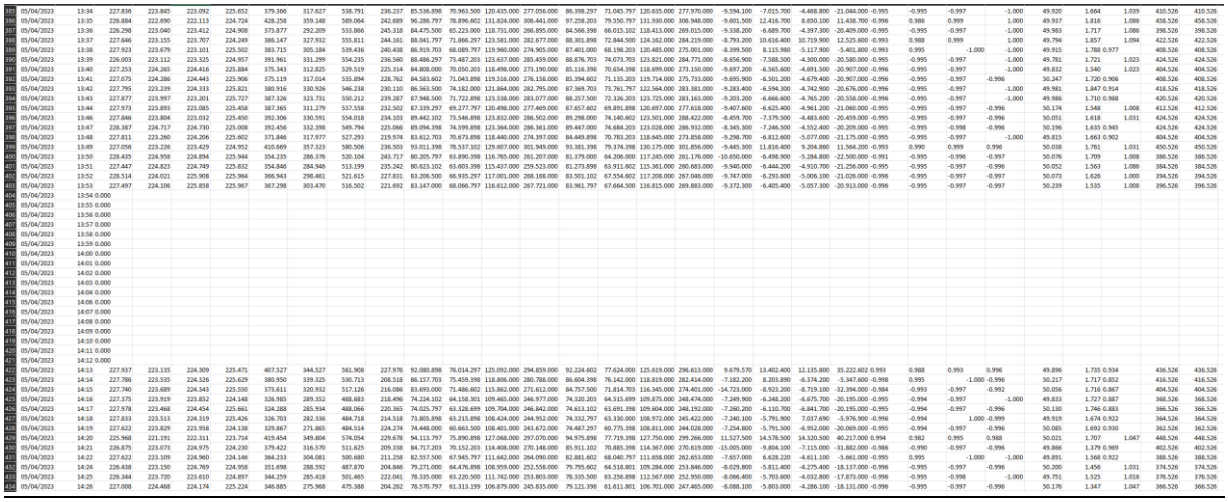
- ✓ -Comprobar la eficiencia del código y su respuesta temporal, Refactorizarlo.

Partiendo de la base que el código es un lenguaje, siempre se puede "optimizar", y es "Variable" en función del Estándar de programación, las necesidades del usuario, el estilo del programador…

Si bien el programa, salvo un corte de internet de 15min, ajeno a nuestro control, demostró registrar los datos cada minuto, es común encontrar soluciones óptimas a nivel de ejecución en cuanto a CARGA del Procesador, lo que se traduce en tiempo/watios/unidad económica.

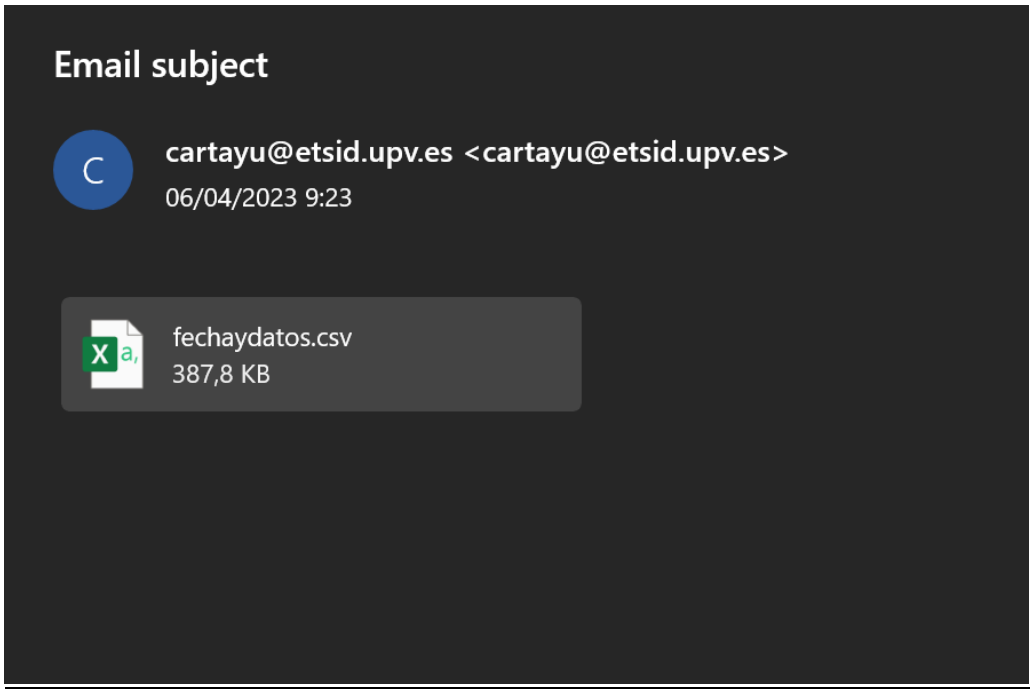
✓ Entre las pruebas realizadas: mostramos a continuación la última prueba realizada;

