COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER’S DEGREE IN INDUSTRIAL ENGINEERING

MASTER’S THESIS

INTEROPERABILITY OF APPLICATIONS IN THE SMART GRIDS CONTEXT

Author: María Lerena Rubio

Academic Supervisor: Néstor Rodríguez Pérez

Industrial Advisor: Eduardo Jiménez Segado

Madrid

August, 2023

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

"INTEROPERABILIDAD DE APLICACIONES EN EL CONTEXTO DE SMART GRIDS"

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico **2022/2023** 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.: MARÍA LERENA RUBIO Fecha: 25/08/2023

[illegible]

Autorizada la entrega del proyecto

EL DIRECTOR ACADÉMICO

[illegible]

Fdo.: NÉSTOR RODRÍGUEZ PÉREZ Fecha: 28/08/2023

EL DIRECTOR EN LA ENTIDAD COLABORADORA

Eduardo Jiménez Segado

Fdo.: EDUARDO JIMÉNEZ SEGADO Fecha: 28/08/2023

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER’S DEGREE IN INDUSTRIAL ENGINEERING

MASTER’S THESIS

INTEROPERABILITY OF APPLICATIONS IN THE SMART GRIDS CONTEXT

Author: María Lerena Rubio

Academic Supervisor: Néstor Rodríguez Pérez

Industrial Advisor: Eduardo Jiménez Segado

Madrid

August, 2023

INTEROPERABILIDAD DE APLICACIONES EN EL CONTEXTO DE SMART GRIDS

Autor: Lerena Rubio, María
Director académico: Rodríguez Pérez, Néstor
Director en la entidad colaboradora: Jiménez Segado, Eduardo
Entidad Colaboradora: Minsait

RESUMEN DEL PROYECTO

1. INTRODUCCIÓN

Las Smart Grids (SGs) son sistemas modernos de distribución de electricidad, vitales para la transición del mercado energético hacia la sostenibilidad y la eficiencia. Las características clave de una SG incluyen la optimización del uso de activos, integración de la generación distribuida y el almacenamiento, garantía de la calidad de la energía, anticipación y respuesta a las perturbaciones, protección contra ataques físicos y cibernéticos y habilitación de la participación del consumidor.

Un diseño efectivo de la arquitectura de red es crucial para administrar la gran cantidad de dispositivos conectados. En las SGs se favorece el flujo bidireccional de energía gracias a tecnologías como los sistemas de control. Al recopilar y transmitir datos de los consumidores, las SGs se asemejan a las redes de telecomunicaciones, beneficiando tanto a los consumidores como a los operadores. Esta infraestructura de energía y comunicaciones se muestra en la ***Figura 1***. Los dispositivos IoT en SG optimizan la gestión, detectan fallos y permiten la interacción usuario-red, maximizando la eficiencia energética. Para evitar la sobrecarga de datos, se propone la aplicación de procesamiento descentralizado a través de Edge Computing.

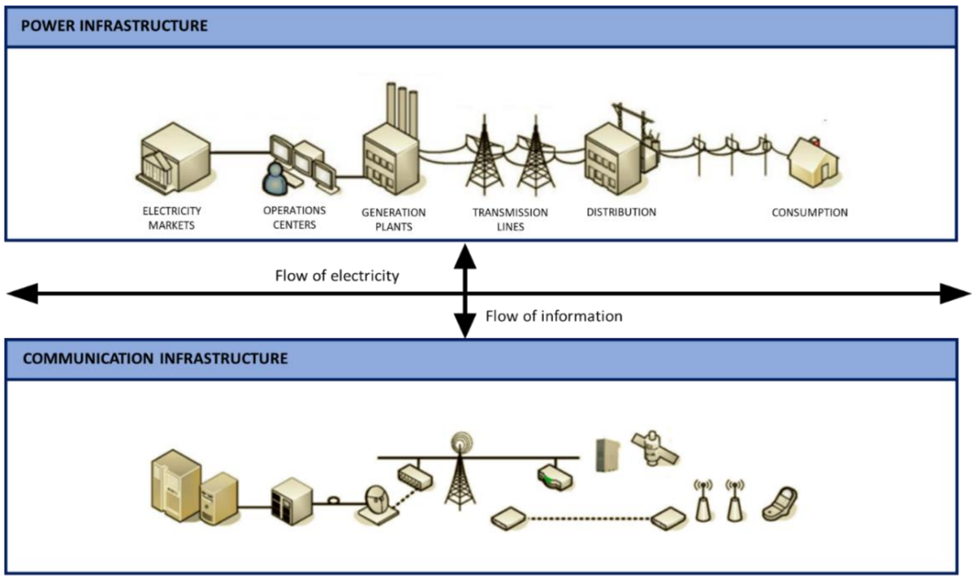


Figura 1. Smart Grid

En la ***Figura 2*** se propone una nueva visión del intercambio de información en los CTs de una SG. El modelo piramidal ilustra el hardware, los Nodos Edge y la plataforma IoT centralizada para el intercambio y control de datos.

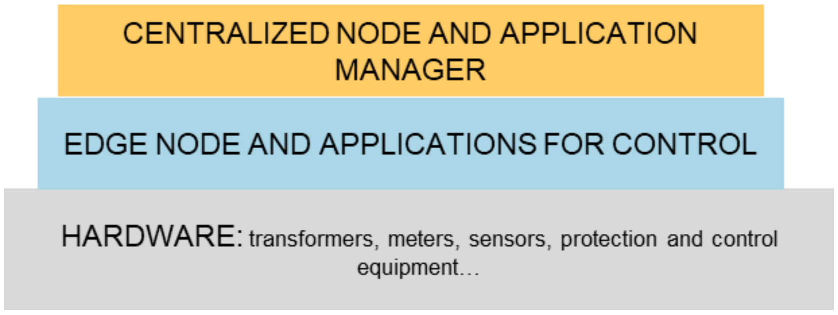

CENTRALIZED NODE AND APPLICATION MANAGER
EDGE NODE AND APPLICATIONS FOR CONTROL
HARDWARE: transformers, meters, sensors, protection and control equipment…

Figura 2. Nueva visión de intercambio de información en CTs de una SG

El desafío clave de las SGs radica en la integración de componentes, sistemas, información y aplicaciones. Las interfaces y funcionalidades deben garantizar la interoperabilidad para permitir procesos de alto nivel. La conexión de todos los componentes de una red eléctrica da lugar a una red interconectada en la que el flujo de información y su análisis se llevarán a cabo en tiempo real.

La Web de las Cosas (WoT) estandariza protocolos y APIs, promoviendo la interacción de dispositivos y el intercambio de datos. Los Thing Descriptors (TD) definen las capacidades, propiedades, e interacciones del dispositivo, fomentando la interoperabilidad y la integración dentro de las aplicaciones.

Los objetivos principales de este proyecto son investigar el papel de las plataformas de gestión de aplicaciones y los nodos Edge en la visión de las SGs, explorar el papel de la virtualización de aplicaciones en la búsqueda de la interoperabilidad, estudiar los estándares del WoT para lograr la interoperabilidad de aplicaciones y crear una especificación de TD para la interoperabilidad entre aplicaciones en el dominio de un CT.

2. ANÁLISIS

2.1. IoT en Smart Grids

Las redes de distribución actuales no fueron diseñadas para gestionar flujos de energía bidireccionales. A medida que la demanda de energía crece y la generación distribuida aumenta, los límites técnicos en las líneas de suministro pueden ser excedidos, surgiendo problemas de calidad de energía. Esto causa un desequilibrio entre la generación y la demanda a nivel local, con respuestas más lentas y pérdidas de calidad.

Ante estos problemas, es necesario encontrar medidas que permitan actuar de manera más rápida ante situaciones de desequilibrio energético. Un desafío adicional es el monitoreo de las redes de distribución. Tradicionalmente, se realiza en redes de AT y MT, dejando una falta de información en BT, donde la mayoría de los consumidores están conectados.

Una alternativa para abordar esto es conectar los transformadores de la red de distribución al centro de control, lo cual implica altos costos de inversión y carga de trabajo. Otra opción es la implementación del Internet Industrial de las Cosas (IIoT), que instala sensores y pasarelas para recopilar y procesar datos localmente, monitoreando la red de manera más profunda.

La adopción del IIoT representa un cambio significativo para las empresas, que tradicionalmente utilizaban sistemas OT, como sistemas SCADA, aislados de otros sistemas y de internet. En el proceso de adoptar el IoT en las SGs, se puede comenzar monitoreando la red de baja tensión, capturando datos de dispositivos físicos y procesando información localmente para generar alertas ante desviaciones de parámetros. Esto reduce la carga de datos transmitida a través de pasarelas, disminuyendo costos de comunicación inalámbrica.

La adopción del IoT en las SGs presenta oportunidades revolucionarias tal y como se muestra en la ***Figura 3***. pero también conlleva retos de seguridad, privacidad e interoperabilidad que deben ser abordados cuidadosamente.

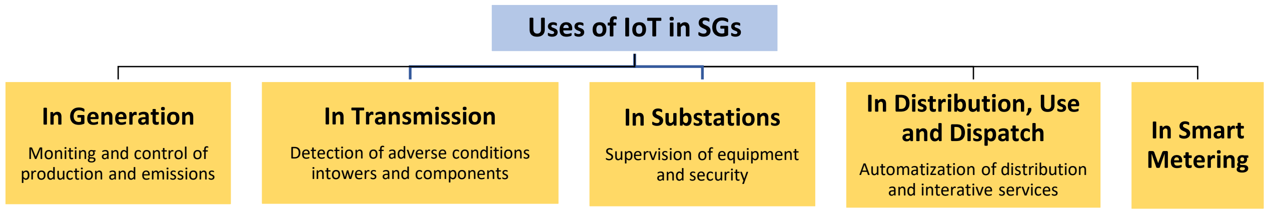

Uses of IoT in SGs
In Generation
Moniting and control of production and emissions
In Transmission
Detection of adverse conditions intowers and components
In Substations
Supervision of equipment and security
In Distribution, Use and Dispatch
Automatization of distribution and interative services
In Smart Metering

Figura 3. Usos del IoT en SGs

2.1.1. Interoperabilidad

Poniendo el enfoque en la interoperabilidad, existen diferentes categorías que describen los diferentes niveles de interoperabilidad que se pueden considerar en el contexto de la comunicación entre sistemas y plataformas en tecnologías de IoT.

En este trabajo y dentro del estudio de la interoperabilidad, nos enfocaremos en las capas de interoperabilidad de red, sintáctica y semántica, para estudiar su aplicación en el contexto de que cualquier aplicación desarrollada pueda funcionar en cualquier plataforma, independientemente del proveedor.

- **Interoperabilidad de red**. Se encarga de transportar información entre diferentes dispositivos que interactúan a través de diferentes redes de comunicación.
- **Interoperabilidad sintáctica**. Es el acuerdo sobre reglas de formato y estructura utilizadas al codificar la información que se intercambia entre dispositivos.
- **Interoperabilidad semántica**. Se encarga de asegurar que los datos transmitidos sean interpretados de manera común entre los diferentes sistemas y aplicaciones. Todos los sistemas involucrados en la transmisión de información, tanto emisores como receptores de datos, deben estar bajo un modelo de información estandarizado que se utilizará como referencia. La forma más común de lograr que los sistemas entiendan el mismo lenguaje es a través de ontologías de

datos, una especie de mapas que explican claramente lo que significa cada cosa, ayudando a que los dispositivos y sistemas comprendan y compartan información correctamente.

A través de las ontologías, la información que explica los datos compartidos por los dispositivos IoT se puede transformar en una secuencia estructurada, lo que ayuda a dar sentido a la información que proviene de los dispositivos y a entenderla correctamente, ya que se establece un contexto claro para su interpretación posterior. Las ontologías también pueden funcionar como un canal para enviar diferentes tipos de instrucciones a los dispositivos. Para ello, proporcionan pautas para garantizar que los comandos se ejecuten correctamente, incluyendo la seguridad y el acceso a la información, así como las operaciones de los dispositivos.

2.1.2. Protocolos de telecomunicación

En el contexto de IoT, es fundamental explorar en profundidad los protocolos de comunicación que permiten una integración eficiente entre diferentes dispositivos y sistemas. En particular, conviene profundizar en el protocolo MQTT en el contexto del IoT, especialmente para este trabajo, debido a lo siguiente:

- Está diseñado para aplicaciones de IoT con baja latencia y ancho de banda limitado, esencial en SGs donde la comunicación rápida y eficiente entre dispositivos IoT y la infraestructura de la red eléctrica es crucial.
- Es ideal en escenarios con conexiones inestables y bajo consumo de energía, asegurando comunicación confiable incluso en condiciones desafiantes, relevantes en SGs con dispositivos en áreas remotas o condiciones ambientales adversas.
- Su topología de publicación y suscripción beneficia a las SGs donde varios dispositivos necesitan monitorear y recibir datos de múltiples fuentes, permitiendo la distribución eficiente de mensajes a varios suscriptores.
- Es adecuado para aplicaciones de monitoreo en tiempo real y control remoto, aspectos importantes en SGs donde la transmisión rápida de datos de sensores y dispositivos IoT permite una gestión efectiva de la red en tiempo real.
- En cuanto a seguridad y privacidad, MQTT simplifica la encriptación de mensajes mediante TLS y la autenticación de clientes usando protocolos modernos como OAuth.

El funcionamiento de MQTT es simple y sigue un patrón de publicación y suscripción, mostrado en la ***Figura 4***.

- El editor crea el mensaje y lo publica en un tema específico.
- El suscriptor recibe mensajes relevantes al tema al que se ha suscrito.
- Un broker comunica con los clientes a través de una red local o conexión a internet.

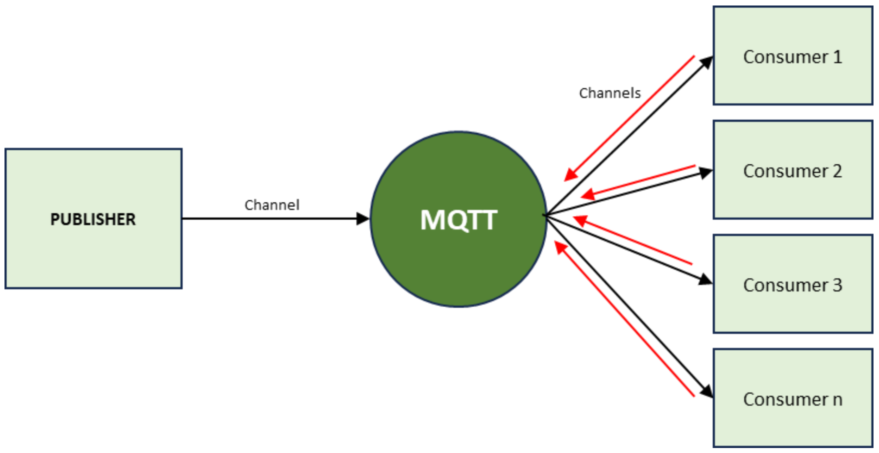


Figura 4. Estructura del MQTT

2.2. Importancia del Edge en SGs

Acompañando al IoT, es común el término Edge. Aplicado a la SG, un nodo edge se ubica en puntos estratégicos donde se genera, transmite o consume datos en las diferentes etapas de generación, transmisión y distribución de energía eléctrica. La función principal de un nodo edge es procesar, analizar y tomar decisiones sobre los datos en tiempo real antes de transmitirlos a un lugar centralizado, como un centro de datos o la nube.

De esta forma, concentradores de datos de medición y supervisores básicos y avanzados en BT permiten la recolección y procesamiento local de datos de consumo de energía y monitoreo del suministro en la red de BT. Esto facilita detectar y resolver problemas en tiempo real sin depender de la conectividad y latencia asociadas con enviar datos a un lugar centralizado.

2.3. Virtualización de aplicaciones

La virtualización de aplicaciones permite ejecutarlas en entornos aislados y virtualizados, separados de la infraestructura física y otras aplicaciones en el sistema. Se crean contenedores o instancias virtuales que encapsulan las aplicaciones y sus dependencias, ofreciendo un entorno consistente y aislado para su ejecución.

En IoT y SGs, la virtualización de aplicaciones tiene beneficios como eficiencia de recursos, gestión simplificada, mínima interrupción y aprovisionamiento rápido. Los hipervisores y contenedores son técnicas de virtualización destacadas. Los contenedores, especialmente Docker, destacan por flexibilidad, portabilidad y rendimiento ágil. En la ***Figura 5*** se muestra una estructura de virtualización basada en contenedores.

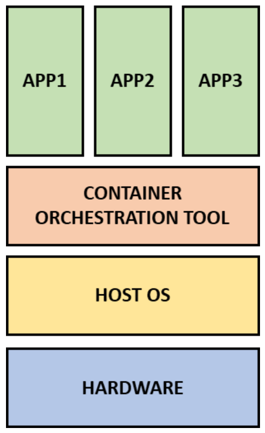


Figura 5. Virtualización basada en contenedores

Docker es elegido por su facilidad, portabilidad y escalabilidad. En Edge Nodes de IoT, Docker brinda ventajas como portabilidad de aplicaciones y rendimiento rápido y ligero en comparación con máquinas virtuales (VMs). Docker permite asignar y limitar recursos de CPU, memoria, red y disco, asegurando una distribución equitativa. Su arquitectura se basa en contenedores con características como "cgroups" y "namespaces", permitiendo administrar, limitar recursos del sistema, y aislar procesos. Estas tecnologías previenen interferencias y aseguran un entorno aislado para cada proceso.

2.4. Gestión centralizada de nodos y aplicaciones

Las plataformas de IoT facilitan el desarrollo y despliegue de sistemas de IoT, permitiendo que los usuarios se centren en lo más importante para ellos, la operación de sus negocios. Con el aumento en el número y dispersión geográfica de despliegues de dispositivos IoT, se hace evidente la necesidad de un sistema de gestión centralizada que proporcione información básica sobre el estado de los dispositivos desplegados y permita actualizaciones de software de forma remota. Las aplicaciones que se ejecutan en los dispositivos deben ser desplegadas de manera remota y segura. La robustez operativa es esencial en cualquier plataforma de IoT, así como la ciberseguridad, especialmente en el ámbito industrial.

Al seleccionar una plataforma de IoT, es útil distinguir entre plataformas genéricas y productos más específicos. Las primeras, como Azure IoT Hub, AWS IoT Core, Google Cloud IoT y Oracle IoT Cloud, ofrecen un amplio ecosistema de herramientas, pero a menudo tienen una curva de aprendizaje pronunciada y no se adaptan a necesidades específicas. Por otro lado, las plataformas específicas, como Onesite Phygital Edge de Minsait, ofrecen capacidades más limitadas pero mejor soporte al usuario y tiempos de desarrollo más cortos.

Los nodos IoT actúan como intermediarios entre dispositivos de captura de datos y sistemas centralizados de procesamiento. Estos nodos se centran en recopilar y transmitir eficientemente los datos a sistemas centrales o la nube para un análisis más profundo. La implementación de los nodos de IoT implica la instalación física y la configuración con la lógica y algoritmos necesarios para su propósito. Estos nodos son donde se ejecutan el sistema operativo y las aplicaciones, ya sea nativamente o en contenedores Docker.

2.4.1. Plataforma Onesite

La Plataforma Onesite de Minsait es amplia y ofrece módulos que permiten a las organizaciones desarrollar soluciones personalizadas, desde gestión y análisis de datos hasta creación de aplicaciones y soluciones de IoT. La plataforma ofrece capacidades como gestión de activos y datos, análisis avanzado, visualización, y creación de aplicaciones.

2.4.2. Solución Phygital Edge

Onesite Phygital Edge es un componente de la Plataforma Onesite que se enfoca en la administración de dispositivos y sistemas de IoT en el borde de la red. Esta plataforma se basa en un sistema de procesamiento cercano a los dispositivos de la red. Una arquitectura de virtualización y contenedor se implementa en estos nodos para distribuir eficientemente inteligencia en forma de microservicios.

La arquitectura de la plataforma Phygital Edge se basa en tres componentes clave, que trabajan juntos para permitir la gestión completa de dispositivos en el campo y ofrecen funcionalidades a través de un agente especializado:

- **Dispositivos de campo**. Estos dispositivos están conectados a la plataforma para permitir su gestión de manera remota.
- **Sistema de Gestión de Aplicaciones y Nodos Edge**. Este centro de gestión global distribuido tiene como función principal comisionar, acceder, configurar, actualizar y gestionar dispositivos de borde. Este componente se muestra en la ***Figura 6***.

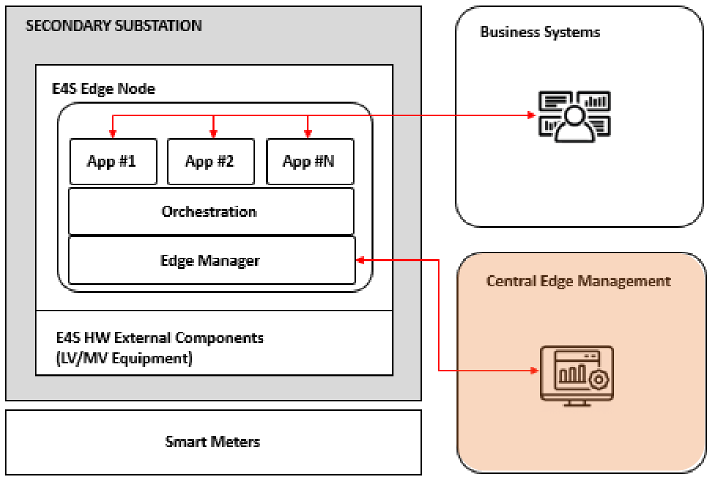


Figura 6. Sistema de Gestión de Aplicaciones y Nodos Edge

La figura del agente es importante, ya que facilita la interacción entre el administrador central y el nodo edge, garantizando las funcionalidades del módulo. La gestión del agente implica controlar y administrar los agentes de software instalados en el nodo edge, que pueden ejecutar comandos, recopilar datos y transmitirlos. Además, el Sistema de Gestión de Aplicaciones y Nodos Edge, debe permitir la instalación, desinstalación, actualización y degradación remotas de agentes en el nodo de borde, y monitorear su estado y rendimiento. También debe proporcionar una API para la gestión de agentes, utilizada por la interfaz gráfica y la línea de comandos.

- **Motor de Borde**: Este conjunto de capacidades desplegadas en nodos a través de contenedores permiten el control remoto de nodos en campo, la adquisición y procesamiento de información local, así como la conexión a la nube empresarial para escalar y realizar análisis más profundos.

En cuanto a la seguridad, la plataforma Phygital Edge emplea un registro de identidad que almacena información sobre dispositivos y módulos autorizados para conectarse. Los dispositivos se autentican en el Sistema de Gestión mediante credenciales almacenadas en el registro, garantizando conexiones seguras y confiables.

2.4.3. Aplicación de esta plataforma al CT

Se han identificado diversas aplicaciones de Phygital Edge en el CT, y se han resumido los casos de uso y requisitos para hacerlos factibles en la ***Tabla 1***.

Tabla 1. Casos del uso de Phygital Edge en un CT

Caso de uso	Descripción	Requisitos de E/S
Concentrador de datos para medidores inteligentes	Esta aplicación recopila datos de contadores inteligentes utilizando la tecnología PLC PRIME. A continuación, envía esos datos a través de las conexiones WAN disponibles en el Centro de datos	Requiere un Nodo Base PRIME conectado a las tres fases y el neutro de la red BT en las subestaciones secundarias
Supervisión avanzada de BT	Esta aplicación monitorea los paneles LV en el CT. Se comunica a través de DLMS con tarjetas de E/S LV	Requiere la instalación de tarjetas de E/S en cada fase del panel LV, para medir valores instantáneos (V, I, P y Q) de cada fase
Identificación de topología de red	Esta aplicación calcula la topología de BT, identificando la línea y la fase para cada medidor inteligente conectado al panel de baja tensión	No requiere E/S adicionales, ya que utiliza datos de aplicaciones anteriores
Identificación de interrupciones	Esta aplicación ha identificado interrupciones en la red de BT, estimando el tamaño de la interrupción y su impacto en los clientes	No requiere E/S adicionales, ya que utiliza datos de aplicaciones anteriores
Monitoreo y Control de la Regulación de Transformadores	Esta aplicación monitoriza los parámetros de BT del transformador y regula la tensión de salida para adaptar la calidad de la energía al estado de la red. Proporciona una interfaz con el controlador del transformador para modificar la salida del transformador	Requiere un módulo de E/S que pueda operar el controlador del transformador
Análisis de vídeo para monitorización visual	Esta aplicación puede cubrir varios casos de uso basados en la señal de video proveniente de una o varias cámaras digitales que capturan diferentes áreas de la subestación	Requiere una o más cámaras de vídeo
Balanceo de carga y perfil de carga de la subestación	Esta aplicación calcula estadísticas de carga de subestaciones secundarias por fase, circuito y transformador para sugerir una mejor distribución al cliente	No requiere E/S adicionales, ya que utiliza datos de otras aplicaciones
Análisis de Predicción de Generación y Demanda	Esta aplicación calcula las predicciones de demanda y generación basadas en datos históricos de contadores inteligentes y paneles de baja tensión	No requiere E/S adicionales, ya que utiliza datos de otras aplicaciones
Gestión de Recursos Energéticos Distribuidos	Esta aplicación opera recursos energéticos distribuidos para responder a situaciones específicas de la red.	Requiere una E/S para comunicarse con los DER

El sistema de orquestación de contenedores, componente esencial de la plataforma, permite la gestión eficiente de aplicaciones y recursos en entornos distribuidos complejos.

