GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

ADQUISICIÓN DE DATOS INDUSTRIALES BASADO EN IOT

Autor: José Gil Fernández

Director: Javier Sánchez Alonso

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Adquisición de datos industriales basado en IoT

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020/21 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: José Gil Fernández Fecha: 16/07/2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Javier Sánchez Alonso Fecha: 16 / 07 / 2021

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ADQUISICIÓN DE DATOS INDUSTRIALES BASADO EN IOT

Autor: José Gil Fernández

Director: Javier Sánchez Alonso

Madrid

Agradecimientos

Me gustaría agradecer en primer lugar a mis padres, por todo el apoyo y ánimos en el trascurso de estos cuatro años. Siempre habéis puesto mi educación como una de vuestras prioridades y quería daros las gracias por todo el esfuerzo y paciencia que os ha supuesto.

En segundo lugar, quería agradecer a mis dos abuelos, Fede y Pepe, que siempre han estado más ilusionados por mi carrera que yo y están muy orgullosos de tener otro nieto ingeniero. También quería dar las gracias a mis abuelas, Pilar y Bea, por todas sus oraciones en mis semanas de exámenes, sin las cuales no sé si estaría aquí y por siempre creer en mí y que podía con ello.

También quería dar las gracias a la empresa Zeus Control, especialmente a Javier y Emilio, por haberme dado esta maravillosa oportunidad para llevar a cabo este Trabajo de Fin de Grado, su dedicación y haberme permitido adentrarme en el mundo de la industria.

Por último, quería dar las gracias a lo mejor que me llevo de estos cuatro años. Mi grupo de amigos, con quienes he compartido el día a día de esta carrera, tardes de estudio, comidas y sobremesas en la cafetería, llantos y alegrías. De no ser por ayuda, apoyo y cariño yo no estaría aquí. Creo que hemos creado un vínculo que hará que nunca olvide estos cuatro años y siempre los recuerde con enorme cariño.

ADQUISICIÓN DE DATOS INDUSTRIALES BASADO EN IOT

Autor: Gil Fernández, José
Director: Sánchez Alonso, Javier.
Entidad Colaboradora: Zeus Control

RESUMEN DEL PROYECTO

Con este proyecto se han investigado una serie de herramientas que nos ofrece la filosofía IoT para poder llevar a cabo una solución aplicable a procesos industriales. Se ha empleado un PLC industrial para el envío de datos a partir del protocolo de comunicación MQTT y la comunicación por enlaces S7. Finalmente se han desarrollado soluciones para la monitorización y adquisición de datos industriales.

Palabras clave: IoT, MQTT, mosquitto, TLS, Node-Red

1. Introducción

En la actualidad, el IoT (Internet of things) se ha convertido en una de las tecnologías más importantes. Cada vez está más presente en nuestra sociedad debido a las numerosas aplicaciones que ofrece, para nuestra vida cotidiana como para su empleo en ámbitos industriales.

La aplicación de esta tecnología en la industria se conoce como IIoT (Industrial Internet of Things) y junto con los avances en Big Data, robótica y Machine Learning, se considera que estamos ante la Cuarta Revolución Industrial, ya que la industria se enfrenta a nuevos modelos de negocio y a una nueva automatización.

Hoy en día ya comienzan a surgir soluciones industriales basadas en IoT que nos permiten llevar a cabo formas de adquisición masiva de datos, monitorización y visualización en tiempo real. Con estas soluciones se puede llevar a cabo un análisis de los datos con los que poder tomar las decisiones más ventajosas. En el futuro, esta tecnología se desarrollará lo suficiente para poder aplicarse en controles y actuadores industriales, con lo que poder emplear la alta conectividad que ofrece el IoT para optimizar la gestión de procesos.

2. Definición del Proyecto

Con este proyecto se pretende llevar a cabo un trabajo de investigación y desarrollo. Para ello se comenzará por adquirir conocimientos generales de IoT y su aplicación en el sector industrial. Posteriormente, se investigarán las posibles herramientas que ofrece el IoT y se desarrollará con ellas unas soluciones IoT de aplicación industrial que puedan servir como base a futuros desarrollos.

Las soluciones desarrolladas en este proyecto se han realizado con herramientas de código abierto de licencia gratuita, y están basadas en el almacenamiento y monitorización de datos en tiempo real, de dispositivos industriales mediante tecnología IoT. Estas soluciones suponen distintas formas de visualización de los datos, y alguna podría incluso suponer una alternativa más económica para soluciones independientes en las que no sea rentable comprar una licencia SCADA.

3. Descripción del modelo/sistema/herramienta

En este modelo se han desarrollado 2 sistemas distintos:

En primer lugar, el de la Ilustración 1. Se quiere usar el protocolo MQTT para realizar la adquisición de datos de un PLC industrial. Para llevar a cabo la comunicación MQTT es necesaria la configuración de un bróker, encargado de gestionar el envío de datos entre clientes publicadores y clientes subscriptores. En este modelo vemos que el bróker configurado será mosquitto, el cliente publicador será el PLC y el cliente subscriptor se encontrará en Node-Red, donde posteriormente se gestionará el almacenamiento de los datos recibidos en una base de datos (InfluxDB). Al mismo tiempo, se empleará la plataforma Grafana para acceder a la base de datos y realizar una visualización de los datos obtenidos. Para simular el envío por MQTT y su uso en visualización, el PLC enviará los datos de 4 señales distintas creadas a partir de una función.

En segundo lugar, el mostrado en la Ilustración 2. El PLC se programará para llevar a cabo una simulación de una línea de pasteurización simple y calcular su OEE (Overall Equipment Effectiveness). Posteriormente, Node-Red se hará cargo de recibir los resultados por S7 y se llevarán a cabo 3 formas distintas de representación. Con Node-Red, Grafana y MQTT Dash. Para este último, será necesario configurar un bróker cloud (MyQttHub), para enviar los datos hasta un cliente subscriptor en un Smartphone.

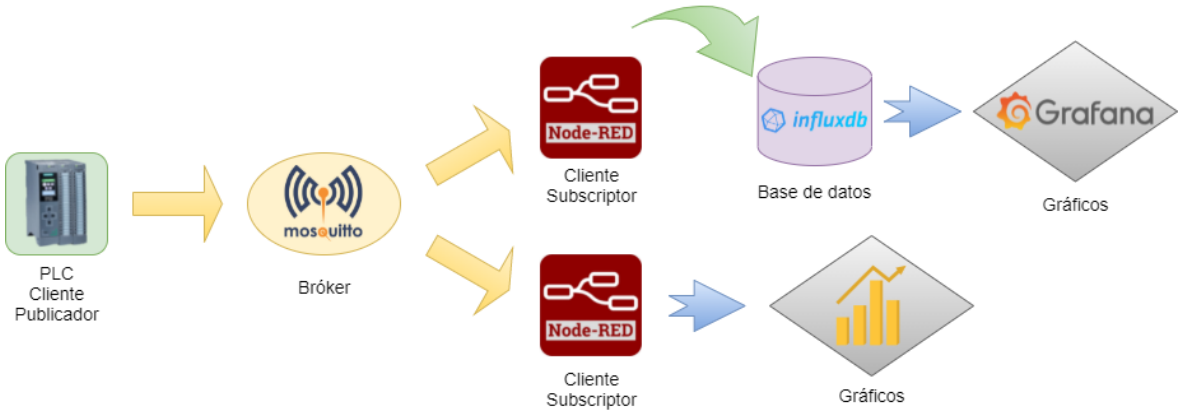


Ilustración 1: Sistema de envío y visualización de señales por MQTT

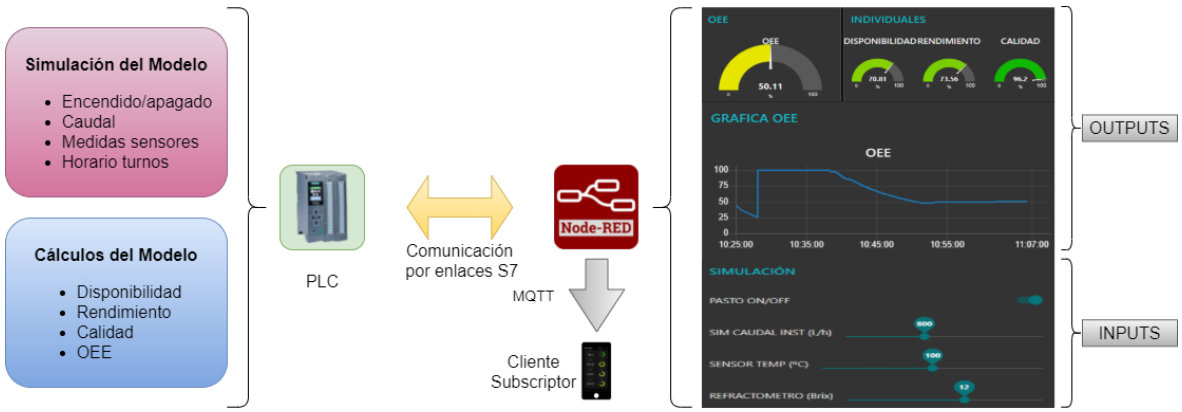


Ilustración 2: Sistema de simulación y visualización del OEE de una línea de pasteurización

4. Resultados

En este proyecto se han conseguido soluciones eficaces y funcionales para la visualización y monitorización de datos industriales. A continuación, se muestran las distintas formas de visualización de los sistemas explicados:

En la Ilustración 3 podemos ver los resultados de visualización y monitorización obtenidos con Grafana de las señales generadas por el PLC y enviadas por el protocolo MQTT.

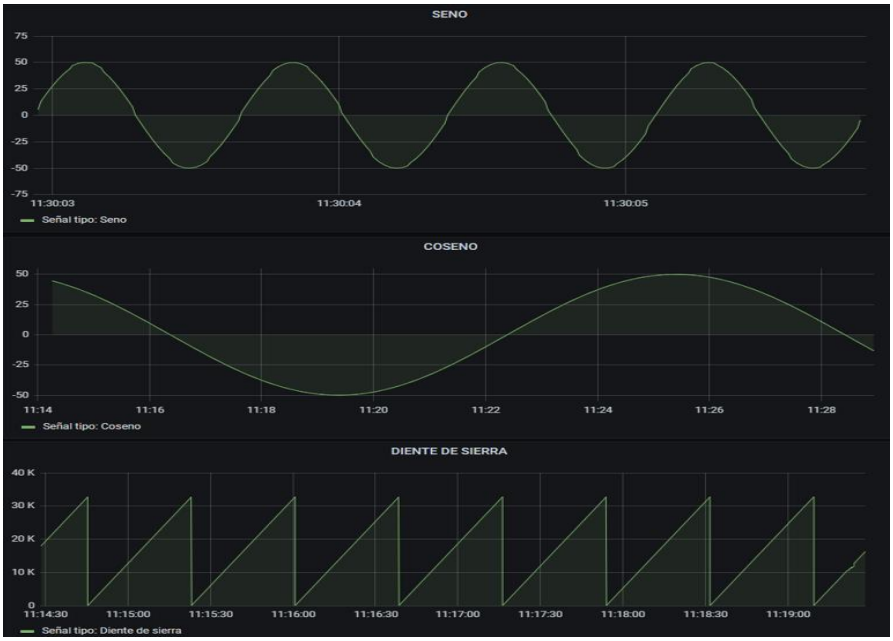


Ilustración 3: Visualización de señales en Grafana

En la Ilustración 4, podemos ver las distintas formas de visualización del OEE de la línea de pasteurización. A la izquierda Node-Red, en el centro Grafana y a la derecha desde MQTT Dash en un Smartphone.

Ilustración 4: Visualizaciones del OEE de la línea de pasteurización

5. Conclusiones

Con la realización de este proyecto se puede concluir que la tecnología IoT está ya muy avanzada en soluciones de adquisición y monitorización de datos. Resultando muy útil para poder llevar a cabo un análisis de los datos en tiempo real de una forma profesional y aplicable a la industria.

También se ha concluido que aún se debe optimizar la función de cliente MQTT del PLC, ya que, aunque sirva para llevar a cabo la visualización de señales, su tiempo de muestreo no es constante. Además, la comunicación S7 ha terminado siendo mucho más cómodo y práctica que el MQTT, ya que se podía tanto recibir como escribir datos desde Node-Red al PLC y no era necesario ordenar los datos previamente, como con el MQTT.

Finalmente, me gustaría mencionar que las herramientas empleadas en este proyecto son muy útiles y siguen en constante desarrollo, de forma que pueden dar píe a mejores soluciones. Node-Red ha resultado ser una herramienta muy versátil para llevar a cabo el diseño y estructura de las soluciones y Grafana es una herramienta excelente para la visualización de datos de forma profesional y tratado de datos.

6. Referencias

[1] "MQTT - The Standard for IoT Messaging", Mqtt.org, 2021. [Online]. Available: https://mqtt.org/. [Accessed: 19- Jul- 2021].

[2] "Eclipse Mosquitto", Eclipse Mosquitto, 2021. [Online]. Available: https://mosquitto.org/. [Accessed: 19- Jul- 2021].

[3] "Node-RED", Nodered.org, 2021. [Online]. Available: https://nodered.org/. [Accessed: 19- Jul- 2021].

[4] "InfluxDB: Purpose-Built Open Source Time Series Database | InfluxData", InfluxData, 2021. [Online]. Available: https://www.influxdata.com/. [Accessed: 19- Jul- 2021].

[5] "Grafana", Grafana.org, 2021. [Online]. Available: https://grafana.com/. [Accessed: 19- Jul- 2021].

[6] "MyQttHub -- Servidor MQTT Cloud -- Alojamiento MQTT", Myqtthub.com, 2021. [Online]. Available: https://myqtthub.com/. [Accessed: 19- Jul- 2021].

IOT-BASED INDUSTRIAL DATA ACQUISITION

Author: Gil Fernández, José.
Supervisor: Sánchez Alonso, Javier.
Collaborating Entity: Zeus Control

ABSTRACT

With this project, a series of tools offered by the IoT philosophy have been investigated in order to carry out a solution applicable to industrial processes. An industrial PLC has been used to send data with the MQTT communication protocol and with the S7 link communication. Finally, some solutions have been developed for the monitoring and acquisition of industrial data.

Keywords: IoT, MQTT, mosquitto, TLS, Node-Red

7. Introduction

Nowadays, IoT (Internet of things) has become one of the most important technologies. It is more often present in our society due to the numerous applications it offers, for our daily lives as well as for its use in industrial fields.

The application of this technology in industry is known as IIoT (Industrial Internet of Things) and together with advances in Big Data, robotics and Machine Learning, we are facing the Fourth Industrial Revolution. This is because Industry is facing new business models and new ways of automatization.

Today, industrial solutions based on IoT are beginning to emerge. These solutions allow us to carry out forms of massive data acquisition, monitoring and visualisation in real time. With these solutions, data analysis can be carried out in order to make the most advantageous decisions. In the future, this technology will be developed enough to be applied to industrial controls and actuators, so that the high connectivity offered by the IoT can be used to optimise process management.

8. Project Definition

The aim of this project is to carry out research and development work. This will start by acquiring general knowledge of IoT and its application in the industrial sector. Subsequently, the possible tools offered by the IoT will be investigated and applied in the development of industrial IoT solutions that can serve as a basis for future developments.

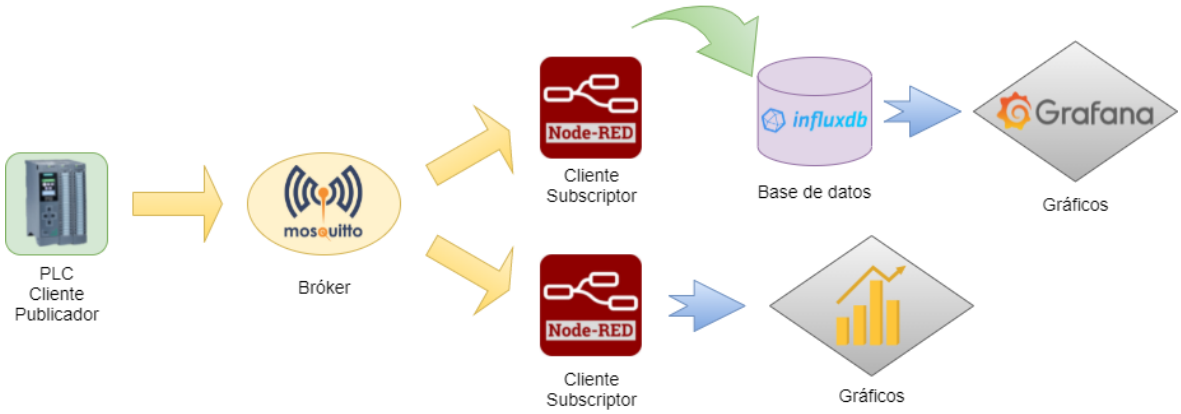
The solutions developed in this project have been developed with free open-source tools and are based on the storage and monitoring of real time data from industrial devices using IoT technology. These solutions involve different ways of visualising data, and some of them could even be a cheaper alternative for stand-alone solutions where it is not cost-effective to buy a SCADA licence.

9. Description of the system

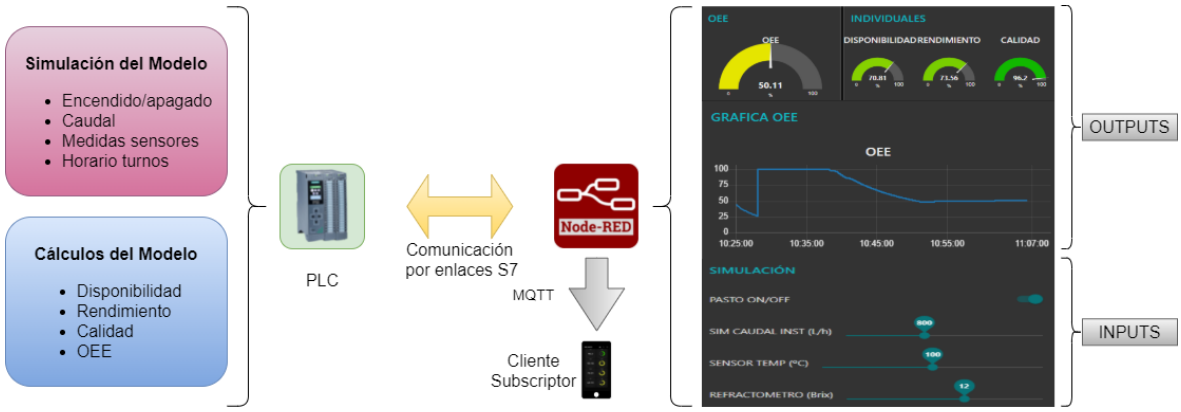
Two different systems have been developed in this model:

Firstly, the one shown in Illustration 1. The idea is to program a function in the PLC in charge of generating four different signals and sending the data of these signals through the MQTT protocol. To do this, the PLC must be programmed as a publishing client and the mosquitto broker must be configured to share the data with the different Node-Network subscriber clients. Subsequently, the storage of the data received will be managed in a database (InfluxDB) and the Grafana platform will be used to access the databases and visualise the data obtained.

Secondly, the one shown in Illustration 2. The PLC will be programmed to carry out a simulation of a simple pasteurisation line and calculate its OEE (Overall Equipment Effectiveness). Subsequently, Node-Red will take care of receiving the results by S7 communication and 3 different ways of representation will be carried out. With Node-Red, Grafana and MQTT Dash. For the last one, it will be necessary to configure a cloud broker (MyQttHub) to send the data to a subscriber client on a Smartphone.



Ilustration 1: Signal sending and display system via MQTT

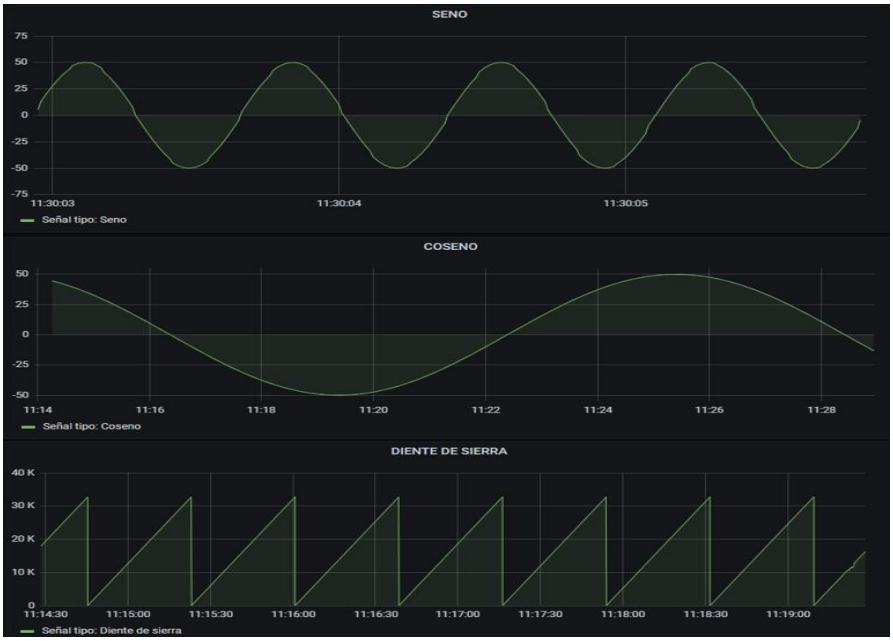


Ilustration 2: System for simulation and visualisation of the OEE of a pasteurisation line.

10. Results

In this project, effective and functional solutions for the visualisation and monitoring of industrial data have been achieved. The different forms of visualisation of the systems explained are shown below:

In Illustration 3 we can see the visualisation and monitoring results obtained with Grafana of the signals generated by the PLC and sent by the MQTT protocol.



Ilustration 3: Signal display in Grafana

In Illustration 4, we can see the different ways of visualising the OEE of the pasteurisation line. On the left Node-Red, in the centre Grafana and on the right from MQTT Dash on a smartphone.

Ilustration 4: Pasteurisation line OEE visualisations

11. Conclusions

With the completion of this project, it can be concluded that IoT technology is already very advanced in data acquisition and monitoring solutions. It is very useful to be able to carry out data analysis in real time in a professional way and applicable to industry.

It has also been concluded that the MQTT client function of the PLC still needs to be optimised, as, although it serves to carry out signal visualisation, its sampling time is not constant. Furthermore, S7 communication has turned out to be much more convenient and practical than MQTT, as it was possible to both receive and write data from Node-Red to the PLC and it was not necessary to sort the data beforehand, as with MQTT.

Finally, I would like to mention that the tools used in this project are very useful and are still under constant development, so that they can lead to better solutions. Node-Red has proved to be a very versatile tool for carrying out the design and structure of the solutions and Grafana is an excellent tool for professional data visualisation and data processing.

12. References

[1] "MQTT - The Standard for IoT Messaging", Mqtt.org, 2021. [Online]. Available: https://mqtt.org/. [Accessed: 19- Jul- 2021].

[2] "Eclipse Mosquitto", Eclipse Mosquitto, 2021. [Online]. Available: https://mosquitto.org/. [Accessed: 19- Jul- 2021].

[3] "Node-RED", Nodered.org, 2021. [Online]. Available: https://nodered.org/. [Accessed: 19- Jul- 2021].

[4] "InfluxDB: Purpose-Built Open Source Time Series Database | InfluxData", InfluxData, 2021. [Online]. Available: https://www.influxdata.com/. [Accessed: 19- Jul- 2021].

[5] "Grafana", Grafana.org, 2021. [Online]. Available: https://grafana.com/. [Accessed: 19- Jul- 2021].

[6] "MyQttHub -- Servidor MQTT Cloud -- Alojamiento MQTT", Myqtthub.com, 2021. [Online]. Available: https://myqtthub.com/. [Accessed: 19- Jul- 2021].

Índice de la memoria

Índice de Ilustraciones

Índice de tablas

Capítulo 1. INTRODUCCIÓN

En la actualidad, el IoT (Internet of things) se ha convertido en una de las tecnologías más importantes. Cada vez está más presente en nuestra sociedad debido a las numerosas aplicaciones que ofrece, para nuestra vida cotidiana como para su empleo en ámbitos industriales.

Fue un profesor del MIT, llamado Kevin Ashton quien usó por primera vez en 2009 el término "*Internet of Things*". El IoT consiste en conectar dispositivos entre sí a través de una red (pública o privada), con el fin de que estos puedan interactuar entre sí sin necesidad de intervención humana, lo que actualmente se conoce como (M2M, Machine to Machine). De esta forma, los dispositivos son capaces de recolectar y usar información por su cuenta. El potencial de la tecnología IoT reside en la adquisición masiva de datos del mundo físico, a partir de sensores u objetos cotidianos con dispositivos integrados. Permitiendo así, la interacción directa entre el mundo físico y digital sin implicación humana.

La aplicación de esta tecnología en la industria se conoce como IIoT (Industrial Internet of Things) y junto con los avances en Big Data, robótica y Machine Learning, se considera que estamos ante la Cuarta Revolución Industrial. Ya que la industria se enfrenta a nuevos modelos de negocio y a una nueva automatización.

Con este proyecto se pretende llevar a cabo un trabajo de investigación y desarrollo. Para ello se comenzará por adquirir conocimientos generales de IoT y su aplicación en el sector industrial. Posteriormente, se investigarán las posibles herramientas que ofrece el IoT y se desarrollará con ellas una solución IoT de aplicación industrial que pueda servir como base a futuros desarrollos. Dicha aplicación consistirá en la monitorización y supervisión del rendimiento de la simulación de una pasteurizadora.

1.1 MOTIVACIÓN DEL PROYECTO

Este proyecto ha sido propuesto por la empresa Zeus Control, dedicada a la automatización y control de procesos industriales. La finalidad de este proyecto es empezar a familiarizarse y abrir el camino a la industria 4.0, la cual plantea una nueva automatización y modelos de negocio. El objeto es estudiar las posibilidades que ofrece el IoT y aplicarlas a soluciones reales de Zeus Control. Viendo así, como esas nuevas tecnologías pueden aportar valor y dar soluciones innovadoras. Se pretende realizar un acercamiento a términos como industria conectada, adquisición masiva de datos y gestión de la información.

Aunque el sector industrial se encuentra en continuo desarrollo, ya tiene soluciones eficaces y robustas ante muchos tipos de necesidades. Dado que ya tiene soluciones optimizadas y seguras, la integración de innovaciones en la industria a gran escala debe aportar una mejora considerable y numerosas ventajas frente a las soluciones anteriores. Por todo esto, se considera que la industria es bastante lenta a la hora de implementar nuevas tecnologías, ya que los cambios pueden llegar a ser arriesgados y dar nuevos problemas desconocidos. No obstante, está obligada a adecuarse a las nuevas tecnologías y necesidades de cada sector.

Es por eso, que antes de incorporar grandes innovaciones en la industria, es necesario llevar a cabo una investigación a fondo de las nuevas tecnologías emergentes. Para ello es necesario estudiar las posibles aplicaciones que se les puede dar y compararlas con las soluciones previas para analizar la rentabilidad del cambio a nuevas soluciones. También es necesario poner a prueba dichas tecnologías con aplicaciones sencillas para verificar su correcto funcionamiento y enfrentarse a los posibles errores y dificultades que pueden plantearse a la hora del desarrollo. Los fabricantes están adaptando sus productos a esta nueva tecnología, pero como integradores debemos garantizar que su aplicación no reduce la seguridad y la calidad de nuestras soluciones.

Con este proyecto se pretende intentar reproducir un sistema, que actualmente se gestionaría con sistemas de control tradicional, utilizando las nuevas herramientas y posibilidades de IoT: conectividad, adquisición de datos, deslocalización, acceso remoto y envío a sistemas cloud. Se busca aportar valor a las soluciones, aumentando prestaciones, accesibilidad o reducción de costes.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este proyecto se han investigado las herramientas y tecnologías que se emplean hoy en día en el desarrollo de soluciones IoT. A continuación, se describirán las tecnologías que han resultado ser más interesantes para su aplicación en el desarrollo del ejemplo de aplicación industrial de este proyecto.

2.1 PROTOCOLO MQTT

(***Message Queuing Telemetry Transport***) El protocolo MQTT es uno de los sistemas de conectividad estándar del IoT, creado por la organización Oasis. Su diseño está centrado en minimizar el ancho de banda de red y recursos requeridos por los dispositivos. Esto hace que sea posible utilizarse en microcontroladores pequeños, lo que lo hace idóneo para el IoT y la conexión de dispositivos remotos.

El protocolo MQTT posee una arquitectura de publicación/subscripción para el transporte de mensajes tal y como se muestra en el ejemplo de la Ilustración 5. Como podemos ver, la estructura consta de cliente publicador, bróker (servidor) y cliente subscriptor. También se puede observar que los mensajes se envían bajo tópicos, estos tópicos asociados a los mensajes sirven para describir y ordenar los contenidos de los mensajes, y para que los subscriptores se subscriban a los tópicos de los mensajes que quieren recibir. En este ejemplo, el cliente publicador envía bajo el tópico “temperature” la información captada por un sensor de temperatura al bróker, después este se encarga de direccionar los mensajes a los clientes subscriptores subscritos al tópico “temperature”.

Otra de las ventajas del MQTT es que permite realizar una comunicación segura mediante cifrado TLS y autenticación de clientes con usuario y contraseña. Además, el protocolo MQTT tiene 3 niveles de calidad de servicio (QoS) para garantizar la entrega de mensajes:

- QoS=0, garantiza que el mensaje se envíe como máximo una vez.
- QoS=1, garantiza que el mensaje se envíe al menos una vez.
- QoS=2, garantiza que el mensaje se envíe exactamente una vez.

En este proyecto utilizaremos MQTT v3.1.1 como uno de los protocolos de comunicación IoT para la realización del desarrollo.