Pérdida de datos de 15 minutos, incontrolable desde mi punto de vista, si no hay conectividad, ya sea de internet, intranet local, o cortes de red eléctrica, el programa dejará de registrar datos.



- Entre las pruebas realizadas, este programa, aprovechando la librería estática compilada "libcurl", es capaz de adjuntar el archivo de texto plano que crea esté código, y lo adjunta desde la propia dirección del alumno, hacia la propia dirección personal del alumno.

  En formato .CSV, archivo de texto PLANO.

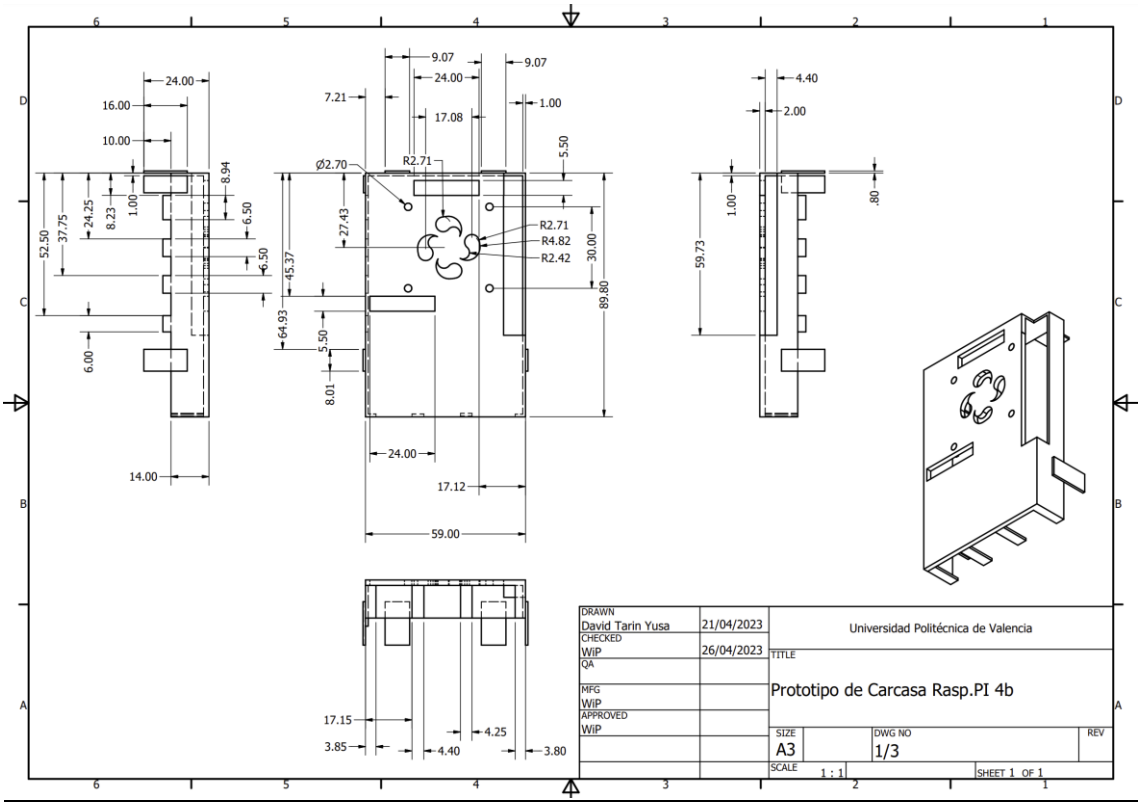


Esta es la imagen del Dashboard de GRAFANA, que muestra un día laboral desde las 8AM aproximadamente, con una duración de 30 horas.

(La ejecución del código y su duración es configurable)

SE ADJUNTAN COMO ANEXOS:

Los planos realizados en Autodesk Inventor, así como las piezas que podrían servir como diseño prototipo para una carcasa de Inyección, que no de Impresión 3d.



Estos tres planos, consisten en tres piezas básicas que, ensambladas, conjuntan una carcasa para la RaspberryPI 4B, que, disponiendo de sus medidas públicas ofrecidas por el fabricante, hicieron posible esta propuesta.

Este prototipo propuesto contempla:

- Un Backup de 4 Baterias AA comúnes, que podrían dar un soporte eléctrico de entre 2h y 6h, en función del BUCK-CONVERTER
  seleccionado, las especificaciones de las baterías, la carga de proceso de la Raspberry..

- Una Tapa con un diseño orgánico para un ventilador pequeño, de 30x30mm, sujeto por cuatro tornillos.

  Esta tapa tiene orificios indispensables para los GPIO y debería ser adecuada para los usuarios experimentales.