El modelo de comunicación diseñado busca tres tipos de intercambio de información entre los componentes de software desplegados en la subestación.

• Tipo de Mensaje 1 - Bajo demanda. La aplicación envía MQTT para solicitar información.
• Tipo de Mensaje 2 – Programado. La aplicación usa MQTT para programar tarea o pedir información, con detalles y URI para acceder luego.
• Tipo de Mensaje 3 – Espontáneo. La aplicación publica datos en tema MQTT, y otras se suscriben para recibir estos datos.

Estos flujos de comunicación deben ser compatibles y se logran utilizando el protocolo MQTT, que se combina con la arquitectura WoT para lograr interoperabilidad y comunicación eficiente entre aplicaciones.

El WoT permite que las aplicaciones desplegadas en diferentes nodos se comuniquen de manera fluida y eficiente. Los TDs definen los dispositivos IoT presentes en el CT, facilitando la integración de nuevos dispositivos y ofreciendo una vista unificada de los activos de la subestación.

La arquitectura WoT-A específica para el CT trata a las aplicaciones como "Cosas", cada una con un conjunto completo de funcionalidades que incluyen lectura de información, actualización de estado, suscripciones a notificaciones y funciones. El WoT Directory es una parte fundamental de esta arquitectura, donde las aplicaciones cuelgan sus tarjetas de identificación. Las comunicaciones entre aplicaciones ocurren directamente sin intermediarios, y un protocolo de comunicación eficiente, como JSON-MQTT, se utiliza para asegurar una interoperabilidad mejorada.

3. DESARROLLO DE LA ESPECIFICACIÓN

En este trabajo, se ha desplegado una aplicación junto a su correspondiente TD, basada en el estándar WoT. El TD programado define las capacidades, propiedades, interacciones y otra información relevante sobre la Cosa, que en este caso representa un elemento de hardware, permitiendo interoperabilidad e integración sin problemas con aplicaciones WoT. Este TD facilita la integración e interacción con dispositivos o servicios IoT por otras aplicaciones o plataformas.

El objetivo final de la aplicación es recibir señales de dispositivos IoT desplegados en un entorno de laboratorio de Minsait, llenar un TD de un Sensor de Temperatura de Transformador con la información de esos mensajes provenientes de este tipo de dispositivo. Luego, otra aplicación lee ese TD y crea alertas cuando la temperatura excede un valor previamente determinado.

El dispositivo hardware seleccionado para llenar el TD propuesto y cuyas mediciones se monitorean posteriormente es un sensor de temperatura inalámbrico. Estos sensores compactos pueden comunicarse sin cables físicos y utilizan protocolos como Bluetooth, Wi-Fi y Zigbee, lo que facilita la transmisión rápida y confiable de datos de temperatura. Además, la tecnología inalámbrica ofrece flexibilidad y escalabilidad, ya que los sensores se pueden desplegar y mover fácilmente, adaptándose a diferentes escenarios de monitoreo. En el contexto de las SGs, estos sensores son muy útiles para monitorear variaciones de temperatura en equipos eléctricos, líneas de transmisión, subestaciones y otros componentes críticos de la red. Este monitoreo continuo permite a los operadores detectar fallos potenciales o problemas en la red causados por el sobrecalentamiento, tomando medidas preventivas para mejorar la confiabilidad y seguridad de la red.

La parte práctica de este trabajo establece varios objetivos fundamentales, incluyendo la comunicación eficiente entre dispositivos y sistemas utilizando el protocolo MQTT, la gestión de datos generados por dispositivos IoT, la extracción de información relevante de los datos recibidos, y la interoperabilidad para facilitar la integración y compatibilidad entre dispositivos y aplicaciones en la Red Inteligente.

Las fortalezas incluyen el uso del protocolo MQTT, ampliamente adoptado y compatible con varios dispositivos, el formato JSON para el intercambio de datos, y los TD personalizados para estructurar y estandarizar información sobre dispositivos y capacidades. Además, en el contexto de las SGs, estas fortalezas permiten una mayor eficiencia energética, gestión de cargas eléctricas, monitoreo en tiempo real y la integración de fuentes de energía renovable.

En la ***Tabla 2*** se presentan las principales herramientas empleadas para el desarrollo de la especificación, junto con su aplicación en el campo de las SGs.

Tabla 2. Herramientas y su aplicación en SGs

Herramienta	Descripción	Aplicación en SGs
Python	Lenguaje de programación versátil y fácil de usar	Facilita el desarrollo de aplicaciones IoT en SGs
	Amplia comunidad y bibliotecas para tareas específicas	Le permite procesar datos y comunicarse con dispositivos en el contexto de IoT
	Multiplataforma, compatible con varios sistemas operativos	Asegura que la aplicación funcione en diferentes entornos
	Flexibilidad para desarrollar una variedad de aplicaciones	Adaptable a diferentes requisitos en el contexto de las SGs
	json: Facilita el procesamiento de datos en formato JSON	Estructurar los datos de TD y temperatura de forma organizada
	paho.mqtt.client: Implementa la comunicación MQTT	Establece una comunicación eficiente entre el termostato y el sistema central
	datetime: proporciona herramientas para trabajar con fechas y horas	Registra la fecha y hora de los datos de temperatura para su análisis y seguimiento
Docker	Plataforma de contenedores que facilita la creación, implementación y administración	Proporciona un entorno aislado y consistente para el desarrollo y la ejecución
	Portabilidad y consistencia en diferentes entornos	Asegura que la aplicación funcione de manera consistente en varias plataformas
	Facilita el trabajo en equipo y evita conflictos de configuración	Agiliza el desarrollo y despliegue de la aplicación en diferentes etapas
	Aislamiento y seguridad en la ejecución de la aplicación	Mejora la seguridad al ejecutar la aplicación en contenedores aislados
MQTT Explorer	Herramienta de visualización y depuración MQTT de código abierto	Verifica la comunicación MQTT entre el termostato y el sistema central
	Proporciona supervisión y depuración de mensajes MQTT	Facilita la identificación y resolución de problemas de comunicación
	Interfaz gráfica intuitiva para representar la comunicación MQTT	Permite visualizar en tiempo real los mensajes y temas enviados y recibidos

3.1. Modelo

En el contexto de un CTi, se utiliza un sensor de temperatura inalámbrico para capturar la temperatura exterior del transformador. Este dispositivo se comunica a través de Zigbee con la Pasarela Zigbee.

La Pasarela Zigbee es un dispositivo que actúa como intermediario entre los dispositivos Zigbee y otros sistemas de comunicación. Básicamente, permite que los dispositivos Zigbee se comuniquen con sistemas y aplicaciones que utilizan otros protocolos de comunicación, como MQTT. La información recopilada de los dispositivos Zigbee, como sensores o actuadores, puede transmitirse a través de MQTT al sistema central.

Una vez que las aplicaciones y dispositivos IoT han realizado sus cálculos o llevado a cabo sus acciones, es crucial que los resultados y datos generados se compartan y comuniquen de manera efectiva. Para lograr esto, se utiliza una Plataforma de gestión de nodos y aplicaciones, que actúa como centro de control y coordinación para la infraestructura IoT. Sin embargo, esta parte de la comunicación con la plataforma no está incluida en el desarrollo práctico de este trabajo.

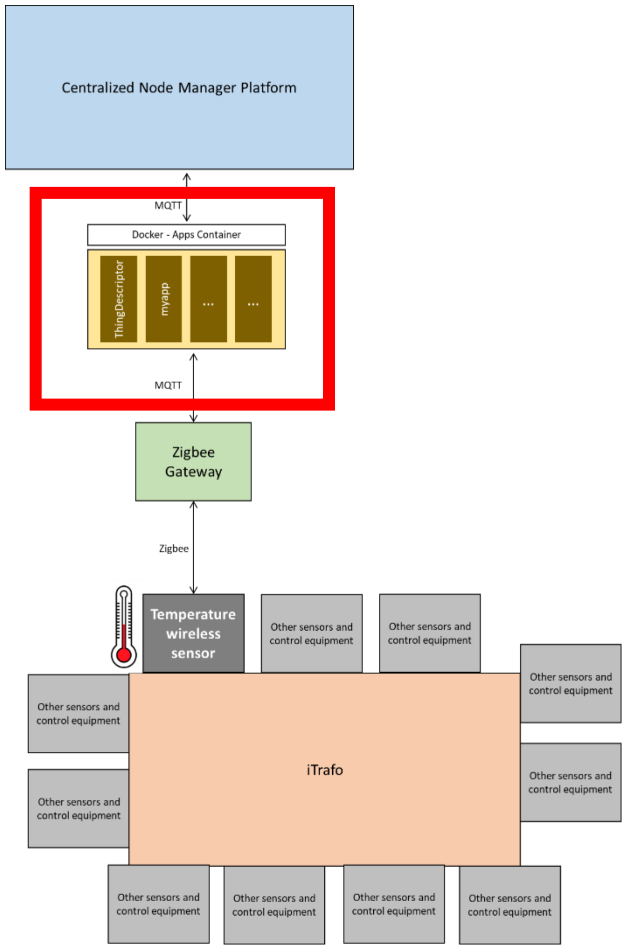
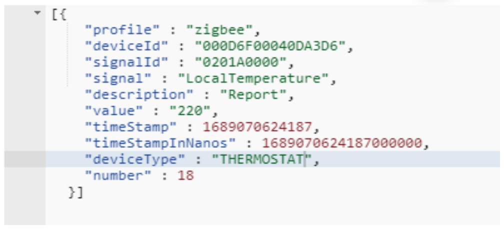

Centralized Node Manager Platform
MQTT
Docker - Apps Container
ThingDescriptor
myapp
...
...
MQTT
Zigbee Gateway
Zigbee
Temperature wireless sensor
Other sensors and control equipment
iTrafo

Figura 7. Modelo de la parte práctica del trabajo

3.2. Resultados

Prueba #1: Se lee un mensaje publicado por MQTT proveniente de un dispositivo. El deviceType es un sensor de temperatura del CT, y cuya temperatura no supera el valor límite de 20ºC, como se muestra en la ***Figura 8***.

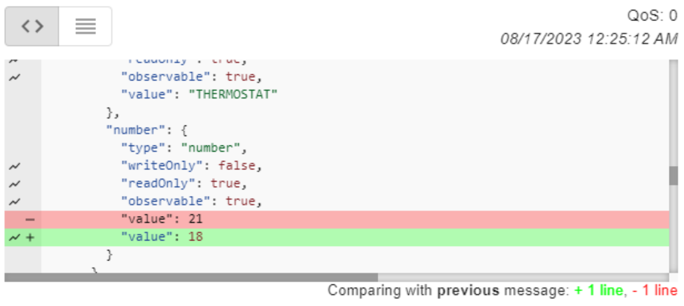
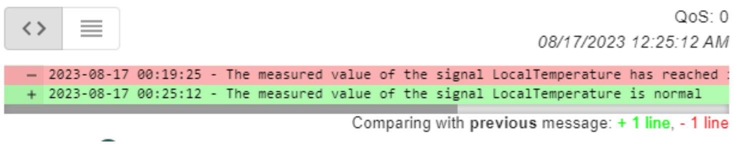


```
[{
    "profile" : "zigbee",
    "deviceId" : "000D6F00040DA3D6",
    "signalId" : "0201A0000",
    "signal" : "LocalTemperature",
    "description" : "Report",
    "value" : "220",
    "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
    "deviceType" : "THERMOSTAT",
    "number" : 18
}]
```

Figura 8. Prueba #1: Mensaje leído proveniente de un termostato

A partir de él, se ha publicado un mensaje que contiene el TD lleno con la información leída. En otro tema MQTT se ha recibido el mensaje correspondiente para la medición de temperatura dentro de los límites. Estos se muestran en la ***Tabla 3***.

Tabla 3. Prueba #1: Mensajes publicados (a la izquierda el TD, a la derecha el mensaje de límite de Tª)

Tema 1	Tema 2
	

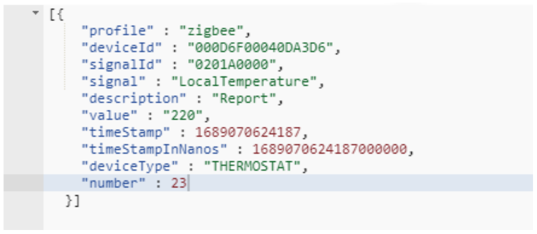
Tema 1:

```
QoS: 0
08/17/2023 12:25:12 AM
          "readOnly": true,
          "observable": true,
          "value": "THERMOSTAT"
        },
        "number": {
          "type": "number",
          "writeOnly": false,
          "readOnly": true,
          "observable": true,
 -        "value": 21
 +        "value": 18
        }
Comparing with previous message: + 1 line, - 1 line
```

Tema 2:

```
QoS: 0
08/17/2023 12:25:12 AM
- 2023-08-17 00:19:25 - The measured value of the signal LocalTemperature has reached
+ 2023-08-17 00:25:12 - The measured value of the signal LocalTemperature is normal
Comparing with previous message: + 1 line, - 1 line
```

Prueba #2: Ahora se lee un mensaje publicado por MQTT proveniente del mismo tipo de dispositivo, pero superando el límite de 20ºC, como se muestra en la ***Figura 9.***

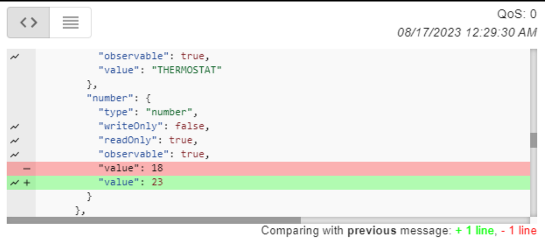
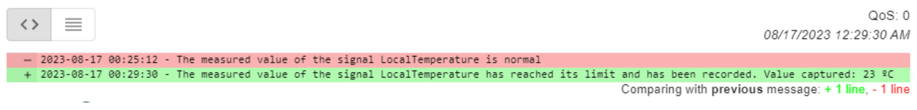


```
[{
    "profile" : "zigbee",
    "deviceId" : "000D6F00040DA3D6",
    "signalId" : "0201A0000",
    "signal" : "LocalTemperature",
    "description" : "Report",
    "value" : "220",
    "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
    "deviceType" : "THERMOSTAT",
    "number" : 23
}]
```

Figura 9. Prueba #2: Mensaje leído proveniente de un termostato

De nuevo, se publican los mensajes correspondientes con la información leída, tal y como se muestra en la *Tabla 4*.

Tabla 4. Prueba #2: Mensajes publicados (a la izquierda el TD, a la derecha el mensaje de límite de Tª)

Tema 1	Tema 2
	

Tema 1:

QoS: 0
08/17/2023 12:29:30 AM

```
        "observable": true,
        "value": "THERMOSTAT"
      },
      "number": {
        "type": "number",
        "writeOnly": false,
        "readOnly": true,
        "observable": true,
-       "value": 18
+       "value": 23
      }
    },
```

Comparing with **previous** message: + 1 line, - 1 line

Tema 2:

QoS: 0
08/17/2023 12:29:30 AM

```
- 2023-08-17 00:25:12 - The measured value of the signal LocalTemperature is normal
+ 2023-08-17 00:29:30 - The measured value of the signal LocalTemperature has reached its limit and has been recorded. Value captured: 23 ºC
```

Comparing with **previous** message: + 1 line, - 1 line

Prueba #3: Se comprueba la correcta contenerización de las aplicaciones desarrolladas. Se han creado tres contenedores en el sistema y todos ellos se ejecutan correctamente.

```
CONTAINER
ID    IMAGE                                                                 COMMAN
D                   CREATED            STATUS            PORTS
                         NAMES
dd2473d82c99   myapp_image:latest
   "python myapp.py"        13 seconds ago   Up 12
seconds                                                      myapp_container
6e72ccb33867   thing_descriptor_app_image:latest
    "python ThingDescrip…"   4 minutes ago    Up 4
minutes                                                       thing_descriptor
_container
fb35bfe64c8a   emslab.onesaitplatform.com/onesait-things/edge-
mqtt:2.0.0   "/docker-entrypoint.…"   2 weeks ago      Up 2
weeks      0.0.0.0:1883->1883/tcp, :::1883->1883/tcp   edge.mqtt
```

La especificación desarrollada marca un hito en la integración de dispositivos IoT dentro del marco de WoT. Con el uso de TDs, se garantiza la interoperabilidad con una amplia gama de dispositivos IoT. Aunque inicialmente se ha definido para un Sensor de Temperatura de Transformador, su adaptabilidad inherente lo convierte en una solución versátil para diferentes dispositivos. Esta flexibilidad permite una integración ágil en el ecosistema de WoT, lo que a su vez agiliza el despliegue y la gestión al contenerizar aplicaciones. Por tanto, este enfoque enriquece la utilidad de los dispositivos IoT al fomentar el intercambio de datos y alertas, promoviendo la interoperabilidad en diversas aplicaciones y plataformas, independientemente del proveedor.

4. IMPACTO ECONÓMICO

El análisis del impacto económico se centra en la computación distribuida y la interoperabilidad en este proyecto.

La incorporación de nodos Edge en la red de BT busca modernizarla, y tiene implicaciones económicas significativas. Estas inversiones se distribuyen en áreas clave, como la instalación de equipos Edge, la plataforma de gestión, certificaciones de interoperabilidad, coste de licencias y aplicaciones, componentes necesarios, operación y mantenimiento. Por otra parte, conlleva eficiencia operativa y ahorros, como la reducción de costos de inspección, reparaciones y compensaciones por interrupciones, entre otros. La adopción de sistemas distribuidos también optimiza recursos y reduce pérdidas técnicas y no técnicas, con un enfoque flexible en nuevas funcionalidades.

Por otro lado, la interoperabilidad de aplicaciones bajo un estándar ofrece ventajas, como la reducción de costos de desarrollo y la eficiencia en el intercambio de datos, fomentando la competencia entre proveedores y evitando dependencias.

5. CONCLUSIONES

- Se confirma la importancia de plataformas de gestión de aplicaciones y nodos Edge en SGs para la gestión y monitoreo. La gestión centralizada se vuelve crucial con la expansión de dispositivos IoT, permitiendo actualizaciones remotas y un ambiente integrado para desarrollo y depuración. La plataforma Onesait Phygital Edge, basada en virtualización y contenedores, es ideal para distribuir información en la cadena de valor energético.
- Se destaca que la tecnología de virtualización promueve eficiencia, gestión simplificada y portabilidad, con Docker como elección respaldada por su facilidad de uso, escalabilidad y adecuación a las demandas cambiantes de las SGs.
- La implementación de conceptos WoT permite una comunicación fluida y eficiente entre aplicaciones, facilitando la interoperabilidad. Los TDs y directorios WoT promueven la colaboración y el patrón de comunicación y la combinación JSON-MQTT mejoran la interoperabilidad y el intercambio de datos.
- Mediante el desarrollo de la especificación, se demuestra la integración fluida de aplicaciones con hardware en subestaciones secundarias, creando aplicaciones para construir TDs de sensores de temperatura y generar alertas. Esta flexibilidad facilita la adaptabilidad a diferentes dispositivos IoT y enriquece el intercambio de datos y cooperación entre aplicaciones y plataformas, sin importar el proveedor.

INTEROPERABILITY OF APPLICATIONS IN THE SMART GRIDS CONTEXT

Author: Lerena Rubio, María
Academic Advisor: Rodríguez Pérez, Néstor.
Industrial Advisor: Jiménez Segado, Eduardo.
Collaborating Entity: Minsait

PROJECT SUMMARY

1. INTRODUCTION

Smart Grids (SGs) are modern electricity distribution systems, vital for the transition of the energy market towards sustainability and efficiency. Key features of an SGs include optimizing asset use, integrating distributed generation and storage, ensuring power quality, anticipating, and responding to disruptions, protecting against physical and cyber-attacks, and enabling consumer participation.

Effective network architecture design is crucial for managing the large number of connected devices. In SGs, the bidirectional flow of energy is favored thanks to technologies such as control systems. By collecting and transmitting consumer data, SGs resemble telecommunications networks, benefiting both consumers and operators. This energy and communications infrastructure is shown in ***Figure 1***. IoT devices in SGs optimize management, detection of faults and enable consumer-network interaction, maximizing energy efficiency. To avoid data overload, the application of decentralized processing through Edge Computing is proposed.

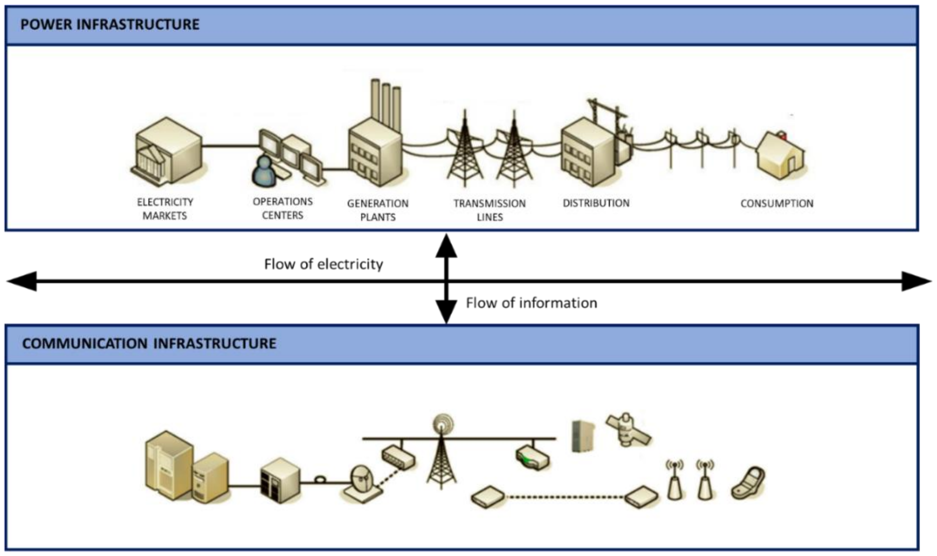


Figure 1. Smart Grid

Figure 2 proposes a new vision of information exchange in the Secondary Substation (SS) of a SG. The pyramid model illustrates the hardware, Edge Nodes, and centralized IoT platform for data exchange and control.

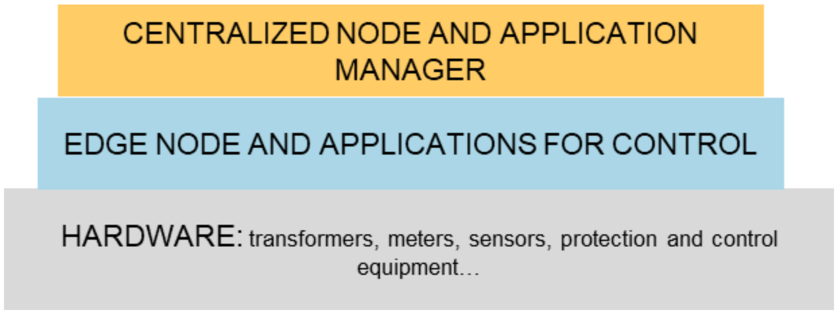

CENTRALIZED NODE AND APPLICATION MANAGER
EDGE NODE AND APPLICATIONS FOR CONTROL
HARDWARE: transformers, meters, sensors, protection and control equipment…

Figure 2. New vision of information exchange in a SS of a SG

The key challenge of SGs lies in the integration of components, systems, information, and applications. Interfaces and functionalities must ensure interoperability to enable high-level processes. The connection of all the components of an electrical network gives rise to an interconnected network in which the flow of information and its analysis will be carried out in real time.

The Web of Things (WoT) standardizes protocols and APIs, promoting device interaction and data exchange. Thing Descriptors (TDs) define device capabilities, properties, and interactions, fostering interoperability and integration within applications.

The main objectives of this project are to investigate the role of application management platforms and edge nodes in the vision of SGs, explore the role of application virtualization in the search for interoperability, study WoT standards to achieve application interoperability and create a TD specification for interoperability between applications in the domain of a SS.

2. ANALYSIS

2.1. IoT in Smart Grids

Current distribution networks were not designed to manage bidirectional energy flows. As demand for energy grows and distributed generation increases, technical limits on supply lines can be exceeded, leading to power quality issues. This causes an imbalance between generation and demand at the local level, with slower responses and quality losses.

Faced with these problems, it is necessary to find measures that allow us to act more quickly in situations of energy imbalance. An additional challenge is the monitoring of distribution networks. Traditionally, it is done in HV and MV networks, leaving a lack of information in BT, where most consumers are connected.
An alternative to address this is to connect the transformers of the distribution network to the control center, which implies high investment costs and workload. Another option is the implementation of the

Industrial Internet of Things (IIoT), which installs sensors and gateways to collect and process data locally, monitoring the network more deeply.

The adoption of the IIoT represents a significant change for utilities, which traditionally use OT systems, such as SCADA systems, isolated from other systems and the internet. The process of adopting IoT in SGs starts by monitoring the LV network, capturing data from physical devices and processing information locally to generate alerts for parameter deviations. This reduces the load of data transmitted through gateways, decreasing wireless communication costs.

The adoption of IoT in SGs presents revolutionary opportunities as shown in ***Figure 3***. But it also brings security, privacy and interoperability challenges that must be carefully addressed.