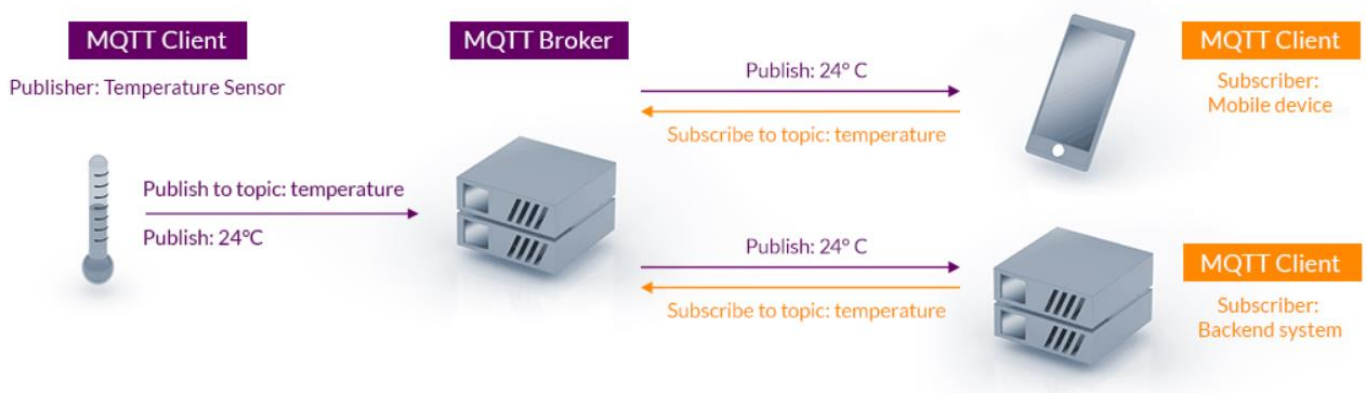


Ilustración 5: Arquitectura del protocolo MQTT (mqtt.org)

2.2 MOSQUITTO

Es un bróker de protocolo MQTT de código abierto creado por Eclipse Foundation. Mosquitto es un bróker que nos permite una amplia libertad de configuración que nos permite obtener un bróker personalizado con las funcionalidades que buscamos. También proporciona clientes MQTT publicadores y subscriptores de la línea de comandos, mosquitto_pub y moquitto_sub.

Vamos a emplear mosquitto v2.0.9 como bróker de las comunicaciones MQTT que realicemos en el desarrollo de este proyecto. Mosquitto es un servidor local que para este proyecto se instalará en un ordenador, aunque también se podría instalar en una RaspBerry Pi. Su función será recibir los datos enviados por los clientes publicadores y enviarlos a los clientes subscritos a los tópicos correspondientes.

2.3 PLC SIMATIC S7-1500

Autómata programable de SIEMENS. Para este proyecto se empleará la CPU 1513-1 PN, 6ES7 513-1AL02-0AB0 con versión de firmware V2,5. En un primer lugar, se programará un generador de distintas señales y estas se irán transmitiendo mediante una librería de SIEMENS por protocolo MQTT. En segundo lugar, se programará para realizar un cálculo simplificado del OEE (Overall Equipment Effectiveness), que es una razón porcentual de la eficiencia productiva de un proceso industrial. También se programará un modelo con el que simular el cálculo del OEE y se enviarán los datos por enlaces S7.

2.4 COMUNICACIÓN POR ENLACES S7

Se trata de una forma de comunicación única y específica de equipos SIEMENS. Esta comunicación nos permite leer y escribir datos con dispositivos conectados en la misma red. En este proyecto se empleará para realizar el envío de datos del PLC en la simulación del OEE para llevar a cabo una monitorización del rendimiento en tiempo real. Se decidió emplear este sistema de comunicación por su simple uso tanto en lectura como escritura de datos.

2.5 TIA PORTAL

Son las siglas de Totally Integrated Automation, un software que es propiedad de SIEMENS y se emplea en la programación e integración de componentes de máquinas para poder llevar a cabo el control de procedimientos y operaciones. En este proyecto se empleará para la programación del PLC S7-1500, con su generador de señales, se descargará e implantará la librería "LMQTT_Client" para el envío de datos a través del protocolo MQTT y se desarrollará el modelo de simulación y cálculo del OEE.

2.6 NODE-RED

Es una herramienta de programación de código abierto perteneciente a JS Foundation basada en flujo, que permite conectar dispositivos de hardware, API y servicios en línea. Tiene un tiempo de ejecución basado en Node.js (Entorno de JavaScript con arquitectura basada en eventos). Su programación de flujo permite obtener una buena representación visual, tal y como se puede ver en la Ilustración 6. Como podemos observar, el flujo está formado por nodos, que son bloques de programación predeterminados con distintas funciones. Además, Node-Red tiene una comunidad donde la gente comparte nodos creados, los cuales se pueden descargar desde la biblioteca Node-Red, lo que permite al usuario programar de una manera rápida y sencilla.

En este proyecto se ha usado Node-Red con su función de cliente subscriptor la bróker para recibir los datos enviados por el PLC por protocolo MQTT. También se han empleado nodos que permiten recibir y enviar los datos del PLC por comunicación S7. Una vez recibidos los datos del PLC, Node-Red nos permitía almacenar los datos en distintas bases de datos e incluso realizar un dashboard para monitorizar el OEE en tiempo real, y llevar a cabo la simulación del modelo.

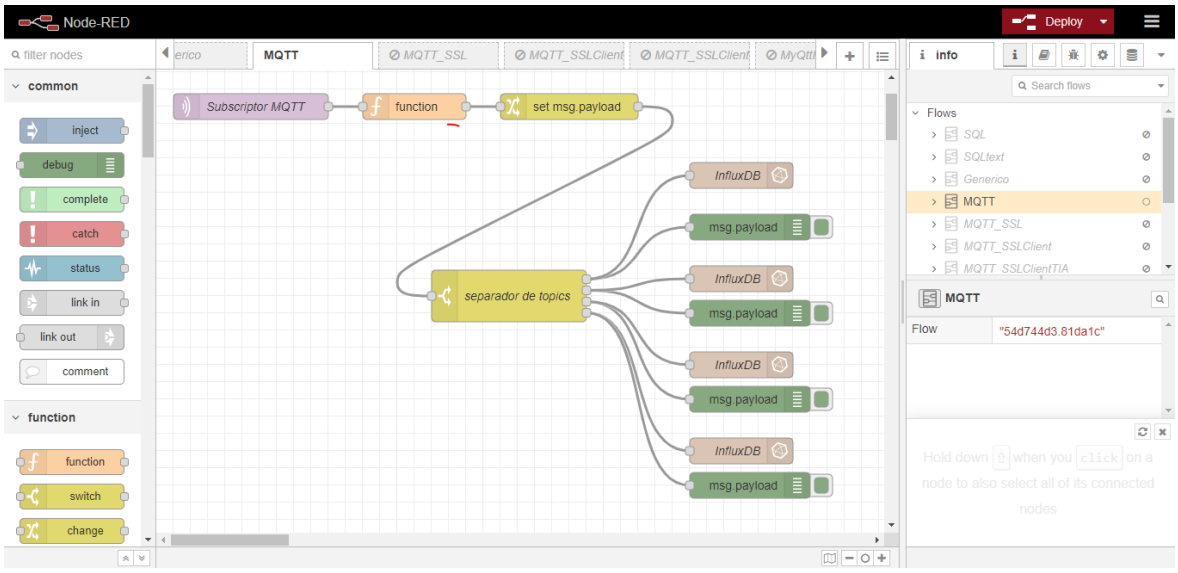


Ilustración 6: Visualización de Node-Red

2.7 GRAFANA

Es una herramienta de software libre desarrollada en Lenguaje GO y Node.js LTS con licencia Apache 2.0. Sirve como medio de compartir y acceder a datos, es una forma de unificar los datos de distintas fuentes y lugares, permitiendo a un equipo tener un acceso global a los datos para poder trabajar juntos con los datos. Ofrece múltiples formas de visualización y control de datos altamente personalizables, permitiendo hacer análisis de datos e incluso establecer alarmas y avisos. En este proyecto se empleará para el acceso y visualización de las bases de datos, tal y como se muestra en la Ilustración 7.

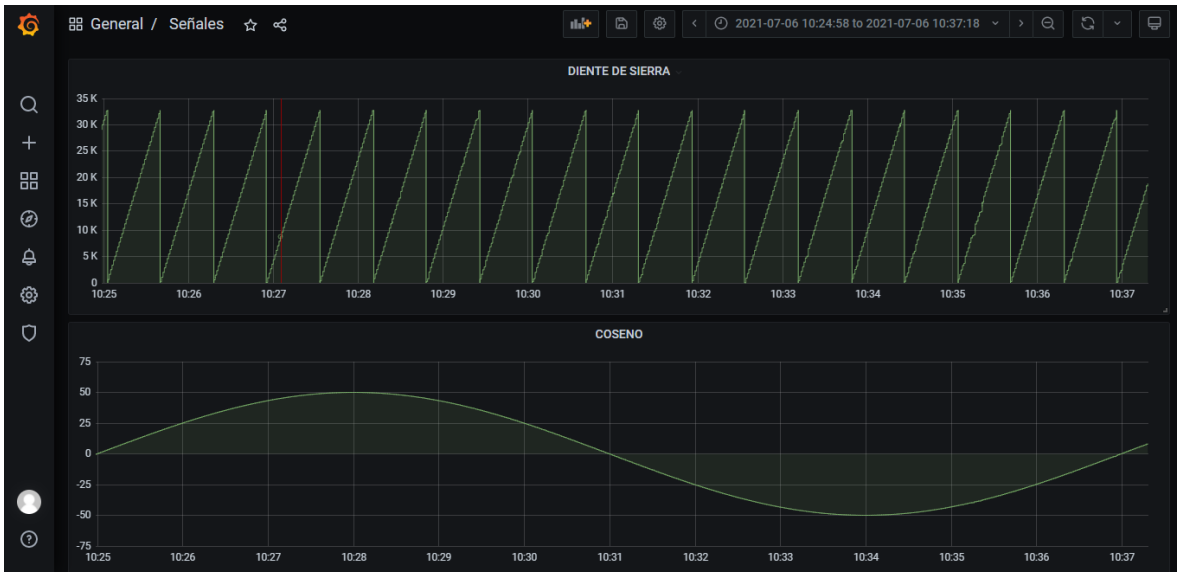


Ilustración 7: Visualización Grafana

2.8 SQL SERVER EXPRESS, INFLUXDB

Estas son las bases de datos que se emplearon en este proyecto. Node-Red se encargaba de introducir los datos en estas, y posteriormente Grafana accedía a las mismas para llevar a cabo la representación y visualización de los datos.

SQL Server Express es la versión gratuita de SQL Server, herramienta de gestión de base de datos creada por Microsoft. Emplea el lenguaje Transact-SQL para la creación de tablas, manipulación y recuperación de datos.

InfluxDB es una base de datos de series de tiempo, que resulta ser mucho más cómoda y práctica para soluciones IoT de monitorización y análisis. Además, está diseñada para grandes volúmenes de datos y distintas fuentes de datos con marca de tiempo.

2.9 OPENSSL

Es un proyecto de software libre, con una licencia que lo permite ser usado con propósito comercial. Está formado por un conjunto de herramientas con todas las funciones de los protocolos TLS (Transport Layer Security) y SSL (Secure Sockets Layer). Además, se emplea generalmente como biblioteca criptográfica.

En este proyecto se empleará OpenSSL para poder establecer una conexión segura mediante el protocolo MQTT. Para ello se usan los protocolos TLS y SSL que tienen como fin encriptar y firmar los mensajes. De esta forma sabemos que nadie será capaz de leer nuestros mensajes y que los mensajes que nos son enviados provienen de donde o quien esperábamos. Esto es de vital importancia si se le quiere dar una aplicación industrial, ya que las empresas no quieren que nadie pueda acceder a sus datos.

2.10 MYQTTHUB

Es una plataforma Cloud MQTT altamente escalable con opciones de soporte profesional. Resulta muy útil para construir tu propia solución IoT, conectando dispositivos e intercambiando mensajes por protocolo MQTT.

MyQttHub ofrece una gran variedad de planes para ajustarse a las necesidades del cliente. Desde subscripciones anuales de 2,5€/mes hasta 39€/mes para usos más profesionales. Estos planes dependen de la cantidad máxima de usuarios, almacenamiento, paneles, conexiones y mensajes.

En este proyecto se empleará la versión gratuita de MyQttHub para poder probar la solución de emplear un bróker cloud, a diferencia del bróker mosquitto, que es local (el bróker es el ordenador). Con esto ya no es necesario realizar una conexión por cable Ethernet entre los dispositivos clientes y el bróker. Se empleará para poder enviar información del rendimiento de una planta, en tiempo real, a un Smartphone que pueda encontrarse en cualquier lugar con conexión a internet.

2.11 MQTT DASH

Es una App gratuita que se puede obtener en Play Store. Esta App permite emplear los dispositivos Smartphone como clientes MQTT, tanto como publicador como subscriptor.

En este proyecto se empleará para configurar un cliente subscriptor en un Smartphone que reciba los datos del rendimiento de la planta, desde el bróker MyQttHub y los represente de una forma atractiva.

Capítulo 3. ESTADO DE LA CUESTIÓN

El IoT es uno de los avances tecnológicos más prometedores hoy en día, el cual consiste en la conexión de múltiples dispositivos mediante internet, para que estos puedan interactuar entre sí sin que sea necesaria intervención humana. Con la llegada de la Industria 4.0 ha aumentado la oferta de "Smart sensors", sensores con circuitos electrónicos integrados, compatibles con los distintos protocolos de comunicación IoT. Esto permite una adquisición masiva de datos que puede ser compartida con todos los dispositivos conectados por IoT. El procesamiento de estos datos es una de las utilidades fundamentales del IoT y con la que se pueden desarrollar infinidad de soluciones y aplicaciones en ámbitos tanto cotidianos como industriales.

En el sector industrial, los datos obtenidos son de vital importancia para poder llevar a cabo análisis del funcionamiento de máquinas, líneas e incluso fábricas completas, desde sistemas de monitorización en tiempo real para tomar decisiones de producción, hasta el uso de Big Data y Machine Learning para un procesado a fondo de los datos con el fin de desarrollar soluciones más complejas que optimicen en gran medida los procesos. Una de las aplicaciones más comentadas del IoT en el sector industrial es el mantenimiento predictivo de máquinas. A través de los sensores, se manda la información a la red, obteniendo una amplia base de datos, la cual se puede procesar y entrenar para predecir cuándo va a haber algún error en la máquina y poder evitarlo antes de tiempo.

El protocolo MQTT es uno de los más empleados en el sector industrial a la hora de enviar datos a la nube debido a que se puede emplear en equipos con funcionalidad mínima para comunicar a través de redes con bajo ancho de banda y alta latencia por TCP/IP. Es un protocolo ligero, con baja carga de transporte que ofrece la posibilidad de encriptar los mensajes por SSL/TLS. Su estructura está formada por un servidor(bróker) y clientes publicadores y subscriptores.

La mayor parte de las empresas relacionadas con el sector de la industria ya son conscientes del gran potencial del IoT, y ofrecen productos y soluciones IoT. Desde hardware y software, hasta soluciones de plataformas IoT ya desarrolladas. Como, por ejemplo:

- IBM Watson IoT
- Amazon AWS IoT Core
- Microsoft Azure IoT Hub

Todas son plataformas IoT ya desarrolladas, de empresas distintas, con la misma función. Un servicio gestionado y alojado en cloud con funciones de registro de dispositivos, visualización rápida y almacenamiento de datos. Conecta los dispositivos IoT a través de MQTT y HTTPS.

En este proyecto se realizará una solución con herramientas de código abierto de licencia gratuita, que sirva como alternativa económica a empresas pequeñas para que puedan beneficiarse de las ventajas del IoT en soluciones particulares. De la misma forma, la solución de monitorización a través del IoT llevada a cabo en este proyecto, podría ser una alternativa para soluciones independientes en las que no sea rentable comprar una licencia SCADA.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como ya se ha mencionado anteriormente, este proyecto comenzó por una fase de investigación y estudio de la tecnología IoT, así como sus herramientas. Una vez realizado dicho estudio, se pretendía valorar algunas de las herramientas con las que poder llevar a cabo una solución de aplicación industrial.

Tras el análisis de las nuevas herramientas y soluciones, se llegó a la conclusión que, hoy en día, donde reside el mayor potencial del IoT, es en soluciones de monitorización y visualización de datos en tiempo real, adquisición y almacenamiento de datos, y la facilidad de compartir dichos datos. Sin embargo, las posibles soluciones para controles y actuaciones en plantas de procesos industriales aún necesitan desarrollos más complejos y llevar a cabo un estudio de su robustez para hacer frente a las soluciones de control tradicionales. A pesar de todo ello, las cualidades de conectividad que ofrece el IoT pueden resultar muy útiles también para aplicarse en este ámbito, y conforme siga avanzando esta tecnología, se llegará a soluciones viables y ventajosas.

En este proyecto se ha desarrollado una solución de almacenamiento y monitorización en tiempo real de datos de dispositivos industriales mediante tecnología IoT. Esta solución está planteada como posible alternativa a una solución SCADA (Supervisory Control And Data Acquisition) ante distintas necesidades.

La solución propuesta en este proyecto supone una alternativa más económica para soluciones sencillas en las que un SCADA no sea rentable. Es cierto que ante soluciones complejas un SCADA es mejor opción, ya que permite la configuración de herramientas para tratamiento de alarmas, creación de sinópticos e historizado de datos sin necesidad de desarrollo adicional.

Esta solución también puede ser útil para llevar a cabo una monitorización e historizado de lecturas sin control del proceso, como la monitorización del consumo de energía, nivel y temperatura de tanques, temperaturas de un almacén, etc. Con la integración de Smart sensors con tecnología IoT esta solución es muy beneficiosa.

Del mismo modo, esta solución puede ser ventajosa para la adquisición de señales o información de un conjunto de máquinas que dispongan de control propio. Algunos ejemplos pueden ser la adquisición de datos del OEE de una línea de fabricación, registro de unidades producidas, medidas de calidad de productos, etc.

Otra ventaja de esta solución es que es muy flexible, se puede emplear en plantas descentralizadas con máquinas de distintos fabricantes e incluso en plantas con maquinaria antigua.

COMPARACIÓN ECONÓMICA ENTRE LA SOLUCIÓN IOT Y SCADA

A continuación, se muestra la comparativa entre una posible oferta para la solución de monitorización y visualización de señales de un PLC mediante las herramientas IoT y la oferta equivalente con la solución SCADA.

Como se puede observar en la Tabla 1, para la solución de este proyecto los componentes necesarios son: Un PLC con su Memory card, la licencia de TIA portal para poder programar el PLC, un PC situado en la planta, encargado de recibir los datos y almacenarlos en bases de datos (Si el PC tiene pantalla también valdría como panel), ya sea por MQTT o por enlace S7 y un ordenador para llevar a cabo el desarrollo e incluso el control remoto de los datos.

En cambio, en la Tabla 5, vemos que además de los componentes ya mencionados, para obtener una solución equivalente, es necesario añadir un panel de operador, así como las licencias de SCADA y de servidor remoto tanto para el uso del cliente, como para su desarrollo. Es aquí donde está la verdadera diferencia económica entre ambas soluciones.

Tanto en la Tabla 2 y Tabla 6 como en la Tabla 3 y Tabla 7 podemos observar que no hay una gran diferencia económica entre ambas soluciones. La única diferencia son las horas de programación del panel, que en la solución IoT están incluidas dentro de las horas de desarrollo IoT con las mismas herramientas de visualización.

En la Tabla 4 y Tabla 8 vemos el coste total de ambas soluciones, resultando la solución SCADA aproximadamente el doble de costoso que la IoT. Por esta diferencia notable, pensamos que la solución IoT puede ser una buena alternativa ante soluciones sencillas y de pequeña escala.

Solución IoT

Suministro	Referencia	Cantidad	Coste T.	Precio U.	Precio T.	
SIMATIC S7-1500, CPU 1513-1 PN	6ES7513-1AL02-0AB0	1	1.291,56 €	1.518,87 €	1.518,87 €	Hardware
SIMATIC S7, Memory Card	6ES7954-8LC03-0AA0	1	40,22 €	47,30 €	47,30 €	
SIMATIC STEP 7 Prof. V16	6ES7822-1AA07-0YA5	1	1.686,71 €	1.983,57 €	1.983,57 €	
			- €	- €	- €	
SIMATIC IPC127E; Atom E3930	6AG4021-0AA11-0AA0	1	470,96 €	553,85 €	553,85 €	PC Local
			- €	- €	- €	
ORDENADOR SOBREMESA		1	550,00 €	646,80 €	646,80 €	Sala control/remoto
MONITOR 24"		1	175,00 €	205,80 €	205,80 €	
Total Materiales			**4.214,45 €**		**4.956,19 €**	

Tabla 1: Suministros solución IoT

Mano de obra	Cantidad	Coste U.	Coste T.	Precio U.	Precio T.
Técnico de Proyecto: PLC	40	39,00 €	1.560,00 €	48,05 €	1.921,92 €
Técnico de Proyecto: IoT	80	39,00 €	3.120,00 €	48,05 €	3.843,84 €
Tecnico eléctrico: Ingeniería	32,0	39,00 €	1.248,00 €	48,05 €	1.537,54 €
MO montaje: horas normales	56,0	29,00 €	1.624,00 €	34,10 €	1.909,82 €
Técnico de Proyecto: horas ingeniería	45	39,00 €	1.755,00 €	48,05 €	2.162,16 €
Técnico de Proyecto: horas extraordinarias	5	39,00 €	195,00 €	62,46 €	312,31 €
Total Mano de Obra			**9.502,00 €**		**11.687,59 €**

Tabla 2: Mano de obra solución IoT

Gastos 2 Puesta en marcha	Cantidad	Coste U.	Coste T.	Precio U.	Precio T.
Dietas técnicos	5	59,00 €	295,00 €	66,08 €	330,40 €
Km coche	700	0,27 €	190,91 €	0,31 €	213,82 €
Hoteles	4	65,00 €	260,00 €	72,80 €	291,20 €
Total gastos 2			**745,91 €**		**835,42 €**

Tabla 3: Gastos implantación IoT

ITEM	PRECIO
Suministro	4.956,19 €
Ingeniería y programación	5.765,76 €
Esquemas eléctricos	1.537,54 €
Montaje de envolventes	1.909,82 €
Montaje en planta	- €
Puesta en marcha	3.309,89 €
Contratas	- €
TOTAL	**17.479,20 €**

Tabla 4: Total solución IoT

Solución SCADA

Suministro	Referencia	Cantidad	Coste T.	Precio U.	Precio T.	
SIMATIC S7-1500, CPU 1513-1 PN	6ES7513-1AL02-0AB0	1	1.291,56 €	1.518,87 €	1.518,87 €	Hardware
SIMATIC S7, Memory Card	6ES7954-8LC03-0AA0	1	40,22 €	47,30 €	47,30 €	
SIMATIC STEP 7 Prof. V16	6ES7822-1AA07-0YA5	1	1.686,71 €	1.983,57 €	1.983,57 €	
			- €	- €	- €	
SIMATIC HMI TP1200 Comfort	6AV2124-0MC01-0AX0	1	1.806,54 €	2.124,49 €	2.124,49 €	Opción Panel de operador
			- €	- €	- €	
SIMATIC WinCC Professional 4096 PowerTags V16	6AV2103-0HA06-0AA5	1	2.066,15 €	2.429,79 €	2.429,79 €	Opción SCADA
SIMATIC WinCC RT Professional, 4096 PowerTags V16	6AV2105-0HA06-0AA0	1	3.902,40 €	4.589,22 €	4.589,22 €	
ORDENADOR SOBREMESA		1	550,00 €	646,80 €	646,80 €	
MONITOR 24"		1	175,00 €	205,80 €	205,80 €	
			- €	- €	- €	
SIMATIC WinCC Server para Runtime Professional Opción para WinCC	6AV2107-0EB00-0BB0	1	2.596,46 €	3.053,44 €	3.053,44 €	Acceso remoto web
SIMATIC WinCC/Web Navigator, 1 licencia de cliente	6AV6362-1AB00-0BB0	1	2.564,87 €	3.016,29 €	3.016,29 €	
Total Materiales			**16.679,91 €**		**19.615,57 €**	

Tabla 5: Suministros solución SCADA

Mano de obra	Cantidad	Coste U.	Coste T.	Precio U.	Precio T.
Técnico de Proyecto: PLC	40	39,00 €	1.560,00 €	48,05 €	1.921,92 €
Técnico de Proyecto: PANEL	40	39,00 €	1.560,00 €	48,05 €	1.921,92 €
Técnico de Proyecto: SCADA	80	39,00 €	3.120,00 €	48,05 €	3.843,84 €
Tecnico eléctrico: Ingeniería	32,0	39,00 €	1.248,00 €	48,05 €	1.537,54 €
MO montaje: horas normales	56,0	29,00 €	1.624,00 €	34,10 €	1.909,82 €
Técnico de Proyecto: horas ingeniería	45	39,00 €	1.755,00 €	48,05 €	2.162,16 €
Técnico de Proyecto: horas extraordinarias	5	39,00 €	195,00 €	62,46 €	312,31 €
Total Mano de Obra			**11.062,00 €**		**13.609,51 €**

Tabla 6: Mano de obra solución SCADA

Gastos 2 Puesta en marcha	Cantidad	Coste U.	Coste T.	Precio U.	Precio T.
Dietas técnicos	5	59,00 €	295,00 €	66,08 €	330,40 €
Km coche	700	0,27 €	190,91 €	0,31 €	213,82 €
Hoteles	4	65,00 €	260,00 €	72,80 €	291,20 €
Total gastos 2			**745,91 €**		**835,42 €**

Tabla 7: Gastos implantación SCADA

ITEM	PRECIO
Suministro	19.615,57 €
Ingeniería y programación	7.687,68 €
Esquemas eléctricos	1.537,54 €
Montaje de envolventes	1.909,82 €
Montaje en planta	- €
Puesta en marcha	3.309,89 €
Contratas	- €
TOTAL	**34.060,50 €**

Tabla 8: Total solución SCADA

4.2 OBJETIVOS

El objetivo principal de este proyecto es utilizar las herramientas y filosofía de la tecnología IoT para obtener una aplicación industrial que permita la adquisición y tratamiento de datos de un proceso productivo. Para ello se pretenden lograr los siguientes propósitos:

- Programar un PLC S7-1500 de SIEMENS para la generación de distintas señales con el fin de evaluar su correcto envío y creación de un módulo de cálculo simplificado del rendimiento general de una máquina (OEE) y respectivo modelo de simulación.
- Creación de una librería para el envío de varias de las señales generadas por el protocolo MQTT de forma paralela y automática.
- Configuración de un bróker local para gestionar la recepción y envío de datos. Que funcione para el envío de datos desde el PLC.
- Prueba de un bróker cloud para el envío de datos por protocolo MQTT sin necesidad de conexión física (Ethernet) a distintos clientes.
- Estudio de distintas opciones para el almacenamiento de datos. También se creará un dashboard/aplicación para la visualización de los datos almacenados en la plataforma cloud. Se configurará con control de datos y acceso a datos para que tenga utilidad comercial.
- Finalmente, se pretende llevar a cabo una aplicación práctica de este desarrollo como base para soluciones reales. En concreto se pretende llevar a cabo la simulación y cálculo del OEE de una línea de pasteurización. El PLC realizará tanto la simulación como el cálculo. Una vez realizados los cálculos oportunos se enviarán los resultados obtenidos por la comunicación de enlaces S7 y el protocolo MQTT, y se probará la aplicación de supervisión para monitorizar el rendimiento de la pasteurizadora en tiempo real desde el ordenador y desde un cliente Smartphone.

4.3 METODOLOGÍA

1. **Investigación:** Se llevará acabo toda la investigación necesaria. Estudiando los distintos tipos de protocolos de comunicación que se pueden emplear. En especial el protocolo MQTT para su posterior aplicación. También se estudiarán las distintas posibilidades para el almacenamiento de datos en el cloud con propósito industrial. Igualmente se compararán las posibilidades para el acceso y supervisión de los datos almacenados en dicho cloud.

2. **Programación PLC:** Se aprenderá a usar el TIA Portal para llevar a cabo la programación de los autómatas programables de la familia S7-1200 o S7-1500. Se programará para el tratamiento de señales y gestión de contadores.

3. **Desarrollo librerías MQTT:** Se desarrollarán librerías para poder estructurar y facilitar las funciones de la comunicación MQTT.

4. **Estudio y uso de bases de datos:** Se valorarán distintas bases de datos para el almacenamiento de estos. La base de datos más adecuada se integrará en la solución.

5. **Desarrollo dashboard:** Se implementará la interfaz para la visualización y control de datos en el cloud. Permitiendo llevar a cabo una solución de monitorización real que se pueda aplicar en el ámbito industrial.

6. **Prueba Conjunta:** Se procederá a integrar los desarrollos mencionados anteriormente en la simulación de una línea de producción. De esta forma, se comprobará la funcionalidad de la tecnología IoT en un ámbito industrial. Para dicha simulación se emplearán una CPU S7-1500, el TIA Portal para la simulación de señales y el cloud y dashboard desarrollados.

7. **Desarrollo de memoria:** Una vez acabado todo el desarrollo del proyecto se procederá a la revisión y finalización del documento escrito.

4.4 PLANIFICACIÓN

La planificación y reparto de tareas de este proyecto, mencionadas en el apartado anterior, se muestran a continuación en el cronograma de la Tabla 9. Donde se muestra los distintos plazos en los que se realizaron las distintas tareas.