- Una carcasa básica con los orificios pertinentes para cada conexión, Ethernet, MicroUSB-C, Micro-HDMI, Jack… USB para HDD… y el indispensable SD que contiene el SO.

  (https://uk.rs-online.com/web/c/semiconductors/power-management-ics/buck-converters/?applied-dimensions=4294506458,4294509042,4294511827,4294508847)

Se adjunta la dirección donde se proponen diferentes alternativas de Buck-Converter para bajar el voltaje de 6v a 5v que requiere la RaspberryPi 4B.

11. Herramientas utilizadas y Middlewares

Copilot plugin Visual Studio 2022

"GitHub Copilot"

(https://docs.github.com/es/copilot/getting-started-with-github-copilot)

Asistente de programación con funciones predefinidas y buscador de funciones y librerías, procesador de búsquedas y sugerencias en distintos lenguajes de programación.

"Copilot fue especialmente útil para encontrar funciones ya elaboradas, funcionales, que sirvieron como fundamento y base, para adaptarlas a este trabajo."

Stack Overflow

(https://stackoverflow.com/)

Página web que ofrece un sistema que funciona como un espacio en el que se formulan y solucionan preguntas relacionadas con ingeniería del software.

"El Know-How de este sitio web es incalculable, es a subrayar que mis preguntas particulares, ya habían sido formuladas y resueltas por otros usuarios de internet, lo cual deja en una posición de relevancia la colaboración desinteresada"

OpenAI

Sistema de "Código abierto" de Inteligencia Artificial enfocado en responder preguntas basándose en un entrenamiento limitado en *crawler*(3) de Internet.

(3) Crawler, conocido como rastreador, es un programa que analiza los documentos de los sitios web. Creando una base de datos con la información recolectada.