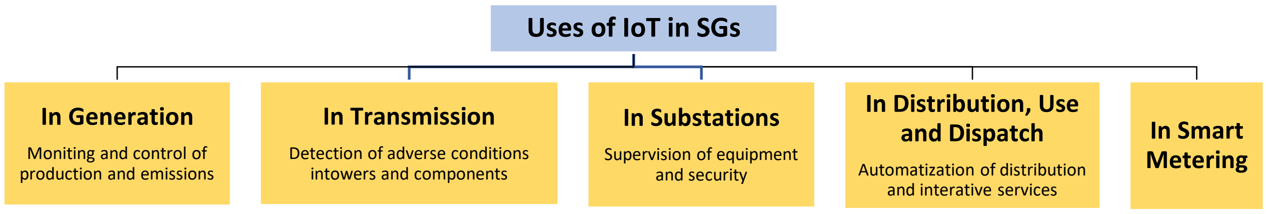

Uses of IoT in SGs
In Generation
Moniting and control of production and emissions
In Transmission
Detection of adverse conditions intowers and components
In Substations
Supervision of equipment and security
In Distribution, Use and Dispatch
Automatization of distribution and interative services
In Smart Metering

Figure 3. Uses of IoT in SGs

2.1.1. Interoperability

Focusing on interoperability, there are different categories that describe the different levels of interoperability that can be considered in the context of communication between systems and platforms in IoT technologies.

In this work and within the study of interoperability, the focus is on the network, syntactic, and semantic interoperability layers, to study their application in the context that any developed application can work on any platform, regardless of the provider.

- **Network interoperability**. It is responsible for transporting information between different devices that interact through different communication networks.
- **Syntactic interoperability**. It is the agreement on format and structure rules used when encoding the information that is exchanged between devices.
- **Semantic interoperability**. It is responsible for ensuring that the transmitted data is interpreted in a common way between the different systems and applications. All actors involved in the transmission of information, both senders and receivers of data, must be under a standardized information model that will be used as a reference. The most common way to get systems to understand the same language is through data ontologies, which are a kind of maps that clearly explain what each thing means, helping devices and systems understand and share information correctly.

  Through ontologies, the information that explains the data shared by IoT devices can be transformed into a structured sequence, which helps to make sense of the information coming from the devices and to understand it correctly, since a clear context is established for its subsequent interpretation. Ontologies can also function as a channel for sending different types of instructions to devices. To do this, they provide guidelines to ensure that commands are executed correctly, including security and access to information, as well as device operations.

2.1.2. Telecommunication protocols

In the context of IoT and Edge Computing, it is critical to explore in depth the communication protocols that enable efficient integration between different devices and systems. In particular, it is convenient to delve into the MQTT protocol in the IoT context, especially for this work, due to the following:

- It is designed for IoT applications with low latency and limited bandwidth, essential in SGs where fast and efficient communication between IoT devices and the electrical grid infrastructure is crucial.
- It is ideal in scenarios with unstable connections and low power consumption, ensuring reliable communication even in challenging conditions, relevant in SGs with devices in remote areas or adverse environmental conditions.
- Its publish-and-subscribe topology benefits SGs where multiple devices need to monitor and receive data from multiple sources, enabling efficient distribution of messages to multiple subscribers.
- It is suitable for real-time monitoring and remote control applications, important aspects in SGs where fast transmission of data from sensors and IoT devices enables effective real-time network management.
- In terms of security and privacy, MQTT simplifies message encryption using TLS and client authentication using modern protocols such as OAuth.

How MQTT works is simple and follows a publish-subscribe pattern, shown in ***Figure 4***.

- The editor creates the message and publishes it to a specific topic.
- The subscriber receives messages relevant to the topic to which he has subscribed.
- A broker communicates with clients through a local network or internet connection.

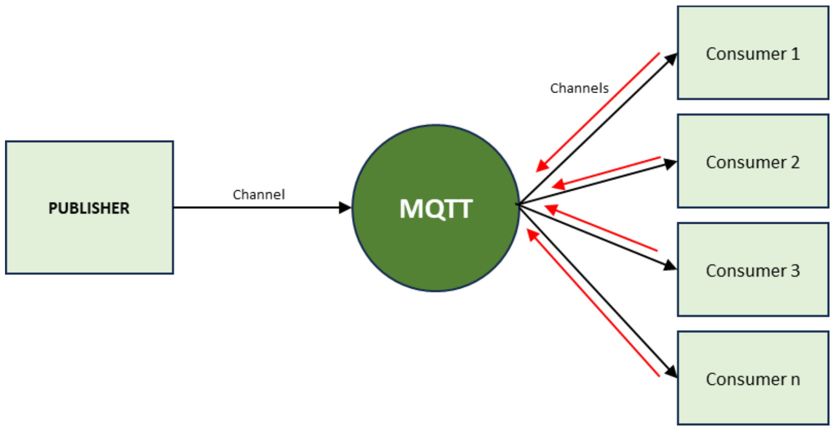


Figure 4. MQTT Structure

2.2. Importance of the Edge in SGs

Along with IoT, the term Edge is common. Applied to SGs, an edge node is located at strategic points where data is generated, transmitted, or consumed in the different stages of generation, transmission, and distribution of electrical energy. The primary function of an edge node is to process, analyze, and make decisions about data in real time before transmitting it to a centralized location, such as a data center or the cloud.

In this way, basic and advanced measurement data concentrators and supervisors in LV enable the local collection and processing of energy consumption data and supply monitoring in the LV network. This makes it easy to detect and resolve issues in real time without relying on the connectivity and latency associated with sending data to a centralized location.

2.3. Application Virtualization

Application virtualization allows applications to run in isolated and virtualized environments, separate from physical infrastructure and other applications on the system. Containers or virtual instances are created that encapsulate applications and their dependencies, providing a consistent and isolated environment for their execution.

In IoT and SGs, application virtualization has benefits such as resource efficiency, simplified management, minimal disruption, and rapid provisioning. Hypervisors and containers are prominent virtualization techniques. Containers, especially Docker, stand out for flexibility, portability, and agile performance. ***Figure 5*** shows a container-based virtualization structure.

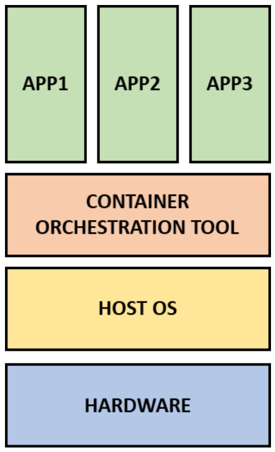


Figure 5. Container-based virtualization

Docker is chosen for its ease, portability, and scalability. In IoT Edge Nodes, Docker provides advantages such as application portability and fast, lightweight performance compared to virtual machines (VMs). Docker allows to allocate and limit CPU, memory, network, and disk resources, ensuring an equitable distribution. Its architecture is based on containers with features such as "cgroups" and "namespaces", allowing to manage, limit system resources, and isolate processes. These technologies prevent interference and ensure an isolated environment for each process.

2.4. Centralized Node and Applications Manager

IoT platforms facilitate the development and deployment of IoT systems, allowing users to focus on what is most important to them, the operation of their businesses. With the increase in the number and geographical dispersion of IoT device deployments, the need for a centralized management system that provides basic information about the status of deployed devices and allows software updates remotely becomes evident. Applications running on devices must be deployed remotely and securely. Operational robustness is essential in any IoT platform, as well as cybersecurity, especially in the industrial area.

When selecting an IoT platform, it is important to distinguish between generic platforms and more specific products. The former, such as Azure IoT Hub, AWS IoT Core, Google Cloud IoT, and Oracle IoT Cloud, offer a broad ecosystem of tools, but often have a steep learning curve and are not tailored to specific needs. On the other hand, specific platforms, such as Minsait Onesite Phygital Edge, offer more limited capabilities but better user support and shorter development times.

IoT nodes act as intermediaries between data capture devices and centralized processing systems. These nodes focus on efficiently collecting and transmitting data to core systems or the cloud for deeper analysis. The implementation of IoT nodes involves physical installation and configuration with the logic and algorithms necessary for their purpose. These nodes are where the operating system and applications run, either natively or in Docker containers.

2.4.1. Onesite Platform

Minsait Onesite Platform is comprehensive and offers modules that allow organizations to develop customized solutions, from data management and analysis to the creation of IoT applications and solutions. The platform offers capabilities such as asset and data management, advanced analytics, visualization, and application creation.

2.4.2. Phygital Edge Solution

Onesite Phygital Edge is a component of the Onesite Platform that focuses on managing IoT devices and systems at the network edge. This platform is based on a processing system close to the devices on the network. A virtualization and container architecture are deployed on these nodes to efficiently distribute intelligence in the form of microservices.

The architecture of the Phygital Edge platform is based on three key components, which work together to enable complete device management in the field and offer functionalities through a specialized agent:

- **Field devices**. These devices are connected to the platform to allow remote management.
- **Application and Edges Management System**. This distributed global management center has as its main function to commission, access, configure, update, and manage edge devices. This component is shown in *Figure 6*.

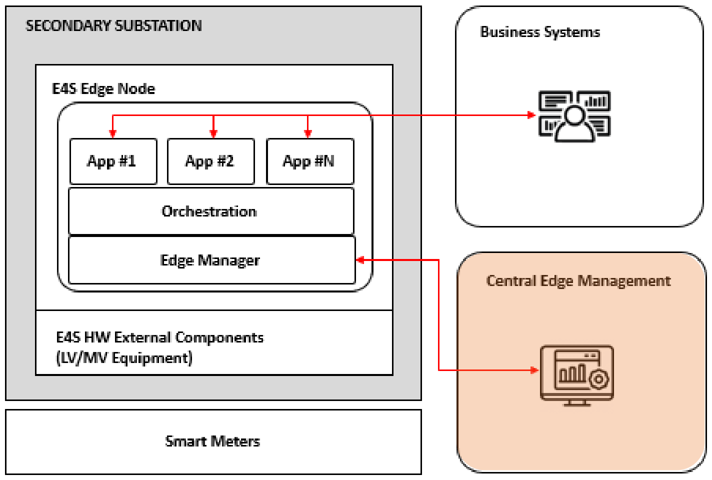


Figure 6. Application and Edges Management System

The figure of the agent is important, since it facilitates the interaction between the central administrator and the edge node, guaranteeing the functionalities of the module. Agent management involves controlling and managing the software agents installed on the edge node, which can execute commands, collect data, and transmit it. In addition, the Edge Node and Application Management System must enable remote installation, uninstallation, upgrade, and downgrade of agents on the edge node, and monitor their status and performance. An API can be provided for agent management, used by the graphical interface and command line.

- **Edge Engine**: This set of capabilities deployed on nodes through containers enable remote control of nodes in the field, local information acquisition and processing, as well as connection to the enterprise cloud to scale and perform deeper analysis.

In terms of security, the Phygital Edge platform employs an identity record that stores information about devices and modules authorized to connect. The devices are authenticated in the Management System using credentials stored in the registry, ensuring secure and reliable connections.

2.4.3. Application of this platform to the SS

Various applications of Phygital Edge have been identified in the SS, and the use cases and requirements to make them feasible have been summarized in *Table 1*.

Table 1. Use Cases of Phygital Edge in a SS

Use Case	Description	I/O Requirements
Data Concentrator for Smart Meters	This application collects data from smart meters using PLC PRIME technology. It then sends that data over the WAN connections available in the Data Hub	It requires a PRIME Base Node connected to the three phases and the neutral of the LV network in the secondary substations
Advanced LV supervision	This application monitors the LV panels on the SS. Communicates via DLMS with LV I/O cards	Requires the installation of I/O cards in each phase of the LV panel, to measure instantaneous values (V, I, P and Q) of each phase
Network Topology Identification	This application calculates the LV topology, identifying the line and phase for each smart meter connected to the low voltage panel. Uses information from Data Concentrator and LV Advanced Supervision	Does not require additional I/O as it uses data from previous applications
Interruptions Identification	This application has identified outages in the LV network, estimating the size of the outage and its impact on customers. Uses information from the three previous applications	Does not require additional I/O as it uses data from previous applications
Transformer Regulation Monitoring and Control	This application monitors the LV parameters of the transformer and regulates the output voltage to adapt the power quality to the state of the grid. It uses information from previous applications, and provides an interface with the transformer controller to modify the output of the transformer	Requires an I/O module that can operate the transformer controller
Video Analysis for visual monitoring	This application can cover several use cases based on the video signal coming from one or several digital cameras that capture different areas of the substation	Requires one or more video cameras
Substation Load Balancing and Load Profile	This application calculates secondary substation load statistics by phase, circuit and transformer to suggest better customer distribution	Does not require additional I/O as it uses data from other applications
Generation and Demand Prediction Analysis	This application calculates demand and generation predictions based on historical data from smart meters and low voltage panels	Does not require additional I/O as it uses data from other applications
Distributed Energy Resources Management (DERMS)	This application operates distributed energy resources to respond to specific grid situations.	Requires an I/O to communicate with the DERs

The container orchestration system, an essential component of the platform, enables efficient management of applications and resources in complex distributed environments.

The designed communication model seeks three types of information exchange between the software components deployed in the substation.

• Message Type 1 - On-Demand. An application sends MQTT to request information.
• Message Type 2 – Scheduled. An application uses MQTT to schedule task or request info, with details and URI to access later.
• Message Type 3 – Spontaneous. An application publishes data in MQTT topic, and others subscribe to receive data.

These communication flows must be compatible and are achieved using the MQTT protocol, which is combined with the WoT architecture to achieve interoperability and efficient communication between applications.

WoT allows applications deployed on different nodes to communicate seamlessly and efficiently. TDs define the IoT devices present in the CT, facilitating the integration of new devices and offering a unified view of the assets of the substation.

The SS-specific WoT-A architecture treats applications as "Things," each with a full set of functionalities including information reading, status updates, notification subscriptions, and features. The WoT Directory is a fundamental part of this architecture, where applications hang their ID cards. Communications between applications occur directly without intermediaries, and an efficient communication protocol, such as JSON-MQTT, is used to ensure improved interoperability.

3. SPECIFICATION DEVELOPMENT

In this work, an application has been deployed together with its corresponding TD, based on the WoT standard. The programmed TD defines the capabilities, properties, interactions, and other relevant information about the Thing, which in this case represents a hardware element, allowing interoperability and seamless integration with WoT applications. This TD facilitates integration and interaction with IoT devices or services by other applications or platforms.

The final objective of the application is to receive signals from IoT devices deployed in a Minsait laboratory environment, fill a TD of a Transformer Temperature Sensor with the information of those messages coming from this type of device. Then another app reads that TD and creates alerts when the temperature exceeds a previously determined value.

The hardware device selected to fill the proposed TD and whose measurements are subsequently monitored is a wireless temperature sensor. These compact sensors can communicate without physical wires and use protocols such as Bluetooth, Wi-Fi and Zigbee, facilitating fast and reliable transmission of temperature data. In addition, wireless technology offers flexibility and scalability, as sensors can be easily deployed and moved, adapting to different monitoring scenarios. In the context of SGs, these sensors are very useful for monitoring temperature variations in electrical equipment, transmission lines, substations, and other critical grid components. This continuous monitoring allows operators to detect potential faults or problems in the network caused by overheating, taking preventive measures to improve the reliability and security of the network.

The practical part of this work establishes several fundamental objectives, including efficient communication between devices and systems using the MQTT protocol, management of data generated by IoT devices, extraction of relevant information from the data received, and interoperability to facilitate integration and compatibility between devices and applications in the SG.

Strengths include the use of the MQTT protocol, widely adopted and supported by various devices, the JSON format for data exchange, and custom TDs to structure and standardize information about devices and capabilities. In addition, in the context of SGs, these strengths enable greater energy efficiency, electrical load management, real-time monitoring and the integration of renewable energy sources.

Table 2 presents the main tools used for the development of the specification, along with their application in the field of SGs.

Table 2. Tools and application in SGs

Tool	Description	SGs application
Python	Versatile and easy-to-use programming language	It facilitates the development of IoT applications in SGs
	Extensive community and libraries for specific tasks	It allows you to process data and communicate with devices in the context of IoT
	Multiplatform, compatible with various operating systems	It ensures that the application works in different environments
	Flexibility to develop a variety of applications	Adaptable to different requirements in the context of SGs
	json: Facilitates data processing in JSON format	Structure the TD and temperature data in an organized way
	paho.mqtt.client: Implements MQTT communication	Establishes efficient communication between the thermostat and the central system
	datetime: Provides tools for working with dates and times	Records the date and time of temperature data for analysis and tracking
Docker	Container platform that makes it easy to create, deploy, and manage	It provides an isolated and consistent environment for development and execution
	Portability and consistency in different environments	It ensures that the application works consistently across various platforms
	It facilitates teamwork and avoids configuration conflicts	It streamlines the development and deployment of the application in different stages
	Isolation and security in the execution of the application	Improves security by running your application in isolated containers
MQTT Explorer	Open source MQTT visualization and debugging tool	Verifies MQTT communication between the thermostat and the central system
	Provides monitoring and debugging of MQTT messages	Facilitates the identification and resolution of communication problems
	Intuitive graphical interface to represent MQTT communication	It allows to visualize in real time the messages and topics sent and received

3.1. Model

In the context of a SS, a wireless temperature sensor is used to capture the outside temperature of the transformer. This device communicates via Zigbee with the Zigbee Gateway.

The Zigbee Gateway is a device that acts as an intermediary between Zigbee devices and other communication systems. Basically, it allows Zigbee devices to communicate with systems and applications that use other communication protocols, such as MQTT. Information collected from Zigbee devices, such as sensors or actuators, can be transmitted via MQTT to the central system.

Once IoT applications and devices have performed their calculations or carried out their actions, it is crucial that the results and data generated are shared and communicated effectively. To achieve this, a Node and Application Management Platform is used, which acts as a control and coordination center for the IoT infrastructure. However, this part of the communication with the platform is not included in the practical development of this work.

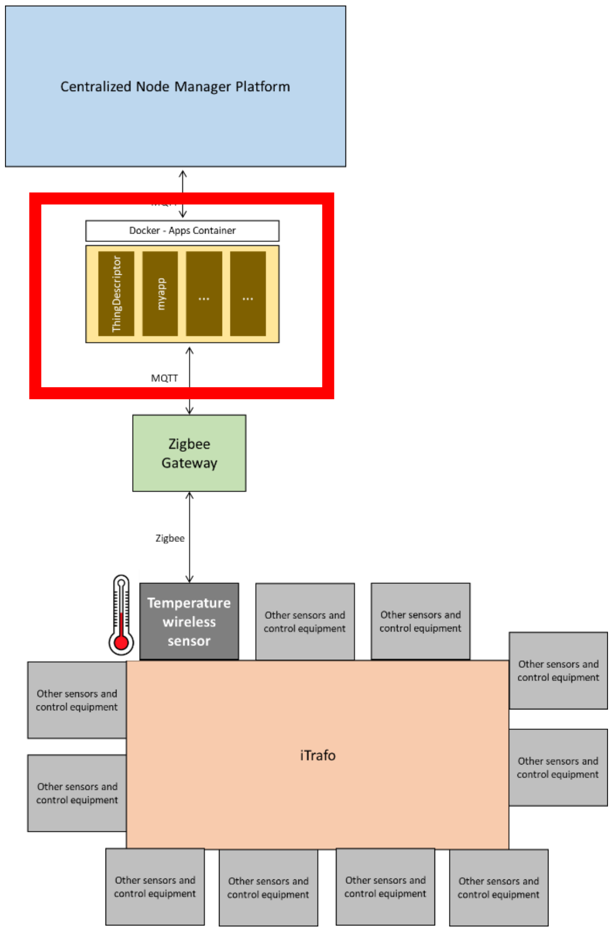
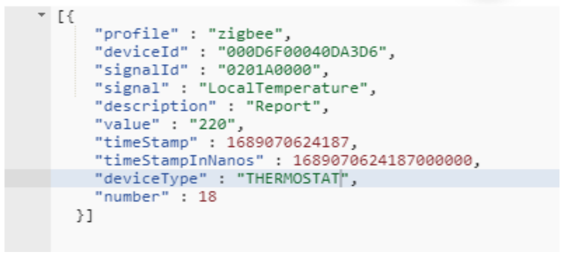

Centralized Node Manager Platform
MQTT
Docker - Apps Container
ThingDescriptor
myapp
...
...
MQTT
Zigbee Gateway
Zigbee
Temperature wireless sensor
Other sensors and control equipment
iTrafo

Figure 7. Model of the practical part of the work

3.2. Results

Test #1: A message published by MQTT is read from a device. The deviceType is a temperature sensor of the SS, and whose temperature does not exceed the limit value of 20ºC, as shown in ***Figure 8***.

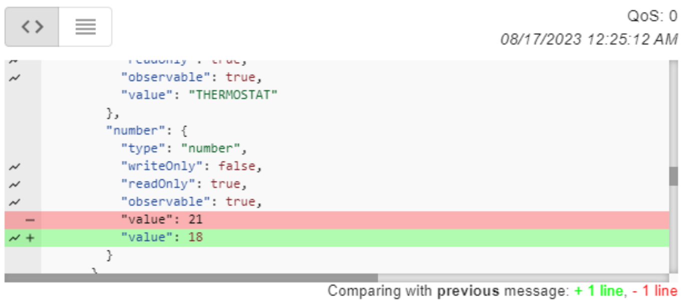
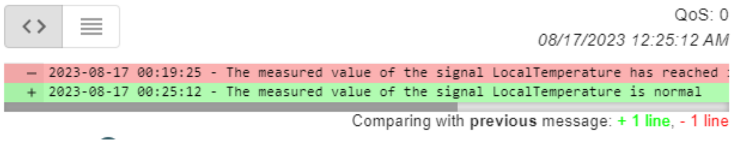


```
[{
    "profile" : "zigbee",
    "deviceId" : "000D6F00040DA3D6",
    "signalId" : "0201A0000",
    "signal" : "LocalTemperature",
    "description" : "Report",
    "value" : "220",
    "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
    "deviceType" : "THERMOSTAT",
    "number" : 18
}]
```

Figure 8. Test #1: Message Read from a Thermostat

From it, a message containing the TD filled with the information read has been published. In another MQTT topic, the corresponding message for temperature measurement within the limits has been received. These are shown in ***Table 3***.

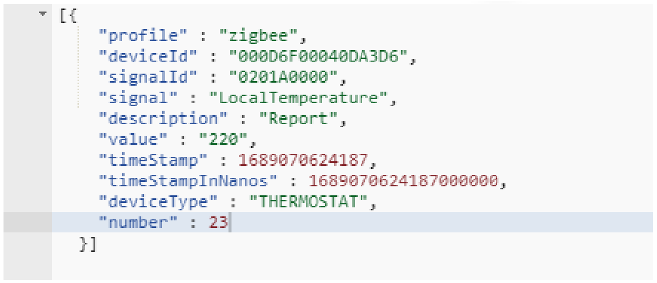
Table 3. Test #1: Published messages (on the left the TD, on the right the limit message of Tª)

Topic 1	Topic 2
	

```
QoS: 0
08/17/2023 12:25:12 AM
~           "observable": true,
            "value": "THERMOSTAT"
          },
          "number": {
            "type": "number",
~           "writeOnly": false,
~           "readOnly": true,
~           "observable": true,
  -         "value": 21
~ +         "value": 18
          }
Comparing with previous message: + 1 line, - 1 line
```

```
QoS: 0
08/17/2023 12:25:12 AM
- 2023-08-17 00:19:25 - The measured value of the signal LocalTemperature has reached
+ 2023-08-17 00:25:12 - The measured value of the signal LocalTemperature is normal
Comparing with previous message: + 1 line, - 1 line
```

Test #2: An MQTT-published message is now read from the same device type, but exceeding the 20°C limit, as shown in ***Figure 9.***

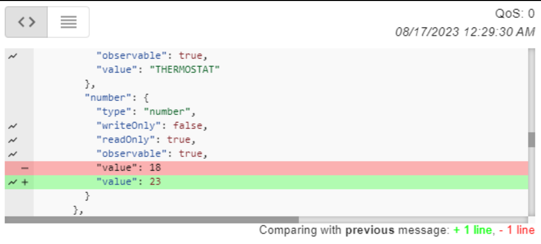
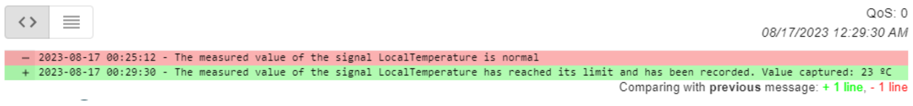


```
[{
    "profile" : "zigbee",
    "deviceId" : "000D6F00040DA3D6",
    "signalId" : "0201A0000",
    "signal" : "LocalTemperature",
    "description" : "Report",
    "value" : "220",
    "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
    "deviceType" : "THERMOSTAT",
    "number" : 23
}]
```

Figure 9. Test #2: Message Read from a Thermostat

Again, the corresponding messages are published with the information read, as shown in ***Table 4***.