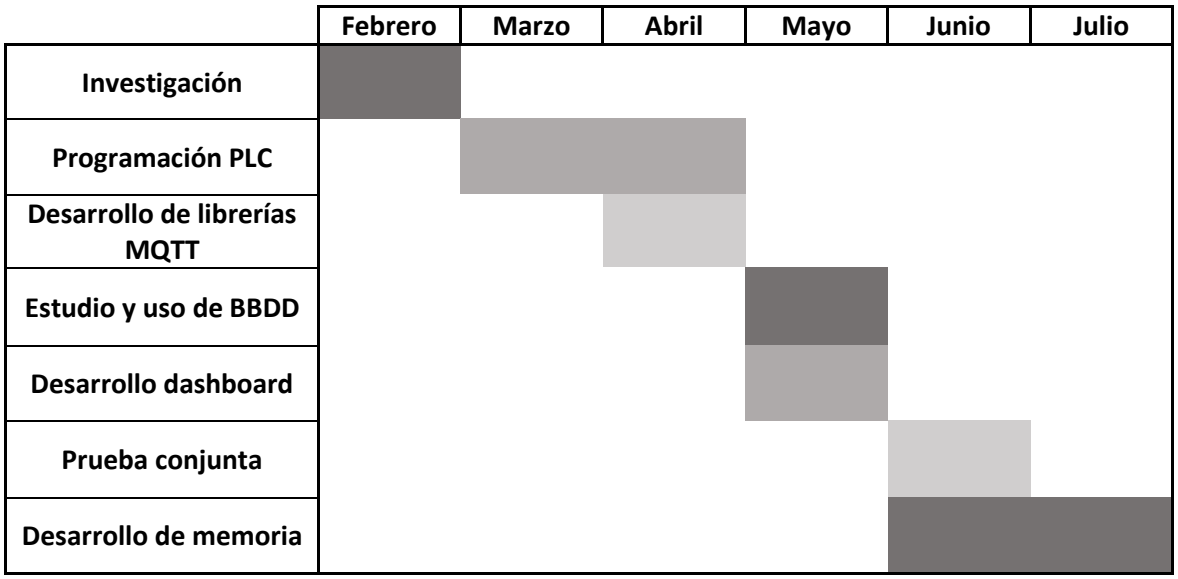


Tabla 9: Cronograma

Capítulo 5. SISTEMA/MODELO DESARROLLADO

En este capítulo se va a explicar de forma detallada, el desarrollo realizado con las tecnologías que han resultado más interesantes para poder llevar a cabo una aplicación industrial con ellas. Se pretende mostrar las configuraciones y procedimientos realizados para que cualquier persona que lea este documento sea capaz de llevar a cabo este desarrollo y tener un proyecto base para futuros desarrollos IoT.

En primer lugar, se explicará el desarrollo de la solución de envío de datos desde el PLC por protocolo MQTT para su almacenamiento en una base de datos y la posterior monitorización de estos desde la plataforma Grafana.

En segundo lugar, se explicará el desarrollo del ejemplo de solución real con la simulación del cálculo del OEE de una pasteurizadora y el envío de datos a través de la comunicación por enlaces S7. Para llevar a cabo la monitorización en tiempo real del rendimiento de la máquina mediante Node-Red y Grafana.

5.1 ANÁLISIS DEL SISTEMA DE ENVÍO DE SEÑALES POR MQTT

El primer modelo realizado en este proyecto hará uso del protocolo MQTT, para la comunicación del PLC con el bróker mosquitto. Para ello será necesario implementar la librería “LMQTT_Client” de SIEMENS. Esta librería ofrece una función de cliente MQTT, que en este modelo usaremos como publicador. La idea es crear una función para generar cuatro señales distintas y enviar los datos de dichas señales mediante la función MQTT a un subscriptor en Node-Red. Posteriormente se gestionará el almacenamiento de los datos recibidos en una base de datos y se empleará la plataforma Grafana para acceder a las bases de datos. Con ella se realizará una visualización de los datos obtenidos. De esta forma, se evaluará la aplicación del protocolo MQTT como forma de monitorización y observabilidad.

5.2 DISEÑO

En este apartado se explicarán paso a paso los procedimientos realizados con cada una de las herramientas para poder llevar a cabo la realización de este modelo.

5.2.1 PROGRAMACIÓN EN TIA PORTAL DEL GENERADOR DE SEÑALES

Para poder poner a prueba la funcionalidad del protocolo MQTT a la hora de monitorizar señales, se decidió enviar distintas señales que fueran interesantes de visualizar. Por ello se decidió programar una función de generación de señales.

Se quería que esta función fuera capaz de generar varias señales de distintos tipos, por lo que previamente se decidió crear un tipo de dato que contuviese las variables necesarias para la generación de dicha señal. Este tipo de dato es T_SEÑAL y se muestra en la Ilustración 8, donde podemos ver las siguientes variables:

- ORD_ON: Orden para activar y desactivar la señal.
- AUMENTAR: Orden de aumentar la señal un incremento (Solo útil en modo manual).
- DISMINUIR: Orden de disminuir la señal un incremento (Solo útil en modo manual).
- SP_INC: Incremento que decide aplicarse a la señal o amplitud de la señal (seno y coseno).
- MODO: Toma un valor del 1-4 para seleccionar el tipo de señal que se desea generar:
    - MODO=1: Señal manual. Se genera mediante las variables AUMENTAR y DISMINUIR, sumando o restando el SP_INC progresivamente.
    - MODO=2: Seno. Genera un seno de amplitud SP_INC.
    - MODO=3: Coseno. Genera un coseno de amplitud SP_INC.
    - MODO=4: Diente de sierra. Genera un diente de sierra con pendiente SP_INC y de valor máximo el introducido en la función.
- SALIDA: Devuelve la salida de la señal generada.

T_SEÑAL

	Nombre	Tipo de datos	Valor predet.	Accesible d...	Escrib...	Visible en ..	Valor de a..
1	ORD_ON	Bool	false	☑	☑	☑	☐
2	AUMENTAR	Bool	false	☑	☑	☑	☐
3	DISMINUIR	Bool	false	☑	☑	☑	☐
4	SP_INC	Int	0	☑	☑	☑	☐
5	SALIDA	Int	0	☑	☑	☑	☐
6	MODO	Int	0	☑	☑	☑	☐

Ilustración 8: Tipo de dato T_SEÑAL

Una vez creado el tipo de dato generamos un vector del tipo T_SEÑAL con un amplio tamaño, ya que de él depende el número de señales que podamos generar con la función. El vector se muestra en la Ilustración 9, y vemos que ya tenemos la herramienta para manejar cada una de las distintas señales. Además, con este vector ya es posible programar la función SEÑALES, que emplea un bucle para recorrer este vector y así generar todas las señales al mismo tiempo.

La función SEÑALES la tenemos representada en la Ilustración 10, donde podemos observar que es una función genérica en la que introducimos el número de señales que deseamos generar y el tamaño digital máximo que queremos permitir que tengan las señales.

DB_SEÑAL (instantánea generada: 15/04/2021 17:12:02)

	Nombre	Tipo de datos	Valor de arranq...	Instantánea	Remanen...	Accesible d...	Escrib...	Visible en ..	Valor de a..
1	Static				☐	☐	☐	☐	☐
2	SEÑAL	Array[0..20] of "T_SEÑAL"			☑	☑	☑	☑	☐
3	SEÑAL[0]	"T_SEÑAL"			☑	☑	☑	☑	☐
4	ORD_ON	Bool	TRUE	TRUE	☑	☑	☑	☑	☐
5	AUMENTAR	Bool	FALSE	FALSE	☑	☑	☑	☑	☐
6	DISMINUIR	Bool	FALSE	FALSE	☑	☑	☑	☑	☐
7	SP_INC	Int	10	10	☑	☑	☑	☑	☐
8	SALIDA	Int	0	0	☑	☑	☑	☑	☐
9	MODO	Int	1	1	☑	☑	☑	☑	☐
10	SEÑAL[1]	"T_SEÑAL"			☑	☑	☑	☑	☐
11	SEÑAL[2]	"T_SEÑAL"			☑	☑	☑	☑	☐
12	SEÑAL[3]	"T_SEÑAL"			☑	☑	☑	☑	☐
13	SEÑAL[4]	"T_SEÑAL"			☑	☑	☑	☑	☐
14	SEÑAL[5]	"T_SEÑAL"			☑	☑	☑	☑	☐
15	SEÑAL[6]	"T_SEÑAL"			☑	☑	☑	☑	☐
16	SEÑAL[7]	"T_SEÑAL"			☑	☑	☑	☑	☐
17	SEÑAL[8]	"T_SEÑAL"			☑	☑	☑	☑	☐
18	SEÑAL[9]	"T_SEÑAL"			☑	☑	☑	☑	☐
19	SEÑAL[10]	"T_SEÑAL"			☑	☑	☑	☑	☐
20	SEÑAL[11]	"T_SEÑAL"			☑	☑	☑	☑	☐
21	SEÑAL[12]	"T_SEÑAL"			☑	☑	☑	☑	☐

Ilustración 9: Vector del tipo T_SEÑAL

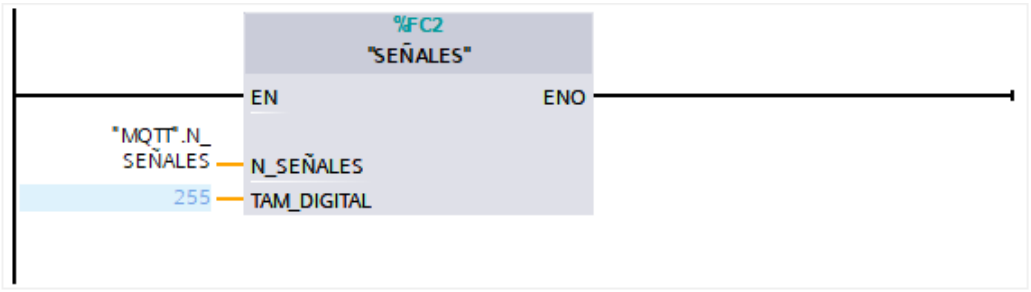


Ilustración 10: Función SEÑALES

Finalmente, se llevó a cabo la programación de un HMI para poder controlar y visualizar las señales generadas. En la Ilustración 11 podemos ver el HMI desarrollado en funcionamiento para el control de una de las señales generadas. Podemos observar que la interfaz tiene inputs para las variables de la señal. Un botón de activación de la señal, botones para seleccionar el tipo de señal que se desea, una barra deslizadora para establecer el incremento o amplitud de nuestras señales y botones de aumento y decremento para el control de la señal manual.

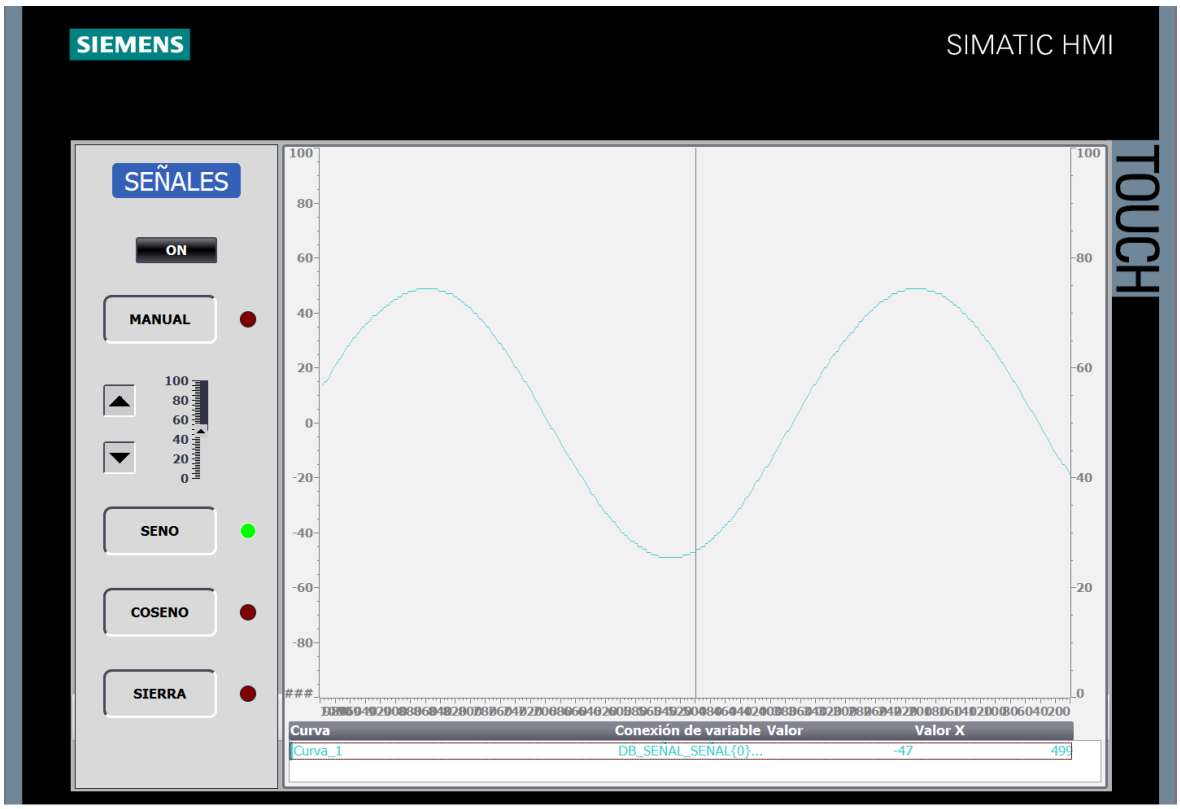


Ilustración 11: HMI para el control de señales

5.2.2 CONFIGURACIÓN DEL BRÓKER MOSQUITTO

Para poder llevar a cabo la comunicación por el protocolo MQTT es necesario tener un bróker/servidor que se encargue de recibir y gestionar él envió de los mensajes a los clientes subscritos a sus respectivos tópicos. En este proyecto se empleará el bróker mosquitto, el cual en este caso se descargará en un ordenador, aunque también podría implantarse en una RaspBerry e incluso en un servicio cloud. El problema es que el servicio cloud tiene un coste considerable y no compensa realizar con él las pruebas de funcionamiento del bróker mosquitto.

En primer lugar, se descarga el bróker mosquitto para un equipo Windows en este caso, aunque también puede hacerse en otros sistemas operativos. La descarga se realiza desde la página oficial: https://mosquitto.org/download/. En este proyecto se emplea la versión v2.0.9. Mosquitto puede descargarse como servidor, aunque para poder iniciarlo con la configuración que deseemos conviene implantarlo en modo manual.

Con la descarga de mosquitto nos encontramos con una serie de ejecutables, los más importantes son:

- *mosquitto.exe*: Lo usaremos para poner en marcha el servidor, indicando la configuración que queremos establecer.
- *mosquitto_pub.exe*: Sirve para enviar mensajes desde un cliente publicador.
- *mosquitto_sub.exe*: Sirve para iniciar un cliente subscriptor.
- *mosquitto_passwd.exe*: Sirve para crear los archivos de usuario y contraseña pertenecientes al servidor.

En este punto ya podemos probar el bróker y enviar y recibir mensaje con los clientes publicador y subscriptor. Sin embargo, esta comunicación se está realizando en local, aun no podríamos realizar la comunicación con otros dispositivos MQTT, como es el caso del PLC. Además, esta comunicación carece de seguridad, cualquier cliente podría publicar o subscribirse a nuestro bróker. Esto no es conveniente, y menos si se le quiere dar una aplicación industrial, ya que las empresas quieren mantener sus datos de forma segura. Por ello estableceremos que los clientes tengan que registrarse con usuario y contraseña.

Primero crearemos el archivo que contenga los usuarios y contraseñas aptos para nuestro servidor. Como se muestra en la Ilustración 12**¡Error! No se encuentra el origen de la referencia.**, tenemos que abrir la línea de comandos desde la dirección de mosquitto y usar el ejecutable *mosquitto_passwd.exe*. Con el comando -c se crea un primer archivo y hemos de facilitar el nombre del archivo (PASSWORD) y el nombre de usuario (Publicador_1), posteriormente se ha de introducir la contraseña y confirmarla. Aunque no se vea lo que se escribe, se está haciendo. Con el comando -b podemos introducir nuevos usuarios con sus respectivas contraseñas en el archivo previamente creado. Una vez acabada la creación de usuarios y contraseñas se puede abrir el archivo, este debería de tener la forma de la Ilustración 13**¡Error! No se encuentra el origen de la referencia.**, con los nombres de usuario seguidos de su contraseña encriptada.

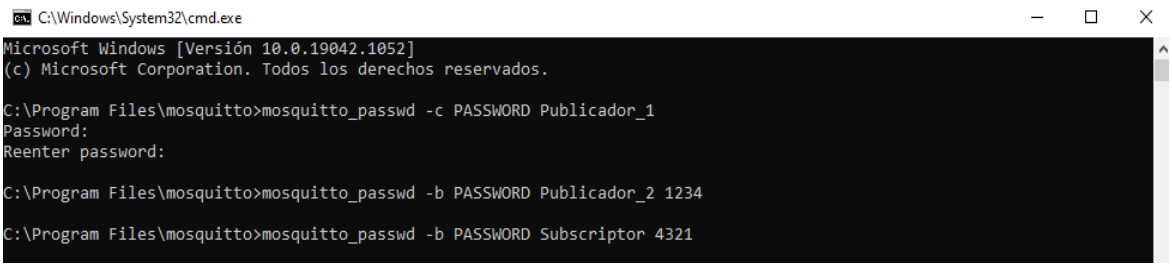


Ilustración 12: Creación del archivo de contraseñas

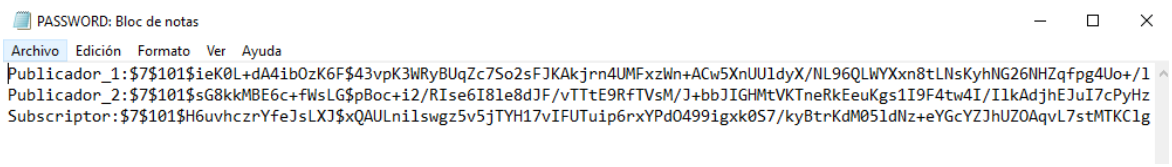


Ilustración 13: Resultado del archivo de contraseñas

Para permitir que mosquitto comunique con otros dispositivos externos es necesario emplear una configuración distinta a la predeterminada, por lo que debemos dirigirnos a la dirección donde hemos descargado mosquitto y editaremos el archivo mosquitto.conf. La edición se realizará eliminando los "#" cuando se desee descomentar alguna línea. En este caso será necesario editar solo 2 líneas:

- *listener 1883* para especificar el puerto en el que se realizará la conexión. También se puede especificar una dirección IP del bróker.
- *allow_anonymous true* de esta forma cualquier cliente puede comunicarse al bróker.

En segundo lugar, se ha de editar la configuración del bróker mosquitto. Para ello nos dirigimos a la dirección donde hemos descargado mosquitto y editamos el archivo mosquitto.conf. La edición se realizará eliminando los "#" cuando se desee descomentar alguna línea. En este caso queremos editar la configuración para poder comunicar el bróker con dispositivos externos y que los clientes tengan que registrarse con usuario y contraseña. Por ello hemos de editar las siguientes líneas:

- *listener 1883,* para especificar el puerto en el que se realizará la conexión. También se puede especificar una dirección IP del bróker. (Ej: *listener 1883 192.168.0.67*)
- allow*_anonymous false,* para que no pueda conectarse cualquier cliente y tenga que registrarse.
- *Password_file C: \Program Files \mosquitto \PASSWORD,* para indicar la dirección donde se encuentra el archivo de contraseñas creado previamente.
- *Log_type error, log_type warning, log_type notice, log_type information,* esto es para poder supervisar las conexiones y su funcionamiento

A continuación, en las siguientes Ilustraciones se muestra un ejemplo de comunicación MQTT con la configuración implantada y los clientes publicador y subscriptor.

En la Ilustración 14 vemos como debemos arrancar el bróker mosquitto, desde la línea de comandos, con el comando mosquitto -c e indicando el archivo de configuración con el que se quiere iniciar el bróker. También vemos que notifica los clientes que se conectan y desconectan del servidor.

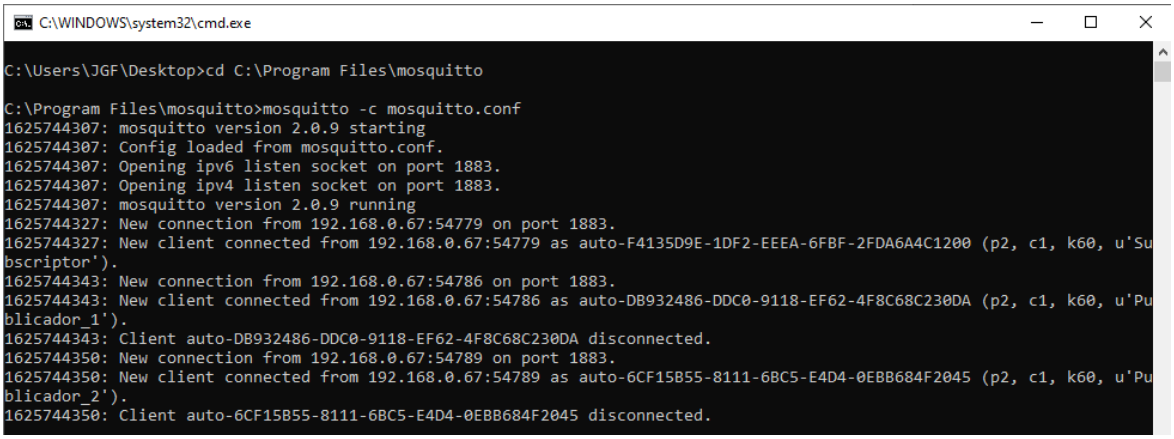


Ilustración 14: Arranque y visualización del bróker mosquitto

En la Ilustración 15, podemos observar cómo conectar un cliente subscriptor al bróker. Para ello empleamos el comando mosquitto_sub -d (habilita mensajes de depurado) -h (dirección IP del bróker, ordenador en este caso) -v (mostrar los mensajes publicados con sus tópicos) -t (# significa subscribirse a todos los tópicos posibles) -u (usuario del cliente) -P (contraseña del cliente). También se puede apreciar la llegada de dos mensajes con tópicos temperatura1 y temperatura2.

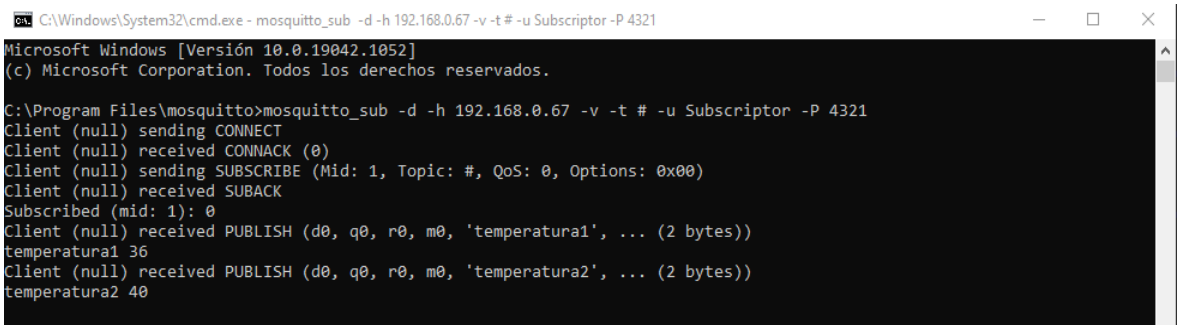


Ilustración 15: Cliente Subscriptor del bróker mosquitto

Finalmente, en la Ilustración 16 podemos ver como enviar mensajes al bróker por medio de un cliente publicador. Mediante el comando mosquitto_pub -d -h -t (tópico deseado) -m (mensaje que se desea enviar) -u -P. Además, podemos observar el envío de los mensajes que llegaron al subscriptor en la .

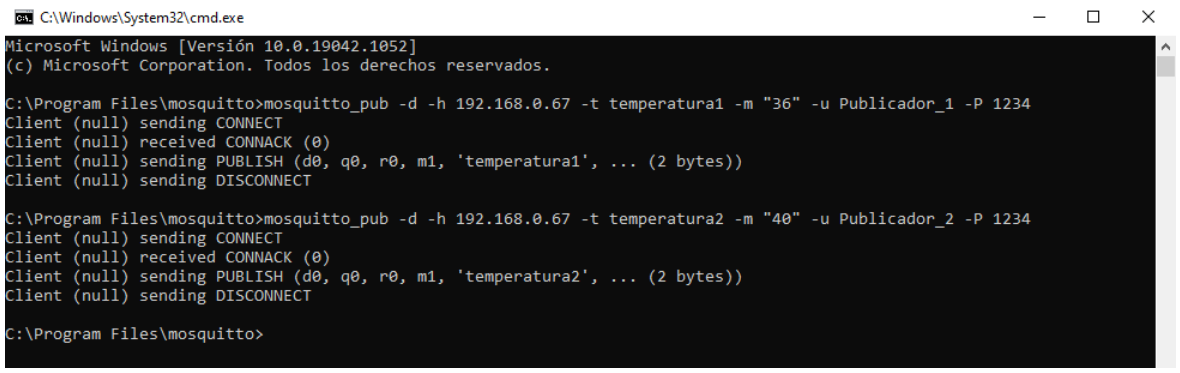


Ilustración 16: Cliente publicador del bróker mosquitto

Por último, cabe mencionar que para que el ordenador pueda realizar las conexiones de mosquitto a través del puerto 1883, es necesario permitir a mosquitto comunicarse a través del Firewall. Si no, no podremos comunicar otros dispositivos con el bróker, en nuestro caso el PLC. Este proyecto se ha realizado con un ordenador con sistema operativo de Windows 10 Pro, Versión 20H2 y para permitir la comunicación a través del Firewall es necesario seguir los pasos de la Ilustración 17. Desde el Panel de control nos dirigimos a Firewall de Windows Defender, dentro de Sistema y seguridad. Allí se encuentra la opción de Permitir a aplicaciones comunicarse a través de Firewall Windows Defender. Una vez allí, activamos las casillas de mosquitto para habilitar la comunicación.

En el caso de que se tengan firewalls externalizados instalados y el firewall por defecto de Windows desactivado, también es necesario definir una excepción que permita a mosquitto comunicarse a través de dichos firewalls.

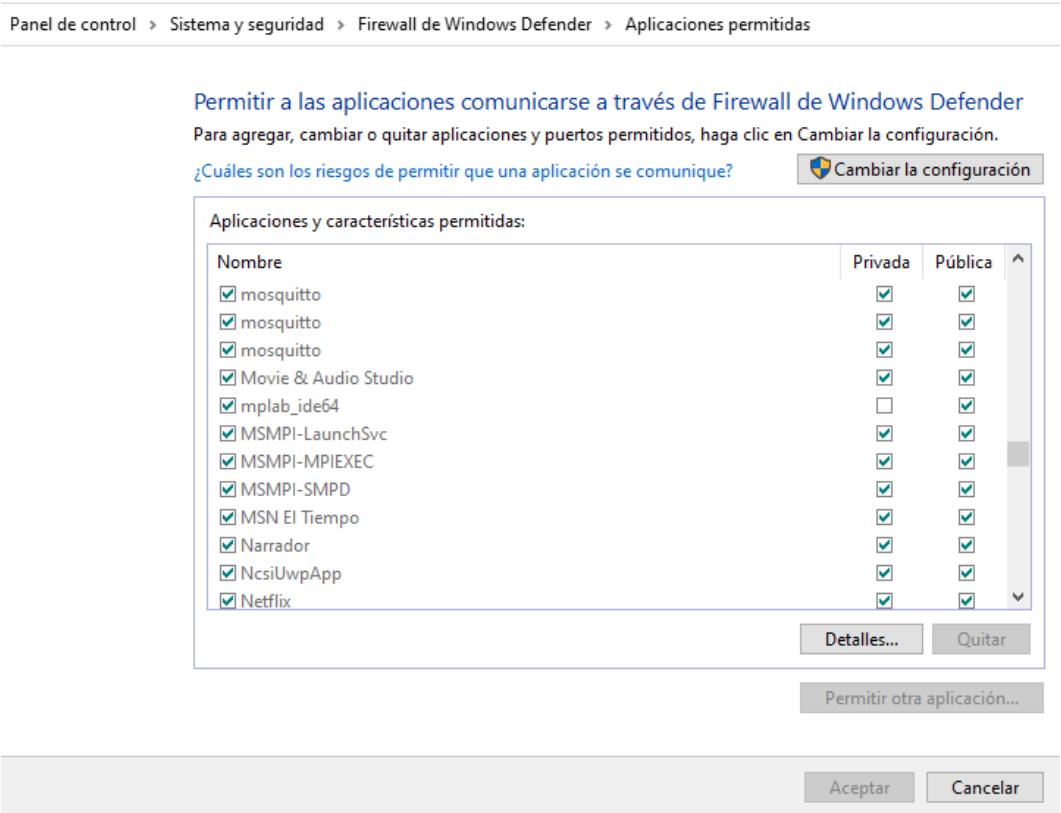


Ilustración 17: Permitir a mosquitto counicarse a través del Firewall

5.2.3 PROGRAMACIÓN DEL PLC COMO CLIENTE PUBLICADOR

Una vez terminada la configuración del servidor mosquitto, ya podemos centrarnos en la programación del S7-1500 como cliente publicador. La programación se realizará con el TIA Portal V16 y se empleará la librería "LMQTT_Client", la cual se puede obtener de la página oficial de siemens: https://support.industry.siemens.com

En primer lugar, descargaremos y abriremos la librería LMQTT. Allí encontraremos la función FB "LMQTT_Client" y los tipos de datos del PLC usados por dicha función. Deberemos de agregar ambos a nuestro proyecto, para ello se puede arrastrar desde la librería y soltar. En el caso de la función, deberemos añadirla a "Bloques del programa" y en el caso de los tipos de datos, se añadirán a "Tipos de datos del PLC".

En segundo lugar, debemos de crear un bloque de datos que almacene los parámetros de conexión TCP, parámetros MQTT, mensaje y tópico que se desea enviar. En la Ilustración 18, se muestra el bloque de datos que se ha empleado en este modelo. A continuación, agregamos la función a nuestro programa y asignamos las variables de nuestro nuevo bloque de datos, a las entradas y salidas de la función; tal y como se muestra en la Ilustración 19.