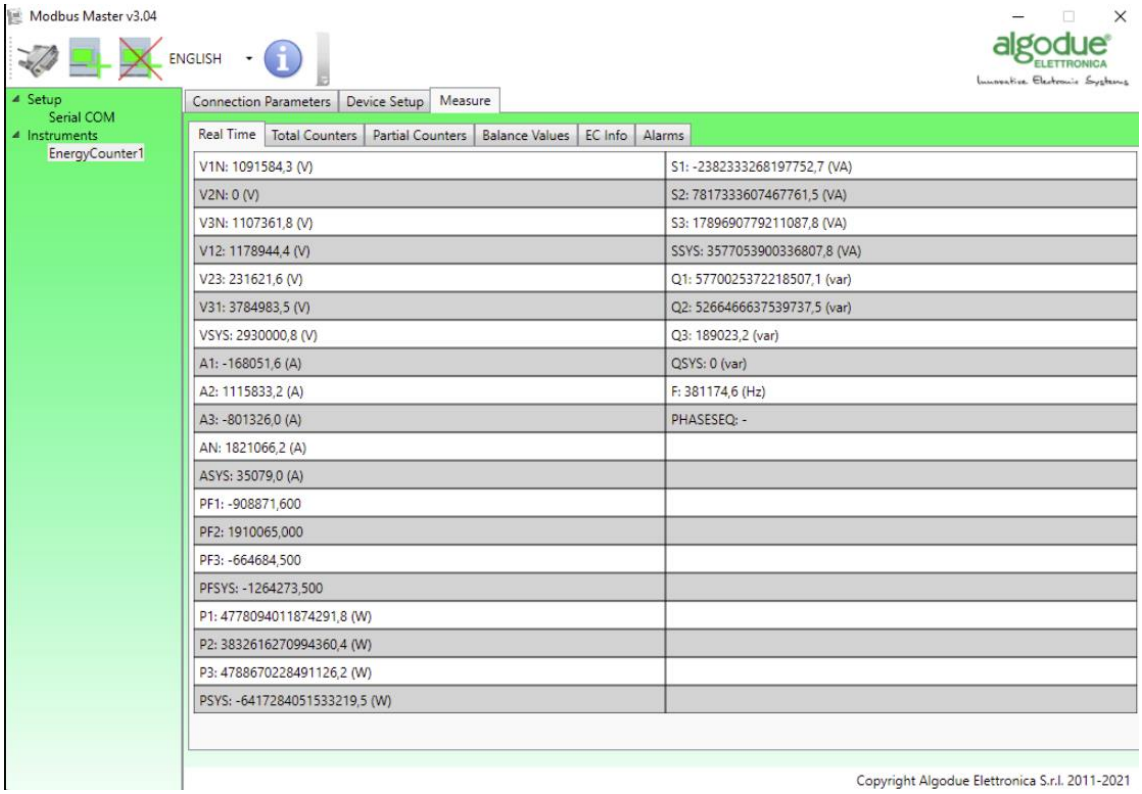
"Las inteligencias Artificiales asistentes de lenguaje, responden de forma precisa a preguntas coherentes… Indico aquí, que tampoco son resolutivas al completo, están en fase inmadura en fecha de este trabajo, ya que ha habido funciones que no fue capaz de sintetizar y requerían de correcciones humanas indispensables, el humano crea, imagina a una velocidad que la AI, a día de hoy, no puede ni ordenar lógicamente nuestras ideas"

Middlewares

Modbus-POLL o CAS Modbus Scanner, Modbus Master o Simply Modbus TCP Client.

"Estos son softwares que facilitaron la consulta específica de los registros del analizador de Red de baja tensión, dichos valores se almacenan en registros que si bien el fabricante "ofreciera" un manual donde referenciar dichos registros, yo no disponía de este ni pude encontrarlo, gracias a estos softwares intermedios, fue posible esta tarea.

Saber en que formato eran presentados, (Little endian Byte Swap), ya fue cuestión empírica"



Esta imagen corresponde a un software intermedio, monetizado, y que ofrece menos funcionalidades que el proyecto actual, sin embargo, resultó útil para discernir en que registro se encontraba cada dato a evaluar en este trabajo.