Table 4. Test #2: Published messages (on the left the TD, on the right the limit message of Tª)

Topic 1	Topic 2
	

```
QoS: 0
08/17/2023 12:29:30 AM
~           "observable": true,
            "value": "THERMOSTAT"
          },
          "number": {
            "type": "number",
~           "writeOnly": false,
~           "readOnly": true,
~           "observable": true,
  -         "value": 18
~ +         "value": 23
          }
        },
Comparing with previous message: + 1 line, - 1 line
```

```
QoS: 0
08/17/2023 12:29:30 AM
- 2023-08-17 00:25:12 - The measured value of the signal LocalTemperature is normal
+ 2023-08-17 00:29:30 - The measured value of the signal LocalTemperature has reached its limit and has been recorded. Value captured: 23 ºC
Comparing with previous message: + 1 line, - 1 line
```

Test #3: The correct containerization of the developed applications is checked. Three containers have been created on the system and all of them are running correctly.

```
4. CONTAINER
   ID   IMAGE                                                          COMMAN
   D                  CREATED           STATUS          PORTS
                           NAMES
5. dd2473d82c99   myapp_image:latest
      "python myapp.py"        13 seconds ago   Up 12
   seconds                                                myapp_container
6. 6e72ccb33867   thing_descriptor_app_image:latest
       "python ThingDescrip…"   4 minutes ago    Up 4
   minutes                                                 thing_descriptor
   _container
7. fb35bfe64c8a   emslab.onesaitplatform.com/onesait-things/edge-
   mqtt:2.0.0   "/docker-entrypoint.…"   2 weeks ago      Up 2
   weeks      0.0.0.0:1883->1883/tcp, :::1883->1883/tcp   edge.mqtt
```

The developed specification marks a milestone in the integration of IoT devices within the WoT framework. With the addition of TDs, interoperability with a wide range of IoT devices is ensured. Although initially conceived for a Transformer Temperature Sensor, its inherent adaptability makes it a versatile solution for different devices. This flexibility enables agile integration into the WoT ecosystem, which in turn streamlines deployment and management when containerizing applications. Ultimately, this approach enriches the usefulness of IoT devices by encouraging the exchange of data and alerts, promoting interoperability across diverse applications and platforms, regardless of vendor.

4. ECONOMIC IMPACT

The economic impact analysis focuses on distributed computing and interoperability in this project.

The incorporation of edge nodes into the BT network seeks to modernize it and has significant economic implications. These investments are distributed in key areas, such as the installation of Edge equipment, the management platform, interoperability certifications, licenses and costs of applications, necessary components, and operation / maintenance. This translates into operational efficiency and savings, such as reduced inspection costs, repairs, and compensation for interruptions, among others. The adoption of distributed systems also optimizes resources and reduces technical and non-technical losses, with a flexible focus on new functionalities.

The interoperability of applications under a standard offers advantages, such as reduced development costs and efficiency in data exchange, encouraging competition between suppliers and avoiding dependencies.

5. CONCLUSIONS

- The importance of application management platforms and edge nodes in SGs for management and monitoring is confirmed, with edge nodes allowing preliminary on-site analysis. Centralized management becomes crucial with the expansion of IoT devices, enabling remote updates and an integrated environment for development and debugging. The Onesait Phygital Edge platform, based on virtualization and containers, is ideal for distributing information in the energy value chain.
- Virtualization technology promotes efficiency, simplified management, and portability, with Docker as a choice backed by its ease of use, scalability, and adaptation to the changing demands of SGs.
- The implementation of WoT concepts allows fluid and efficient communication between applications, facilitating interoperability. TDs and WoT directories promote collaboration and communication pattern and JSON-MQTT combination improve interoperability and data exchange.
- Through the development of the specification, the seamless integration of applications with hardware in secondary substations is demonstrated, creating applications to build TDs of temperature sensors and generate alerts. This flexibility facilitates adaptability to different IoT devices and enriches data exchange and cooperation between applications and platforms, regardless of the provider.

6. BIBLIOGRAPHY

[1] O. Vermesan, Advancing IoT Platforms Interoperability. New York: River Publishers, 2022.

[2] A. Goudarzi, F. Ghayoor, M. Waseem, S. Fahad, y I. Traore, "A survey on IoT-enabled smart grids: Emerging, applications, challenges, and outlook", Energies, vol. 15, núm. 19, p. 6984, 2022.

[3] 'Phygital Minsait', Minsait.com. [Online]. Available: https://www.minsait.com/es/aceleradores/phygital.

[4] "Grupo de trabajo de CT Inteligente - FutuRed. Plataforma española de redes eléctricas", FutuRed. Plataforma española de redes eléctricas, 24-mar-2021. [oniline]. Available in: https://www.futured.es/grupo-trabajo-ct-inteligente/.

[5] "MQTT - the standard for IoT messaging", Mqtt.org. [online] online in: https://mqtt.org/.
W. Shi, J. Cao, Q. Zhang, Y. Li, y L. Xu, "Edge Computing: Vision and Challenges", IEEE Internet Things J., vol. 3, núm. 5, pp. 637–646, 2016.

[6] S. Singh y N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology", en 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804–807.

[7] "Documentation - web of things (WoT)", Www.w3.org. [online]. Available in: https://www.w3.org/WoT/documentation/.

[8] "Web of Things (WoT) architecture 1.1", Www.w3.org. [online]. Available in: https://www.w3.org/TR/2023/PR-wot-architecture11-20230711/.

[9] European Platform Initiative, "ADVANCING IoT PLATFORMS INTEROPERABILITY", Iot-epi.eu, 2018. [online]. Available in: https://iot-epi.eu/wp-content/uploads/2018/07/Advancing-IoT-Platform-Interoperability-2018-IoT-EPI.pdf.

TABLE OF CONTENT

INDEX OF FIGURES

INDEX OF TABLES

GLOSSARY OF TERMS

ADM	Advanced Distribution Management
ADMS	Advanced Distribution Management System
AEMS	Application and Edges Management System
AI	Artificial Intelligent
AIOTI	Alliance for Internet of Things Innovation
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ARP	Address Resolution Protocol
ASN.1	Abstract Syntax Notation One
CAAS	Communication as a Service
CAPEX	Capital Expenditure
CIM	Common Information Model
CoAP	Constrained Application Protocol
DCU	Data Concentration Unit
DER	Distributed Energy Resources
DERMS	Distributed Energy Resource Management System
DG	Distributed Generation
DSO	Distribution System Operator
ETSI	European Telecommunications Standards Institute
EV	Electric Vehicle
ESS	Energy Storage System
ESPs	Energy Service Providers
ETI	Energy Information Framework
ETSI	European Telecommunications Standards Institute
FSD	Fault Step Detector
FTP	File Transfer Protocol
GPS	Global Positioning System
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
IEC	International Electrotechnical Commission
IIoT	Industrial Internet of Things
IoT	Internet of Things
IED	Intelligent Electronic Device
ISS	Intelligent Secondary Substation
IP/IPv6	Internet Protocol / Internet Protocol version 6
JWT	JSON Web Token
LAN	Local Area Network
LV	Low Voltage
LXC	Linux Containers

ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
M2M	Machine to Machine
MV	Medium Voltage
NDN	Named Data Network
NFC	Near Field Communication Protocol
NMS	Network Management System
OPEX	Operating Expenditure
OS	Operating System
OT	Operational Technology
PDS	Power Distribution System
PLC	Power Line Communication
PMU	Phasor Measurement Unit
SaaS	Software as a Service
SAS	Substation Automation System
SCADA	Supervisory Control and Data Acquisition
SG	Smart Grid
SGAM	Smart Grid Architecture Model
SOAP	Simple Object Access Protocol
SNMP	Simple Network Management Protocol
SS	Secondary Substation
SSP	Secondary Substation Platform
SSL	Secure Sockets Layer
TD	Thing Descriptor
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VM	Virtual Machine
WAN	Wide Area Network
WoT	Web of Things
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

INTRODUCTION

This work has been developed in collaboration with Minsait, in the department of "Phygital and Product Unit". The contribution of ideas, documentation, experience, and professional talent by Minsait has been fundamental in the realization of this project.

To begin and contextualize this work, the environment in which it develops will be explored, examining the state of the art of key concepts. The underlying motivation driving the implementation of the project will then be discussed. Last section in this introduction, the objectives pursued towards its fulfillment will be detailed.

Context

SGs are electricity distribution systems that use digital and advanced technologies for the intelligent and safe monitoring and management of the energy transport from generation sources to end users, according to the definition provided in the EIT Monthly Webinar. [1]

The growing demand for SGs drives a transformation of the energy market. This entails a transition towards an economically, socially, and environmentally sustainable framework, capable of managing resources more efficiently. SGs are an essential part of the transformation, enabling distributed or decentralized generation, the use of smart devices, and a two-way flow of energy and communications. These challenges will transform the way the electricity grid operates, from a hierarchical approach to one of distributed control, thus allowing to optimize the use of energy resources, obtain information on generation and demand trends in dynamic environments, and offer new services to end users.

One of the outstanding characteristics of SGs compared to traditional grids is the ability to integrate renewable facilities of different sizes, thus optimizing energy generation and consumption. In this context, users can act both as consumers and producers of energy. This is what is known as prosumers [37].

In terms of network monitoring and control, SGs make use of advanced communication technologies, smart meters, sensors, and automated control systems. Using these technologies, greater monitoring of the electrical system and more efficient management of energy demand are achieved. In addition, with the aim of improving the reliability and resilience of the system, fault detection systems and automatic response to interruptions are implemented. [2]

Grouping SGs features based on a functionality approach, these include the following represented in ***Figure 1*** [1].

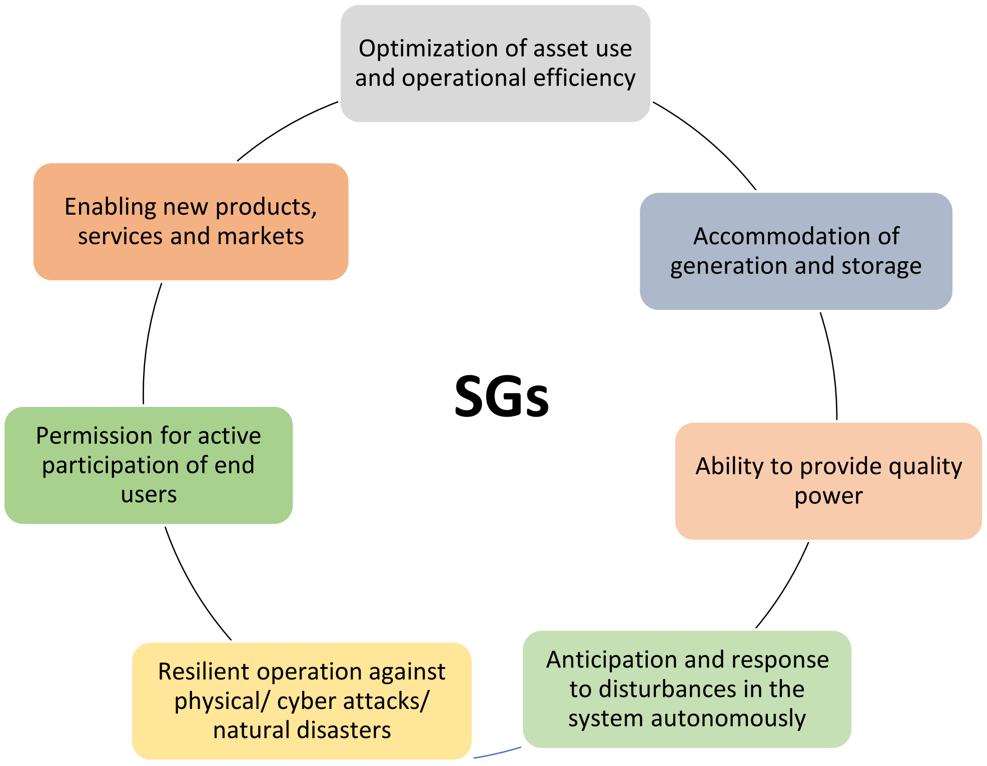

Optimization of asset use and operational efficiency
Accommodation of generation and storage
Ability to provide quality power
Anticipation and response to disturbances in the system autonomously
Resilient operation against physical/ cyber attacks/ natural disasters
Permission for active participation of end users
Enabling new products, services and markets
SGs

Figure 1: SGs features based on a functionality approach

- **Optimization of asset use and operational efficiency**. Through real-time monitoring and data collection, SGs enable efficient management of energy assets.

- **Accommodation of generation and storage**. SGs can integrate different sources of power generation, known as DG. On the other hand, they allow the efficient incorporation of storage systems to store the excess energy produced during those moments of low demand and use it when needed.

- **Ability to provide quality power**. With the incorporation of electronic devices and digital technology, SGs guarantee a stable and quality electricity supply. It thus helps to avoid fluctuations and outages that adversely affect economic activity and end users.

- **Anticipation and response to disturbances in the system autonomously**. SGs allow to identify and anticipate network problems by collecting and analyzing network information in real time. In addition, in case of disturbances, the grid can be automatically reconfigured to restore power supply in affected areas, minimizing the impact.

- **Resilient operation against physical and cyberattacks, and natural disasters**. Advanced infrastructure and safety protocols help maintain power integrity and continuity.

- **Permission for active participation of end users**. SGs allow consumers to play an active role in managing energy consumption, while improving grid stability. This is achieved through technologies such as smart meters and mobile applications for real-time consumption monitoring, adjustment of usage patterns and participation in smart demand programs.

- **Enabling new products, services, and markets**. SGs open the door to the creation of new energy-related products and services, fostering innovation and competition in the energy market. Some of these services are offering more flexible tariff plans based on demand, and allowing users to sell the excess energy they generate to the grid.

In this context, one of the crucial challenges is to discover how to use contemporary ICTs to improve the reliability, stability, and energy efficiency of the electricity grid, as well as meet the needs of consumers. The increasing incorporation of renewable energy sources and other distributed generation sources is a growing trend, and unlike the traditional electricity grid, this type of energy supply has a less centralized and unpredictable infrastructure. This impacts the performance of the power grid, making the use of ICTs for monitoring and control essential. [13]

SGs require a good network architecture design to manage the numerous connected devices and sensors at all points of the network, such as power generation, distribution, and consumption. A smart and robust grid enables two-way and real-time communication, collecting and analyzing data to optimize performance and energy efficiency. The network architecture must be able to support the growing demand for connectivity, in addition to ensuring the interoperability of the devices and systems involved [3].

- Firstly, and unlike the traditional one-way flow from generation to consumption represented in ***Figure 2***, Smart Grids allow electricity to flow in both directions, from generation to consumption, and from consumption to the grid, as represented in ***Figure 3***. This is possible thanks to the implementation of technologies such as smart meters and control systems. These facilitate the injection of electricity generated by distributed sources into the grid to be used by other consumers.

- In addition, the Smart Grid shares similarities with telecommunication networks in terms of the bidirectional transmission of information. As in telecommunication networks, SGs use smart meters and other monitoring devices to collect data on electricity consumption, power quality, demand, and other relevant parameters from the grid. This data is transmitted bidirectionally to network operators, as shown in ***Figure 3***, allowing them to monitor and control the network in a more efficient manner. In addition, consumers also benefit from the two-way transmission of information, being able to access data on energy consumption and prices in real time, allowing them to make more informed decisions and actively participate in energy management.

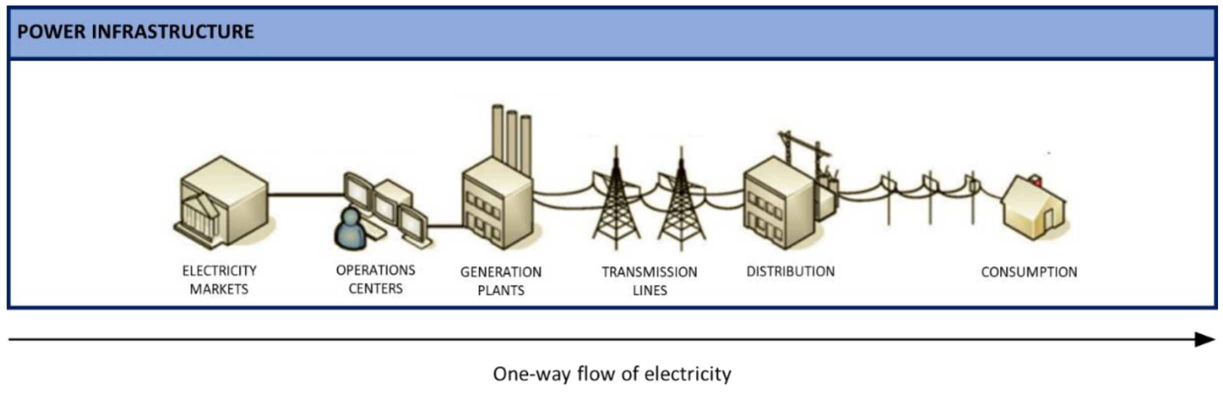


Figure 2: Traditional grid flow of electricity

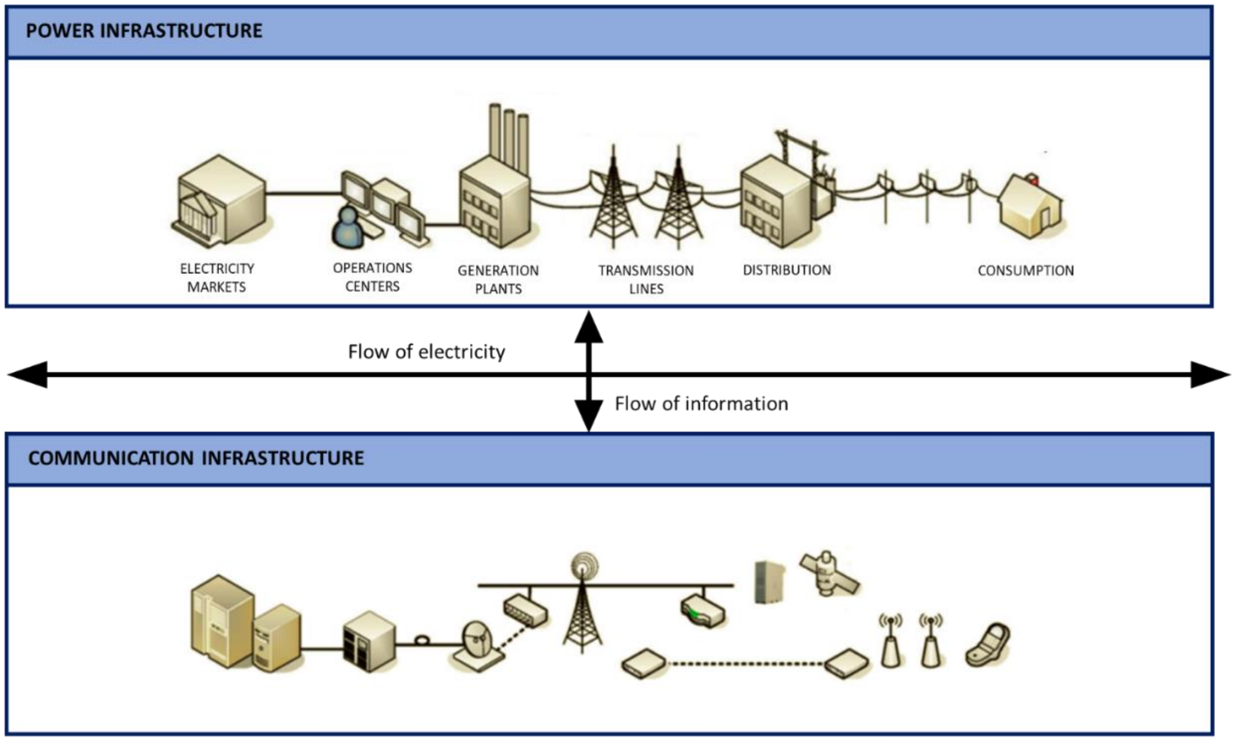


Figure 3: Smart Grid Flow of electricity and information

The approach of loading all information captured from the grid into a centralized system like SCADA can lead to data overload. For this reason, a vision of deployment at the local level that can communicate with the centralized control center is proposed. This is where the concepts of distributed computing and platforms based on IIoT come into play. By placing data processing and analysis capacity close to the source, latency is reduced and the load on the centralized infrastructure is minimized. This allows faster and more efficient decision-making, contributing to better management of energy generation, distribution, and consumption in real time. It is important to assimilate the underlying principles and technologies of IIoT to apply them in the field of SGs, adapting them through appropriate application interfaces and protocols. The IIoT can play an important role in addressing these smart grid interoperability challenges. With the use of IIoT technologies, devices and data, smart grids can achieve seamless communication, integration and interoperability between various components and stakeholders.

IoT and IIoT concepts will be used interchangeably throughout the rest of the document.

Figure 4 shows a schematic representation of the new vision of information exchange in the SS of a SG. A three-level pyramid describing the structure and components involved in data exchange and application management is presented.

- At the base of the pyramid are **hardware components**, which include transformers, sensors, and control and protection equipment, among others. These elements are fundamental for the collection of real-time data and the monitoring of the electrical infrastructure in the transformation center.

- The next level of the pyramid represents the **Edge node and applications for control**. This is where edge nodes are deployed, which act as connection and data processing points close to the devices and sensors that capture the data in the transformer center. These edge nodes enable faster data processing and analysis capabilities, helping to make real-time decisions and perform local control actions.

- At the top of the pyramid is the **Centralized node and application manager**. This level represents the IoT platform where centralized management applications and tools are deployed for the control and monitoring of energy infrastructure. This centralized platform allows efficient management of applications deployed in the SS, facilitating interoperability and data exchange with other systems and networks.

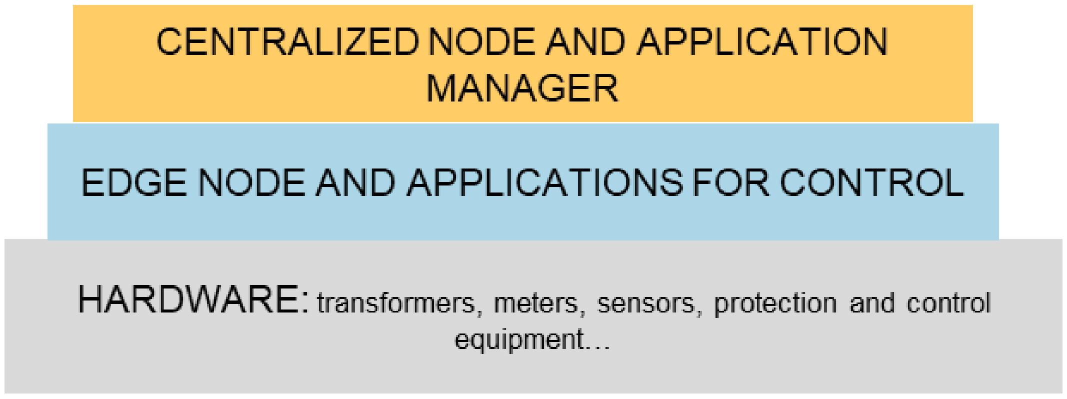

CENTRALIZED NODE AND APPLICATION MANAGER
EDGE NODE AND APPLICATIONS FOR CONTROL
HARDWARE: transformers, meters, sensors, protection and control equipment…

Figure 4: New vision of information models and information exchange

Continuing with communication networks in the context of SGs, they must be able to support the large number of connected IoT devices. The network architecture should contemplate communication technologies and protocols that enable efficient and low-latency communication between IoT devices and management systems. Robust security and privacy mechanisms should be considered to protect the integrity and confidentiality of the transmitted data.

The diversity and incompatibility of protocols is one of the biggest problems in the integration of IoT with grid communication systems. IoT communications rely on specialized protocols for this, such as MQTT [38], CoAP [39] and AMQP [40], while typical electrical communications use IEC 61850 [41], DNP3 [42] or IEC 60870-5-104 [43].

Minsait is working on a standard ontological model for allowing system integration and information exchange as part of several initiatives to unify both systems, as there is currently no common communication language between the IoT world and SGs.

The fundamental architecture that enables this integration is achieved by defining subscribers, such as MQTT. It is an open, compact, and lightweight messaging protocol used in data transmission, perfecto to use in remote locations when a small amount of network bandwidth is required. This protocol will be explained in greater depth and used throughout this work.

Virtualization also plays a critical role in network architecture for SG with IoT, as virtualizing network resources and services achieves greater flexibility and scalability in the infrastructure, making network management more efficient and agile. This also includes the ability to easily deploy and deploy new IoT devices, and the ability to adapt to changes in energy demand or grid conditions in real time.

Digital substations are a key component in SGs with IoT, since they allow the integration of smart devices and sensors at those strategic points of the electrical network, to collect data and thus improve the supervision and control of the network. The network architecture must ensure reliable and secure connectivity between digital substations and central systems, to facilitate data transmission and coordination of operations throughout the network.

The main objective of this work is to highlight the importance of developing interoperable applications in the context of SGs with IoT platforms. Interoperability refers to the ability of different communication devices and systems to exchange and use data in an effective and transparent manner. In this context, it is essential that applications and services designed for SGs are independent of the provider of the IoT platform used.

Interoperability is essential in SGs due to the need to integrate the different components, technologies and communication protocols used by the different manufacturers that make it up. Interoperability allows applications to be deployed and operate efficiently on different IoT platforms, no matter who the vendor is. This fosters competitiveness and mass adoption of SGs, as users and businesses can freely choose the platforms that best suit their needs, without being limited by vendor-specific support.

WoT plays a crucial role in ensuring this interoperability by providing standardized protocols such as HTTP [44], CoAP and MQTT, and APIs that enable interaction and communication between IoT devices and services. These standardized protocols and APIs allow devices from different manufacturers, operating on different platforms, to interact and exchange data in a consistent and interoperable way.

To demonstrate this interoperability under the WoT standard, an application with its corresponding Thing Descriptor will be deployed. This TD is a key component of the WoT architecture, being a metadata representation that provides a standardized description of a physical or virtual entity in the IoT ecosystem. This is called a Thing. The TD defines its capabilities, properties, interactions, and other Thing data, facilitating interoperability and integration with WoT applications.

State of the art

In the context previously exposed, it is worth mentioning the state of IoT platforms, research on interoperability in devices and network components, and the current relationship between utilities and the IIoT.