MQTT

	Nombre	Tipo de datos	Valor de arranque	Remanen...	Accesible d...	Escrib...	Visible en ...	Valor de a...
1	Static							
2	enable	Bool	false		✓	✓	✓	
3	tcpConnParam	"LMQTT_typeTcpConnParamData"			✓	✓	✓	
4	mqttParam	"LMQTT_typeParamData"			✓	✓	✓	
5	publishData	"LMQTT_typePublishData"			✓	✓	✓	
6	subscribeToTopic	"LMQTT_typeSubscribeData"			✓	✓	✓	
7	subscriptionMessage	"LMQTT_typeSubscriptionData"			✓	✓	✓	
8	tcpEstablished	Bool	false		✓	✓	✓	
9	mqttEstablished	Bool	false		✓	✓	✓	
10	done	Bool	false		✓	✓	✓	
11	error	Bool	false		✓	✓	✓	
12	StatusID	Int	0		✓	✓	✓	
13	i	Int	0	✓	✓	✓	✓	
14	N_SEÑALES	Int	4	✓	✓	✓	✓	
15	N_TOPIC	WString[250]	WSTRING#''		✓	✓	✓	
16	AUX_F1	Bool	false		✓	✓	✓	
17	AUX_F2	Bool	false		✓	✓	✓	
18	FLANCO_DONE	Bool	false		✓	✓	✓	

Ilustración 18: Bloque de datos MQTT

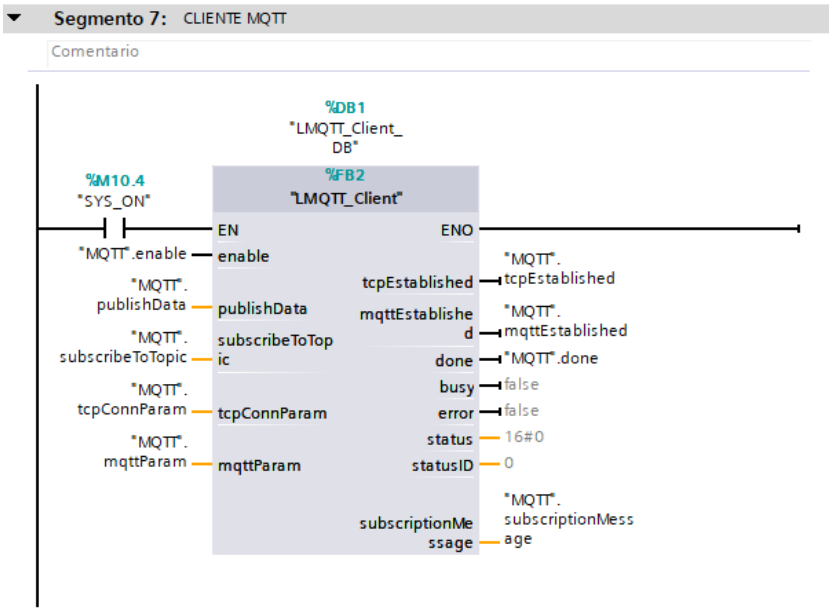


Ilustración 19: Función "LMQTT_Client"

Por último, para terminar con la configuración de la función, nos dirigimos al bloque de datos y configuramos los parámetros de conexión TCP y MQTT para permitir la comunicación. Como podemos observar en la Ilustración 20 e Ilustración 21, hay una gran cantidad de parámetros para definir. Sin embargo, en este modelo se emplearán únicamente los oportunos para obtener una solución funcional.

En la Ilustración 20 vemos que es necesario introducir el “hwidentifier” y “connId”, estos son el identificador de hardware de la interfaz PN/IE y el ID de conexión por los que se establece la conexión. También hemos de especificar la dirección IP del bróker MQTT y puerto por el que establecer la conexión. En este proyecto se está usando el bróker mosquitto instalado en el ordenador, por lo que añadimos la dirección IP del ordenador. El puerto 1883 es el puerto específico de comunicación MQTT sin encriptar. El puerto local es opcional, si se pone a 0 sigue funcionando.

En la Ilustración 21, se puede observar los parámetros para habilitar al cliente publicador de usuario y contraseña. Vemos que es necesario establecer previamente los flag de usuario y contraseña a "true" para activarlo y después agregar el usuario y contraseña correspondientes. Este usuario y contraseña del cliente, debe coincidir con uno de los configurados en el bróker mosquitto. De no ser así el envío de datos no será permitido.

9	tcpConnParam	"LMQTT_typeTcpCo...		☐	☑	☑	☑	☐
10	useQdn	Bool	false	☐	☑	☑	☑	☐
11	hwIdentifier	HW_ANY	64	☐	☑	☑	☑	☐
12	connectionID	CONN_OUC	16#10	☐	☑	☑	☑	☐
13	qdnAdressBroker	String[254]	''	☐	☑	☑	☑	☐
14	ipAdressBroker	Array[0..3] of USInt		☐	☑	☑	☑	☐
15	ipAdressBroker...	USInt	192	☐	☑	☑	☑	☐
16	ipAdressBroker...	USInt	168	☐	☑	☑	☑	☐
17	ipAdressBroker...	USInt	0	☐	☑	☑	☑	☐
18	ipAdressBroker...	USInt	67	☐	☑	☑	☑	☐
19	localPort	UInt	2000	☐	☑	☑	☑	☐
20	mqttPort	UInt	1883	☐	☑	☑	☑	☐
21	activateSecureConn	Bool	false	☐	☑	☑	☑	☐
22	validateSubjectAlt...	Bool	false	☐	☑	☑	☑	☐
23	idTlsServerCertifica...	UDInt	0	☐	☑	☑	☑	☐
24	idTlsOwnCertificate	UDInt	0	☐	☑	☑	☑	☐

Ilustración 20: Parámetros de conexión TCP

25	mqttParam	"LMQTT_typeParam...		☐	☑	☑	☑	☐
26	connectFlag	"LMQTT_typeConn...		☐	☑	☑	☑	☐
27	cleanSession	Bool	true	☐	☑	☑	☑	☐
28	will	Bool	false	☐	☑	☑	☑	☐
29	willQoS1	Bool	false	☐	☑	☑	☑	☐
30	willQoS2	Bool	false	☐	☑	☑	☑	☐
31	willRetain	Bool	false	☐	☑	☑	☑	☐
32	password	Bool	true	☐	☑	☑	☑	☐
33	userName	Bool	true	☐	☑	☑	☑	☐
34	keepAlive	UInt	30	☐	☑	☑	☑	☐
35	clientIdentifier	String[23]	'Siemens SIMATIC'	☐	☑	☑	☑	☐
36	willTopic	String[100]	''	☐	☑	☑	☑	☐
37	willMessage	String[100]	''	☐	☑	☑	☑	☐
38	userName	String[100]	'zc'	☐	☑	☑	☑	☐
39	password	String[200]	'zciiyc'	☐	☑	☑	☑	☐

Ilustración 21: Parámetros de conexión MQTT

Con esta configuración ya se puede realizar un envío de mensajes desde el PLC al bróker, para ello es necesario emplear los parámetros de "publishData" mostrados en la Ilustración 22, donde vemos las variables de mensaje (36), tópico (SIGNAL0), QoS (2) y publishData (realiza el envío cada flanco). Sin embargo, esto no es útil, ya que necesitamos un proceso automático para el envío continuo de datos.

MQTT

		Nombre	Tipo de datos	Valor de arranque	Remanen...	Accesible d...	Escrib...	Visible en ..	Valor de a..
1		Static			☐	☐	☐	☐	☐
2		publishData	"LMQTT_typePublis...		☐	☑	☑	☑	☐
3		publishMessage	Bool	false	☐	☑	☑	☑	☐
4		publishTopic	WString[250]	WSTRING#'SIGNAL0'	☐	☑	☑	☑	☐
5		publishMessageDat	WString[800]	WSTRING#'36'	☐	☑	☑	☑	☐
6		publishQoS	Int	2	☐	☑	☑	☑	☐

Ilustración 22: Parámetros de envío de datos

ENVÍO CONTINUO DE CUATRO SEÑALES

En este sistema queremos realizar el envío continuo de los datos de 4 señales distintas. Para ello, se realiza un bucle infinito que vaya enviando progresivamente un dato de cada una de las cuatro señales.

En primer lugar, se utiliza la función generadora de señales explicada en el 5.2.1. Dicha función genera a su salida tantas señales como deseemos, en este caso 4. Una vez tenemos los datos que deseamos enviar, debemos prepararlo junto con su tópico, para enviarlos como variables tipo WString. Como se puede observar en la Ilustración 23, se coge una de las señales de la función generadora y se convierte de Real a WString para establecerse como mensaje de envío en la función LMQTT_Client. Lo mismo se hace con el tópico, salvo que para distinguir las señales y que no se mezclen sus datos en la base de datos, es necesario asociarle tópicos distintos a cada señal. Por eso vemos que también se añade un número identificador a cada señal que resulta ser el mismo iterador usado para recorrer el vector de señales. Finalmente se concatenan ambos WString resultando el tópico de envío en la función LMQTT_Client. (Ej: SIGNAL0)

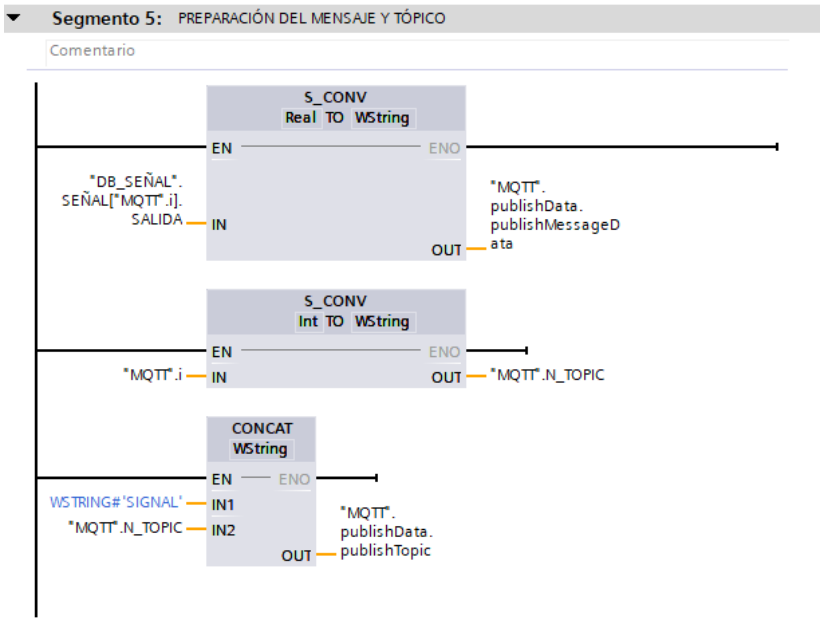


Ilustración 23: Preparación del mensaje y tópico

En la Ilustración 24 e Ilustración 25 podemos ver la gestión de la variable publishData para permitir el envío progresivo de las distintas señales. Para ello es necesario que la variable sufra un flanco por envío, por lo que en la Ilustración 24 vemos que la activamos siempre que esté desactivada, se desactive la función o la conexión y en la Ilustración 25 vemos que se resetea cada vez que recibimos un flanco de conexión establecida o mensaje anterior enviado correctamente (done).

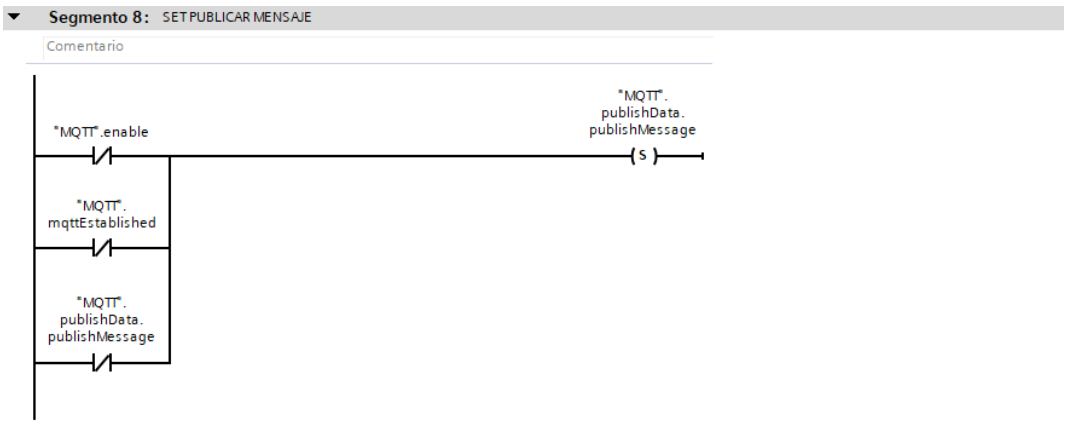


Ilustración 24: Set de la variable de envío

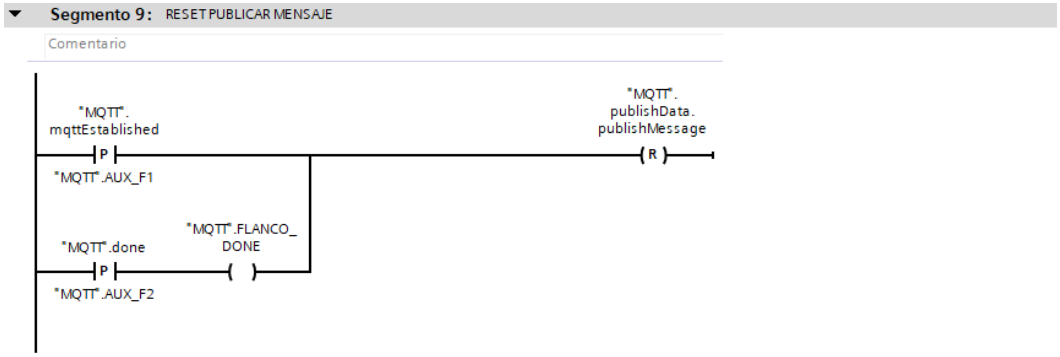


Ilustración 25: Reset de la variable de envío

A continuación, se muestra en la Ilustración 26 el bloque empleado para dar un valor a la variable error, así como el Estatus de dicho error para saber su origen y poder arreglarlo.

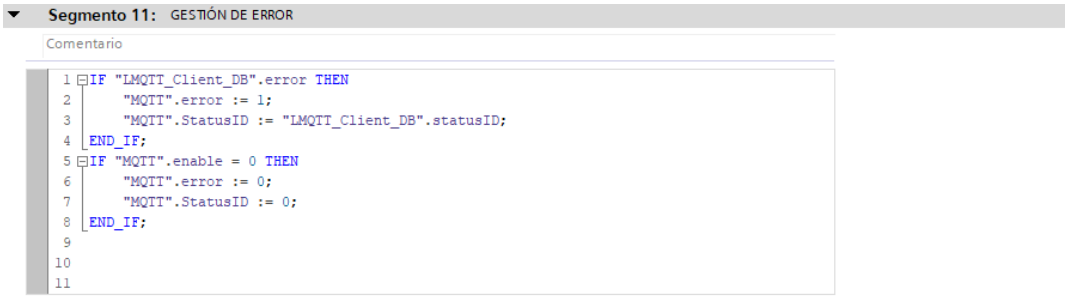


Ilustración 26: Gestión de error

Finalmente se muestra en la Ilustración 27 la gestión de la variable iteradora con la que se recorre el vector de señales y se asigna el número de la señal para el tópico. Podemos observar que cada ciclo se incrementa una unidad el iterador con la intención de envíar uno a uno los datos de todas las señales, hasta que una vez se han enviado de las cuatro, se vuelve a comenzar por la primera. En esta solución se envía de la señal 0 a la señal 3.

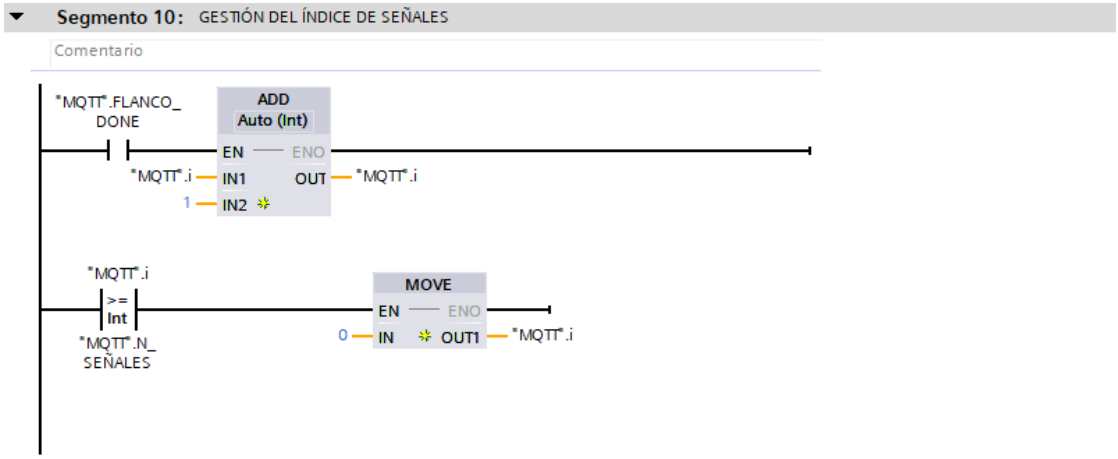


Ilustración 27: Gestión del Índice de señales

5.2.4 BASES DE DATOS

En este proyecto se emplearon dos bases de datos distintas y se realizó un desarrollo para poder almacenar datos de las señales del PLC en ambas. Inicialmente se probó con SQL Server Express, no obstante, esta base de datos no era capaz de almacenar datos a la misma velocidad que el PLC los enviaba. Además, para poder realizar posteriormente una representación gráfica, era necesario llevar una gestión adicional del timestamp de cada dato. Por todo esto se decidió que InfluxDB era la mejor opción que emplear, ya que era capaz de almacenar datos a la velocidad de envío del PLC y automáticamente asociaba un timestamp a los datos según se introducían en la base de datos.

Podemos descargar InfluxDB desde el siguiente link: https://mosquitto.org/download/. Una vez descargada y descomprimida la carpeta, encontraremos los ejecutables mostrados en la Ilustración 28. De los cuales se emplearán *influx.exe* e *influxd.exe*. *Influxd.exe* se usará inicialmente para arrancar el servidor e *influx.exe* se usará para poder hacer consultas y operaciones con las bases de datos.

Ilustración 28: Ejecutables InfluxDB

En la dirección donde hemos descargado InfluxDB se deben crear unas nuevas carpetas llamadas data, meta y wal, quedando nuestra carpeta con la apariencia de la Ilustración 29. Posteriormente nos dirigimos al archivo de configuración *influxdb.conf* y añadimos las direcciones de dichas carpetas, tal y como se muestra en la Ilustración 30.

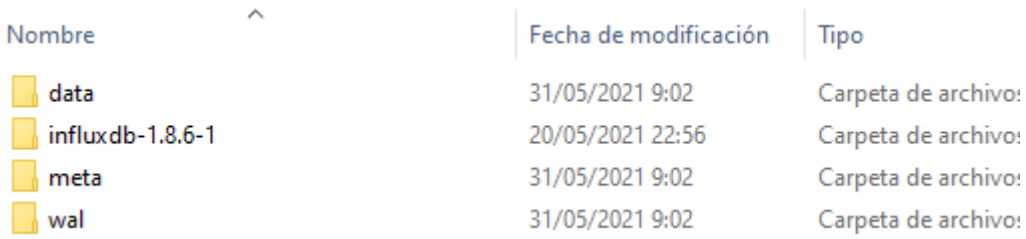


Ilustración 29: Carpeta descarga InfluxDB

```
[meta]
  # Where the metadata/raft database is stored
  dir = "C:\Users\JGF\Desktop\MQTT\INFLUXDB\meta"

[data]
  # The directory where the TSM storage engine stores TSM files.
  dir = "C:\Users\JGF\Desktop\MQTT\INFLUXDB\data"

  # The directory where the TSM storage engine stores WAL files.
  wal-dir = "C:\Users\JGF\Desktop\MQTT\INFLUXDB\dir"
```

Ilustración 30: Edición influxdb.conf

Finalmente, para poder comenzar a almacenar datos, hemos de iniciar InfluxDB ejecutando *influxd.exe* desde el símbolo del sistema y después ejecutar *influx.exe* para crear la base o bases de datos en las que deseamos almacenar nuestros datos. En la Ilustración 31 podemos ver cómo crear una base de datos nueva y mostrar las existentes. Con InfluxDB ejecutándose y la base de datos ya creada, solo queda comenzar a introducir los datos y eso se hará mediante Node-Red.

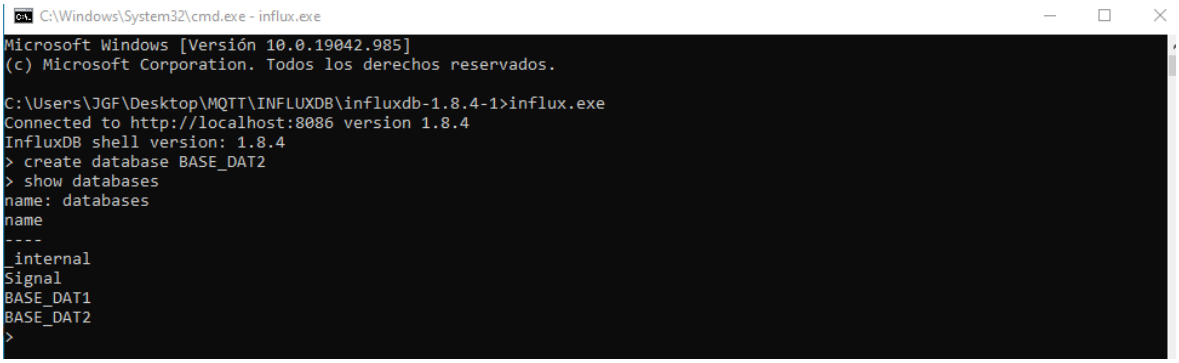


Ilustración 31: Creación de base de datos InfluxDB

5.2.5 NODE-RED

En primer lugar, se ha de instalar Node.js desde el siguiente link: https://nodejs.org/en/. Para acabar con la instalación de Node.js, se descargan las herramientas necesarias mediante el archivo descargado *install_tools.bat*. Por último, ejecutamos este comando para descargar Node-Red: *npm install -g –unsafe-perm node-red.*

Ya podemos usar Node-Red, para iniciarlo solo es necesario escribir node-red en el símbolo del sistema tal y como se muestra en la Ilustración 32. Podemos ver que se facilita una dirección donde está funcionando el servidor. Introducimos esa dirección en nuestro navegador y ya tendremos Node-Red funcionando sin necesidad de conexión a internet.

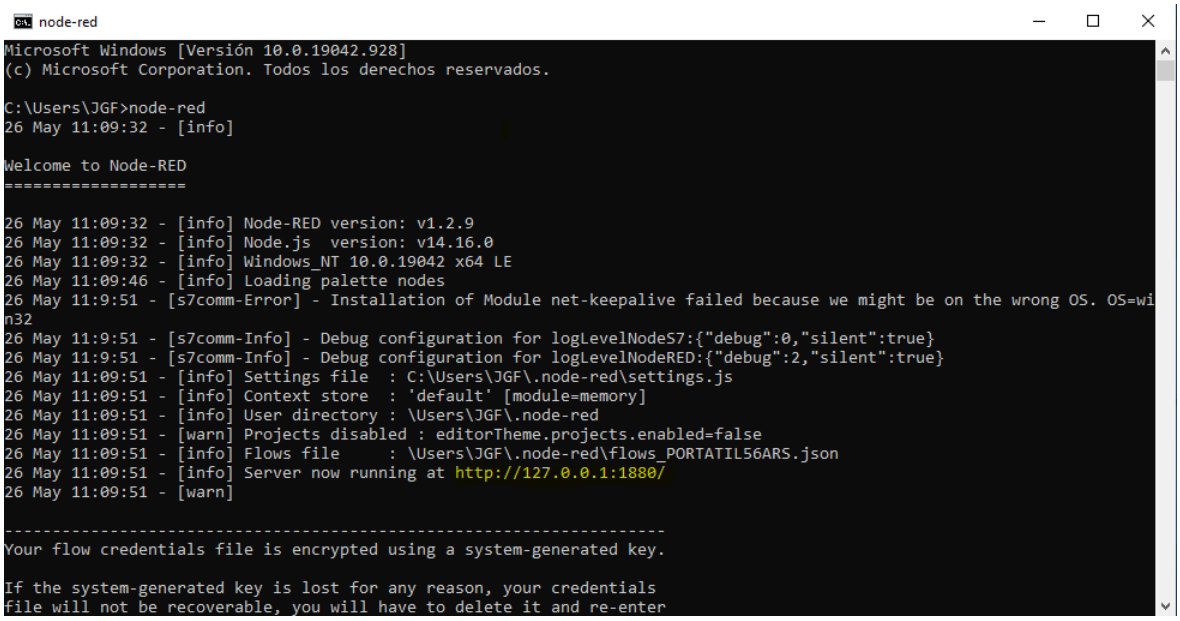


Ilustración 32: Arranque Node-Red

Antes de poder realizar nuestro diseño en Node-Red, es necesario descargar un nodo adicional de la librería de nodos. Este nodo será el encargado de introducir los datos en nuestra base de datos InfluxDB. Para obtener esta librería solo hay que ejecutar el siguiente comando: *npm install node-red-contrib-influxdb*, si aún no tenemos el nodo, hemos de reiniciar Node-Red.

A continuación, se muestra en la Ilustración 33 el diseño en Node-Red para recibir los datos enviados por el PLC mediante MQTT con un cliente subscriptor e introducir los datos de las señales en InfluxDB. En primer lugar, tenemos el nodo de cliente subscriptor MQTT el cual está configurado tal y como se muestra en la Ilustración 34 e Ilustración 35. Podemos ver que está subscrito a cualquier tópico (#), ya que son 4 señales con 4 tópicos distintos, usamos QoS 2 para asegurarnos que el mensaje es recibido una sola vez. También introducimos la dirección del bróker mosquitto, que como está en el mismo ordenador, podemos usar localhost y el puerto 1883 correspondiente a mosquitto. Por último, añadimos uno de los usuarios y contraseñas configurados en mosquitto para poder recibir los datos.

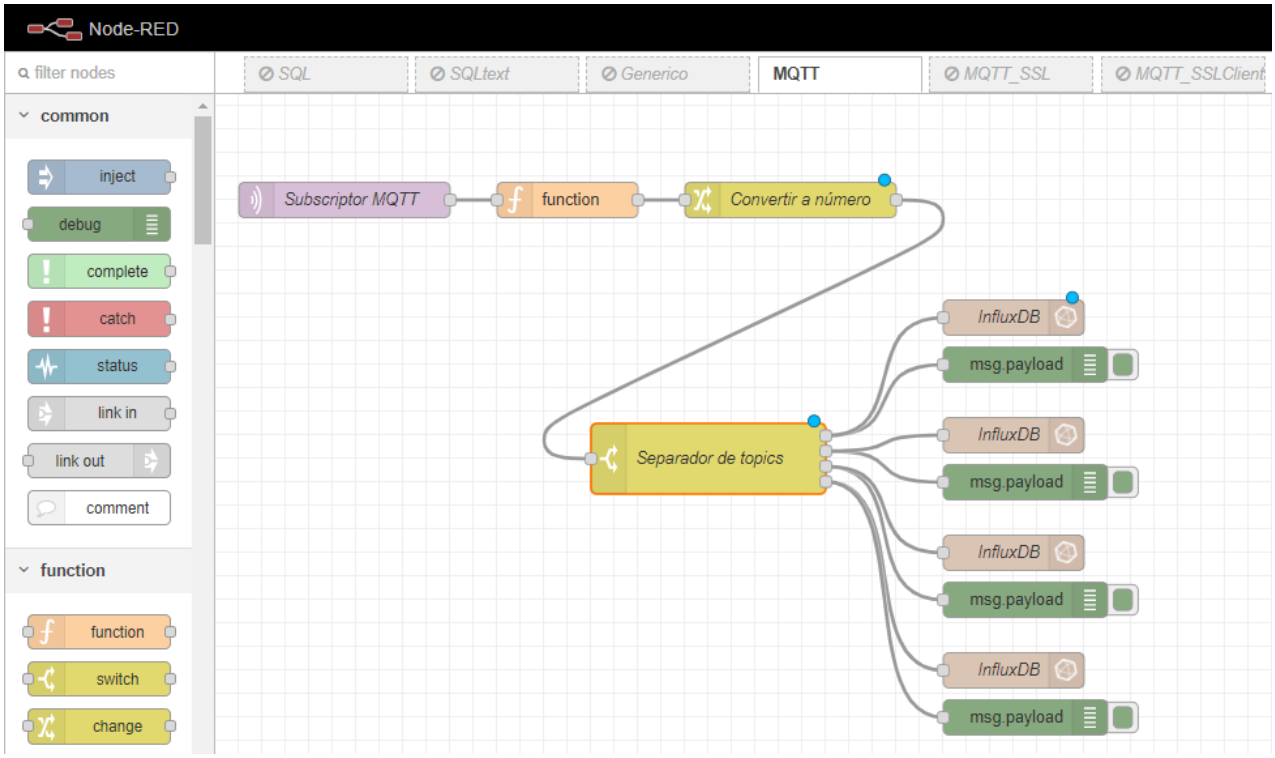


Ilustración 33: Diseño de Node-Red (MQTT-InfluxDB)

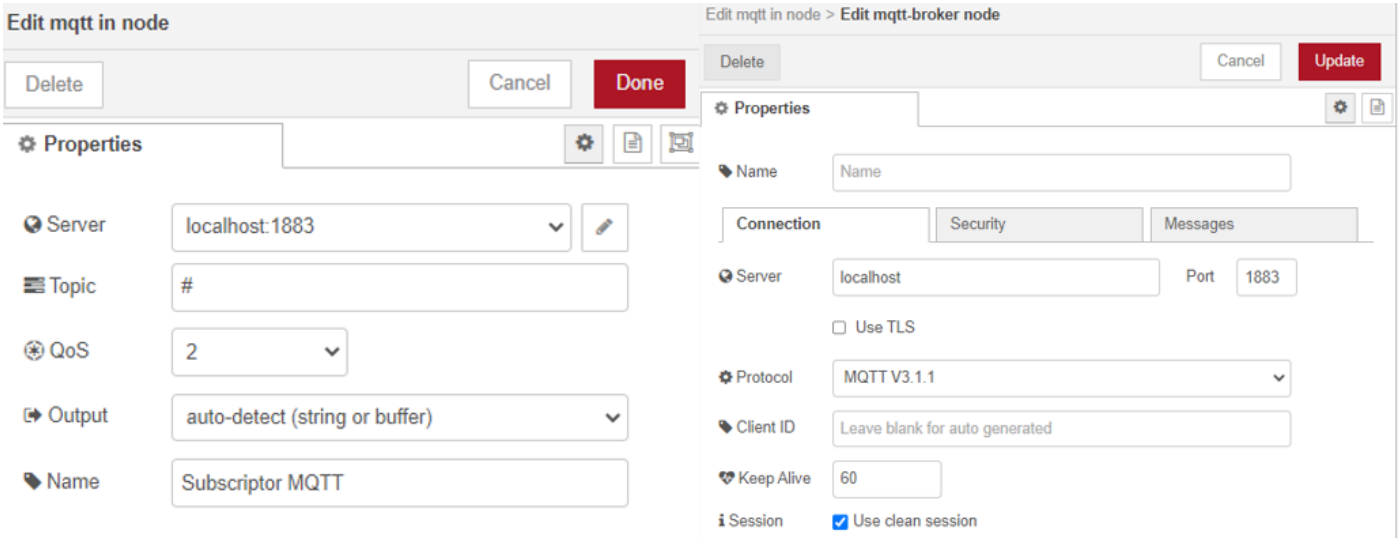


Ilustración 34: Nodo Subscriptor MQTT

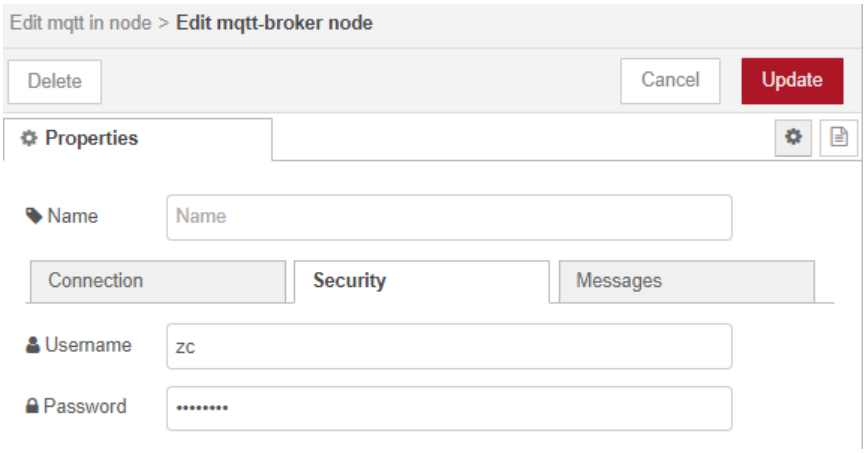


*Ilustración 35: Nodo Subscriptor MQTT**

El nodo "function" es un nodo personalizado para tratar los mensajes y tópicos recibidos desde el PLC, ya que se recibían como " 36" y "SIGNAL 0" y se querían eliminar dichos espacios innecesarios. Para ello usamos la función trim y sustituimos los espacios del tópico tal y como se muestra en la Ilustración 36. El nodo "Convertir a número" se usa para cambiar el formato del mensaje a número, de forma que se pueda luego trabajar con los datos almacenados. Su configuración se muestra en la Ilustración 37.