Tx = 0: Err = 0: ID = 1: F = 03: SR = 1000ms
No Connection

	Alias	00050	Alias	00100	Alias	00150
0				222.757141		1.#INF00
1						
2				37.781311		0.000000
3						
4				53083.519531		240.487549
5						
6				58513.628906		0.954153
7						
8				50304.007813		0.979862
9						
10						1.600764
11						
12						1.557977
13						
14						1.256108
15						
16						1.027432
17						
18						0.988763
19						
20						1.081708
21						
22						8.562078
23						
24				-3979.788818		6.550443
25						
26				-15255.744141		6.348936
27						
28				-0.990696		1.495323
29						
30		225.093719		-0.997690		1.501479
31						
32		225.000336		-0.996885		1.195185
33						
34		226.530075		-0.995590		0.962229
35						
36		225.541382		50.008705		0.945702
37						
38		389.762451		0.678252		1.023393
39						
40		391.865662		0.538254		8.538434
41						
42		390.612671		1.000000		6.536990
43						
44		390.746918		1.008215		6.302944
45						
46		238.043335		1.008215		0.571362
47						
48		260.662201		1.008215		0.415759

Read/Write Definition

Slave ID: 1
Function: 03 Read Holding Registers (4x)
Address: 80
Quantity: 125
Scan Rate: 1000 ms
Read/Write Enabled

OK
Cancel
Apply
Read/Write Once

View
Rows: 10 20 50 100
Hide Alias Columns
Address in Cell
Display: Float
PLC Addresses (Base 1)

12. Líneas de Trabajo futuras/en desarrollo

- Se propone una mejora de este TFM en forma de ejecución en tiempo real en un servidor dedicado, con un "SAI", sistema de apoyo de energía o similar, así como una redundancia del mismo en otro contendor de Docker

"Si bien la RaspberryPI 3b, e Incluso la 4b, resultan más que suficientes y presentan autonomía para controlar un solo analizador de red YA en este trabajo, es más que probable que un complejo de edificios como lo puede ser la U.P.V. requiera de un array de Raspberry's, con total seguridad, para asegurar una funcionalidad segura y de calidad."

- Dada la potencialidad de centralización, es evidente que este trabajo abre puertas a conectividad con "Google Alerts" o similar, interconexión entre plataformas, dispositivos… con su correspondiente estudio sobre parámetros relevantes, definidos por las necesidades del técnico"

- Se propone también un sistema Activo, o mejor dicho, Reactivo, que conecte y desconecte los dispositivos necesarios para corregir la CALIDAD de la Red. Baterías de condensadores…

- "Crear una IA capaz de injerir en los procesos de estudio de la red."
  Puede que este sea el trabajo mas ambicioso que ahora mismo podría plantearse, pues la IA se estima que debería saber: qué, cuándo, cómo y porqué, debe actuar por si misma, de forma autónoma"

  Pudiera partir de la base de obtener "ineficiencias" y clasificarlas de forma lógica, en base a la experiencia.

- "PM2 Y CRON", *podrían facilitar, mostrando herramientas útiles para gestionar, temporizar acciones futuras….*

- "Jenkins y Rancher" igualmente, consisten en cadenas de procesos conectados de forma tal que la salida de cada elemento es la entrada del próximo. Permiten la comunicación y sincronización entre procesos.

  A priori, podrían ser herramientas más que válidas para las futuras líneas de desarrollo.

13. Bibliografía

- Manual GAVAZZI WM4096DS
- Commonly used Power and Converter Equations By Daniel W Hart
- LaNormaISO50001 Eduardo Musitani
- Power Electronics Handbook Fourth Edition
- Manual de baja tensión Siemens

Enlaces de la web:

- https://baseconvert.com/ieee-754-floating-point
- https://es.wikipedia.org/wiki/IEEE_754
- https://es.wikipedia.org/wiki/Endianness
- https://es.rs-online.com/web/p/raspberry-pi/1822098
- https://docs.github.com/es/copilot/getting-started-with-github-copilot
- https://uk.rs-online.com/web/c/semiconductors/power-management-ics/buck-converters/?applied-dimensions=4294506458,4294509042,4294511827,4294508847