Firstly, this section focuses on IoT platforms aimed at facilitating the implementation, management and use of connected devices and systems on the IoT. These platforms allow to connect, communicate, collect, and analyze data from IoT devices, as well as deploy applications and services based on them.

IoT refers to the network of interconnected devices and the technology that allows them to communicate with each other and with the cloud. This allows everyday objects to collect data and respond intelligently to users. [45]

Within the concept of IoT, the IIoT arises. This concept is important as it focuses on IoT applications in an industrial environment, as opposed to traditional IoT technology which is applied in different use cases and different sectors. The IIoT is a system in which devices, sensors and autonomous equipment are connected via the Internet in industrial applications. Unlike IoT, which focuses on consumer services, IIoT focuses on increasing security and efficiency in production environments. The IIoT is characterized by five main elements as shown in ***Figure 5*** [14]:

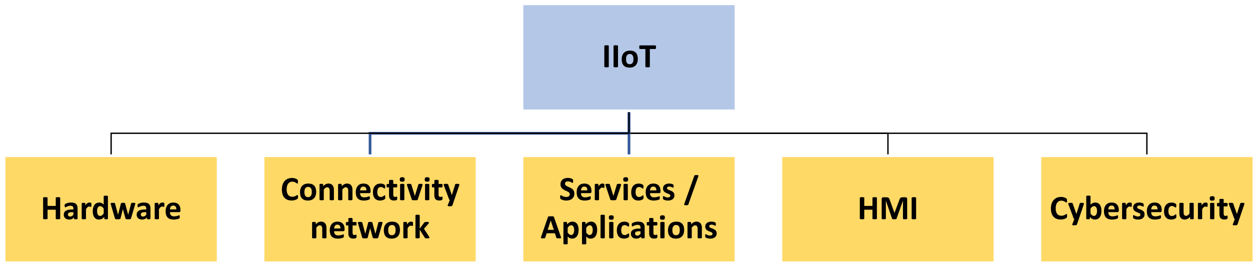


Figure 5: Main elements of IIoT

- **Hardware**. These include sensors and other connected devices through which data is collected.
- **Network**. There is a connectivity network to communicate devices and servers, for which technologies such as Cloud Computing and Edge Computing are used.
- **Services**. These are the computer applications which are deployed to analyze the collected data and subsequently process it. In this way, control, maintenance, and management services can be offered.

- **Content**. It is the interface with the human operator.

- **Cybersecurity**. Any IoT platform must be robust from a cybersecurity standpoint, which is especially important in the industrial world. Security is one of the barriers to the adoption of information technologies by established companies with a strong OT heritage. Cybersecurity solutions include data encryption, limitation of network services, port cancellation or authentication services among others.

As a consequence of the energy transition, and the increasing incursion of renewable energy sources, challenges arise for companies in the energy sector. These include:

- **Need for optimization of networks and infrastructures**. Needed to create a more resilient network and maximize availability.

- **Integration of storage devices**. Necessary to promote the efficiency and fusion of multiple sources of electricity generation.

- **Incorporation of digital technologies** such as AI and IoT to predict and optimize the management and supply of energy to the different traffic networks.

The IIoT [15] has become the key technology to carry out the transition from traditional electricity grids to SGs, focusing on the third challenge previously mentioned.

Among the IoT platforms on the market we highlight AWS IoT Core [16], Azure IoT Hub [17], Google Cloud IoT [18] and Oracle IoT Cloud [19]. All of them are cloud services designed to facilitate the connection, management and analysis of IoT devices. These platforms offer end-to-end solutions for secure device connectivity, ingestion and processing of data, integration with cloud services and enterprise applications. With the implementation of these platforms, organizations can harness the potential of IoT devices to make strategic decisions, improve operational efficiency, and develop new applications and services based on that data.

Apart from the previous ones mentioned that we will consider more general and less oriented to the requirements of Smart grids, the solutions of Barbara IoT [20], Bosch [21] and Minsait (Onesait Phygital Edge) [22] have focused on the field of SG and offer capabilities for Edge Node and Application Management. These platforms typically offer better user support and much shorter development and deployment times.

In this work we focus on the Phygital Edge platform from Minsait and its requirements. Phygital Edge [22] is a software component which provides distributed workload management functions for Edge Computing, enabling a managed and secured virtual environment of applications packaged in containers, allowing the deployment of distributed applications connected in real time to devices and sensors in the field, via standard protocols.

From these IoT platforms and their use in SGs, challenges arise such as interoperability and cybersecurity. Interoperability in that any application can be deployed independently of the platform provider will be studied in this paper.

Continuing with interoperability concept, there are already several definitions suggested by the various literature sources that serve as a starting point for this project [26-30]. In general, it can be defined as the ability of two or more systems to communicate and share information between them.

Interoperability is typically presented in interoperability frameworks.

The European Interoperability Framework (EIF) [34] defines 3 levels of interoperability: technical, semantic, and organizational interoperability, briefly described in ***Figure 6.***

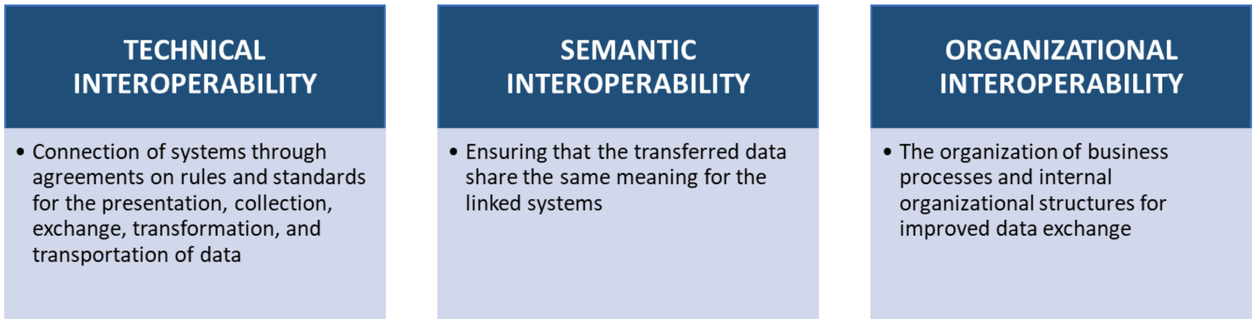

TECHNICAL INTEROPERABILITY
• Connection of systems through agreements on rules and standards for the presentation, collection, exchange, transformation, and transportation of data
SEMANTIC INTEROPERABILITY
• Ensuring that the transferred data share the same meaning for the linked systems
ORGANIZATIONAL INTEROPERABILITY
• The organization of business processes and internal organizational structures for improved data exchange

Figure 6: Interoperability levels defined by EIF

ETSI and AIOTI present a more specific classification for IoT [35], defining 4 levels of interoperability: technical, syntactic, semantic, and organizational interoperability, briefly described in ***Figure 7***.

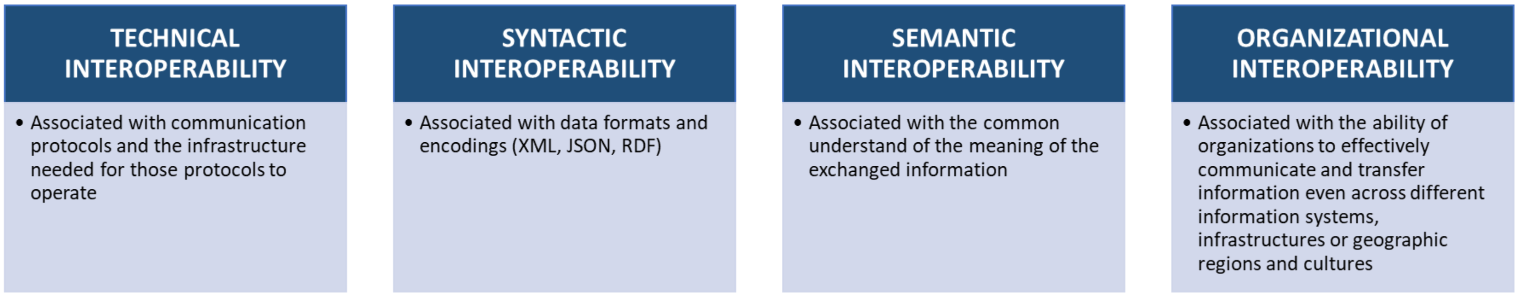

TECHNICAL INTEROPERABILITY
• Associated with communication protocols and the infrastructure needed for those protocols to operate
SYNTACTIC INTEROPERABILITY
• Associated with data formats and encodings (XML, JSON, RDF)
SEMANTIC INTEROPERABILITY
• Associated with the common understand of the meaning of the exchanged information
ORGANIZATIONAL INTEROPERABILITY
• Associated with the ability of organizations to effectively communicate and transfer information even across different information systems, infrastructures or geographic regions and cultures

Figure 7:Interoperability levels defined by ETSI and AIOTI

In research in the field of interoperability, it is important to mention SGAM [33]. It is an architecture model inspired by the model developed by the GridWise Architecture Council. It is used to describe and organize the elements of a SG, providing a hierarchical structure by which it divides the SG into layers, each one representing a specific aspect of network infrastructure and operation.

This model is composed of five interoperability layers, which ordered from bottom to top include: Component, Communication, Information, Function and Business Layer, as represented in *Figure 8*.

- **Business layer**. It focuses on components related to the exchange of business information inside the SG, including functional departments, business processes and organizational capabilities.
- **Function layer**. It shows the relationship of functions and services from an architectural perspective, defining the interactions between the different functions of the SG.
- **Information layer**. It shows the way of sharing information between SG services and components, by defining the necessary protocols and methods.
- **Communication layer**. It is responsible for efficient and secure communication between the different elements of the SG. It includes those protocols and methods described in the information layer.
- **Component layer**. It includes physical components such as network devices and equipment, sharing information to ensure the correct functioning of the SG.

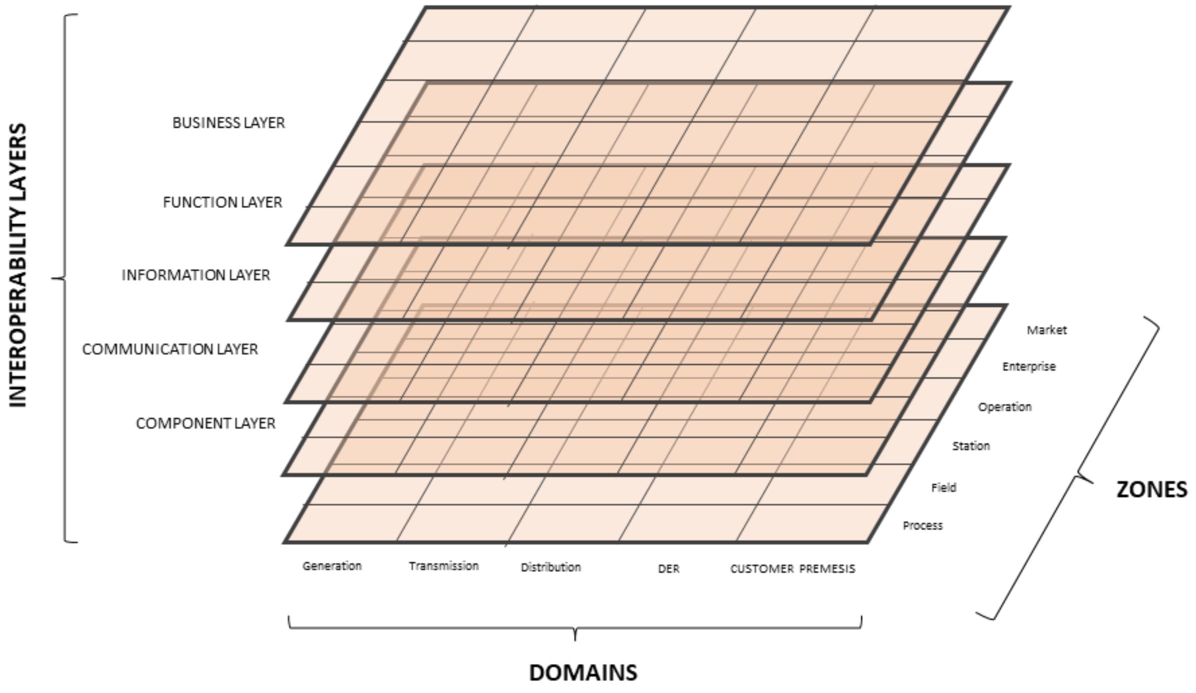


Figure 8: SGAM Model

It is important to mention WoT when talking about interoperability. Starting from the IoT and with the increasing number of devices connected to the internet, to unfold the full potential of this new vision, devices will have to speak the same language and communicate using common channels and protocols. And that is what the WoT comes to solve, which will be key in this work.

Regarding the current relationship between the IIoT and the SG, in terms of the actual application of these platforms in utilities, currently, on a large scale, no distributor has implemented the use of node management platforms effectively. In addition, the demonstrations carried out so far have been carried out with several suppliers at once. The fact that distributors carry out demonstrations with different suppliers is in itself a valid reason to work on interoperability, since, if they finally choose a specific solution, they will be able to continue to take advantage of the applications they already had on other platforms, which guarantees greater flexibility and avoids the fear of “vendor lock-in".

Motivation

The focus of the project consists of studying interoperability based on the idea that any developed application can work on any platform, regardless of the provider.

The key challenge of SG lies in the integration of components, systems, information, and applications. Interfaces and functionalities must ensure interoperability to enable high-level processes. Connecting all the components of an electrical network gives rise to an interconnected network in which the flow of information and its analysis will be carried out in real time. [9]

The "Cross Platform Access" pattern [11], represented in ***Figure 9***, is essential in the pursuit of interoperability in an IoT context. This pattern refers to the ability to hide differences and complexities between different platforms, thus allowing an application or service to access features and resources in a transparent and consistent manner.

The main objective of this pattern is to provide a common interface for different applications or services to interact with any IoT platform. In this way, allow applications to connect and use resources from different platforms through a single interface specification, instead of having to develop platform-specific interfaces.

To achieve that goal, it is necessary to address the challenge that platforms, which can come from different vendors, must use common interfaces and data formats to ensure smooth communication. This involves establishing common standards and agreements between platform providers, so that platforms can understand each other and share data between them consistently. [12]

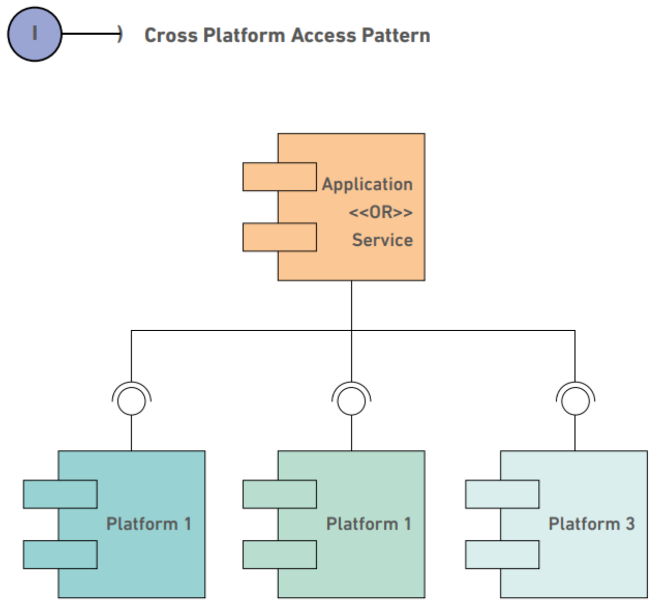


Figure 9: Cross Platform Access Pattern [11]

WoT provides an architecture and a set of standards that allow the integration and interaction of the various devices and services present on the web in a transparent way. Its adoption plays an important role in the search for interoperability in the context of applications deployed on edge nodes.

The application of WoT principles in IoT platforms promotes the use of standardized interfaces and protocols, simplifying the development and deployment of interoperable applications. In addition, it allows the exposure of devices as web resources, facilitating their discovery and access by other applications.

With the possibility of deploying applications on different platforms without having to make major modifications, greater flexibility and reuse of applications is achieved. This promotes innovation by allowing different actors to contribute their solutions without being limited by compatibility restrictions.

As mentioned previously, in the absence of effective implementation of node management platforms by utilities, this motivates to prioritize interoperability to take advantage of applications from various platforms and avoid exclusive dependence on one provider.

It is important to keep in mind that, in many cases, the developer of an application is not usually the same provider of the edge node. It can be a company specialized in software that is responsible for creating interoperable applications for different hardware vendors. Therefore, interoperability is also of interest to application developers, as it allows them to create solutions that are compatible with a wide range of edge nodes.

In addition, it is common for utilities to request the development of multiple alternatives for specific components, such as smart meters or data concentrators, from different manufacturers. This practice is carried out to mitigate risks, such as the bankruptcy of a manufacturer. In this sense, the advantage of focusing on software-based solutions is that they are more economical compared to hardware and software development combined.

Objectives

The main objectives to achieve along this project are the following:

- Study the role of application management platforms and Edge nodes to fulfill the vision of SGs.
- Study Application Virtualization as an interoperability driver.
- Research on the standards defined by WoT to achieve interoperability of applications.
- To develop a specification of a Thing description to enable interoperability of applications in the SS domain.

RELEVANT THEORY

As a starting point, sections of relevant theory to the realization of this work are included. All the concepts will be landed especially for this work in the **ANALYSIS** section.

Intelligent Secondary Substation (ISS)

Introducing the ISS [23] will allow to appreciate the benefits and potential of smart solutions in the transition towards a smarter and more sustainable electricity grid. The ISS concept leverages emerging technologies such as automation, IoT, Data Analytics and AI.

The new ISS must respond to the future needs of LV network management in a scenario of DER, mainly DG, EVs, ADM and ESS, integrated into the grid. It is assumed that the control capacity of the DERs can be managed and coordinated to avoid saturations in the network and manage current assets in an efficient way.

The levers that drive towards the transformation of low voltage management are mainly three, as represented in ***Figure 10***:

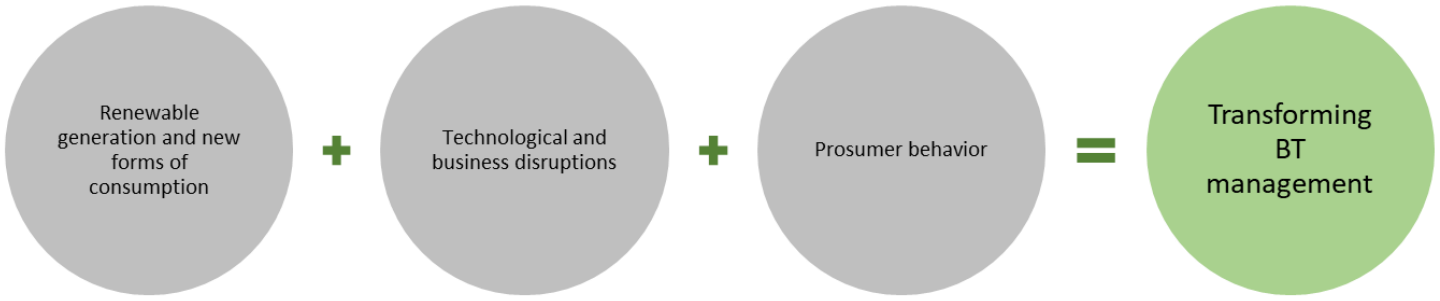

Renewable generation and new forms of consumption
+
Technological and business disruptions
+
Prosumer behavior
=
Transforming BT management

Figure 10: Levers for the transformation of low voltage management

- **Renewable generation and new ways of consumption**. Related to the awareness of climate change and the need to reduce CO_2 emissions, the generation of energy from renewable sources arises. Decentralized generation also emerges, allowing users to produce their own energy, in addition to the possibility of selling the surplus to the grid. New ways of consumption, such as the use of intelligent vehicles, or efficient appliances, drive towards the need for smarter management in the low voltage network.

- **Technological and business disruptions**. The emergence of disruptive technologies such as EVs, flexibility and microgrids, causes the way the network is managed and operated to change. Smart devices, such as demand controls and storage systems, appear to cope with energy flexibility. Therefore, the need arises to take advantage of these technologies and adapt to the new business models that appear as a result of these disruptions.

- **Prosumer behavior**. The behavior of the prosumer in this new situation, being able to act as a consumer and producer, implies greater interaction on its part with the electricity grid and having the ability to contribute to the surplus produced, in addition to adjusting its consumption according to the needs of the market. Therefore, the need for communication and coordination with prosumers arises to allow their active participation in energy management and facilitate the integration of distributed generation into the grid.

The ISS can be defined as an agile and flexible platform that, in real time, can detect and address risk situations in the electricity supply, ensuring its continuity and quality. [31]

After having explained the levers towards the transformation of low voltage management, challenges arise in low voltage, which are grouped in ***Table 1***, in addition to the ISS response to them.

Table 1: LV challenges and ISS responses

LV Challenge	ISS Response
Need for monitoring and integration of distributed resources within the network operation	DER management systems leveraging Edge capabilities
Need for operation closer to design limits	Resolution at the Edge
New physical and cyber security threats	Implementation of artificial vision algorithms in SS, generating intrusion alarms, and implementing cybersecurity measures
Need for agility in locating incidents in LV and restoring service	Real-time management
Limited physical space where the switchgear has risen gradually in recent years	Simplification of switchgear by incorporating the hub node and the LV supervisor
Efficiency losses due to load imbalance	The node achieves efficient load balancing management, allowing balances to be applied
Difficulties in forecasting flows in the network, which makes it difficult to plan the operation	ISS enables accurate planning and more efficient response times
Risk of quality deterioration and loss of control in voltage profiles.	Increased quality by performing operations in real time
Pressure on OPEX in a regulatory framework that only encourages investment.	Possibility of remote maintenance and automatic deployment that achieve savings in OPEX

The following table, ***Table 2***, outlines the key ISS elements which enhance the overall functionality and communication within the power distribution network, categorized into MT, BT, auxiliary devices, and communications.

Table 2: ISS elements in MT, BT and auxiliary devices

MV	Remote Control Unit (RTU) Fault Step Detector (FSD) Transformer on-load tap-changer Transformer MV Protection MV Line Protection MT Broadband PLC Modem (MV B-PLC)
LV	Measurement Data Concentrator Basic BT Supervisor BT Advanced Supervisor LV Frame Automation
Auxiliaries and communications	Power supply/battery charger Surveillance systems Cellular/xDSL/wireless/FO router Modem PLC BT Weather station. Various sensors of humidity, presence, interior temperature etc.

ISS structure is represented in ***Figure 11***, and main elements are later described.

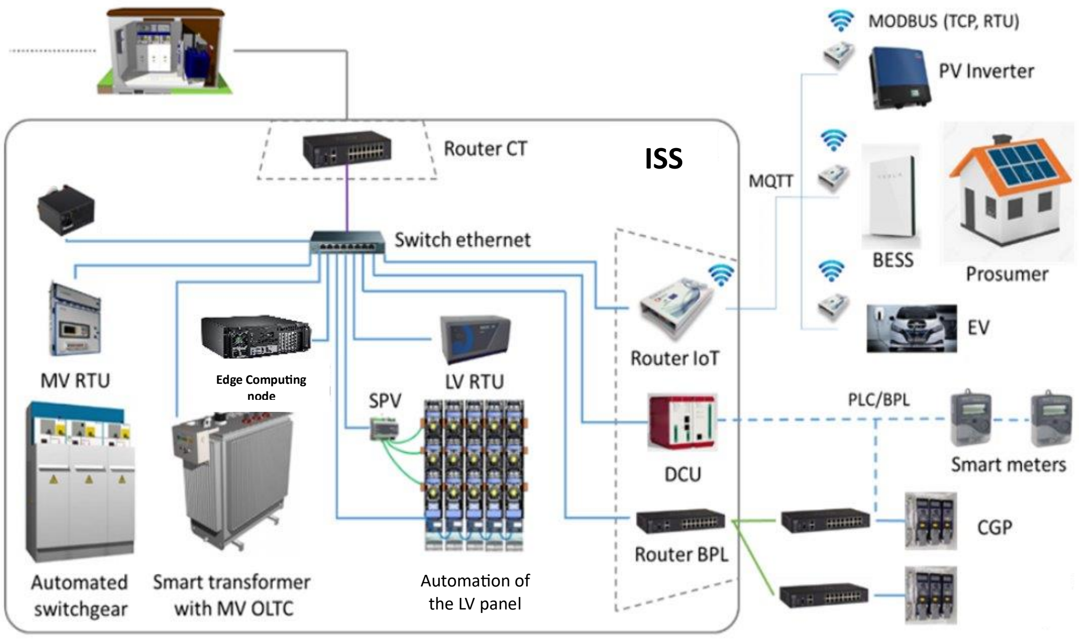


Figure 11: ISS Structure

Next, some of the main elements of the MV part in an ISS are described:

- **MV RTU**. It is a component strategically placed in the MV of the distribution network, and whose function is to collect real-time data on various variables, such as intensity, voltage, transformer temperature, in addition to also monitoring the performance of the transformer and thus detect anomalies. The MV RTU processes the collected data and transmits it to centralized control systems, such as SCADA, which allows operators to monitor and manage the power grid efficiently and quickly. It can also receive control commands from the central system, allowing adjustments to the transformer operation to be made remotely.

- **FSD in MV**. This device acts as a protective barrier against situations of overload, short circuit or network failures. It monitors currents and voltages in the transformer windings constantly, and if it detects an abnormal condition, it can take protective measures, such as disconnecting the transformer from the grid to prevent further damage or interruptions in the power supply.