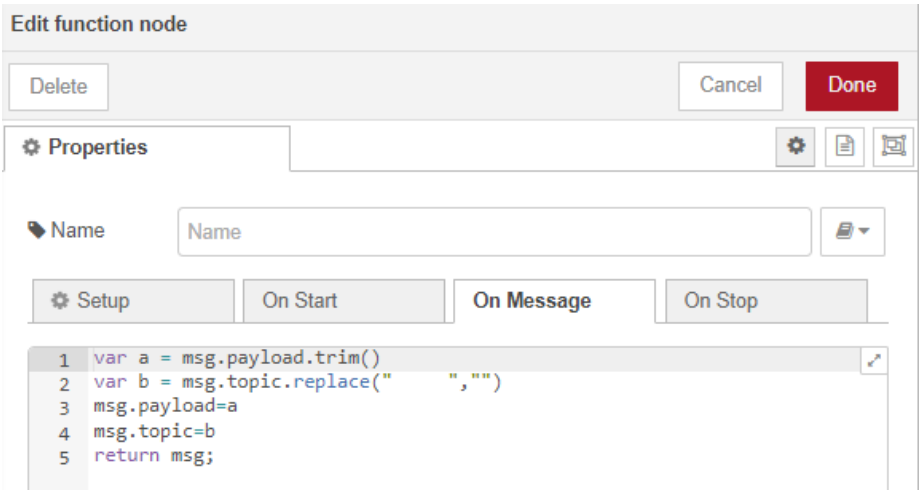


Ilustración 36: Nodo función para eliminar espacios

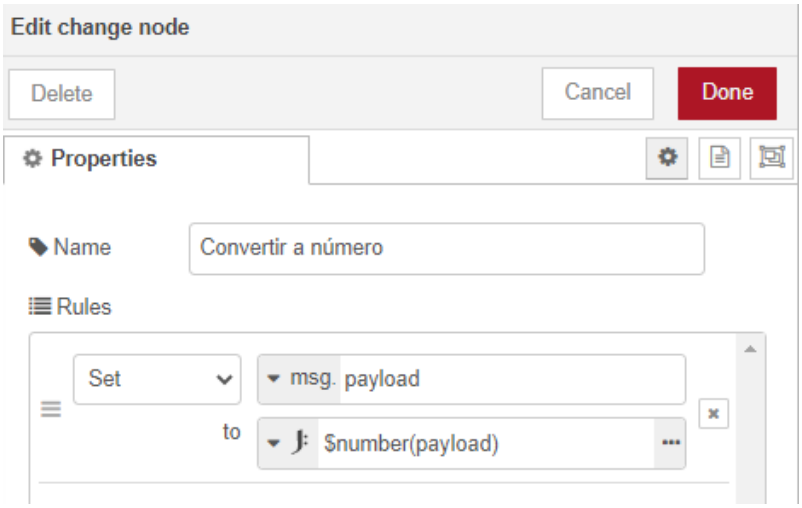


Ilustración 37: Nodo Convertir a número

Finalmente se emplean 2 últimos nodos. El "Separador de topics", el cual separa los datos según su tópico para poder introducir las señales en InfluxDB como series de datos distintas. En la Ilustración 38 tenemos la configuración del nodo de tipo "switch" en el que se establece según su tópico una salida distinta del nodo y en la Ilustración 39 tenemos la configuración de uno de los 4 bloques InfluxDB que aparecen en el diseño. Vemos que es necesario indicar el nombre de la base de datos en la que queremos almacenar las señales, así como la dirección del servidor, que, al ser el mismo ordenador, nos sirve localhost con el puerto específico de InfluxDB (8086). También es necesario introducir el nombre de la medición en la que se quieren introducir los datos, por lo que en el diseño se emplean los 4 nodos, cada uno con una medición para cada una de las señales. Cabe mencionar que para que este diseño funcione, hemos de tener en funcionamiento el bróker mosquitto e InfluxDB.

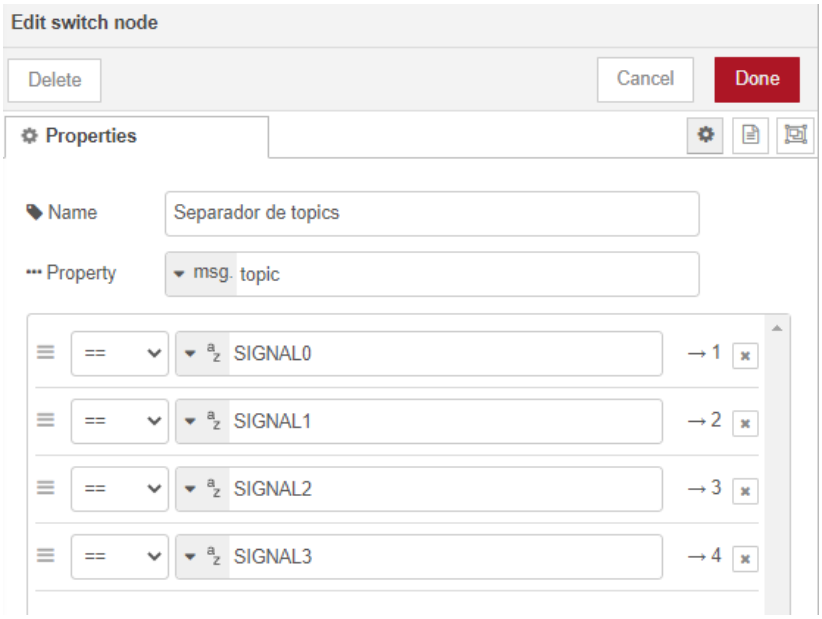


Ilustración 38: Nodo Separador de topics

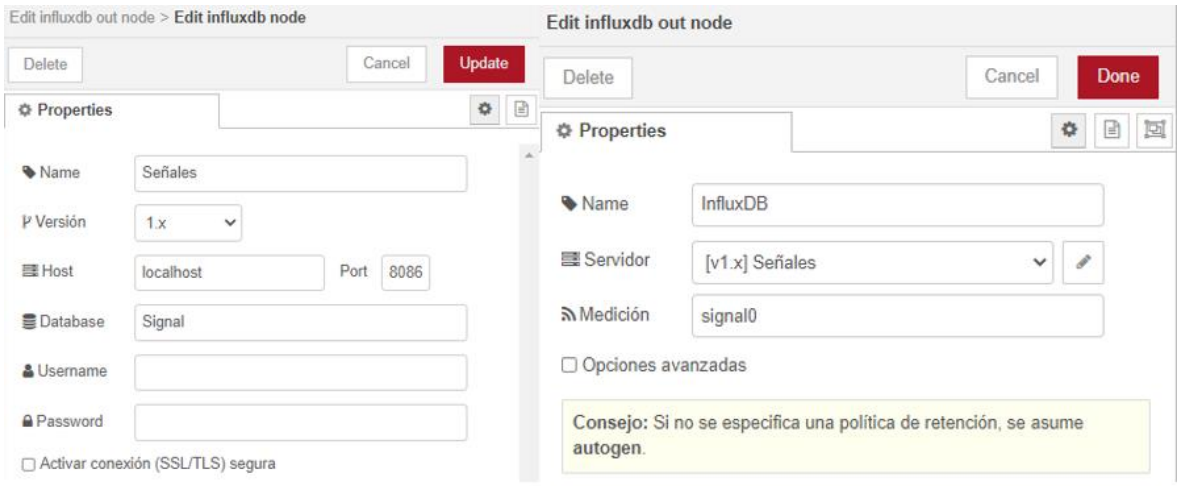


Ilustración 39: Nodo InfluxDB

5.2.6 GRAFANA

Al igual que Node-Red, Grafana también funciona desde el navegador y sin necesidad de internet. Se puede descargar para Windows desde el siguiente link:

https://grafana.com/grafana/download?edition=oss&pg=get&plcmt=selfmanaged-box1-cta1&platform=windows.

Una vez finalizada su descarga, ejecutamos desde la línea de comandos *grafana-server.exe* para arrancar el servidor de Grafana y nos dirigimos desde el navegador al puerto de Grafana http://localhost:3000/. Se nos pedirá un usuario y contraseña que en ambos casos es "admin". Seguidamente se permite cambiar dicha contraseña.

Grafana ya se encuentra totalmente configurado, de forma que ya se pueden crear los dashboards y formas de visualización de datos deseadas. El primer paso es añadir nuestra base de datos InfluxDB de la que queremos extraer los datos que se van a representar. Para ello nos dirigimos desde el menú a Configuración > Bases de datos y ahí podemos añadir una nueva base de datos. En la Ilustración 40 vemos los campos necesarios para añadir la base de datos, indicamos que se trata de una base de datos InfluxDB, el lenguaje de consulta, la dirección de nuestro InfluxDB y el nombre de la base de datos a la que queremos acceder, en este caso "Signal" que contiene las señales que queremos representar.

Finalmente nos dirigimos a Crear > Dashboard para crear los dashboards donde se representarán las 4 señales generadas. Ahí podemos añadir los paneles que deseamos y procedemos a editarlos, en este caso queremos una visualización de tipo gráfico y tenemos una amplia variedad de opciones de display, ejes, leyendas, etc. Debemos añadir un "Query" para realizar la consulta que deseamos de la base de datos. En la Ilustración 41, podemos ver la Query empleada en este desarrollo, es necesario indicar el nombre de la base de datos configurada anteriormente en Grafana, que serie de datos (señal) se desea representar, así como operaciones de selección interesantes como la media, mediana, percentil, máximos, mínimos, etc. Así creamos las gráficas de las cuatro señales.

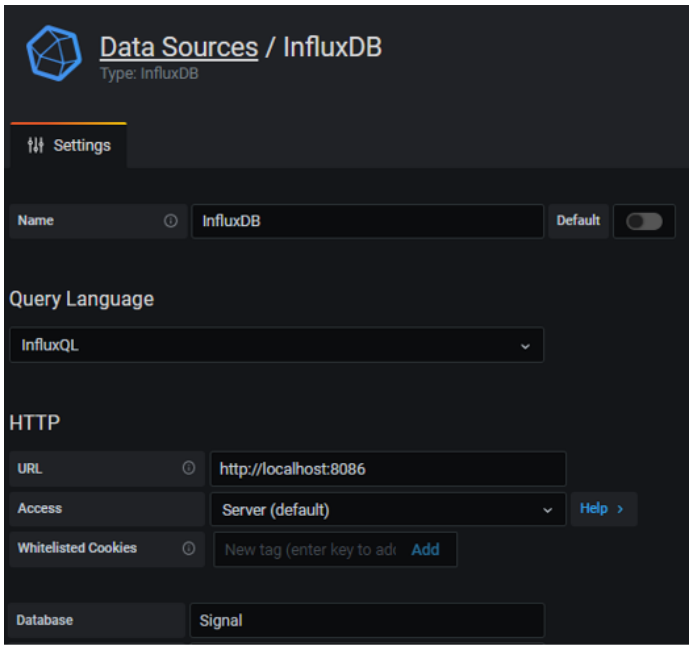


Ilustración 40: Añadir Base de datos en Grafana

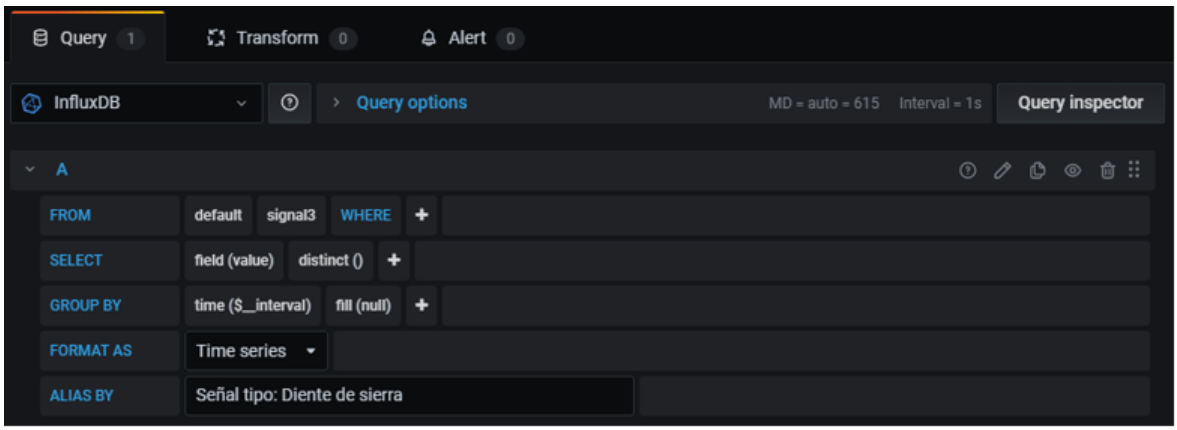


Ilustración 41: Configuración Query Grafana

5.2.7 OPENSSL

Para poder establecer una conexión MQTT segura, es necesario configurar una conexión encriptada. La encriptación consiste en garantizar que nuestros mensajes no son alterados, no han sido leídos y son enviados a quien queremos. Para ello se hace uso de keys (llaves) que son números de 128 bits normalmente, los cuales se combinan con el mensaje mediante un algoritmo con el propósito de encriptar el mensaje (sea indescifrable) y firmarlo (saber quién ha mandado el mensaje). Los certificados sirven para asociar las keys a personas o entidades, de forma que podemos saber con certeza que la key pertenece a quien dice, estos certificados son verificados por Autoridades de Certificados (CA) de confianza que se encargan de asegurar que los certificados son reales. Como para este desarrollo solo necesitamos una solución sencilla, usaremos OpenSSL para crear nuestras propias keys y certificados y firmarlos como nuestra propia Autoridad de Certificados.

Existen 2 posibilidades para encriptar la conexión, la primera es crear únicamente key y certificado para el bróker, mientras que la segunda es crear tanto key y certificado para el bróker como para los clientes que se vayan a conectar al mismo. Empezaremos por la primera ya que para la segunda hay que seguir los mismos primeros pasos.

En primer lugar, hemos de crear la key y certificado de la CA que usaremos para verificar el resto de los certificados. Para ello nos dirigimos al símbolo del sistema desde la carpeta de bin de nuestra carpeta OpenSSL descargada y ejecutamos los comandos mostrados en la Ilustración 42 e Ilustración 43. Con la creación de la key es necesario introducir una contraseña y confirmarla. En la creación del certificado es necesario introducir una serie de datos representativos del CA y firmar el certificado con la key creada.

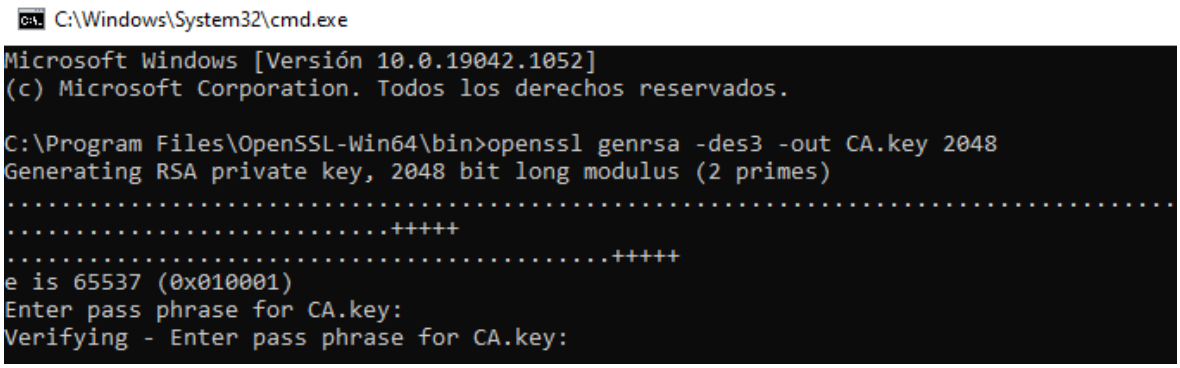


Ilustración 42: Creación key del CA

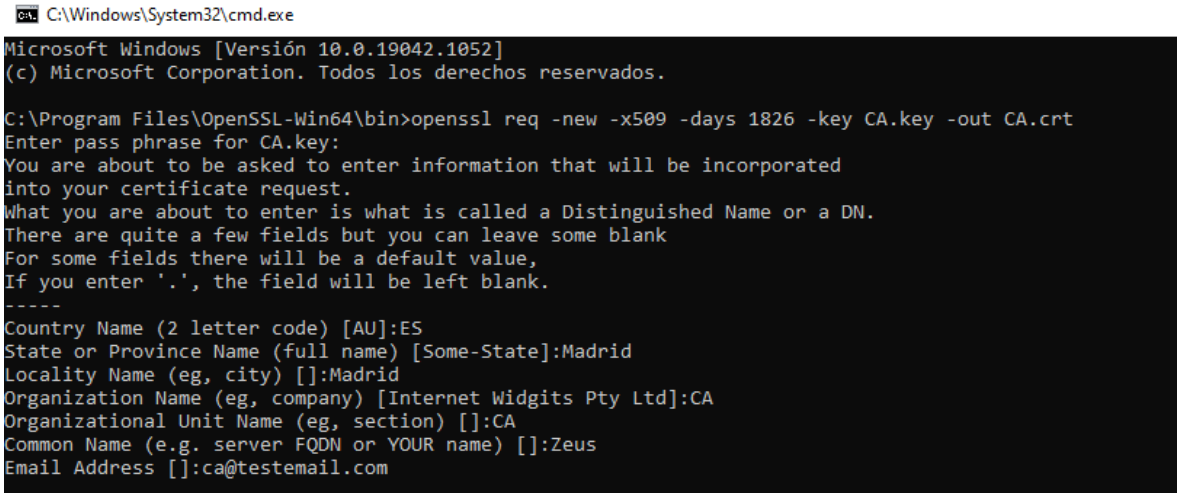


Ilustración 43: Creación certificado del CA

En segundo lugar, se crea una key sin contraseña para nuestro bróker (mosquitto) y una solicitud de certificado firmada por esta key. En la Ilustración 44 se puede ver el comando necesario para crear la key sin contraseña. En la Ilustración 45, vemos como se crea una solicitud de certificado en la que es necesario introducir información del bróker. El campo "Common Name" es de vital importancia para que funcione la comunicación, ya que los clientes deberán de identificar al bróker con el mismo nombre. En este desarrollo se ha usado la dirección IP del ordenador (con mosquitto) como Common Name.

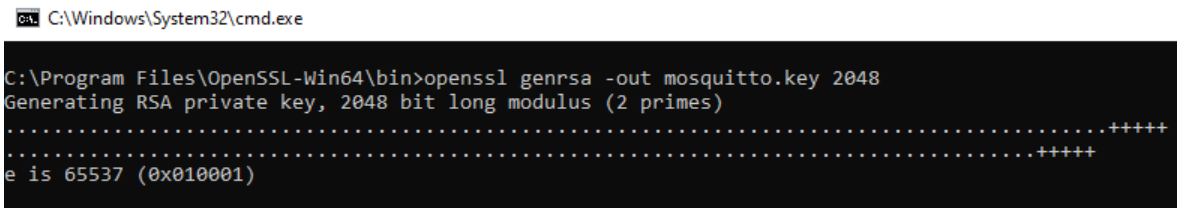


Ilustración 44: Creación key de mosquitto (sin contraseña)

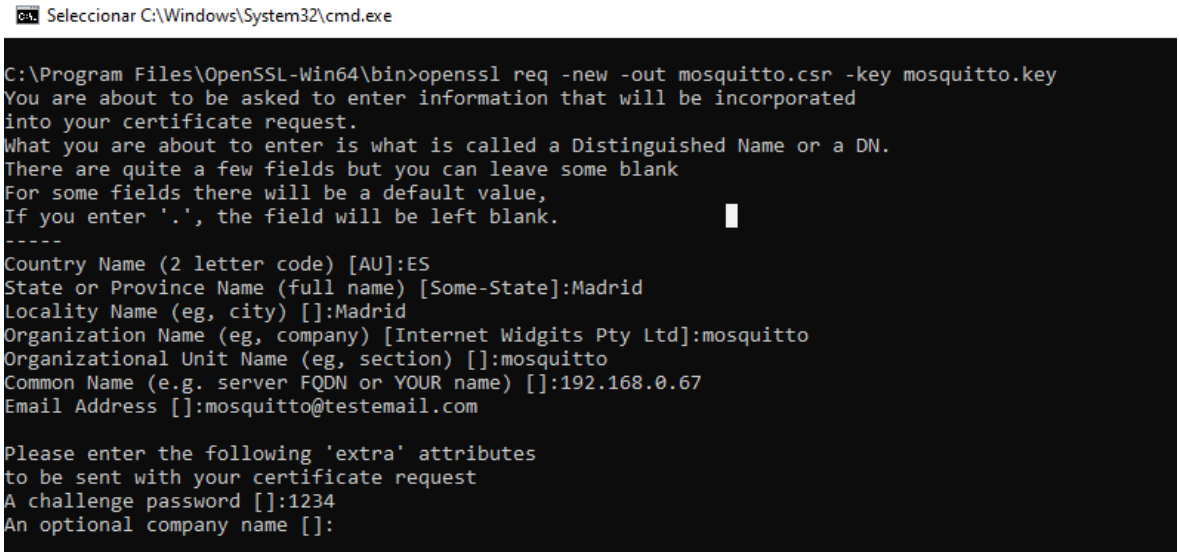


Ilustración 45: Creación solicitud de certificado de mosquitto

El propósito de una solicitud de certificado es enviarla al CA para que este verifique que efectivamente el certificado pertenece a quien corresponde y se cree un certificado firmado por el CA como garantía de que es real. Como en este desarrollo tanto el servidor como los clientes son gestionados por nosotros, seremos nuestro propio CA para firmar la solicitud y crear el certificado tal y como se muestra en la Ilustración 46. Vemos que es necesario introducir la contraseña de la key del CA.

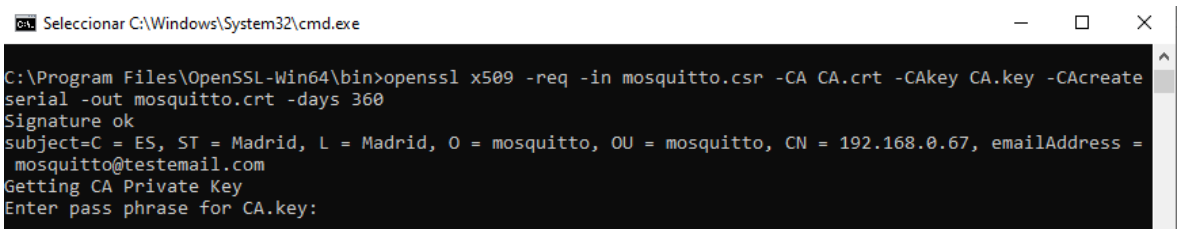


Ilustración 46: Creación de certificado de mosquitto verificado

Una vez terminada la creación de keys y certificados necesitamos copiar los archivos: *CA.crt, mosquitto.crt, mosquitto.key* y añadirlos a una nueva carpeta dentro de nuestra carpeta de mosquitto. En este proyecto se ha llamado a dicha carpeta "certs". Posteriormente, hemos de editar el archivo de configuración de nuestro bróker, cambiando el puerto de escucha al 8883, que es el específico de la comunicación encriptada y añadir las direcciones de la key y certificado de nuestro bróker y del certificado del CA, tal y como se muestra en la Ilustración 47.

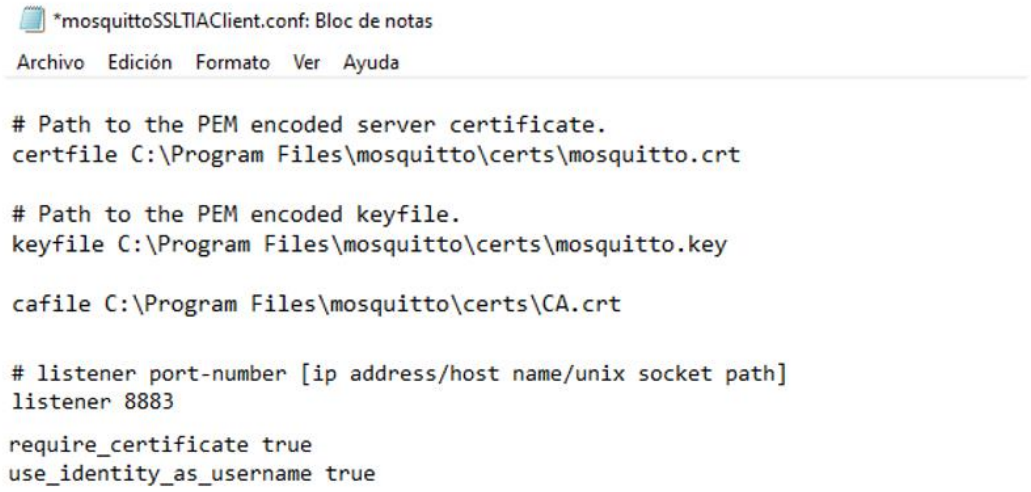


Ilustración 47: Edición mosquitto.conf encriptado

Ahora ya la comunicación entre el bróker mosquitto y sus clientes está encriptada y la única diferencia de configuración para los clientes es añadir el certificado del CA para garantizar la conexión, de no ser así la comunicación no será llevada a cabo por seguridad. La segunda forma de conexión encriptada es crear también una key y certificado para los clientes que vayan a comunicarse con el bróker. Esto se realiza del mismo modo que para los del bróker, se crea una key sin contraseña, se crea con dicha key una solicitud de certificado y se crea un certificado firmado por la CA. Cabe mencionar que, si el certificado del bróker y del cliente no están firmados con la misma key del CA, esta conexión no funcionará.

En la Ilustración 48, se muestra la configuración necesaria para el cliente subscriptor de Node-Red que recibirá los datos de las señales de forma encriptada. Es necesario que el servidor coincida con el establecido como "Common Name" en el certificado, cambiamos el puerto a 8883 y habilitamos la casilla TLS. Introducimos el certificado del CA y la key y certificado del cliente en caso de que también se hayan creado, de no ser así podríamos emplear la primera forma de encriptación, únicamente con el certificado del CA.

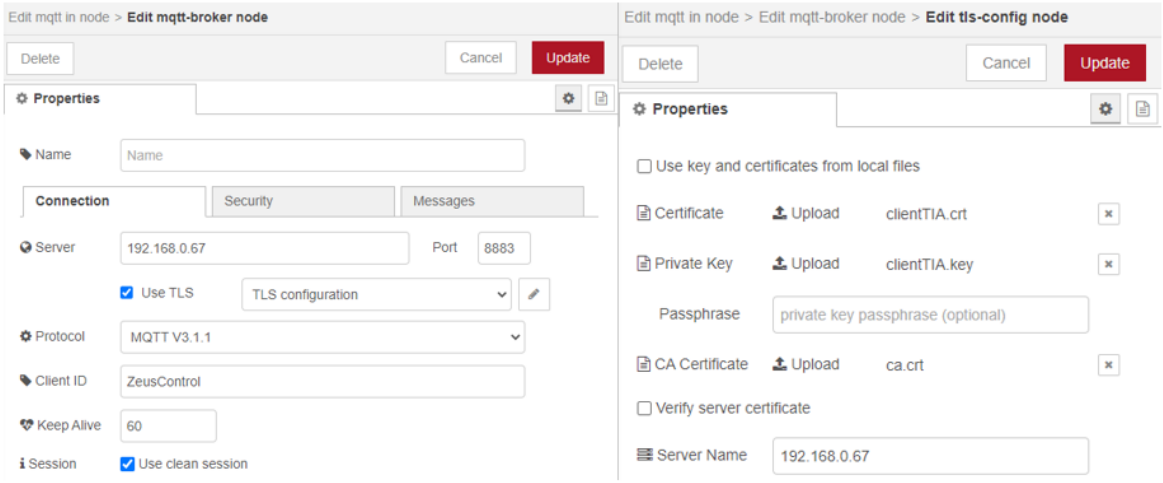


Ilustración 48: Nodo Subscriptor MQTT (encriptado)

Finalmente, para poder realizar nuestra solución de forma encriptada, también es necesario introducir los certificados en nuestro cliente publicador del PLC. Para ello nos dirigimos en el TIA Portal a las propiedades del PLC, donde encontraremos una sección de "Administrador de certificados" y habilitaremos la casilla de la Ilustración 49. Después hemos de "proteger este proyecto" desde las opciones de seguridad, donde tendremos que introducir un usuario y contraseña. Ahora ya podemos registrarnos como administradores y usar las opciones de seguridad. Nos dirigimos en el menú al "Administrador de certificados" y allí importamos el certificado del CA como "Autoridades de certificación raíz y certificados acreditados" y la key y certificado del cliente unificados en formato P12 como "Certificados de dispositivos". En la Ilustración 50, se muestra como pasar a formato P12.

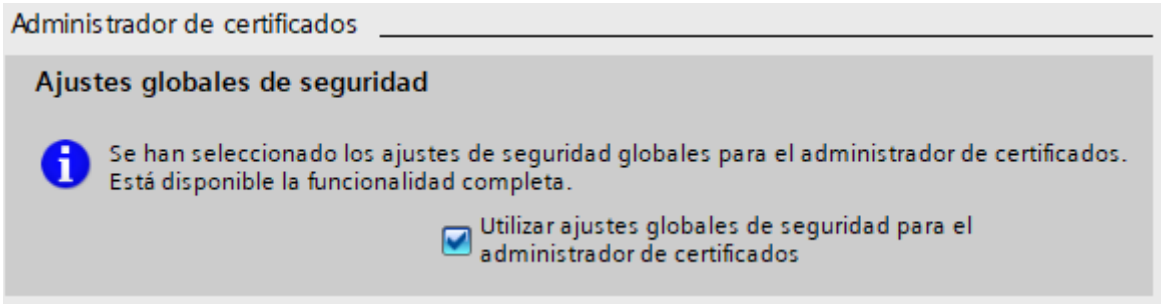


Ilustración 49: Habilitar el Administrador de certificados

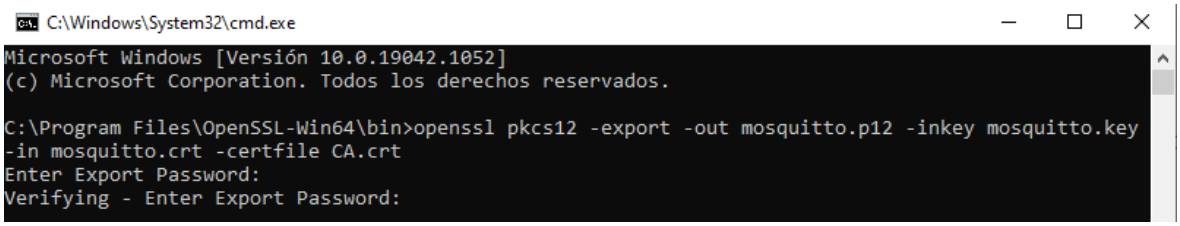


Ilustración 50: Conversión a archivo P12

Con los certificados importados al Administrador global de certificados, ya se pueden cargar al PLC desde "Certificados de dispositivo" en las propiedades del PLC. Una vez introducidos el certificado del CA y el archivo P12 con la key y certificado del cliente, ya solo queda cambiar el puerto e introducir los ID asociados a los certificados en la función de cliente MQTT tal y como se muestra en la Ilustración 51.

MQTT

	Nombre	Tipo de datos	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ...
4	tcpConnParam	"LMQTT_typeTcpCo...		☐	☑	☑	☑
5	useQdn	Bool	false	☐	☑	☑	☑
6	hwIdentifier	HW_ANY	64	☐	☑	☑	☑
7	connectionID	CONN_OUC	16#10	☐	☑	☑	☑
8	qdnAdressBroker	String[254]	''	☐	☑	☑	☑
9	ipAdressBroker	Array[0..3] of USInt		☐	☑	☑	☑
10	localPort	UInt	2000	☐	☑	☑	☑
11	mqttPort	UInt	8883	☐	☑	☑	☑
12	activateSecureConn	Bool	true	☐	☑	☑	☑
13	validateSubjectAlter...	Bool	false	☐	☑	☑	☑
14	idTlsServerCertificate	UDInt	5	☐	☑	☑	☑
15	idTlsOwnCertificate	UDInt	6	☐	☑	☑	☑

Ilustración 51: Parámetros de conexión TCP (encriptado)

5.3 IMPLEMENTACIÓN

Es necesario cargar el programa en el PLC y conectarlo con el ordenador mediante Ethernet. Una vez arrancados mosquitto con la configuración adecuada, Node-Red, InluxDB y Grafana, ya podemos visualizar las señales tal y como se muestra en la Ilustración 52, donde vemos representadas señales del tipo seno y coseno con distintas frecuencias y del tipo diente de sierra.

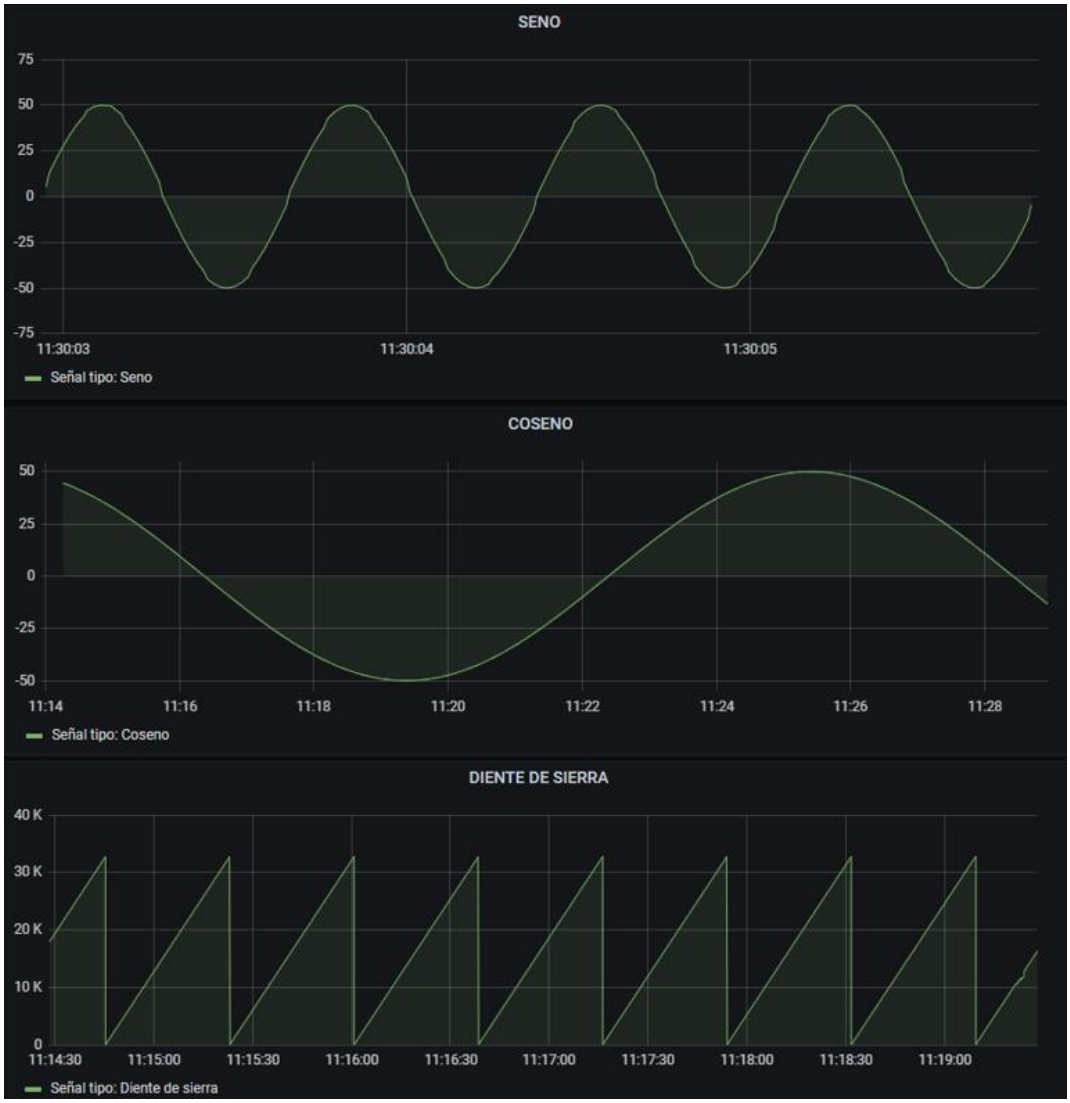


Ilustración 52: Gráficas de las señales en Grafana

5.4 ANÁLISIS DEL SISTEMA DE CÁLCULO DEL OEE Y MONITORIZACIÓN DE UNA PASTEURIZADORA

El segundo modelo realizado en este proyecto realizará la comunicación entre el PLC y Node-Red mediante la comunicación por enlaces S7, la cual nos permite no solo leer los datos del PLC, si no también introducir datos en el mismo gracias a Node-Red.

El PLC se programará para llevar a cabo una simulación de una línea de pasteurización simple y calcular su OEE (Overall Equipment Effectiveness). Posteriormente, Node-Red se hará cargo de recibir los resultados por S7 y llevará a cabo 3 formas distintas de representación. En primer lugar, a través de una librería de dashboard que ofrece Node-Red, la cual da la posibilidad de integrar interruptores y cursores para llevar a cabo conjuntamente la simulación y monitorización. En segundo lugar, se emplearán las mismas herramientas que en la anterior solución, InfluxDB y Grafana, con nuevos dashboards personalizados y una alarma configurada para notificar vía email en caso de que el OEE total de la línea este por debajo del 50%. Por último, se configurará un bróker cloud (MyQttHub), para enviar desde un cliente publicador en Node-Red, los datos hasta un cliente subscriptor en un Smartphone.

El modelo de la línea será como el representado en la Ilustración 53, supondremos una máquina pasteurizadora que podremos encender, apagar y regular su caudal de funcionamiento. A esta máquina, llega la leche y se la calienta hasta la temperatura de pasteurización, para asegurar la eliminación de bacterias contenidas en la misma. En este desarrollo se ha considerado una temperatura de pasteurización de 72ºC. Además, en este proyecto, vamos a considerar que hay un termómetro a la salida de la pasteurizadora encargado de activar una recirculación a la pasteurizadora en caso de que no se llegue a alcanzar la temperatura de pasteurización. Finalmente, consideraremos también que hay un refractómetro encargado de determinar la calidad de la leche para que, en el caso de no estar dentro del margen admisible, se vierta como desecho para su reciclado. En caso de estar dentro del margen admisible, se almacenará en un depósito, como producto útil.

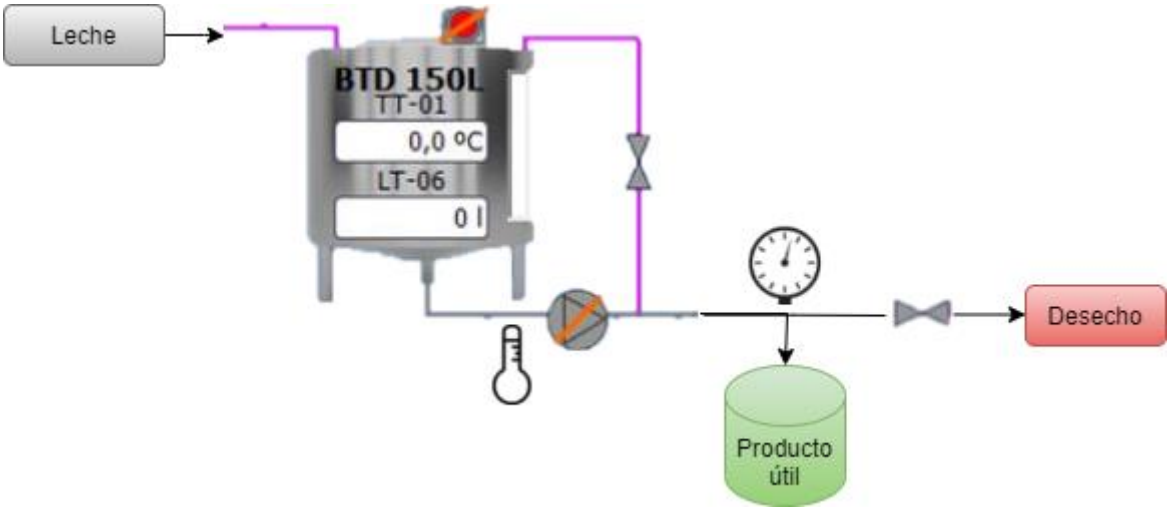


Ilustración 53: Modelo de la línea de pasteurización

5.5 DISEÑO

En este apartado se explicarán paso a paso los procedimientos realizados con cada una de las herramientas para poder llevar a cabo la realización de este modelo.

5.5.1 PROGRAMACIÓN EN TIA PORTAL DEL MODELO DE LA LÍNEA DE PASTEURIZACIÓN

En este proyecto, se quería simular el funcionamiento continuo de una línea de producción con hasta 3 turnos por jornada, con la intención de reiniciar las variables acumuladas en cada cambio de turno. Para ello, se creó un tipo de variable llamado "T_TURNO" con la que poder establecer un vector de 3 turnos en una jornada de producción. Este tipo de variable contiene las horas de inicio y fin del turno, así como los resultados acumulados finales de los cálculos de OEE, disponibilidad, rendimiento y calidad.

Se creó una función genérica para el cálculo del OEE. Esta función se muestra en la Ilustración 55, y se encarga de realizar los cálculos del OEE a partir de los datos del sistema, es decir, de los datos que podríamos obtener directamente de la planta, como por ejemplo cuándo la máquina está funcionando, cuál es el caudal nominal, caudal instantáneo, cuándo el producto se está desechando, etc.

Esta función se encarga de gestionar una serie de contadores, como el tiempo de operación, el tiempo de operación planificado, la cantidad de litros producidos y la cantidad de litros útiles almacenados. Los contadores se actualizan con cada ciclo, y se hace uso del tiempo de ciclo para llevar a cabo los cálculos oportunos. La gestión de contadores se reinicia con cada fin de turno. Como se puede ver en la Ilustración 54, las variables de los contadores son de tipo real, lo cual no supone ningún impedimento de saturación de los contadores ante turnos de 8 horas, sin embargo, para turnos más largos, sería necesario emplear variables Dint y realizar los cálculos con el tiempo de ciclo en milisegundos para evitar así que saturen los contadores y poder llevar a cabo turnos de hasta 24 horas.

DB_OEE

	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ...	Valor de a...
1	Static				☐	☐	☐	☐	☐
2	OEE_ACUM	Real	0.0	0.0	☐	☑	☑	☑	☐
3	DISP_ACUM	Real	4.0	0.0	☐	☑	☑	☑	☐
4	REND_ACUM	Real	8.0	0.0	☐	☑	☑	☑	☐
5	REND_INST	Real	12.0	0.0	☐	☑	☑	☑	☐
6	CALIDAD_ACUM	Real	16.0	0.0	☐	☑	☑	☑	☐
7	TURNO	Array[0..2] of "T_TU...	20.0		☐	☑	☑	☑	☐
8	N_TURNO	Int	176.0	0	☐	☑	☑	☑	☑
9	TIEMPOCICLOACTUAL	Real	178.0	0.0	☐	☑	☑	☑	☐
10	TIEMPOCICLOMEMORIA	LReal	182.0	0.0	☐	☑	☑	☑	☐
11	TPO	Real	190.0	0.0	☐	☑	☑	☑	☐
12	TO	Real	194.0	0.0	☐	☑	☑	☑	☐
13	LITROS_ACUM	Real	198.0	0.0	☐	☑	☑	☑	☐
14	LITROS_TANQUE	Real	202.0	0.0	☐	☑	☑	☑	☐
15	FECHAHORA_ACTUAL	DTL	206.0	DTL#1970-01-01-(	☐	☑	☑	☑	☐
16	RET	DTL	218.0	DTL#1970-01-01-(	☐	☑	☑	☑	☐

Ilustración 54: Bloque de datos de la función OEE

Además de la gestión de los contadores, con cada ciclo la función realiza los cálculos del OEE, para los que necesita los valores acumulados de cada turno.

En la Ecuación 1 podemos ver el cálculo de la Disponibilidad acumulada desde el inicio del turno, calculando el cociente del TO entre el TPO, teniendo en cuenta que puede que la máquina no haya estado encendida todo el tiempo que se tenía pensado debido a errores o imprevistos.

$$E.1 \quad Disponibilidadad = \frac{Tiempo\ total\ de\ Operación\ (TO)}{Tiempo\ de\ Operación\ Planificado\ (TPO)} \cdot 100$$

En la Ecuación 2 vemos el cálculo del Rendimiento, con el cociente de los litros totales producidos entre los que se hubiesen producido durante el tiempo de operación si la máquina hubiese funcionado en condiciones nominales. Al rendimiento le pueden afectar 2 aspectos: En primer lugar, el caudal real de funcionamiento, que no siempre es el nominal. Y, en segundo lugar, la temperatura de pasteurización, ya que, si está por debajo de la adecuada, comienza una recirculación con la que el rendimiento instantáneo es nulo, lo que afecta altamente al rendimiento acumulado de dicho turno.

$$E.2 \quad Rendimiento = \frac{Litros\ totales\ producidos}{Litros\ que\ se\ podrían\ haber\ producido} \cdot 100 = \frac{Litros\ totales\ producidos\ (L)}{Caudal\ nominal(L/h) \cdot TO(s)/3600} \cdot 100$$

En la Ecuación 3 tenemos el cálculo de la Calidad, a través del cociente de los litros de producto útiles entre los litros totales producidos, teniendo en cuenta que puede que haya producto que no fuese útil y haya sido desechado. La medida del refractómetro es la que determina cuando el producto es útil o se desecha, lo que quiere decir que es nuestra forma de simular la Calidad acumulada de la línea.

$$E.3 \quad Calidad = \frac{Litros\ de\ producto\ útil}{Litros\ totales\ producidos} \cdot 100$$

Finalmente, en la ecuación 4 tenemos el cálculo del OEE acumulado en el turno, el cual se obtiene como producto de los resultados de las 3 ecuaciones anteriores. El OEE es información muy útil para evaluar el funcionamiento de una planta y con la adquisición de estos datos, se permite saber dónde están los errores y poder tomar medidas para optimizar el funcionamiento de la producción.

$$E.4 \quad OEE = Disponibilidad \cdot Rendimiento \cdot Calidad/10000$$

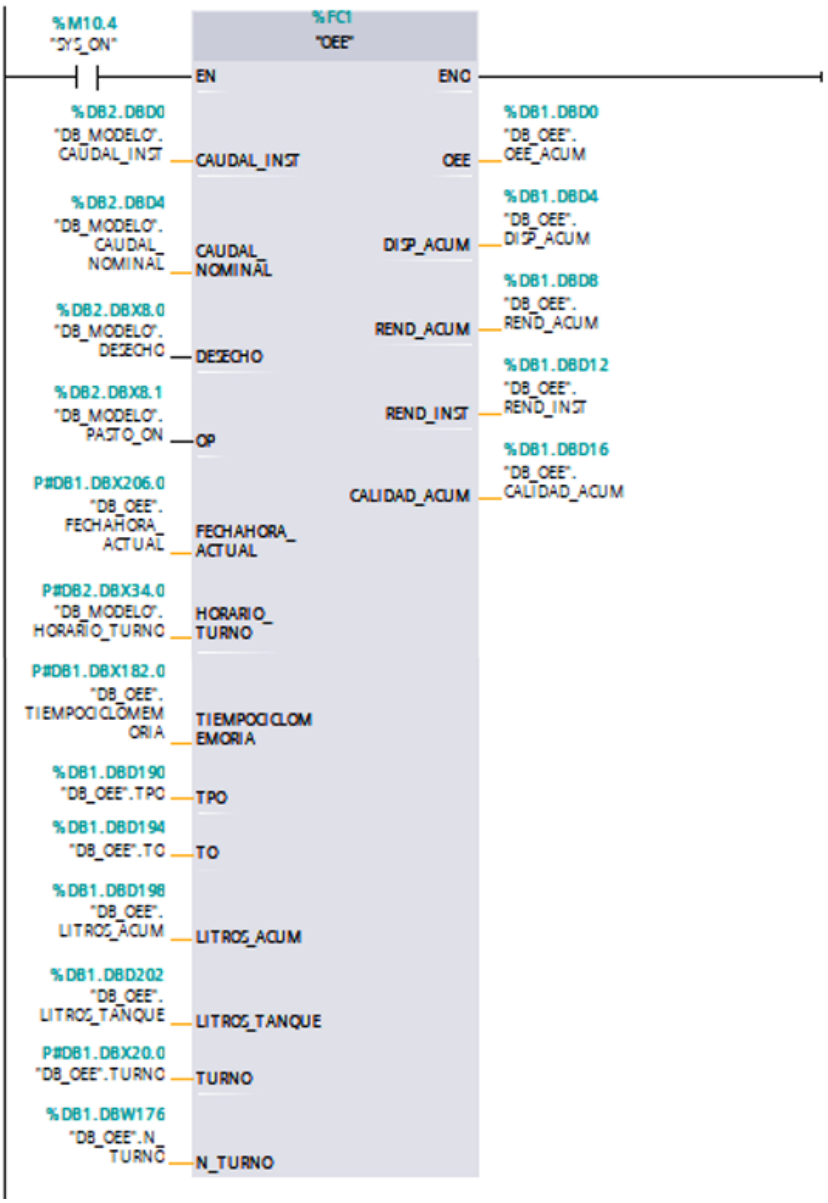


Ilustración 55: Función de cálculo del OEE

Para poder simular los cambios de turnos del modelo y verificar su correcto funcionamiento, se emplearon las funciones predeterminadas de la Ilustración 56 e Ilustración 57. Estas nos permiten leer la fecha actual del PLC para poder compararlo con los horarios de los turnos y escribir (cambiar) la fecha para poder realizar los cambios de turno sin tener que esperar 8 horas.

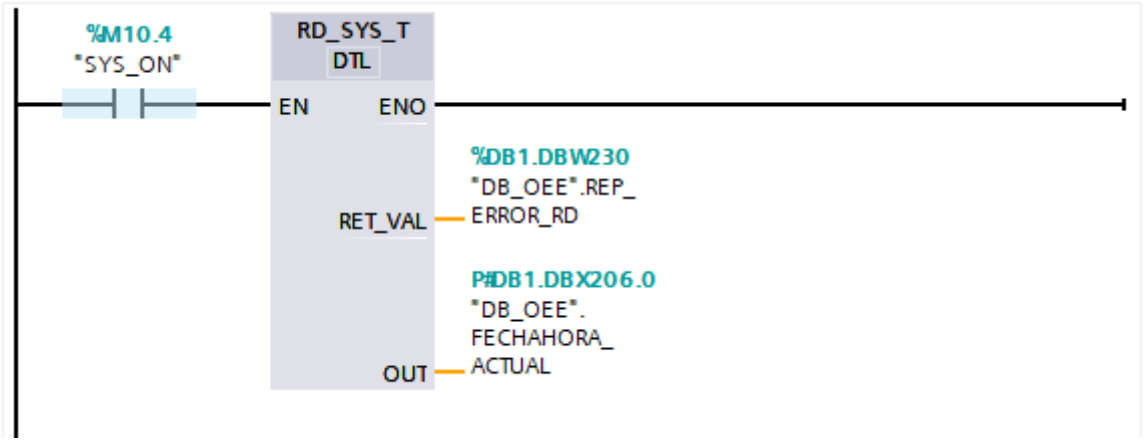


Ilustración 56: Lectura de fecha actual

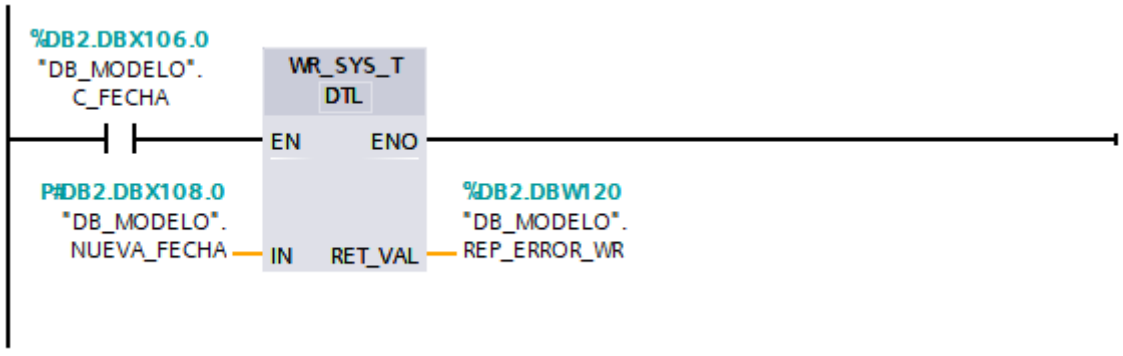


Ilustración 57: Escritura de fecha

Finalmente se creó una función con la que poder simular el funcionamiento de la línea. Esta función es la mostrada en la Ilustración 58, y se encarga de establecer el caudal instantáneo que se desea simular, si hay recirculación, en cuyo caso el caudal instantáneo es nulo y si el producto es útil o debe ser desechado.

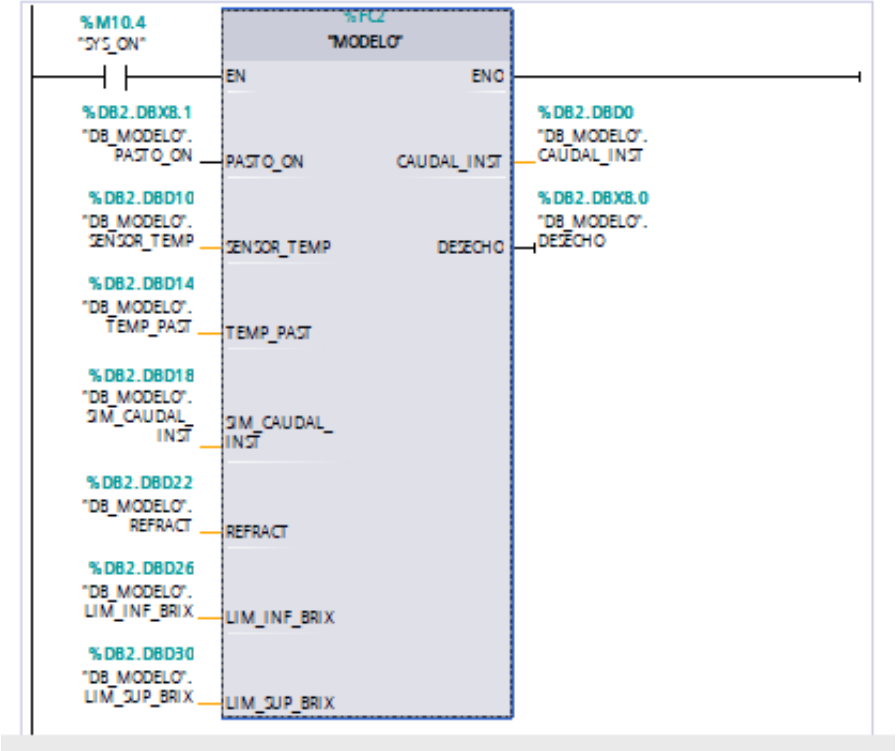


Ilustración 58: Función de simulación del modelo

La configuración de la comunicación S7 es muy sencilla, ya que no se necesita realizar ninguna programación adicional, ni usar funciones como en el caso del MQTT. Tan solo es necesario permitir la comunicación PUT/GET, tal y como se muestra en la Ilustración 59, y asegurarnos que los bloques del programa tienen inhabilitada la opción de acceso optimizado al bloque, como se puede observar en la Ilustración 60. Esto es debido a que la comunicación es antigua y estamos usando un PLC bastante posterior. Con la comunicación S7, Node-Red puede acceder directamente a los valores de las variables lo cual es realmente cómodo ya que se puede acceder a todas las variables a la vez, sin necesidad de ordenar los datos para su posterior envío.