- **Transformer on-load tap-changer**. This mechanism allows to adjust the voltage to the output of the transformer while it is operating. When there are variations in power demand, the on-load tap-changer can change the transformer ratio to keep the output voltage within proper limits.

- **Broadband PLC modem in MT (MV B-PLC)**. This component uses MV power lines as a means of communication to transmit data at high speed. It uses PLC technology to send information between devices and control systems. In this way a reliable and fast communication network can be established between different points of the electrical network. This technology enables data transmission for real-time monitoring, remote control, and efficient management of the power grid. It can be used to send data to and from the MV RTU, DPF and other devices.

Next, we are going to focus on LV elements.

- **Measurement data concentrator**. This device is crucial in collecting and storing information about power consumption on a LV ISS. It is responsible for centralizing data from multiple counters and meters located on the ISS. By collecting this data, the concentrator provides a detailed view of the behavior of the electrical grid in the LV, being important in planning, optimization, and decision-making on energy management.

- **Basic LV supervisor**. It has a very important role since it is responsible for ensuring the stability and security of the electrical network in the ISS. Although its capabilities are more limited compared to more advanced systems, this component constantly monitors the overall health of the network and can detect overload conditions, short circuits or other abnormal situations that may affect operation. In response, automatic protective actions come into play, such as disconnecting network segments to prevent damage and outages.

- **Advanced LV Supervisor**. In addition to the basic monitoring functions, this component uses more complex algorithms for in-depth analysis of measurement data, being able to identify consumption patterns over time, detect unusual behaviors that could mean problems in the network and predict future load conditions. For these reasons, it makes it possible to make more informed decisions regarding network management.

- **Automation of the LV panel**. It incorporates automatic devices and systems in the LV electrical panel of the ISS. It includes the installation of circuit breakers and protection relays that allow to respond quickly and accurately to the signals emitted by supervisors and centralized control systems. Automation can enable automatic reset after a temporary disconnection, reducing service interruption time.

In addition, there are other very important elements involved in an ISS.

- **Edge Computing Node**. It refers to a device that performs data processing and analysis close to the data source, rather than sending all the data to a central cloud to be processed. In the context of an ISS, an edge computing node could be a device located in the CTI that performs analysis and preliminary processing of the data collected from the devices and sensors in real time, allowing faster and more efficient decisions to be made in place, without relying on the latency of communication with a distant cloud.

- **DCU**. It is a control and communication unit that collects data and controls devices at the site of their installation. It is designed to supervise and control operations within the ISS, being able to interact with other external monitoring and control systems. In some cases, an edge computing node can be incorporated into a DCU to perform local data analysis before transmitting information to centralized monitoring or control systems.

- **Power supply and battery charger**. These are essential components for maintaining continuous operation and power availability on the ISS.

- **Surveillance systems**. They consist of cameras and sensors that constantly monitor the ISS environment, helping to detect intruders, prevent theft, and provide visual evidence in case of them.

- **Cellular/xDSL/wireless/FO router**. Routers are devices that allow network connections to be established inside and outside the ISS. These routers facilitate the communication of the systems within the ISS with remote control systems or external monitoring centers, allowing a more efficient management and supervision of the electrical infrastructure.

  These routers can be replaced by IoT Routers [46], for several reasons derived from the use of IoT in SGs, which will be discussed in more depth in later sections of this work.

  - In terms of connectivity, IoT routers are designed specifically for IoT-related devices and applications. These IoT devices are typically sensors, actuators, and other components that collect and transmit data in an automated environment.

  - IoT routers are prepared to handle large volumes of connected devices in an IoT network, achieving efficient management of low-power connections, ability to process amounts of data quickly and connect to multiple communication protocols used in IoT devices.

  - Since IoT devices can present security challenges, IoT routers are typically equipped with specific security features to protect connected devices and transmitted data.

  - IoT routers can offer centralized management tools and platforms to monitor and manage connected devices in real time, and thus have complete control over the network of connected devices.

  - In addition, IoT routers are designed to operate efficiently in terms of power consumption, as many IoT devices run on batteries or limited power sources.

- **LV PLC Modem**: Allows communication through LV power lines, establishing connection between different components within the ISS, such as supervisors, protection devices and automation systems.

- **Weather stations**. It collects data on weather conditions at the ISS location, measuring parameters such as temperature, humidity, wind speed and direction, or precipitation. This data is valuable for understanding how weather conditions can affect the operation of electrical infrastructure. In addition, they allow preventive measures to be taken in case of adverse weather conditions.

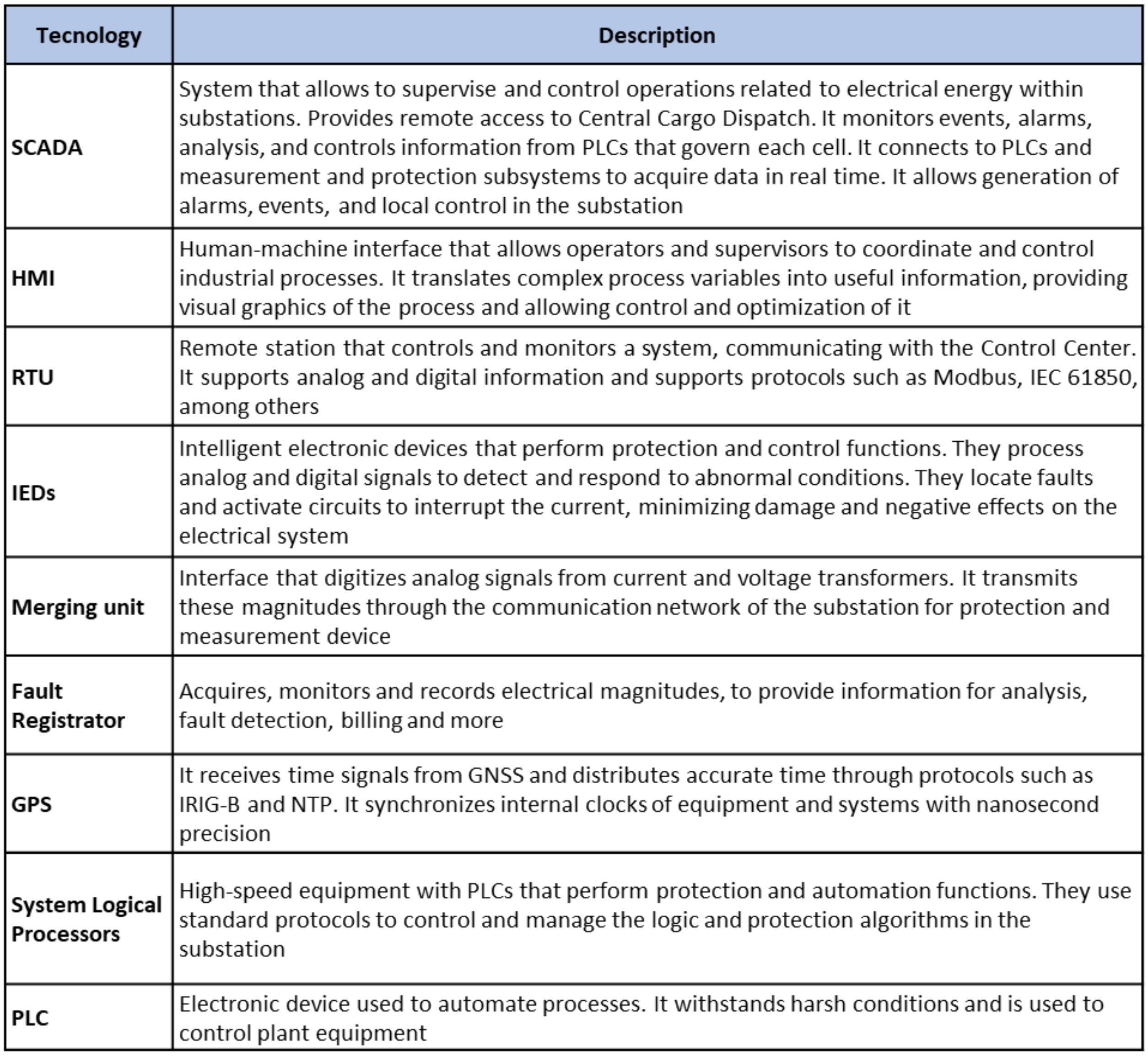
- **Miscellaneous sensors**. Humidity, temperature, presence, and other sensors are included to monitor aspects of the environment in and around the ISS.

Primary Substation (PS)

While the work is primarily focused on the Secondary Substation (SS), the significance of virtualization, as will be seen later, is of crucial importance in the domain of the primary substation (PS) as well.

Utilities are currently researching and implementing various SG enabling technologies with the idea of improving energy system efficiency and utilization. These technologies are constantly used in a substation environment. The following table, ***Table 3***, presents these technologies along with a brief description of them.

Table 3: SG Enabling technologies



Tecnology	Description
SCADA	System that allows to supervise and control operations related to electrical energy within substations. Provides remote access to Central Cargo Dispatch. It monitors events, alarms, analysis, and controls information from PLCs that govern each cell. It connects to PLCs and measurement and protection subsystems to acquire data in real time. It allows generation of alarms, events, and local control in the substation
HMI	Human-machine interface that allows operators and supervisors to coordinate and control industrial processes. It translates complex process variables into useful information, providing visual graphics of the process and allowing control and optimization of it
RTU	Remote station that controls and monitors a system, communicating with the Control Center. It supports analog and digital information and supports protocols such as Modbus, IEC 61850, among others
IEDs	Intelligent electronic devices that perform protection and control functions. They process analog and digital signals to detect and respond to abnormal conditions. They locate faults and activate circuits to interrupt the current, minimizing damage and negative effects on the electrical system
Merging unit	Interface that digitizes analog signals from current and voltage transformers. It transmits these magnitudes through the communication network of the substation for protection and measurement device
Fault Registrator	Acquires, monitors and records electrical magnitudes, to provide information for analysis, fault detection, billing and more
GPS	It receives time signals from GNSS and distributes accurate time through protocols such as IRIG-B and NTP. It synchronizes internal clocks of equipment and systems with nanosecond precision
System Logical Processors	High-speed equipment with PLCs that perform protection and automation functions. They use standard protocols to control and manage the logic and protection algorithms in the substation
PLC	Electronic device used to automate processes. It withstands harsh conditions and is used to control plant equipment

Authentication servers	They provide authentication and authorization to devices in a substation
Data Concentrators	They collect data from IEDs at the substation and communicate it to enterprise-level systems, enabling analysis, performance monitoring, fault diagnosis, and more
Synchrophasor Measurement System	Visualize and analyze data from PDCs in real time and historical. It stores past events and provides trends
Network Management Systems (NMS)	Monitor and control managed devices using SNMP. They are used for devices such as switches
Power Quality Management	Records, detects anomalies and analyzes trends of electrical variables in real time and historically. It is used to optimize the network and ensure regulatory compliance
Engineering Workstations	Data records for configuration, firmware, adjustments and documentation necessary to operate and maintain systems in the substation
Security and Surveillance System	It provides access control, fire detection, security systems and surveillance. Accessible via a web browser or thin client on a secure VLAN
Substation Gateway	It converts protocols and collects measurements, status, and event data from devices at the substation. It provides tunneling and advanced automation services. Communicate locally or remotely with a SCADA master station

In an electrical substation, the automation and control infrastructure of the substation is divided into levels. This is represented in ***Figure 12***. Each level has a specific purpose and relates to the functions and operating hierarchy in the substation and the electrical system as a whole.

- **Level 0: Intelligent Switchyard, Sensors, I/O**

  This level is the lowest in the hierarchy and is in the physical environment of the substation, that is, closest to the actual components of the electrical system. It includes equipment and devices directly related to the process of generation, transmission, and distribution of electrical energy, such as switches, transformers, current and voltage measurement sensors, as well as I/O that captures process data.

- **Level 1: Bays IEDs**

  At this level, there are the IEDs installed in each bay of the substation, responsible for controlling and supervising the equipment in each bay, such as switches, transformers, and other devices. They collect real-time information from sensors and act to ensure the safe and efficient operation of the substation. In addition, these IEDs usually communicate with the higher level, or Level 2.

- **Level 2: SAS**

  At this level, the SAS oversees supervising and controlling the substation bays, collecting and processing data from Level 1 IEDs and managing operations in real time. It also performs energy control and monitoring functions in the substation. In addition, it facilitates communication with Level 3.

- **Level 3: Dispatch Center**

  This is the top level in the hierarchy and is in the control and dispatch center of the electrical network. Multiple substations are monitored and controlled, and strategic decisions are made to ensure the supply of electrical energy throughout the grid.

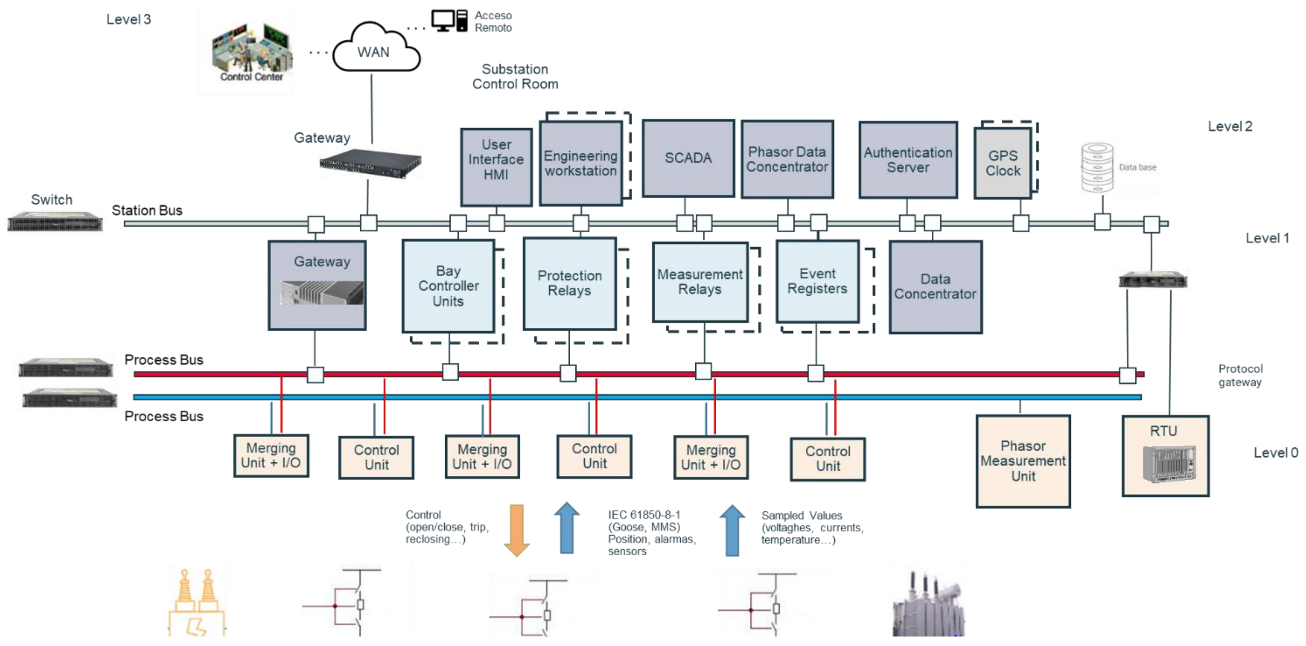


Figure 12: Schematic of a SAS

There are hundreds of devices in a substation. For the substation to function properly, they must all be securely installed, connected to each other via correct links, and have well-defined data transfer schemes. Each device can come from a different manufacturer, with different requirements on power supplies, mechanical construction, or data connectors, among others.

ISS & PS enabling technologies

Due to the growing sensorization and monitoring at all levels of the network, and therefore the increase in the amount of information available by several orders of magnitude, the need arises for greater computing capabilities and deployment of more advanced control algorithms, which in some cases are deployed on distributed platforms. These systems result from the convergence of 3 groups of technologies:

- **IoT**. It allows the management of millions of devices and their information flows in a scalable and cost-effective way.
- **OT**. It consists of providing direct integration and control capabilities of electrical equipment and energy management devices in real time.
- **Virtualization and CaaS management**, allowing:
  - Efficient encapsulation of management, analysis, and control functions.
  - An open and secure ecosystem on which to extend functionality.
  - Centralized management, maintenance, and deployment platform for distributed compute nodes.

Virtualization [47] emerges as an essential component in the network architecture for IoT-enabled SGs. This technique provides a dynamic and efficient solution to address the complex challenges that arise in the management and operation of modern power grids. By virtualizing network resources and services, a flexible and scalable environment is established that optimizes the infrastructure, facilitating network administration and ensuring adaptability to changes and emerging demands.

The biggest benefit of virtualization lies in the elimination of many different hardware devices from multiple vendors. In addition, the power consumption of a pair of redundant virtual servers is much lower than that of many separate devices they replace.

In this context, advanced analytics, AI or ML are easy to deploy and update at the edge, without the need for significant rebuilding of the network infrastructure. This added functionality of software capabilities enables a new era of predictive and prescriptive analytics that does not exist in traditional substation infrastructure.

To achieve this, it is necessary to build:

- A platform as a virtualized platform, so that it abstracts from the hardware. In case of hardware failures, or better hardware is available, the platform should migrate to this new hardware with ease.
- Containerized applications.
- Ability to deliver and update an application or collection of applications to one or groups of substations from the centralized management station.

In this context, virtualization in SGs with IoT goes beyond the consolidation of physical resources but allows the creation of virtual environments that simulate hardware and software resources, resulting in greater agility and efficiency in the management of the electrical network. Key aspects of virtualization in this environment are summarized in ***Figure 13***.

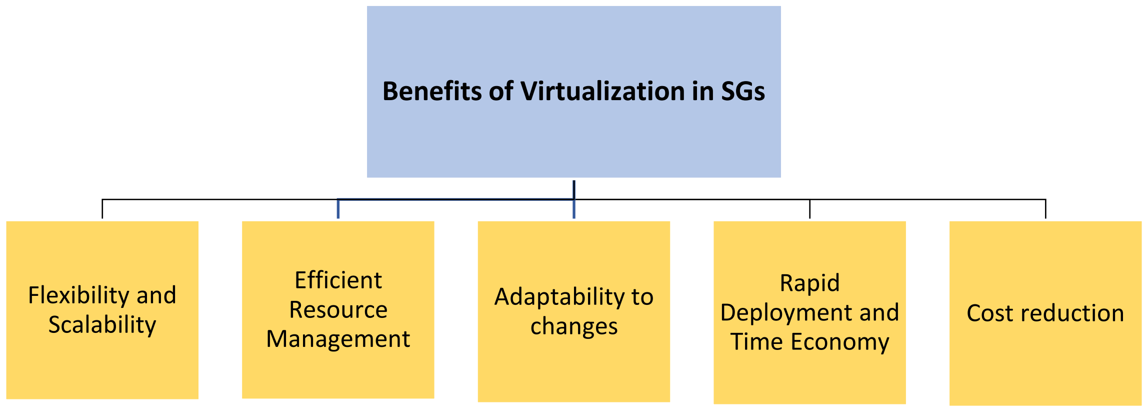

Benefits of Virtualization in SGs
Flexibility and Scalability
Efficient Resource Management
Adaptability to changes
Rapid Deployment and Time Economy
Cost reduction

Figure 13: Key aspects of virtualizations in SGs

- **Flexibility and Scalability**

  Virtualization makes it possible to create multiple virtual instances from a physical infrastructure. This translates into the ability to easily deploy and manage new IoT devices on the power grid, without the need to make large investments in physical hardware. The flexibility inherent in this technique allows you to dynamically adjust the resources allocated to each virtual instance based on changing network needs.

- **Efficient Resource Management**

  In a SG where optimizing energy consumption and distribution are crucial, virtualization allows resources to be allocated according to current demand, minimizing waste, and ensuring that resources are available where and when they are needed.

- **Adaptability to changes**

  Virtualization provides the ability to quickly adapt to changing SG conditions, dynamically scaling virtual instances to ensure a reliable power supply.

- **Rapid deployment and time economy**

  Virtualization significantly reduces the time to deploy new devices in the SG. Traditionally, physical hardware must be acquired, configured, and deployed. However, virtual devices can be created and put into operation quickly.

- **Cost reduction**

  By minimizing investment in physical hardware, operating costs, and optimizing resource utilization, virtualization brings significant economic value.

IoT

The current focus of IoT research [48] is on enabling common objects to be able to perceive the physical world on their own, interact with it, and share those observations with others. This changes to the vision that supervision and decision-making can increasingly be delegated to the machine, freeing up humans for more strategic tasks.

IoT architecture is described in ***Figure 14*** below:

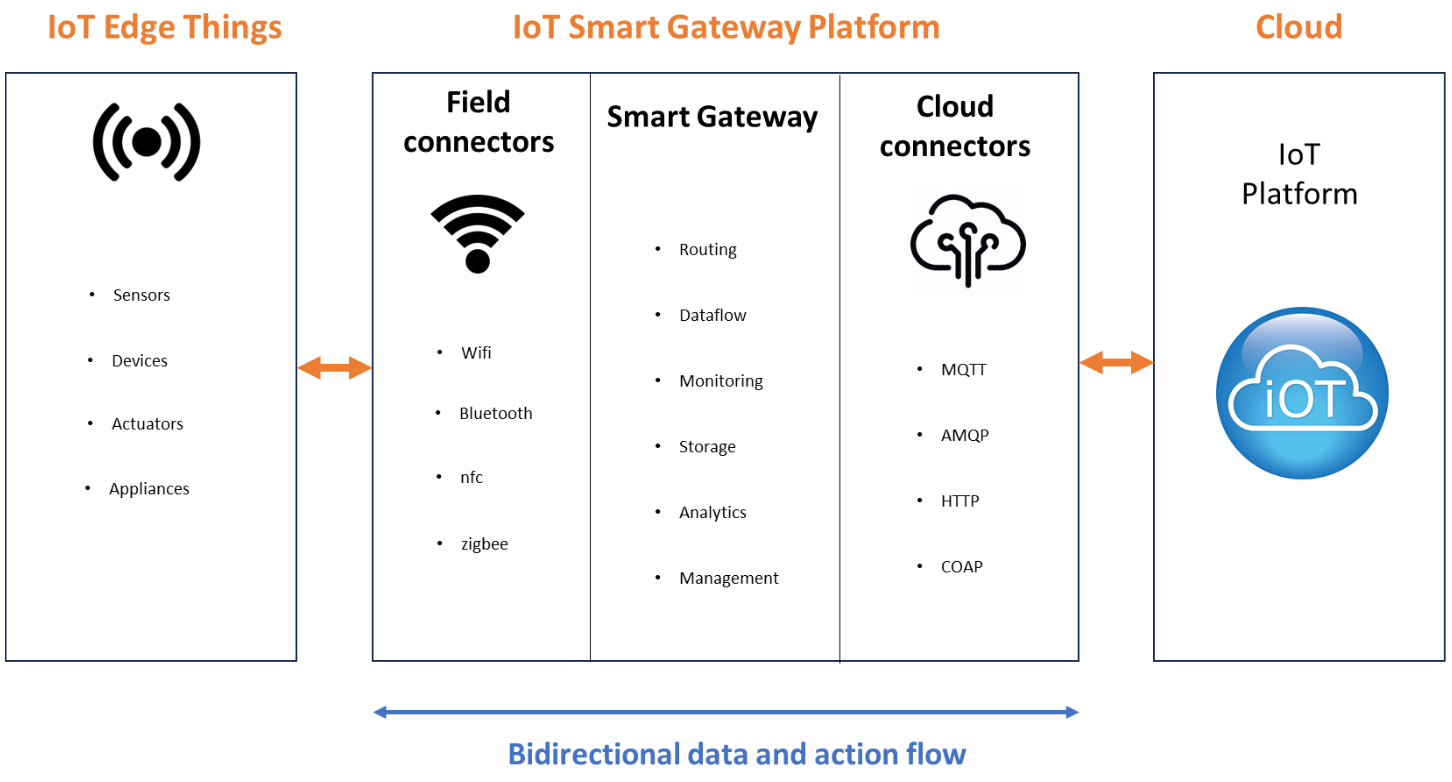


Figure 14: IoT Architecture

- **Edge Things**

  At the edge of the network are what are called "things". These can be sensors, and actuators, among others. The term Edge refers to the edge of the network, as close as possible to its origin. The importance of processing at the edge lies in executing data processing at this point and making decisions in real time.

- **Field Protocols**

  Since the so-called "things" are located at the edge, they must communicate with each other and with the Gateway that will be described below. These communications are based on field protocols, such as Bluetooth [49], Zigbee [50], WiFi [51] and NFC [52].

- **Smart Gateway de IoT**

  The key function of the IoT Gateway is to enable communication from the edge to the cloud, in addition to having capabilities for routing, data flow, management, monitoring and storage of data. The gateway establishes communications between the "things" described above and the cloud services, in addition to managing their interactions. You need to understand field protocols and turn them into cloud protocols.

- **Cloud Protocols**

  Most IoT solutions, even those at the edge, need to integrate with cloud services. For this, cloud protocols such as MQTT, AMQP, CoAP and HTTP are used, which are briefly described below.

  - **MQTT.** It is an efficient communication channel that connects IoT devices with cloud services. It resembles a message line in which devices can post messages to the cloud, and devices and cloud services can subscribe to them, view them, and respond as needed. MQTT is useful when multiple devices require seamless communication both with each other and with the cloud.