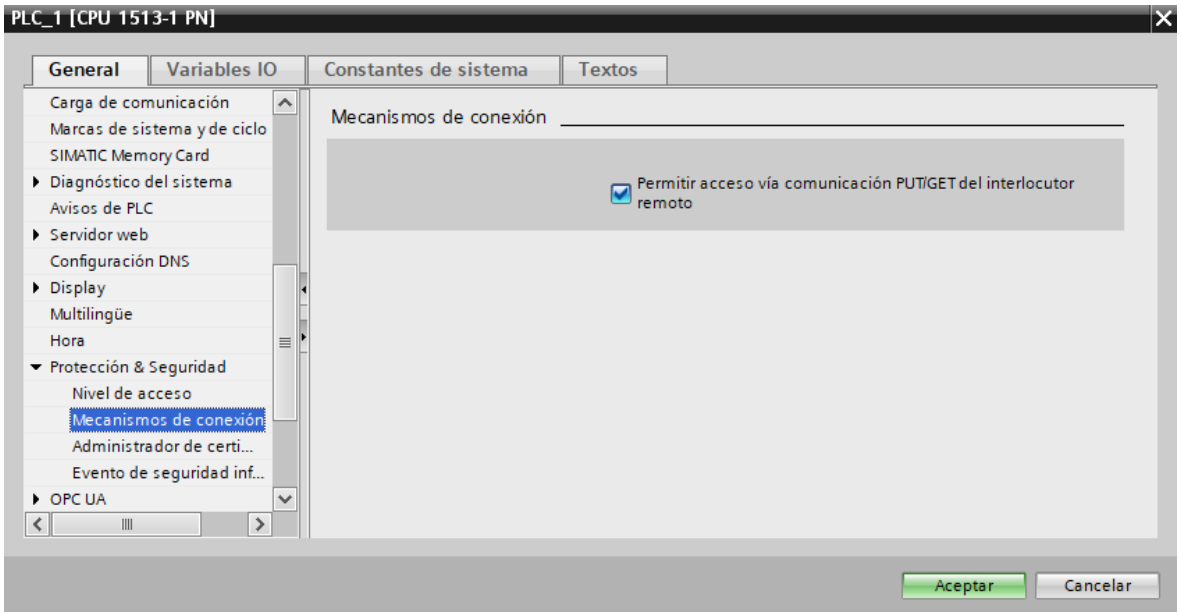


Ilustración 59: Permitir comunicación PUT/GET

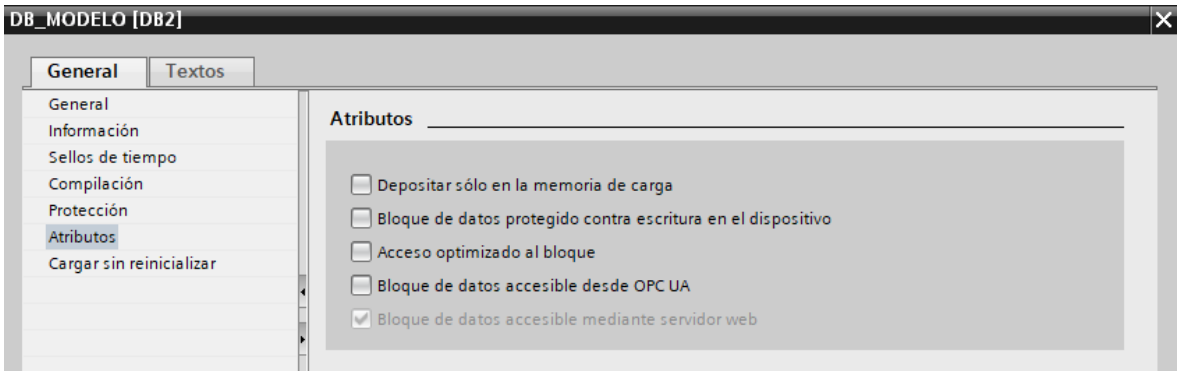


Ilustración 60: Inhabilitación de acceso optimizado al bloque

5.5.2 NODE-RED

Para poder llevar a cabo la comunicación S7 con Node-Red, es necesario instalar una librería adicional. Esta es node-red-contrib-s7 y se instala con el siguiente comando:

npm install node-red-contrib-s7

También instalaremos la librería de node-red-dashboard para la visualización y simulación del modelo. Se instala con el siguiente comando:

npm install node-red-dashboard

Como se puede ver en la Ilustración 61, el diseño realizado en esta solución está inspirado en el desarrollo anterior de la comunicación MQTT. La diferencia principal está en la recepción de datos por S7 a través del nodo "Enlace S7 con PLC" y la presencia de nodos para introducir mensajes en el PLC. Estos nodos se configuran tal y como se muestra en la Ilustración 62 e Ilustración 63, estableciendo las direcciones de las variables del PLC a las que se desea acceder, la dirección IP del PLC, así como su puerto de comunicación PUT/GET (102) y el Rack y Slot en el que se encuentra. También vemos que se puede establecer el ciclo de lectura o escritura, si deseamos hacerlo solo cuando el valor ha cambiado y el número de variables introducidas que queremos a las que queremos acceder de todas las introducidas.

En este caso, se accede a los valores de los cálculos del OEE y se dividen según su tópico, que, en este caso, es el nombre que le hemos asociado a la variable. Una vez divididos, se usan los nodos de dashboard para personalizar las gráficas y formas de visualización de los rendimientos. También los introducimos en series de datos distintas en InfluxDB, como se hizo anteriormente. Con los nodos de la librería dashboard también introducimos un interruptor para encender y apagar la pasteurizadora y cursores con los que establecer: la temperatura a la salida de la pasteurizadora, la medida del refractómetro y el caudal de simulación.

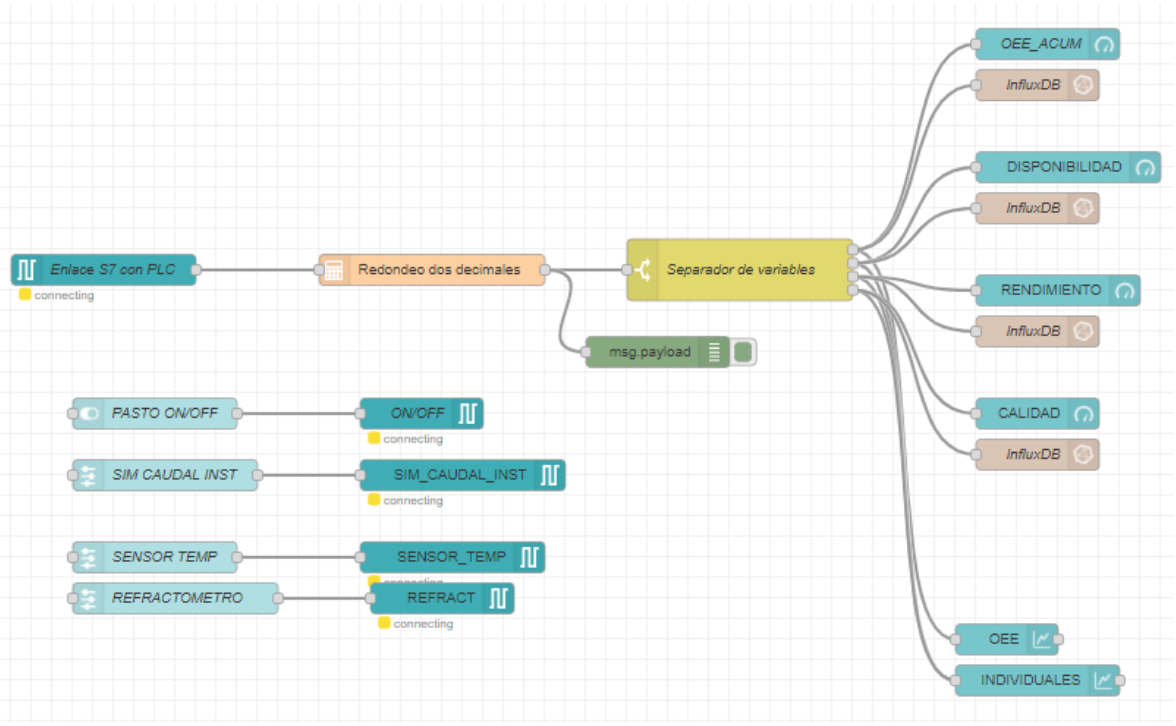


Ilustración 61: Diseño Node-Red (S7)

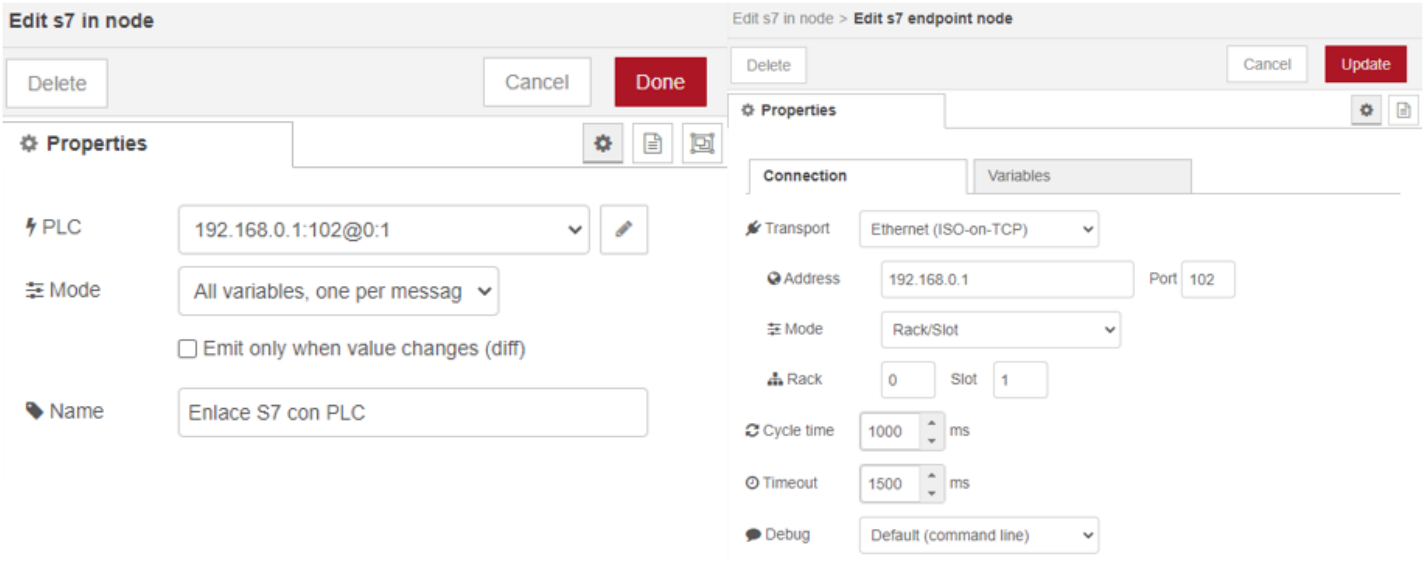


Ilustración 62: Nodo Enlace S7

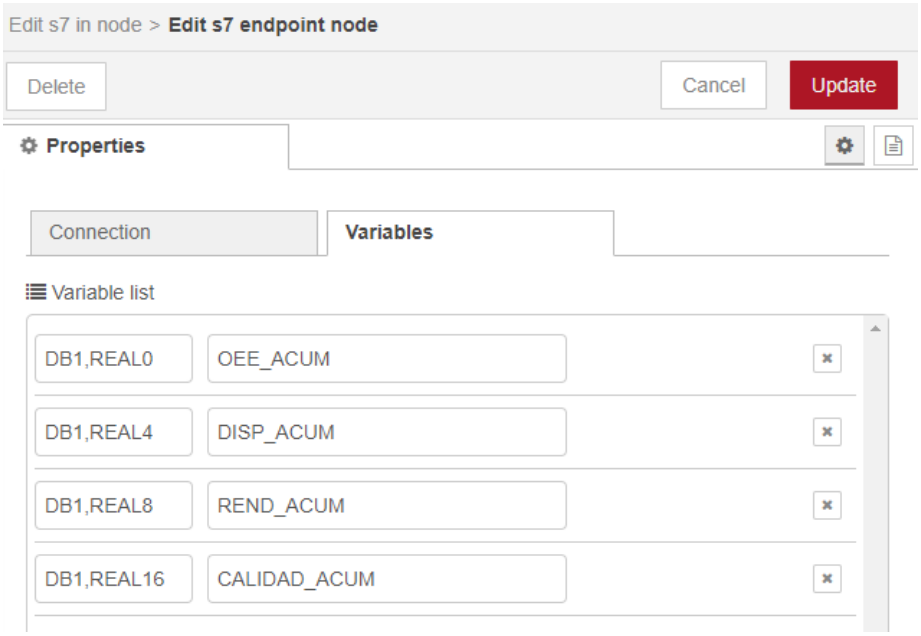


*Ilustración 63: Nodo Enlace S7**

5.5.3 GRAFANA

En esta solución se siguen los mismos pasos que en la solución anterior para configurar la base de datos de la que se quieren extraer los datos, con la diferencia que en esta ocasión se ha creado una representación distinta y personalizada. Se han representado: OEE, Disponibilidad, Rendimiento y Calidad acumulados en cada instante, una gráfica para el poder ver el OEE a lo largo del turno y otra para ver Disponibilidad, Rendimiento y Calidad juntos a lo largo del turno. Estas representaciones pueden verse en el capítulo 5.6.

También se ha configurado una alarma, para que sí el OEE es inferior al 50%, se mande un email notificando de la incidencia. Para ello, hay que configurar un correo que tenga habilitado el acceso a aplicaciones poco seguras. En segundo lugar, hemos de dirigirnos al archivo de configuración de Grafana y configurarlo tal y como se muestra en la Ilustración 64, vemos que es necesario introducir el correo y contraseña. En este caso solo queremos usar una cuenta de Gmail, de forma que se ha configurado para que se envíe la alerta a sí misma.

```
################################### SMTP / Emailing ##################
[smtp]
enabled = true
host = smtp.gmail.com:587
user = joxgil99@gmail.com
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
password = 1234
cert_file =
key_file =
skip_verify = true
from_address = joxgil99@gmail.com
from_name = Grafana
ehlo_identity =
startTLS_policy =

[emails]
welcome_email_on_sign_up = false
templates_pattern = emails/*.html
```

Ilustración 64: Edición grafana.conf para Alarmas

Una vez editada la configuración, nos dirigimos a Grafana y en "Alerting" configuramos un nuevo "Notification Channel", tal y como se muestra en la Ilustración 65, vemos que es necesario seleccionar el tipo Email e introducir el correo que se introdujo en la configuración.

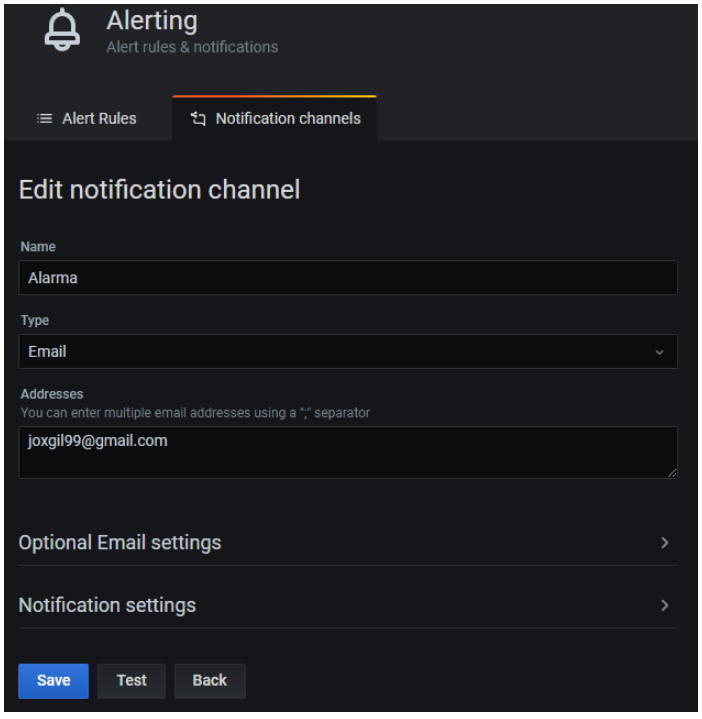


Ilustración 65: Creación de Notification Channel

Finalmente, dentro de la edición de la gráfica del OEE introducimos la alarma que deseamos, tal y como se puede observar en la Ilustración 66. Vemos que se puede configurar la condición que deseemos, en este caso, cuando la media de los datos del último minuto sea menor a 50. También podemos configurar como en este caso, que la alerta se evalúe cada minuto durante un minuto. Por último, escribimos el mensaje que deseamos que se envíe y el "Notification Channel" por el que mandar la notificación.

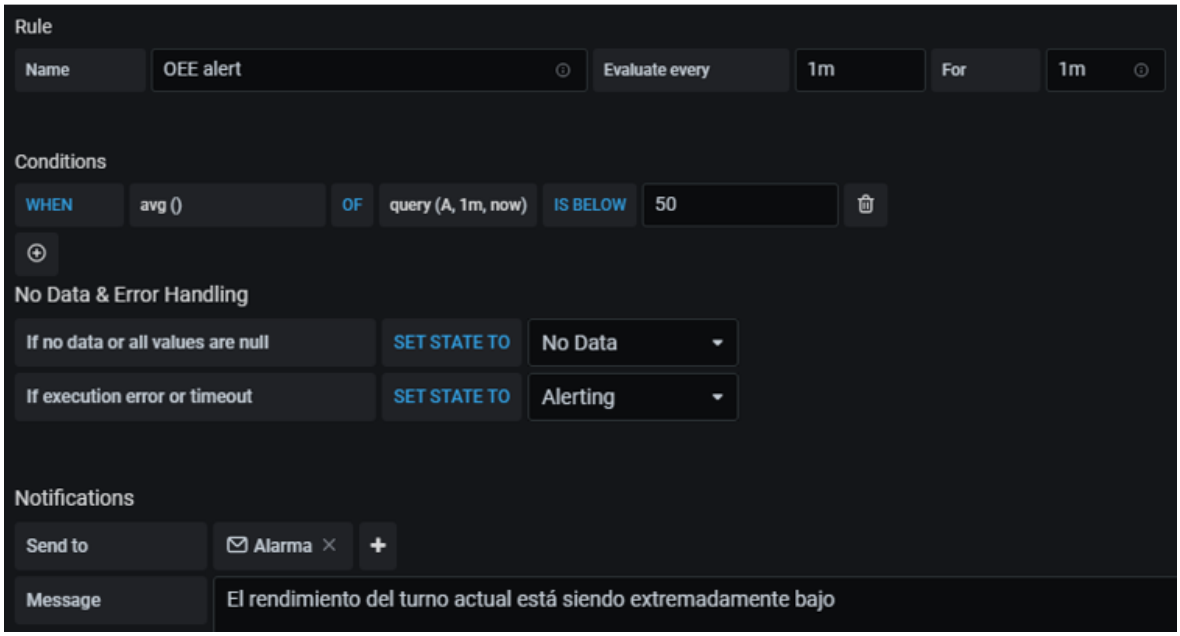


Ilustración 66: Creación de Alarma

5.5.4 MYQTTHUB

Desde la página oficial de MyQttHub, https://myqtthub.com/ se puede uno dar de alta y tener un servidor cloud gratuito con límites de cantidad de dispositivos y mensajes. Una vez dado de alta, se puede acceder a nuestro bróker y configurar los dispositivos que vamos a emplear, en este caso, como se puede observar en la

Ilustración 67, queremos un cliente publicador (Pub_OEE), para enviar desde Node-Red los distintos datos con tópicos diferentes y un cliente subscriptor (Subscriptor), para el dispositivo que va a recibir los datos, en este caso, un Smartphone.

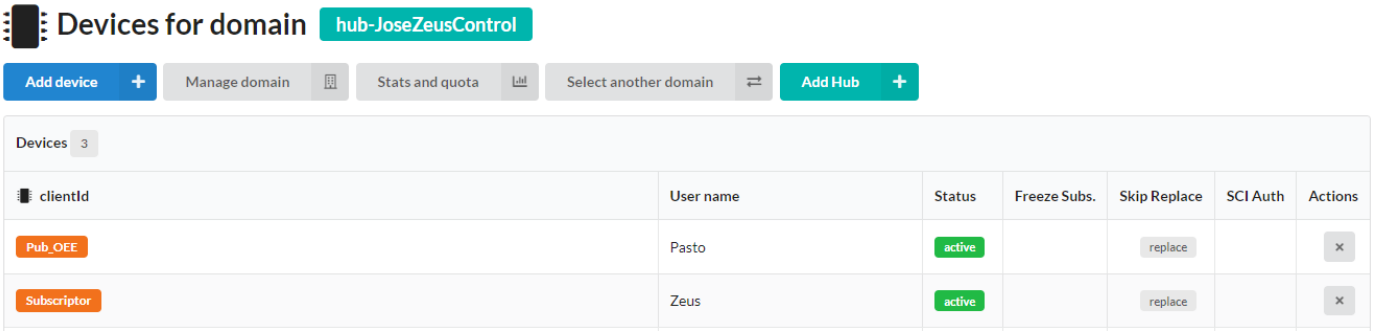


Ilustración 67: Dispositivos del bróker MyQttHub

Finalmente, se modifica el diseño de Node-Red con los nodos del cliente publicador para enviar los mensajes por MQTT al bróker en vez de almacenarlos en InfluxDB como anteriormente, tal y como se muestra en la Ilustración 68.

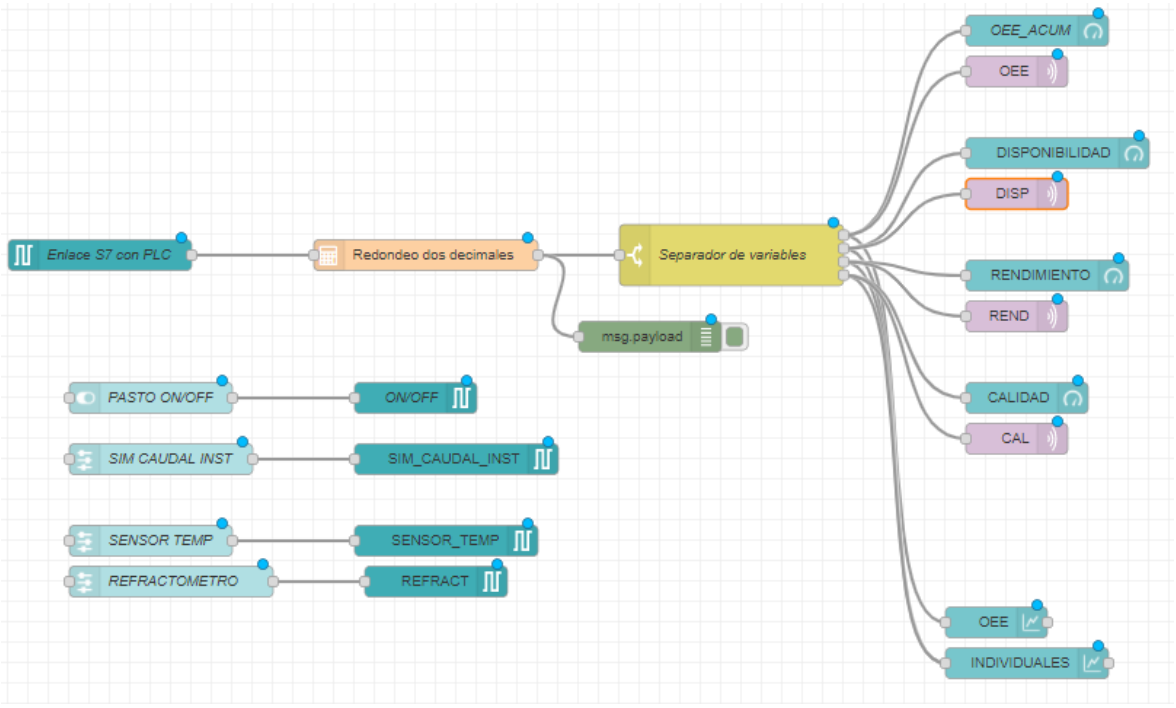


Ilustración 68: Diseño Node-Red (S7+MyQttHub)

5.5.5 MQTT DASH

Esta aplicación puede descargarse desde el Play Store y nos permite configurar un cliente MQTT en un Smartphone. En la Ilustración 69, podemos observar que la configuración de un cliente en esta aplicación es igual a la de cualquier otra. Es necesario introducir la dirección del bróker, en este caso *node02.myqtthub.com* y el puerto de comunicación MQTT 1883. Finalmente hemos de introducir el ID, usuario y contraseña del cliente creado en MyQttHub.

Una vez establecida la conexión con el bróker, se pueden crear distintas formas de representación de los datos, tal y como se muestra en la Ilustración 70, que se muestra cómo crear una forma de representar los datos del OEE, como en el capítulo 5.6.

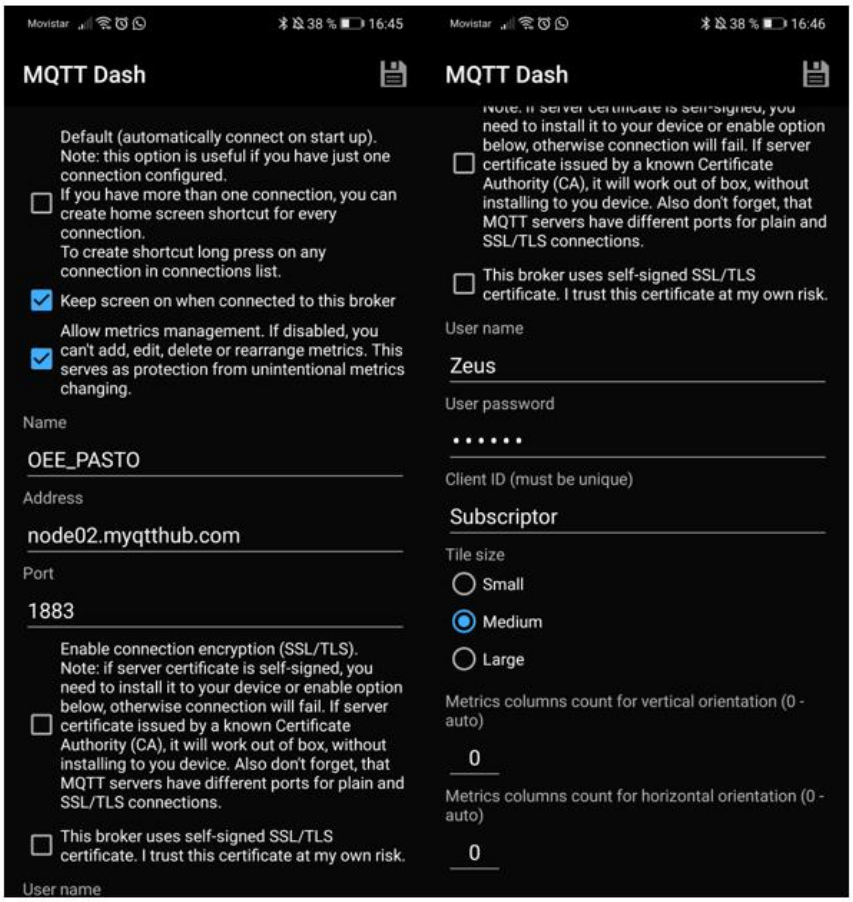


Ilustración 69: Configuración de un cliente en MQTT Dash

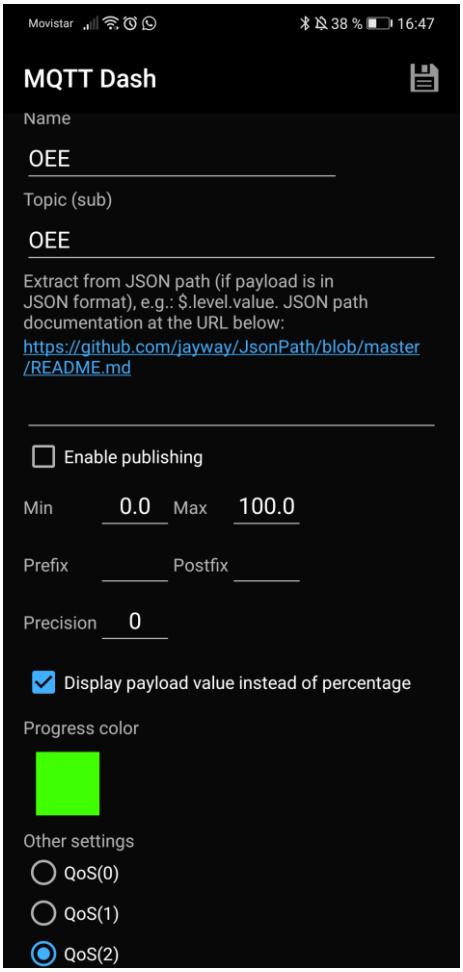


Ilustración 70: Creación de representación de datos MQTT Dash

5.6 IMPLEMENTACIÓN

A continuación, se muestra la puesta en conjunto de todas las herramientas, para dar fruto a la solución final. Para esta solución, es necesario conectar el PLC con el ordenador por Ethernet y arrancar Node-Red, InfluxDB y Grafana con los desarrollos indicados anteriormente.

En la Ilustración 71 podemos ver el dashboard de visualización de Node-Red, donde podemos ver las distintas formas de representación de los datos del OEE de la pasteurizadora. Además, se puede observar que se pueden variar los parámetros de la simulación mediante los inputs incorporados en el dashboard. Esto es muy cómodo, ya que se puede llevar a cabo la simulación sin la necesidad de TIA Portal.

En la Ilustración 72 podemos observar el dashboard de Grafana que resulta más estético y profesional que el de Node-Red, además este ofrece muchas opciones distintas de personalización y herramientas para el tratado de datos.

En la Ilustración 73 podemos ver un ejemplo del uso de alarmas. Vemos que evalúa cada minuto la situación y en primer lugar salta un “warning”, representado con la barra naranja y luego salta la alerta, representada con una barra roja. Este sistema de “warning” previo a la alerta puede resultar muy útil como forma de “histéresis”, para no enviar notificaciones constantemente. En la Ilustración 74, vemos la llegada de 2 correos, uno notificando la incidencia y otro informando que ya se ha salido del estado de alerta, lo cual se representa con la barra verde de la Ilustración 73. La Ilustración 75 nos muestra el contenido del correo, con el mensaje de alerta establecido y el valor de la infracción. En este caso nos muestra el valor calculado de la media, como se especificó en su configuración. Por último, en la Ilustración 76, podemos observar un historial de alertas con el que poder analizar los incidentes.

Finalmente, en la Ilustración 77, podemos observar la visualización configurada en el cliente Smartphone para monitorizar los cálculos del OEE de la planta.

Ilustración 71: Dashboard Node-Red

Ilustración 72: Dashboard Grafana

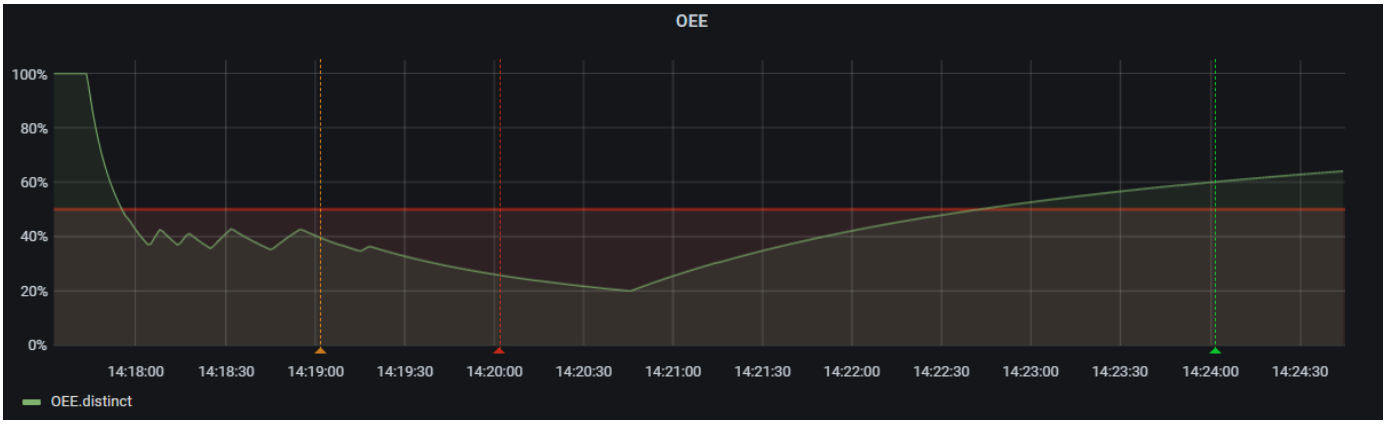


Ilustración 73: Demostración de Alarma (Grafana)

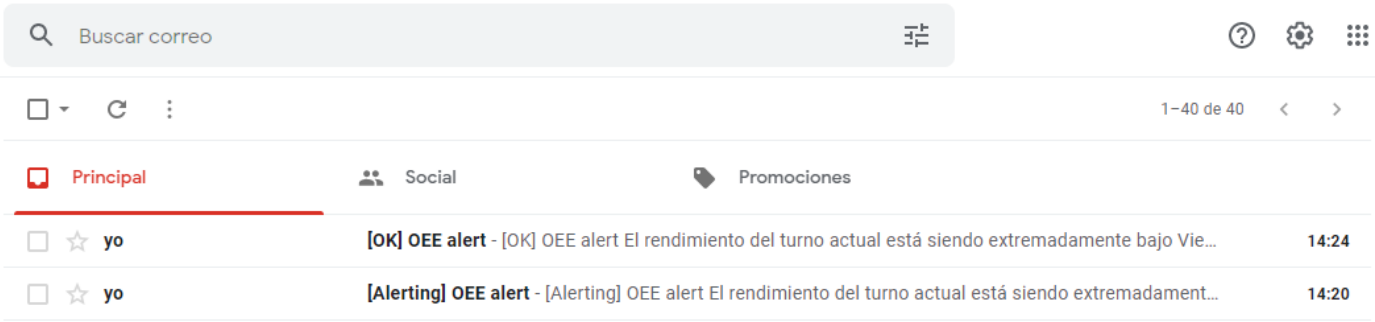


Ilustración 74: Notificación de Alarma por correo

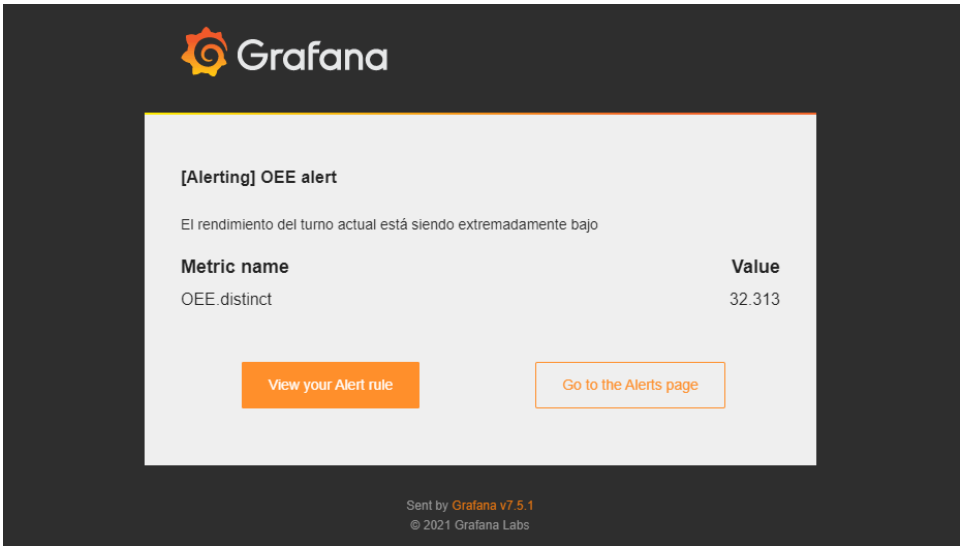


Ilustración 75: Correo de Alerta

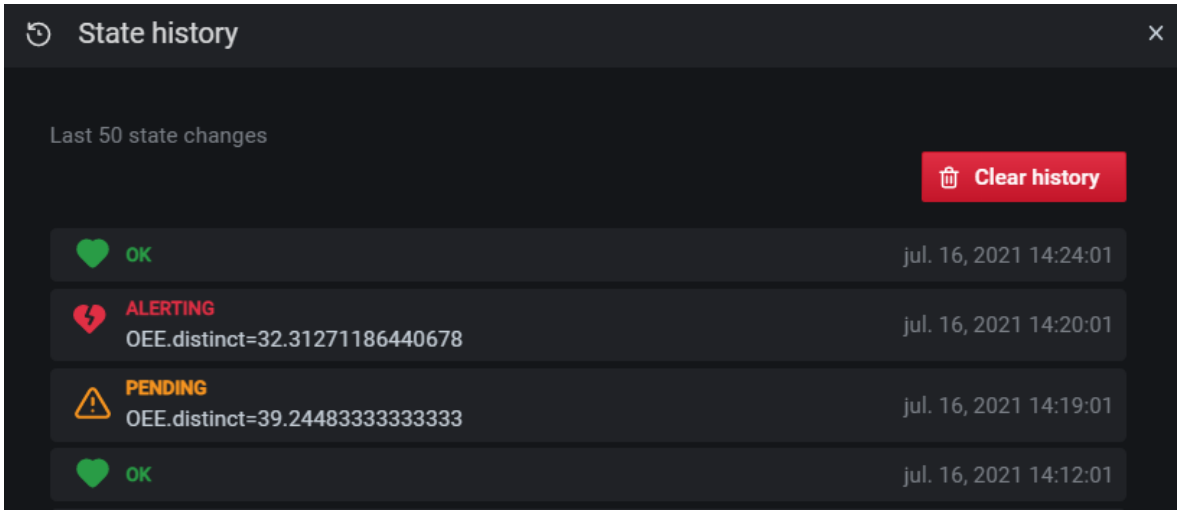


Ilustración 76: Historial de Alertas

Ilustración 77: Visualización MQTT Dash

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se va a analizar el envío de datos desde el PLC por el protocolo MQTT. Esta solución llevaba a cabo el envío de los datos de 4 señales distintas de forma consecutiva. Una vez terminado el desarrollo de dicha solución y visto su correcto funcionamiento se procedió a realizar pruebas y analizar el resultado.

En primer lugar, se creó una nueva base de datos con InfluxDB y se cronometró durante un minuto el envío de datos por MQTT. Posteriormente se realizó un recuento de los datos almacenados en la base de datos para cada una de las cuatro señales. Los resultados fueron los siguientes:

- Signal0: 3281 datos
- Signal1: 3280 datos
- Signal2: 3279 datos
- Signal3: 3279 datos

Lo que resulta un total de 13118 mensajes enviados en un intervalo de 1 minuto, con un tiempo de envío aproximado de 5ms. Este resultado demuestra las grandes velocidades de conectividad que ofrece el protocolo MQTT, las cuales exceden las necesidades de visualización y monitorización en tiempo real. Normalmente estas soluciones ofrecen actualización visual cada 5 segundos, dado que, para su función no son necesarios tiempos de actualización menores.

A la vista de estos datos también podemos observar que los datos no siguen el orden de envió deseado, ya que no todas las señales tienen el mismo número de datos, aunque sí muy parecido.

En segundo lugar, se decidió revisar los datos de la señal tipo "Diente de sierra", ya que era la más sencilla para poder analizar sus datos. Esta función tiene configurada un incremento de una unidad por ciclo del PLC. Sin embargo, al ver los datos almacenados se observó que los datos de la señal tipo "Diente de sierra" tenían un incremento constante de 16 unidades, en vez del incremento constante de 4 unidades esperado (Se envía el dato de una señal por ciclo del PLC y hay 4 señales). Esto nos demuestra que el PLC no es capaz de enviar un mensaje por MQTT cada ciclo del PLC.

Finalmente se representaron los datos en Grafana del intervalo de 1 minuto, tal y como podemos ver en la Ilustración 78, los datos no se envían de forma regular, ya que en algunos momentos se envían 2 datos más seguidos. Esto puede observarse mejor en la Ilustración 79, que vemos señalizados 3 datos consecutivos entre los que hay dos tiempos distintos, en concreto, estos tiempos son de 1 y 2 centésimas de segundo. Estos tiempos son redondeados por Grafana.

A la vista de estos datos podemos determinar que sería conveniente optimizar la función de envío de datos MQTT para que se envíen de forma regular. Esta solución realizaba el máximo envío de datos posible. Si se ajusta el envío de datos mediante su activación tras 1 segundo, solucionaríamos el problema. Aun así, este problema no afecta a nuestra solución, ya que se sigue obteniendo la representación correcta y en tiempo real de los datos. Solo cambia el muestreo de las señales.

Ilustración 78: Datos diente de sierra (1min)

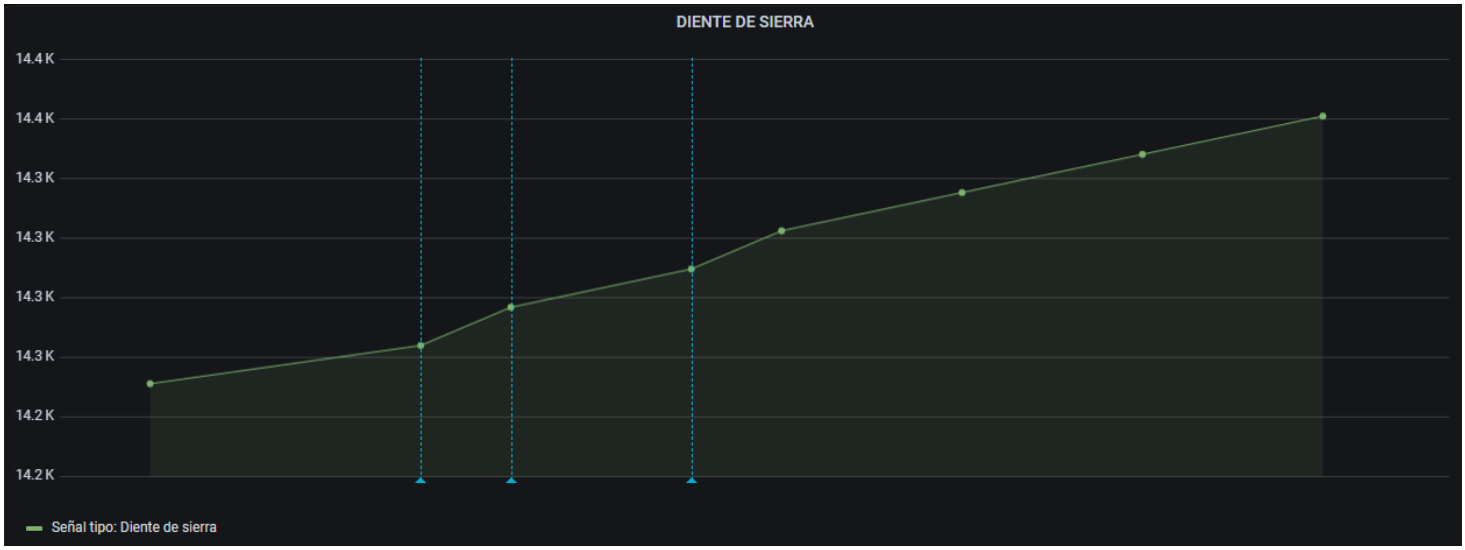


Ilustración 79: Datos diente de sierra (zoom)

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En conclusión, el IoT es una tecnología que gracias a la alta conectividad que ofrece resulta de gran utilidad para numerosas aplicaciones. Su aplicación en soluciones de adquisición y almacenamiento de datos, visualización y monitorización en tiempo real ya están desarrollas, aunque seguirán progresando. Con futuros avances, esta tecnología llegará a emplearse también en el control de procesos y actuadores en plantas industriales.

En este proyecto hemos conseguido realizar el envío progresivo de señales desde un PLC industrial mediante el protocolo MQTT para almacenarlas en una base de datos y obtener una representación gráfica de las señales en tiempo real. Estas gráficas podrían ser compartidas a través de Grafana Cloud.

La segunda solución desarrollada en este proyecto ha sido el cálculo del OEE de una simulación de una línea de pasteurización. Los resultados son enviados por comunicación por enlaces S7 (específica de SIEMENS) y posteriormente se visualizan los datos en distintos dashboards con una alarma de prueba configurada. También se ha usado un bróker cloud para compartir por MQTT los datos con un cliente Smartphone.

De estos desarrollos hemos concluido que:

- El MQTT es un protocolo de comunicación con mucho potencial en la adquisición masiva de datos, especialmente con la compañía de los Smartsensors.
- El envío de datos desde el PLC por S7 es mucho más cómodo que el MQTT, ya que por MQTT hay que programar la función y ordenar los datos para su correcto envío, mientras que por S7 se puede acceder directamente a los valores instantáneos de las variables que deseemos. Además, la comunicación S7 también nos permite enviar del mismo modo datos al PLC.

Para trabajos futuros se plantea la posibilidad de optimizar la función de envío de datos por protocolo MQTT, para que lo haga de forma uniforme. También la prueba y desarrollo de la función como cliente subscriptor para mandar así datos al PLC. Otra opción viable es implantar el bróker mosquitto en distintos dispositivos RaspBerry Pi y puentearlos mediante la opción Bridge que ofrece mosquitto, aunque sabemos que la aplicación de dispositivos RaspBerry Pi en la industria es un tanto arriesgada, ya que: carecen de la robustez necesaria, no son idóneos para comunicaciones industriales y son altamente sensibles a pérdidas de tensión.

Por último, también sería muy provechoso combinar la adquisición de datos y conectividad de estos desarrollos con conocimientos de Big Data y Machine/Deep Learning para llevar a cabo un tratado de los datos y conseguir nuevas soluciones beneficiosas.

Capítulo 8. BIBLIOGRAFÍA

[1] SIMATIC IOT2000 S7-Communication. SIEMENS, 2021. Available: https://support.industry.siemens/

[2] MQTT_Client for SIMATIC S7-1500 and S7-1200. SIEMENS, 2021. Available: https://support.industry.siemens/

[3] "MQTT - The Standard for IoT Messaging", Mqtt.org, 2021. [Online]. Available: https://mqtt.org/. [Accessed: 19- Jul- 2021].

[4] "Eclipse Mosquitto", Eclipse Mosquitto, 2021. [Online]. Available: https://mosquitto.org/. [Accessed: 19- Jul- 2021].

[5] "Node-RED", Nodered.org, 2021. [Online]. Available: https://nodered.org/. [Accessed: 19- Jul- 2021].

[6] "InfluxDB: Purpose-Built Open Source Time Series Database | InfluxData", InfluxData, 2021. [Online]. Available: https://www.influxdata.com/. [Accessed: 19- Jul- 2021].

[7] "Grafana", Grafana.org, 2021. [Online]. Available: https://grafana.com/. [Accessed: 19- Jul- 2021].

[8] "MyQttHub -- Servidor MQTT Cloud -- Alojamiento MQTT", Myqtthub.com, 2021. [Online]. Available: https://myqtthub.com/. [Accessed: 19- Jul- 2021].

[9] S. Cope, "Mosquitto SSL Configuration -MQTT TLS Security", Steve´s internet Guide, 2021. [Online]. Available: http://www.steves-internet-guide.com/mosquitto-tls/. [Accessed: 19- Jul- 2021].

[10] S. Cope, "Creating and Using Client Certificates with MQTT and Mosquitto", Steve's internet Guide, 2021. [Online]. Available: http://www.steves-internet-guide.com/creating-and-using-client-certificates-with-mqtt-and-mosquitto/. [Accessed: 19- Jul- 2021].

ANEXO I: ALINEACIÓN CON LOS OBJETIVOS PARA EL DESARROLLO SOSTENIBLE (ODS)

El IoT permite a la tecnología progresar en su interacción con el mundo físico, de forma que aporta numerosas aplicaciones para distintos sectores. Todo sector que utilice tecnología para interactuar con el mundo real puede beneficiarse ampliamente del IoT para mejorar su funcionamiento e incluso abrir las puertas a nuevas soluciones. Son numerosos los sectores en los que se puede aplicar, tales como: hostelería, comercio, agricultura, industrial, doméstico, salud y ciudades inteligentes.

La innovación y progreso de la tecnología IoT en todos estos sectores puede permitirnos alcanzar niveles más elevados de productividad económica. Además, su implantación podría aportar nuevas soluciones asequibles para empresas pequeñas y medianas, favoreciendo su crecimiento. Contribuyendo así al Objetivo de Desarrollo Sostenible de "Trabajo decente y crecimiento económico".

El IoT también colabora en gran medida con el Objetivo de Desarrollo Sostenible de "Producción y consumo responsable". Ya que se está empleando en la hostelería para la gestión de los almacenes de comida, supervisando la caducidad de los alimentos y realizando pedidos de los productos que sean necesarios. De esta forma se puede ajustar mejor el consumo necesario y prevenir en muchas ocasiones las grandes cantidades de desechos generados por este sector.

Otro Objetivo de Desarrollo Sostenible a los que contribuye el IoT, es el de "Industria, innovación e infraestructuras". Ya que su aplicación supone una modernización de la infraestructura, en la cual se está cobrando mucha importancia los avances en nuevos sensores. Estos son de muy bajo consumo y permanecen conectados a la red recopilando información de todo tipo del mundo real para su posterior análisis.

En conclusión, a través del IoT se pueden optimizar y mejorar la gran mayoría de las tecnologías relacionadas con los Objetivos de Desarrollo Sostenible, pudiendo dar lugar a un mejor sistema de limpieza y saneamiento del agua, mejoras en el sector de las energías y salud, y fomentando el desarrollo de ciudades inteligentes. Contribuyendo así a los siguientes Objetivos de Desarrollo Sostenible: "Agua limpia y saneamiento", "Energía asequible y no contaminante", "Salud y Bienestar" y "Ciudades y comunidades sostenibles".

ANEXO II: CÓDIGO DEL GENERADOR DE SEÑALES

```
FOR #i:= 0 TO #N_SEÑALES DO
         IF "DB_SEÑAL".SEÑAL[#i].ORD_ON THEN
            CASE "DB_SEÑAL".SEÑAL[#i].MODO OF

                     //SEÑAL MANUAL
 1:
#ANTERIOR_SALIDA := "DB_SEÑAL".SEÑAL[#i].SALIDA;
IF "DB_SEÑAL".SEÑAL[#i].AUMENTAR THEN
 IF "DB_SEÑAL".SEÑAL[#i].SP_INC > (INT_TO_REAL(#TAM_DIGITAL) -
"DB_SEÑAL".SEÑAL[#i].SALIDA) THEN
     "DB_SEÑAL".SEÑAL[#i].SALIDA := INT_TO_REAL(#TAM_DIGITAL);
 ELSE
     "DB_SEÑAL".SEÑAL[#i].SALIDA := #ANTERIOR_SALIDA +
"DB_SEÑAL".SEÑAL[#i].SP_INC;
 END_IF;
END_IF;
#ANTERIOR_SALIDA := "DB_SEÑAL".SEÑAL[#i].SALIDA;
IF "DB_SEÑAL".SEÑAL[#i].DISMINUIR THEN
 IF "DB_SEÑAL".SEÑAL[#i].SP_INC > "DB_SEÑAL".SEÑAL[#i].SALIDA THEN
     "DB_SEÑAL".SEÑAL[#i].SALIDA := 0;
 ELSE
     "DB_SEÑAL".SEÑAL[#i].SALIDA := #ANTERIOR_SALIDA -
"DB_SEÑAL".SEÑAL[#i].SP_INC;
 END_IF;

END_IF;

                     //SEÑAL SENO
 2:
IF "DB_SEÑAL".SEÑAL[#i].CONTADOR>=6.283185 THEN
  "DB_SEÑAL".SEÑAL[#i].CONTADOR := 0.0;
END_IF;
#"ANTERIOR CONTADOR" := "DB_SEÑAL".SEÑAL[#i].CONTADOR;
"DB_SEÑAL".SEÑAL[#i].SALIDA := "DB_SEÑAL".SEÑAL[#i].SP_INC *
SIN(+"DB_SEÑAL".SEÑAL[#i].CONTADOR);
"DB_SEÑAL".SEÑAL[#i].CONTADOR := #"ANTERIOR CONTADOR" + 0.01;

                    //SEÑAL COSENO
 3:
IF "DB_SEÑAL".SEÑAL[#i].CONTADOR >= 6.283185 THEN
  "DB_SEÑAL".SEÑAL[#i].CONTADOR := 0.0;
END_IF;
#"ANTERIOR CONTADOR" := "DB_SEÑAL".SEÑAL[#i].CONTADOR;
"DB_SEÑAL".SEÑAL[#i].SALIDA := "DB_SEÑAL".SEÑAL[#i].SP_INC *
COS(+"DB_SEÑAL".SEÑAL[#i].CONTADOR);
"DB_SEÑAL".SEÑAL[#i].CONTADOR := #"ANTERIOR CONTADOR" + 0.00001;
```

```
                    //SEÑAL SIERRA
 4:
#ANTERIOR_SALIDA := "DB_SEÑAL".SEÑAL[#i].SALIDA;

IF "DB_SEÑAL".SEÑAL[#i].SP_INC > (INT_TO_REAL(#TAM_DIGITAL) -
"DB_SEÑAL".SEÑAL[#i].SALIDA) THEN
  "DB_SEÑAL".SEÑAL[#i].SALIDA := 0;
ELSE
  "DB_SEÑAL".SEÑAL[#i].SALIDA := #ANTERIOR_SALIDA +
"DB_SEÑAL".SEÑAL[#i].SP_INC;
END_IF;


 ELSE
"DB_SEÑAL".SEÑAL[#i].SALIDA := 0.0;

           END_CASE;



        ELSE
            "DB_SEÑAL".SEÑAL[#i].SALIDA := 0.0;
            "DB_SEÑAL".SEÑAL[#i].CONTADOR := 0.0;

        END_IF;


END_FOR;
```

ANEXO III: CÓDIGO DE SIMULACIÓN DE LA PASTEURIZADORA

```
//CUANDO LA TEMPERATURA A LA SALIDA DE LA PASTEURIZADORA NO ES LA
ADECUADA SE RECIRCULA EL CAUDAL Y SI LA MAQUINA ESTA APAGADA NO TENDREMOS
CAUDAL.
IF #SENSOR_TEMP <#TEMP_PAST OR #PASTO_ON = 0 THEN
    #CAUDAL_INST := 0.0;
ELSE
    #CAUDAL_INST := #SIM_CAUDAL_INST;
END_IF;
//SI LA MEDIDA DEL CAUDAL NO ES CONSIDERADA DE PRODUCTO BUENO, ESE CAUDAL
SE DESECHARÁ
IF #REFRACT <= #LIM_SUP_BRIX AND #REFRACT >= #LIM_INF_BRIX THEN
    #DESECHO := FALSE;
ELSE
    #DESECHO := TRUE;
END_IF;
```

ANEXO IV: CÓDIGO DE LA FUNCIÓN OEE

```
//CÁLCULO DEL TIEMPO DE CICLO DEL PROGRAMA
#TIEMPOCICLOACTUAL                                                    :=
LREAL_TO_REAL(RUNTIME(#TIEMPOCICLOMEMORIA));//Segundos
// #TIEMPOCICLOACTUAL := #TIEMPOCICLOACTUAL * 1000.0;//Milisegundos
//
//TPO = TPO(anterior) + TIEMPOCICLOACTUAL
#TPO := #TPO + #TIEMPOCICLOACTUAL;
//TO  =  TO(anterior)  +  TIEMPOCICLOACTUAL  (CUANDO  LA  MÁQUINA  ESTÁ
FUNCIONANDO)
IF #OP THEN
    #TO := #TO + #TIEMPOCICLOACTUAL;
END_IF;

////DISPONIBILIDAD
//DISP_ACUM = TO / TPO * 100
#DISP_ACUM := #TO / #TPO * 100;

////RENDIMIENTO
//REND_INST = CAUDAL_INST / CAUDAL_NOMINAL * 100
#REND_INST :=#CAUDAL_INST /#CAUDAL_NOMINAL * 100;
//LITROS_ACUM = LITROS_ACUM(anterior) + (CAUDAL_INST(l/h) / 3600(conv:h-
>s) * TIEMPOCICLOACTUAL(s))
#LITROS_ACUM := #LITROS_ACUM + #CAUDAL_INST * #TIEMPOCICLOACTUAL / 3600.0;
//REND_ACUM  =  LITROS  PRODUCIDOS  /  LITROS  TOTALES  QUE  PODIAMOS  HABER
PRODUCIDO
#REND_ACUM := #LITROS_ACUM / (#CAUDAL_NOMINAL * #TO / 3600.0) * 100.0;

////CALIDAD
//LITROS DE PRODUCTO ACUMULADO
//LITROS_TANQUE   =   LITROS_TANQUE(anterior)   +   (CAUDAL_INST(l/h)   /
3600(conv:h->s) * TIEMPOCICLOACTUAL(s))
IF #DESECHO = FALSE THEN
    #LITROS_TANQUE := #LITROS_TANQUE + #CAUDAL_INST * #TIEMPOCICLOACTUAL
/ 3600.0;
END_IF;
//CALIDAD_ACUM = LITROS PRODUCTO BUENO / LITROS TOTALES
#CALIDAD_ACUM := #LITROS_TANQUE / #LITROS_ACUM * 100.0;

////OEE
#OEE := #DISP_ACUM * #REND_ACUM * #CALIDAD_ACUM / 10000.0;
```

```
////INTRODUCCI?"N DE HORARIO
FOR #I := 0 TO 2 DO
    #TURNO[#I].HORA_INICIO.HOUR := #HORARIO_TURNO[#I].HORA_INICIO.HOUR;
    #TURNO[#I].HORA_INICIO.MINUTE                                    :=
#HORARIO_TURNO[#I].HORA_INICIO.MINUTE;
    #TURNO[#I].HORA_INICIO.SECOND                                    :=
#HORARIO_TURNO[#I].HORA_INICIO.SECOND;
    #TURNO[#I].HORA_FIN.HOUR := #HORARIO_TURNO[#I].HORA_FIN.HOUR;
    #TURNO[#I].HORA_FIN.MINUTE := #HORARIO_TURNO[#I].HORA_FIN.MINUTE;
    #TURNO[#I].HORA_FIN.SECOND := #HORARIO_TURNO[#I].HORA_FIN.SECOND;
END_FOR;

IF #TURNO[0].HORA_FIN > #TURNO[1].HORA_INICIO THEN
    #TURNO[1].HORA_INICIO := #TURNO[0].HORA_FIN;
END_IF;

IF #TURNO[1].HORA_FIN > #TURNO[2].HORA_INICIO THEN
    #TURNO[2].HORA_INICIO := #TURNO[1].HORA_FIN;
END_IF;

IF #TURNO[2].HORA_FIN.HOUR > #TURNO[0].HORA_INICIO.HOUR THEN
    #TURNO[0].HORA_INICIO.HOUR := #TURNO[2].HORA_FIN.HOUR;
END_IF;

////INFORMACI?"N DE TURNO
IF #FECHAHORA_ACTUAL >= #TURNO[#N_TURNO].HORA_FIN THEN
    #TURNO[#N_TURNO].DISP := #DISP_ACUM;
    #TURNO[#N_TURNO].REND := #REND_ACUM;
    #TURNO[#N_TURNO].CALIDAD := #CALIDAD_ACUM;
    #TURNO[#N_TURNO].OEE := #OEE;

    #N_TURNO := #N_TURNO + 1;
END_IF;

IF #N_TURNO = 3 THEN
    #N_TURNO := 0;
    FOR #Y := 0 TO 2 DO
        #TURNO[#Y].HORA_INICIO  :=  (T_ADD(IN1  :=  #TURNO[#Y].HORA_INICIO,
IN2 := T#1d));
        #TURNO[#Y].HORA_FIN := (T_ADD(IN1 := #TURNO[#Y].HORA_FIN, IN2 :=
T#1d));
    END_FOR;

END_IF;

////REINICIO DE VARIABLES CON EL FIN DE TURNO
IF #FECHAHORA_ACTUAL >= #TURNO[#N_TURNO].HORA_INICIO AND #FECHAHORA_ACTUAL
<= (T_ADD(IN1 := #TURNO[#N_TURNO].HORA_INICIO, IN2 := T#2s)) THEN
    #TPO := 0.0;
    #TO := 0.0;
    #LITROS_ACUM := 0.0;
    #LITROS_TANQUE := 0.0;
END_IF;
```