  - **AMQP**. Each device can leave messages in a kind of virtual room, and other devices can access it, read them, and contribute their information. It is effective in cases where organized and structured communication between many devices is needed.

  - **CoAP**. It allows devices to transmit messages to the cloud and receive corresponding responses to them. It is useful for smaller devices that need to maintain communication with the cloud, but do not have the ability to handle large volumes of data.

  - **HTTP**. It is widely used in web browsing, but also between IoT devices and the cloud. Devices query and receive responses from cloud services. This protocol is used especially when occasional and direct communication with cloud services is required.

<u>Advantages and Disadvantages of IoT</u>

First, the advantages of IoT include the following:

- **Improves connectivity**. IoT enables the interconnection of devices and objects in a network, allowing real-time remote access and control from any location.

- **Enables efficient Automation**. IoT devices allow automating repetitive tasks and complex processes, saving time and increasing efficiency.

- **Improves decision Making**. By collecting and analyzing data in real time, it facilitates informed and accurate decision making.

- **Optimizes resources**. The IoT allows a more efficient use of resources in the field it is applied, such as energy, water, and raw materials, contributing to sustainability.

- **Allows remote monitoring and maintenance**. Allows remote monitoring of devices and systems. This simplifies preventive maintenance and troubleshooting.

IoT also has a number of disadvantages, explained below.

- **Presents security and privacy risks**. The interconnection of devices increases the risk of cyberattacks and the exposure of personal data. This creates security and privacy concerns.

- **Depends on connectivity**. The IoT depends on a stable Internet connection, allowing problems to be caused if network outages occur.

- **High initial investment**. The implementation of IoT solutions may require significant investments in devices and infrastructure, which can be expensive.

- **Lack of compatibility and standards**. The lack of uniform standards can hinder interoperability between devices and systems from different manufacturers.

Table 4 below provides a summary of the sales and disadvantages of IoT discussed above.

Table 4: Summary of the advantages and disadvantages of IoT

IoT Advantages	IoT Disadvantages
Improved Connectivity	Lack of Security and Privacy
Efficient Automation	Dependency on Connectivity
Improved Decision Making	Start-up costs
Resource Optimization	Lack of Compatibility and Standards
Remote Monitoring and Maintenance	

Cloud computing and Edge computing

Cloud computing is an innovative technology that has revolutionized the way we interact with information and online services. An example of this is SaaS.

SaaS are computer tools offered to users, over the Internet and without having to download and install software on devices. One example is Google Apps, which includes tools such as Gmail, Google Drive or Google Docs. Cloud computing allows these applications to run on remote servers, being able to access them from any device that has an Internet connection, and without the need to install software locally.

IoT interconnects everyday devices and objects, generating a large amount of data in our daily lives. This vision leads to the cloud, where objects generate and share information in real time. Many applications can be deployed at the edge of the network to process this data quickly and efficiently.

Cloud computing [55] offers a centralized and scalable infrastructure for storing and processing data. The schema of Cloud Computing is shown in ***Figure 15***, representing data producers transmitting data to the cloud, and from the other side, data users sending requests of data to the cloud, that send results to that request back to them. It is very beneficial for applications and services that require intensive processing and scalable resources, as well as when you need to access data and applications from different devices and locations, allowing for smoother and more accessible collaboration.

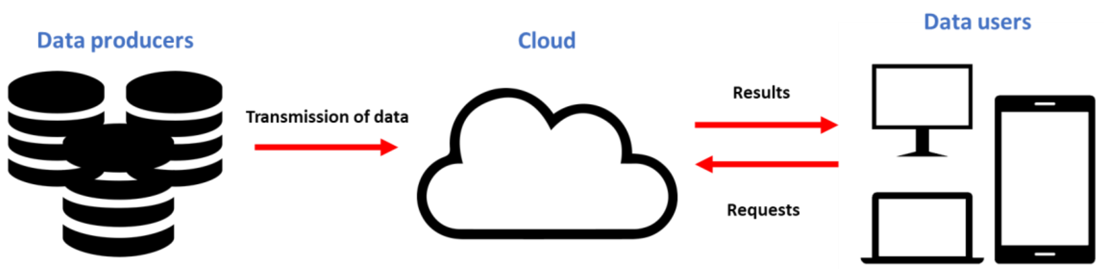


Figure 15: Cloud Computing

With a large number of IoT devices connected to the Internet, there is a landscape in which devices constantly interact and generate data. In this context, the notion of processing engines and scalable infrastructure technologies is crucial. The need to process and analyze these large volumes of data generated becomes essential to which **Edge computing** [55] emerges as a solution.

Centralizing cloud computing tasks is efficient for processing data, as the computing power in the cloud exceeds the capacity of devices at the edge. In contrast, network bandwidth has reached a point of stagnation, and with the increasing amount of data generated at the edge, the speed of data transport becomes the bottleneck for the cloud-based computing paradigm.

In the case of sending all the generated data to the cloud for processing, the response time would be very long. For this reason, data must be processed at the edge for shorter response time, more efficient processing, and less pressure on the network.

The Edge computing structure features a bidirectional flow of computation, as shown in ***Figure 16***. Devices are both consumers and producers of data. At the edge, devices request services and content from the cloud and perform compute tasks from the cloud. The edge can perform compute offloading, data storage, caching, and processing. It is also capable of distributing requests and providing services from the cloud to the user. The edge must be well designed to efficiently meet service requirements such as reliability, security, and privacy protection.

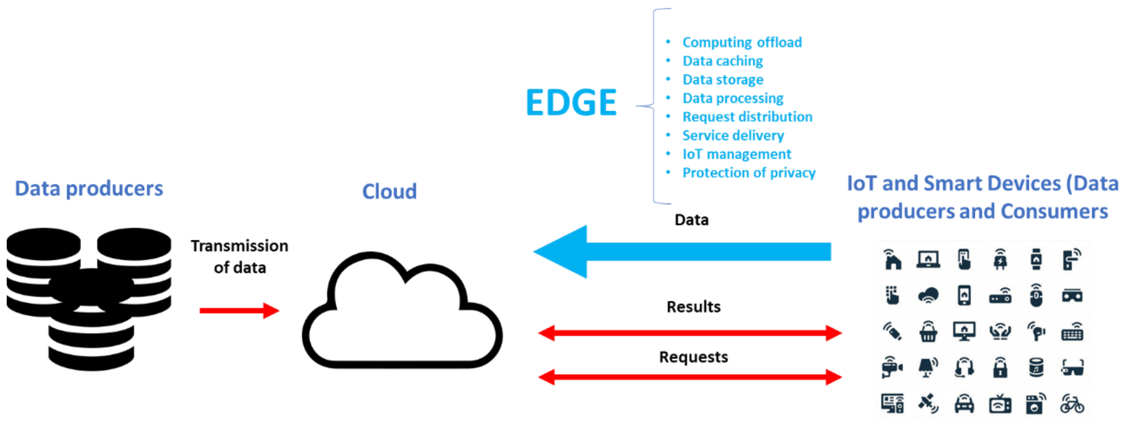


Figure 16: Edge Computing

There are several applications where edge computing is an optimal option. These include programmability, nomenclature, data abstraction, service management, privacy and security, and optimization metrics.

- **Programmability.** In cloud computing, users write programs and they run in the cloud. In contrast, in edge computing, processing is done on nearby devices, instead of in a central location. The devices at the edge are varied and therefore have different rates of execution, which makes it more complicated to write programs that work well for everyone.

  To address this problem, computational flow is proposed. Instead of performing all computing tasks in one place as is the case with the cloud, in the compute flow, that work is broken down into smaller parts and distributed in different places, such as devices at the edge and in the cloud. The main idea is that each part of the calculation process is done where it is most efficient and has less delay, avoiding sending all the data to one place to process it.

- **Nomenclature.** In edge computing there are a lot of "things", placed everywhere. On top of these edge devices, there are many applications working, each running its job. Calling those things correctly is very important for things to work properly, communicate and know where they are.

A way of naming things at the edge that is flexible and works well with the changing nature of these devices is required. IP addresses work well for many networks, but not for things at the edge that can move and have few resources. There are some new ways of naming things that could be employed, such as a so-called NDN, MobileFirst, or TDs.

- **NDN** [53] is especially useful in environments where data is in constant motion and where location can change, such as in edge computing. It is a communication model focused on naming data, rather than its locations. Each piece of information has a unique name, and devices can request that data without worrying about the physical location of the servers.

- **MobileFirst** [54] is useful for devices that can move around a lot, such as mobile devices on a network. This is an approximation that separates the name from the network address in communication. This allows greater mobility and flexibility in communication, adapting to the changing nature of devices in the edge computing environment.

- A TD is used to identify and manage devices on the network more efficiently. This is the structured description of a device in an IoT environment. It contains key information about the device, such as its name, location, features, and capabilities.

The system at the edge would give each thing a kind of identification number and a way to communicate, making it much easier to manage things.

- **Data Abstraction.** In the Edge computing system, several applications can run on the operating system used at the edge, both collecting data from devices and providing services by communicating through control signals.

  Edge computing allows edge nodes to process data and proactively communicate with users. This involves performing data preprocessing at the gateway, for noise removal, event detection, or privacy protection. However, there are several challenges related to this:

  - **There is a diversity of data**. Data from different devices can be in different formats, affecting their subsequent use. Applications at the gateway must extract relevant knowledge from a standardized data table.

  - **It is difficult to abstract the data**. Deciding how much processed information should be kept is difficult. Filtering too much can affect application learning, but maintaining data could present storage issues.

- **Service Management.** Four essential characteristics must be incorporated to ensure system reliability, which are differentiation, extensibility, isolation, and reliability.

  - **Differentiation**. With the increasing expansion of IoT, there is a need for deployment of services at the edge of the network. These services will have different priorities that must be respected.

  - **Extensibility**. Things in IoT are dynamic, so integrating new ones or replacing worn-out devices is straightforward. The design of the service management layer must be flexible and adaptable to handle these changes.

  - **Isolation**. Different applications may need to share data resources, and that in case of failure in the application this does not affect the entire system. It is also important to isolate the user's private data from third-party applications to protect security and privacy.

  - **Reliability**. The edge operating system must track the network topology and allow components to report their status to identify the causes of a failure.

- **Privacy and Security.** At the edge of the network, it is important to take care of data privacy and security. One solution would be to remove certain private details from the data before it is processed.

  Some challenges related to this include detecting unusual usage patterns, identifying attack attempts, or establishing authentication and authorization methods.

- **Optimization Metrics.** In Edge computing, there are many layers, each with different computing capabilities. Distributing tasks effectively is important, and for this there are assignment strategies for completing tasks. To make the optimal decision, metrics such as latency, bandwidth, power, and cost are considered.

  - **Latency**. Speed of response is very important, especially in interactive applications. To reduce it, the best solution is to finish the tasks in the nearest layer with sufficient capacity.

  - **Bandwidth**. High bandwidth speeds up transmission, especially in the case of large data. For short transmissions, high-speed wireless access can be established. If the edge handles the tasks, latency is improved, and bandwidth is saved.

  - **Energy Consumption**. Moving tasks to the edge saves energy, but the balance between computational and transmission consumption is important. Edge computing is preferred if transmission is less than calculations. For the entire edge process, the total energy cost includes layers, and the consumption of each layer is the sum of calculations and transmission.

  - **Cost**. Edge computing reduces latency and consumption, thereby improving the experience. Because of this, more revenue can be generated by handling the same amount of workload.

Hypervisor and container-based virtualization

Virtualization [56] is a technology that allows to create virtual versions of hardware and software resources, such as servers, networks, operating systems, or storage. These virtual versions run in isolation on the same physical hardware, which maximizes the use of available resources and optimizes the administration and flexibility of the systems. Application virtualization is a specific form of virtualization that focuses on encapsulating and running applications and their dependencies in isolated environments, separate from the host operating system. This will be explained later in this paper.

There are two important approaches to virtualization [56] are hypervisor-based virtualization and container-based virtualization.

First, hypervisor-based virtualization involves utilizing a software layer called a hypervisor, which acts as an intermediary between physical hardware and virtual machines, a schematic shown in ***Figure 17***. The instructions of the virtual machines are directed to the hardware through the hypervisor, which allocates the necessary resources. Each virtual machine behaves like a complete operating system, with its own kernel and isolated resources. This approach is secure and highly isolated but can lead to duplication of resources due to the presence of entire operating systems in each virtual machine.

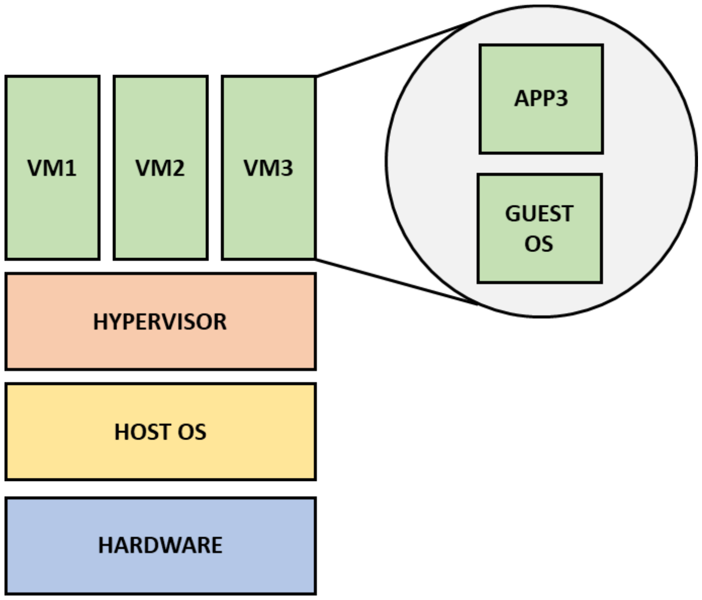


Figure 17: Virtualization structure based on hypervisors

As shown in the image, the hypervisor-based virtualization structure consists of:

- **Physical hardware**. It represents the physical components of the machine, such as the CPU, RAM, hard disk, and other peripherals. Virtual machines will be built and run on top of these resources. These virtual machines will use these resources to run and run applications. Virtualization allows you to optimize the use of these resources by effectively allocating them to virtual machines according to your needs, resulting in better hardware utilization and greater flexibility in system administration.

- **Host OS**. This operating system is the first to boot when the physical machine is started. It manages the physical resources of the hardware and provides essential services such as file management, device management, and memory allocation. In addition, it acts as an intermediary between the hardware and the virtual machines that run on it.

- **The next level is the Hypervisor**. This is a specialized software layer designed to create and manage multiple virtual machines. It ensures proper isolation and resource allocation for each virtual machine, allowing multiple operating systems and applications to coexist on a single physical server.

- **Virtual machine**. Each virtual machine is a virtualized, isolated instance of an entire operating system. Within each of them, a complete hardware architecture is simulated, allowing operating systems and applications to function as if they were on separate physical hardware.

  - **Guest OS**. Each virtual machine runs a guest operating system, which is separate and isolated from other operating systems on the same physical server. It provides a complete environment for applications to run and communicate, allowing applications that require different environments to coexist without interference.

  - **Applications.** Each virtual machine contains specific applications that run within the guest operating system. Virtualization allows applications in different virtual machines to operate independently, without affecting each other.

On the other hand, container-based virtualization offers a different approach, shown in ***Figure 18***. A container is an isolated, lightweight unit that contains an application along with its dependencies. Instead of running multiple full operating systems, containers share a single host operating system. This allows for a more efficient use of resources and greater agility in the deployment of applications. As a disadvantage, due to this greater sharing, containers are less isolated than virtual machines.

The ability to efficiently create, manage and deploy containers aligns perfectly with the challenges and opportunities present in IoT and Edge Computing. Container-based virtualization is an essential tool for optimizing resource management, improving performance, and enabling quick and easy deployment of applications.

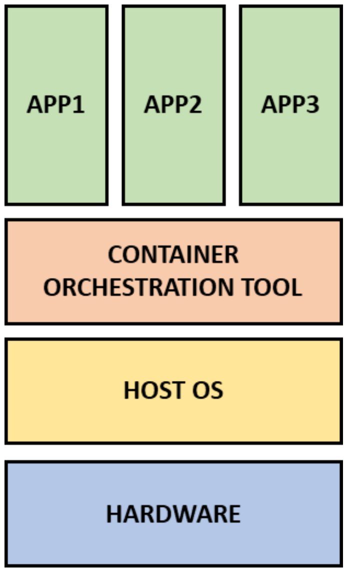


Figure 18: Virtualization structure based on containers

In a container-based virtualization framework like the one shown in the previous image, the key components are the Hardware, the Host OS, the container orchestration tool, and the applications.

- First, the **physical hardware** on which the containers will be built and run.

- Second, the **Host OS**, responsible for enabling containerization functionality, providing an environment in which containers can operate and isolate from each other.

- Next, the **container orchestration tool**. It is a crucial part of this virtualization framework and enables automated management and coordination of multiple containers. It can be Docker, Kubernetes, or another similar tool. Docker provides a platform for building, distributing, and running containers. Kubernetes, however, enables large-scale container orchestration, making it easier to manage applications and services in complex environments.

- Finally, **applications**. They are the programs and services that run inside the containers. Containers encapsulate an application along with all the libraries and dependencies needed for it to function independently and consistently in any environment. Each container hosts a specific application, allowing for high portability and scalability.

The choice between virtualization approaches will depend on factors such as resource efficiency, portability, scalability, and security requirements. Specific needs of the project must be considered before deciding. It will be studied in the **Application Virtualization** section of this work.

WoT

Normally in the world of IoT there is a heterogeneous technological landscape in which there are various IoT systems and devices from different manufacturers and suppliers. This diversity leads to variations in communication protocols, data models used in payload data exchange, and security requirements. IoT applications are usually developed for a specific use case. Traditionally, it is difficult to expand, maintain or reuse these applications throughout the life of these applications.

First, some cases are presented which are increasing in complexity. From them, the understanding and adoption of WoT [57] and its components is motivated, which are described below.

- **Telemetry.** It consists of the monitoring of remote devices, such as sensors that send data to be processed. This transmission can occur through different communication channels, such as GSM, Bluetooth, WiFi or Ethernet networks.

- **Device Controllers.** It is a use case where a local device is controlled by a remote controller, such as an electronic device managed by an application on a mobile. One device can act as a server and respond to commands, and the other as a client and send those commands.

- **From thing to thing.** These are cases where changes are detected, and messages are sent to another device to perform an action. In this case, both devices can act as both clients and servers in the interaction.

- **Edge Devices.** An Edge Device can perform local calculations and connect different protocols. It is used in industrial solutions to preprocess and filter data from connected devices.

- **Digital Twins.** Digital twins are virtual representations of devices in the cloud or on an edge device, which can be used to simulate applications and services prior to their deployment on real devices.

These examples described below, show how devices interact in various contexts. The WoT architecture [58] provides a flexible framework for these devices to communicate and collaborate more effectively and efficiently. By implementing the WoT architecture, greater interoperability, efficiency, and flexibility can be achieved in the design and operation of IoT systems.

The W3C [59], or World Wide Web Consortium, is an international organization dedicated to developing standards and technical protocols for the evolution and growth of the WWW. Its main objective is to ensure the interoperability and continuous evolution of the web, ensuring that web technologies are developed in a consistent manner and thus be used by anyone anywhere.

The W3C [59], through its Web of Things Working Group, leads the creation of standards and technologies that enable the principles of the web to be applied to the field of the IoT. The W3C work contributes to the creation of a more connected, accessible, and compatible WoT ecosystem with the global web.

WoT [57] consists of a set of standardized technological building blocks that help simplify the development of IoT applications, following the web paradigm. This approach increases interoperability and flexibility, especially for cross-domain applications (applications that are not limited to a single sector or application area, but cross different industries, devices, and services), and allows the reuse of established standards and tools. WoT unlocks business potential that is limited by fragmentation in the IoT.

WoT Architecture

WoT is fundamentally composed of three components [58]: Things, TDs, and Links.

- A **"Thing"** is an abstraction of a physical or virtual entity, which is described with standardized metadata. The Consumer is the entity that can interpret that standardized metadata to communicate with it.

  A Web Thing is composed of four architectural aspects of interest as depicted in ***Figure 19***, behavior, Interaction Affordances, security settings, and Protocol Bindings.

  The behavior of a Thing includes autonomous behavior and handlers for Interaction Affordances. Interaction Affordances provide the interaction model between consumers and the Thing through abstract operations, and without reference to a specific network protocol or data encoding. Finally, the protocol binding adds the additional detail needed to map each Interaction Affordance to specific messages from a specific protocol.

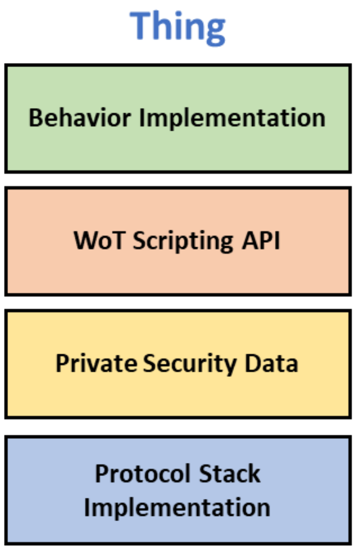


Figure 19: Architectural Aspects of a Thing

- Next, the **TD** is the standardized, machine-readable metadata format that allows Consumers to discover and interpret the capabilities of that Thing. TD components are shown in the following figure, ***Figure 20***, and described below.

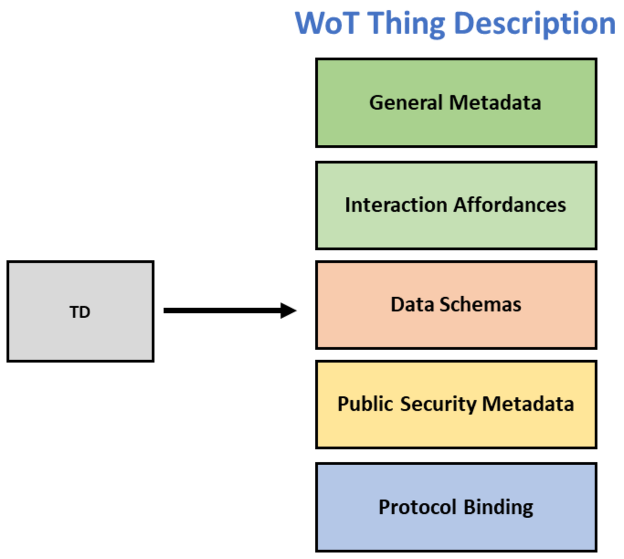


Figure 20: Thing Description components

TDs can be in a JSON format, or in a more advanced one called JSON-LD. Using JSON in TDs offers an efficient, readable, and widely supported way to describe device characteristics and properties in a format that is easy for both developers and machines to understand and process. For more information about JSON text format, see **ANNEX 1: JSON**.

The latter allows machines to better understand information and perform smarter tasks with it. These descriptions are unique to each Thing.

WoT introduces a simple interaction abstraction that is based on properties, events, and actions, as shown in ***Figure 21***. This abstraction can be used to describe any IoT network interface. Through it, applications have a common reference point for obtaining the metadata of an IoT service, as well as understanding how to access the data and functions of that IoT service.

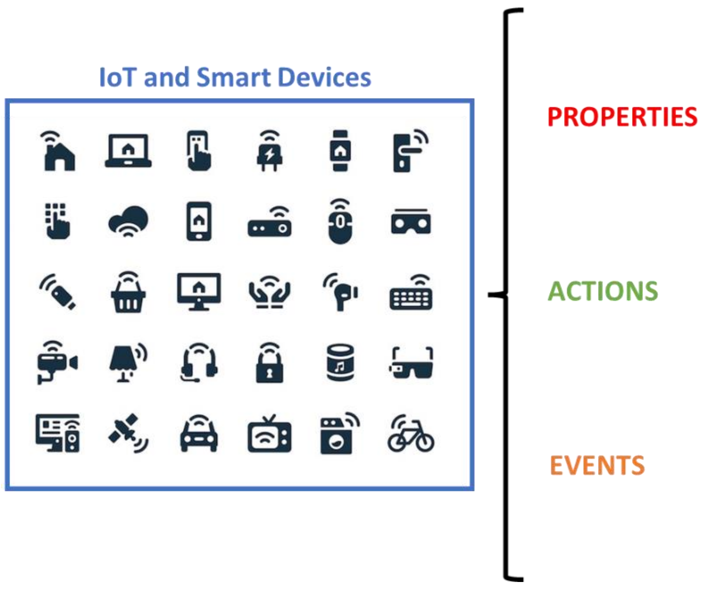


Figure 21: Abstraction based on properties, actions, and events

- **Properties** are facilities of interaction which expose the state of the Object. They can be read-only values, such as sensor values or calculation results, or read and write values such as stateful actuators or configuration parameters.

- **Actions** are facilities of interaction that invoke a function of the Object. This Action can be used to manipulate a state that is not directly exposed such as object properties. Invoking an Action can be used to trigger a process on the Object that manipulates its state over time through actuators. In case the data format is not specified by the Handshake employed, Actions may contain data schemas for input parameters and output results.

- **Events** are facilities of interaction which describe an event source that pushes data asynchronously from the Object to the Consumer. It is not the state that is communicated, but the state transitions. Events can be triggered due to conditions that are not exposed as Properties. If the data is not fully specified by the Handshake employed, the events can include data schemas for event data and subscription control messages.

The **Data Schemas** section defines how the data handled by the Thing is structured and presented. In case "Thing" measures temperature, this section would specify how to present this property in the description.

The **Public Security Metadata** section includes details about the security of the thing. This is information about how the Thing is protected against unauthorized access and malicious attacks.

Finally, Protocol Binding describes the way the thing communicates with other devices or systems through specific protocols. In case the thing used protocols such as HTTP, CoAP or MQTT to communicate, this would be detailed in this part.

- Finally, the **links** represent the relationships between Things and the models of Things, as shown in ***Figure 22***.

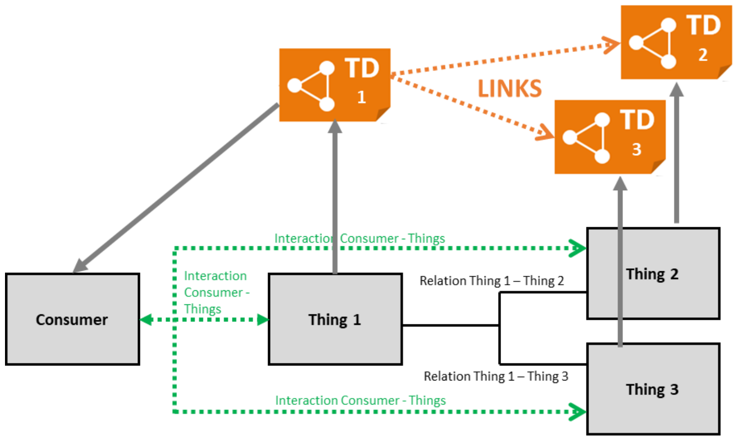

TD 1
TD 2
TD 3
LINKS
Interaction Consumer - Things
Interaction Consumer - Things
Interaction Consumer - Things
Relation Thing 1 – Thing 2
Relation Thing 1 – Thing 3
Consumer
Thing 1
Thing 2
Thing 3

Figure 22: WoT links

In addition to these concepts, in the architecture of WoT there are **intermediaries**. They act as mediators between devices and users, simplifying their interaction. These is represented in ***Figure 23***. They have their own description similar to the original devices but point to their own WoT interface. They can add features or combine devices to create a Virtual Thing.

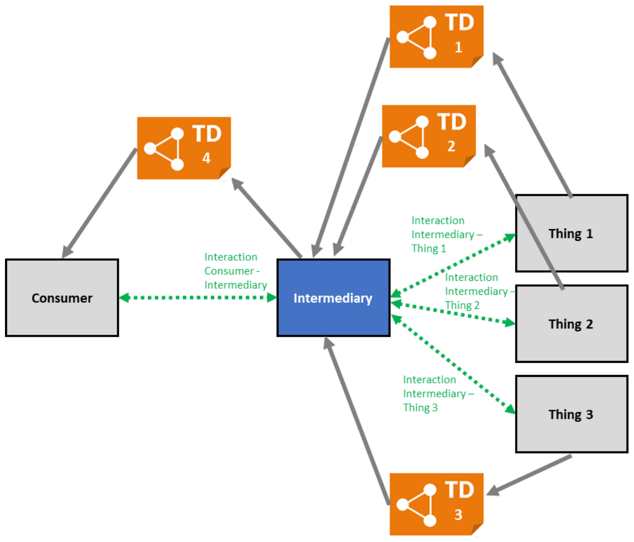

TD 1
TD 2
TD 3
TD 4
Interaction Consumer - Intermediary
Interaction Intermediary – Thing 1
Interaction Intermediary – Thing 2
Interaction Intermediary – Thing 3
Consumer
Intermediary
Thing 1
Thing 2
Thing 3

Figure 23: WoT intermediaries

Security Requirements in WoT

Security is an essential issue in WoT and its implementations. Although the abstract architecture presented alone cannot guarantee security, there are general guidelines to help secure specific WoT implementations.

First, it is important to protect TDs. When describing a device or service in WoT, only authorized people can access its description, so it is advisable to avoid mixing public and private information in the description and to keep private data separate from public data. When describing how a device communicates, it is important to follow best practices to configure the security of that communication. This refers to data encryption, authentication mechanisms, digital signatures, and access controls, among others.

When using scripts to make devices do things, it is important to make sure that those programs cannot be used to damage the system. That is, if there are several programs running at the same time, it is convenient to separate those that handle sensitive information. This reduces the risk of information being exposed if one of the programs is compromised.

If multiple devices rely on each other to communicate, make sure that this trust is necessary and not overshared. Therefore, it is necessary to maintain the relationships of trust as tight as possible. If devices are being used on a private network, consider using a separate network for IoT devices. This decreases the risk that someone from outside can access them. If you need your devices to communicate securely over the Internet, make sure your information is properly encrypted to protect it.

Privacy Requirements in WoT

Privacy is important in all WoT components and implementations. The information contained in a TD can be very sensitive.

TDs can potentially contain personally identifiable information of various types. To protect this information, it is necessary to limit the presence of personally identifiable information in TDs as much as possible. If it is necessary to include this type of information in the TD, it must be stored and transmitted securely, provided only to authorized users, cached for limited times, deleted upon request, and comply with all requirements for the use of personally identifiable information.

Aside from the risks of revealing personally identifiable information through metadata, the data returned by Things can also be sensitive. In the event that the data returned by a Thing is monitoring the location or health of a specific person, this information is sensitive. Therefore, this kind of thing should have some kind of access control. Access controls are generally most effective when secure transportation is also used.

ANALYSIS

IoT in Smart Grids

Current distribution networks were not designed for the management of bidirectional energy flows. As discussed in the introduction to this work, they were designed for the transmission of energy in one direction, from generation to consumption.

With the growth in energy demand and the increase in distributed generation, technical limits may be exceeded in supply lines, in addition to the appearance of problems related to power quality [60].

Due to new generation trends, the balance between energy generation and demand is now becoming a problem at the local level, having slower response times, and even quality losses.

Derived from the problems addressed, there is a need to find measures that allow acting in much faster times responding to situations of imbalance between generation and demand of energy.

Another big problem that arises in distribution networks is about their monitoring. Traditionally, it is mainly done in high and medium voltage networks. Most consumers are connected to low voltage, which lacks monitoring capacity, causing a lack of information at this level.

One alternative to this problem is to connect the transformers of the distribution network to the control center of this, assuming a high investment cost since it would be necessary to create point-to-point connections where they are needed. In addition, all that data would cause an increase in the workload in the control center.

Another alternative is linked to the IIoT. This strategy consists of the installation of sensors and gateways which use wireless communications to concentrate and process data locally.

An IIoT-based approach enables monitoring the grid deeper, applying innovative techniques designed to efficiently interact with the growing number of power generation sources. In addition, the introduction of new elements in the networks, such as storage techniques, will optimize the balance between the generation and consumption.

The incorporation of IIoT is a big change for utilities that have traditionally worked with OT approaches, such as SCADA systems, which tend to be completely isolated from other IT systems of the company and the internet. In addition, IIoT adoption concern utilities about the security derived from devices connected to the internet controlling the networks.

In a process of adopting IoT in SGs, the first step would involve monitoring the LV network, capturing data from physical devices, and processing this information locally to generate alarms when operating parameters exceed established operating limits. One alternative is to monitor those alerts existing systems such as ADMS. It can also be done with specific systems integrated to provide a continuous view of the network at all voltage levels. To accomplish this, gateways are employed with the necessary sophistication to dynamically adjust the data sampling frequency in response to evolving network conditions. Additionally, they enhance data capture when deviations from the predefined standard operating conditions occur. The data is sent to the monitoring centers once the problem has been detected. The ability to generate information locally therefore reduces the amount of data sent through the gateways, and thus reduces the costs associated with wireless communications.

Uses and barriers of IoT in Smart Grids

From a technical point of view, the function of the smart grid IoT network deals with these three areas, the collection of information, its transmission and processing.

The role of IoT in various parts of the smart grid: power generation, transmission lines, distribution, and consumption, is summarized in ***Figure 24*** [13].

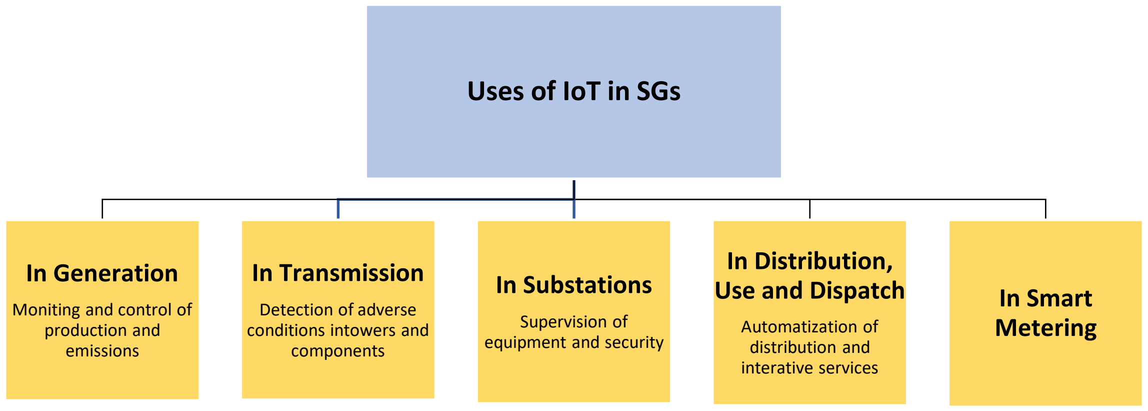

Uses of IoT in SGs
In Generation
Moniting and control of production and emissions
In Transmission
Detection of adverse conditions intowers and components
In Substations
Supervision of equipment and security
In Distribution, Use and Dispatch
Automatization of distribution and interative services
In Smart Metering

Figure 24: Summary of uses of IoT in SGs

- **IoT in power generation.** Through the IoT, it is possible to monitor electricity production in generation plants, controlling parameters such as emissions, consumption, and storage levels. This facilitates decision-making based on collected data to alert and report failures, changes, and forecasts, providing reliable and real-time monitoring.

- **IoT in power transmission.** In the transmission part, to protect it, information on the environmental, mechanical, and operating conditions of transmission line towers and high-voltage electrical components can be detected and monitored through the use of sensors. Using a combination of information and communication networks, it is possible to perform information processing, transmission, and evaluation functions.

- **IoT in the substation.** The IoT monitors the operation of the substation equipment and provides its environmental safety.

- **IoT in distribution, use and dispatch.** In this part of the grid, IoT enables the automation of electricity distribution, enabling two-way interactive services such as smart energy use, data collection, access to distributed generation, and charging and unloading of electric vehicles.

- **IoT in smart metering.** Smart meters communicate data to energy service providers (ESPs) and cloud services through appropriate interfaces. With the use of IoT, more data can be monitored, and as a result, the chances of responding to network failures will improve.

The incorporation of IoT in SGs brings revolutionary opportunities in optimizing energy distribution and improving efficiency in operation. However, there are also challenges that need to be considered when adopting these solutions in the field of SGs.

- In terms of **security**. With the incorporation of IoT into SGs, cybersecurity concerns arise. Vulnerabilities in technology infrastructure can expose the network to threats such as unauthorized access and theft of sensitive data. Protecting the electricity grid from these risks is a priority, as any interruption in the electricity supply has significant consequences.

- In relation to **privacy**. In the context of SGs, the large number of smart devices present generates a large amount of data. This raises concerns about consumer privacy, as the data collected could be misused. Proper management of this data is essential to ensure the privacy of users.

- **Interoperability** is another fundamental barrier that must be addressed in the implementation of IoT in SGs. Due to the diversity of devices, platforms and systems present in a smart grid, ensuring that they can communicate and collaborate correctly is essential to maximize their benefits.

IoT reference architectures applicable to Smart Grids

When talking about different architectures of SGs enabled for IoT, it refers to the structures used to implement SG systems that take advantage of IoT technology, in terms of forms of organization and interconnection of the electrical network infrastructure, devices, sensors and management systems that allow communication and data exchange in real time.

Among the architectures available for IoT-enabled smart grid systems, the following stand out [61-62]:

- **Smart Grid Architecture Model (SGAM)**

  This model has been introduced in the **State of the art** section of this work.

- **Three-Layer Architecture**

  The three-layer architecture can be represented as a pyramid with the following layers as shown in ***Figure 25***.

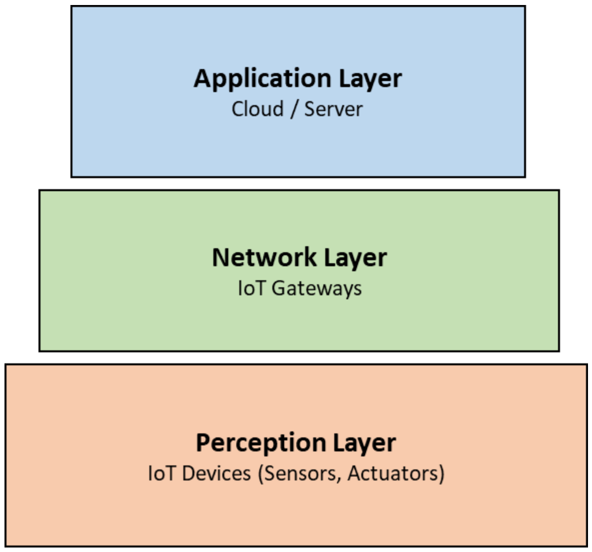


Figure 25: Three-layered IoT architecture applicable to SGs

  - At the base the **Perception layer**. It consists of sensors that monitor the network, capturing information in real time. This information is sent to the management system to analyze various parameters and take appropriate action in emergency situations.

  - Secondly, the **Network layer**. It is based on telecommunications networks. Its function is to map to the corresponding telecommunications protocol, of the information that has been collected in the previous layer of Perception. In this way there is a transmission of data from the Perception layer to the next Application layer.

In this layer, **IoT gateways** [63] play an important role. These are devices that act as intermediaries between the devices of the Perception layer and the Application layer, and whose main function is to facilitate bidirectional communication between network devices and management systems. They act as endpoints or data concentrators, where devices in the Perception layer connect and send their data. In addition to their collection function, they have data processing capacity, transforming them into formats compatible with the communication protocols used in the Network layer, security management and routing data to its destination in the Application layer.

- At the top of the pyramid, the **Application layer**. When data reaches this layer, it uses the information collected from the Network layer to track and take action in real time. Its main objective is to solve problems and optimize the operation of the electrical network.

  This layer can implement algorithms and AI systems to analyze data in real time and make automated decisions, in addition to providing relevant information to operators or control systems.

- **Cloud-Based Architecture**

Cloud computing platforms are used for data storage, modelling and computational analysis. Combining IoT and cloud solutions support data capture with real-time control and intelligence monitoring. This architecture schema is represented in ***Figure 26***.

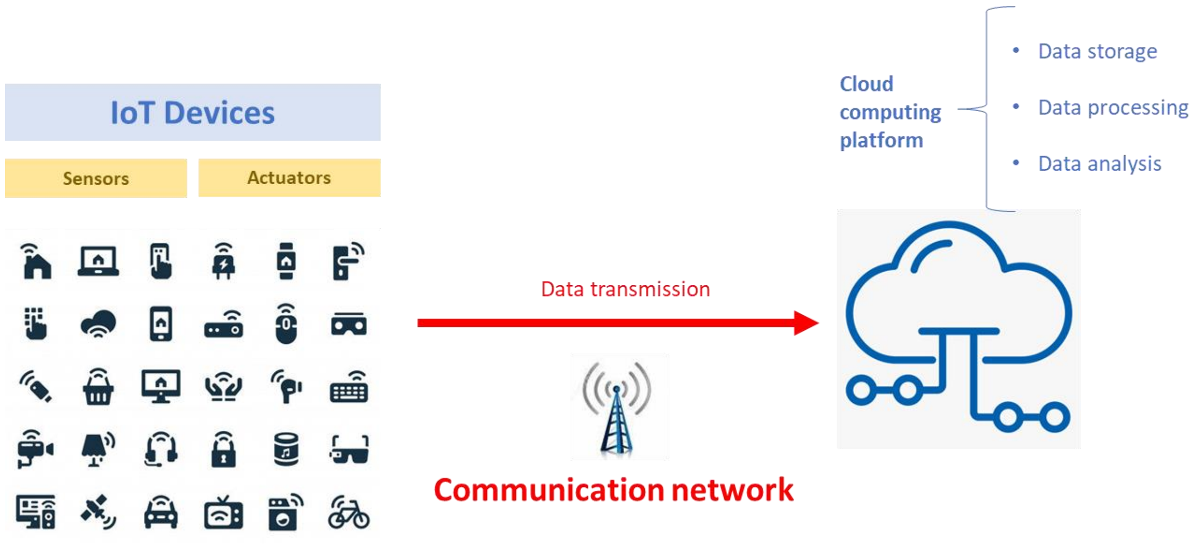


Figure 26: Cloud based architecture

As shown in the image for a cloud based IoT architecture, IoT devices are used in an SG to collect real-time data. These are connected through communication networks, which allow the transmission of data from IoT devices to the cloud, where they are stored and processed.

Cloud IoT architecture uses cloud computing platforms. This platform features storage and processing capabilities, allowing you to manage large volumes of data generated by IoT devices in real time, and uses data analysis techniques to extract useful insights from the collected data. In this way, it manages to improve the efficiency and management of the electricity grid.

On the cloud platform, applications and services are developed for SG management and control, including remote monitoring and control of devices, energy demand management, detection and response to abnormal events, optimization of energy distribution, among others.

- **Smart Grid Architecture based on WoT**

This architecture focuses on the interconnection and communication of devices using standard web technologies. The architecture schema is shown in ***Figure 27*** below.

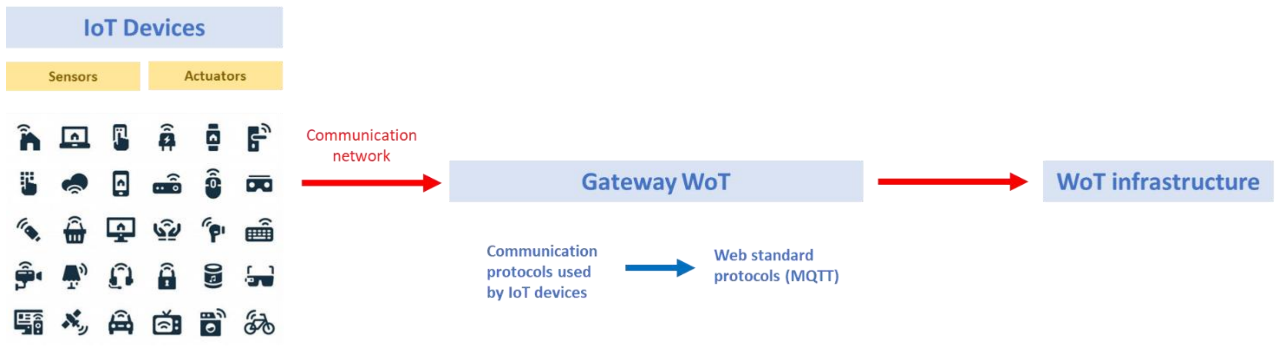


Figure 27: WoT based architecture

A WoT-based SG must first and, as in any architecture, have measurement and control devices to collect data.

In addition, it must have a communication network, responsible for providing the communication infrastructure to connect the devices. This network can be based on existing network technologies, such as local area networks (LANs), wide area networks (WANs), or the Internet, or use wireless technologies, such as Wi-Fi, Bluetooth, or cellular communication.

Thirdly, it must have a WoT Gateway, which acts as an intermediary between the devices of the electrical network and the WoT infrastructure, allowing the translation of the communication protocols used by the electrical devices to standard web protocols, such as MQTT, to facilitate interoperability.

A Data Management Platform collects, processes, and stores the data generated by the devices of the electrical network, which may include database management systems, data analysis tools and information visualization systems. In this way, efficient monitoring and analysis of the electrical network is allowed.

The different applications deployed use the data collected and processed to offer services, such as in other architectures.

The User Interface can be a website, for example, which allows users to interact with the electricity grid and access the services offered.

Since the WoT-based smart grid involves the transmission and storage of sensitive data, robust security and privacy mechanisms are essential. This includes device authentication, data encryption, access control, and protection against cyber-attacks.

- **Combination of Cloud and WoT-based Architectures**

  Cloud and WoT-based architectures can be combined by leveraging the strengths of both to create more complete and scalable solutions. In this way, the IoT devices present in the SG can communicate and send data through local and nearby communication networks (WoT) as well as through the cloud (cloud-based architecture). IoT devices could interact with each other and send data to a cloud computing platform for further storage and processing.

Focus on Interoperability

Taking as a reference the interoperability framework defined in [61], there are different categories describing the different levels of interoperability that can be considered in the context of communication between systems and platforms in IoT technologies. These are summarized in ***Figure 28***.

- **Basic connectivity**. It focuses on the digital exchange of data between systems, and on establishing a reliable communication path. This is achieved through compliance with specifications for data transport medium description, data encoding and transmission rules. Physical interoperability standards include Ethernet [66] and Wi-Fi.

- **Network interoperability**. It is responsible for the problems of transporting information between the different devices that interact through different communication networks. The protocols used at this level are independent of the information transferred. Protocol interoperability standards include FTP [67], TCP [68], UDP [69], IP/IPv6 [70] and ARP [71].

- **Syntactic interoperability**. It is the agreement on rules of formats and structure used when encoding the information that is exchanged between devices. When the syntax is appropriate, it is allowed to decompose the content into common schemas or grammars, but without implying a meaning of the content. Syntactic interoperability standards include HTML [72], XML [73], ASN.1 [74], SOAP [75], and SNMP [76].

- **Semantic interoperability**. It is about breaking down the concepts contained in the message. It includes rules that govern the definition of things, concepts, and their relationships. Examples of semantic interoperability include the Common Information Model (CIM) for energy, SAREF ontology (Smart Appliances REFerence ontology), which will be discussed later in this point, and the substation automation standard IEC 61850 [77].

- **Business context**. It includes roles of actors involved in the interaction, rules and other specific restrictions on the information exchanged, and for the business sector in which it applies.

- **Business Procedures.** The organizations involved must have processes and procedures that are compatible at their interface boundaries to achieve effective interoperability of information.

- The layers of **Business Objectives** and **Economic and Regulatory Policy** involve procedures for aligning stakeholder objectives.

In this work and within the study of interoperability we will focus on the layers of network interoperability, syntactic and semantics, to study their application in the context of that any developed application can work on any platform, regardless of the provider.

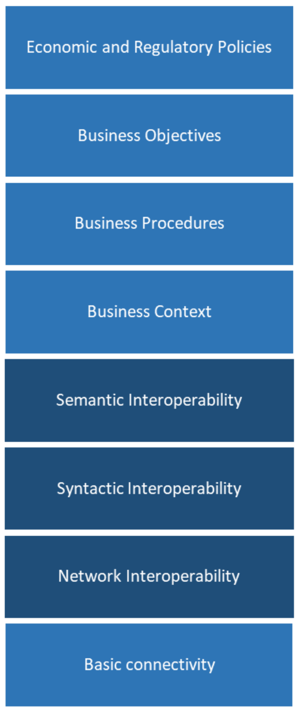


Figure 28: Interoperability levels proposed

The different layers and categories of SGAM and interoperability frameworks allow for a structured and consistent approach to addressing the complexity of interoperability in SGs and related IoT applications. The following image, ***Figure 29***, relates the SGAM model to the previously defined interoperability framework.

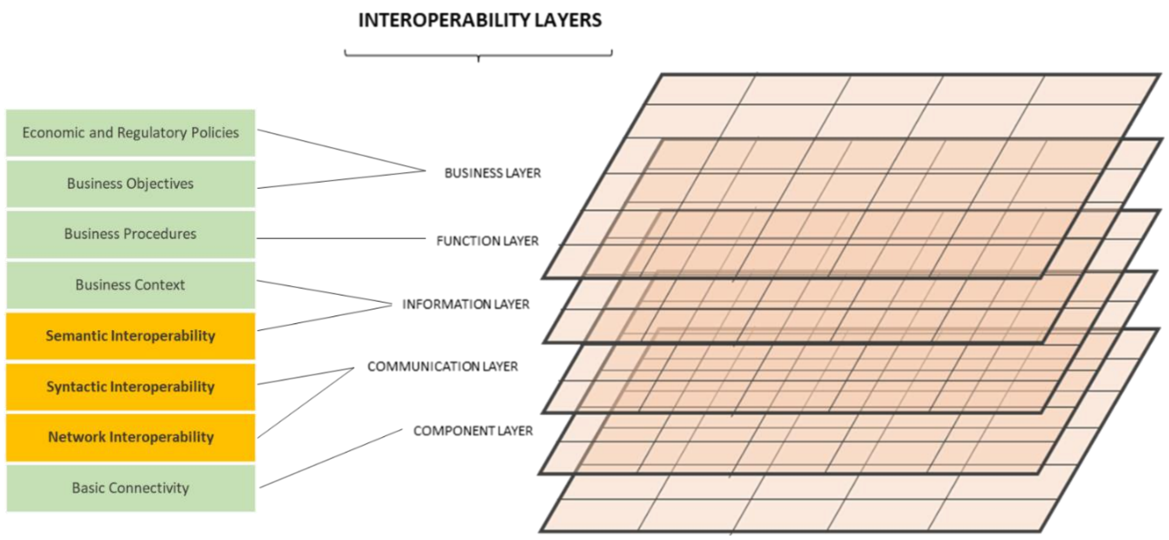


Figure 29: SGAM model and interoperability framework relation

In turn, these layers can be related to Functional Layers, as depicted in the following image, ***Figure 30***.

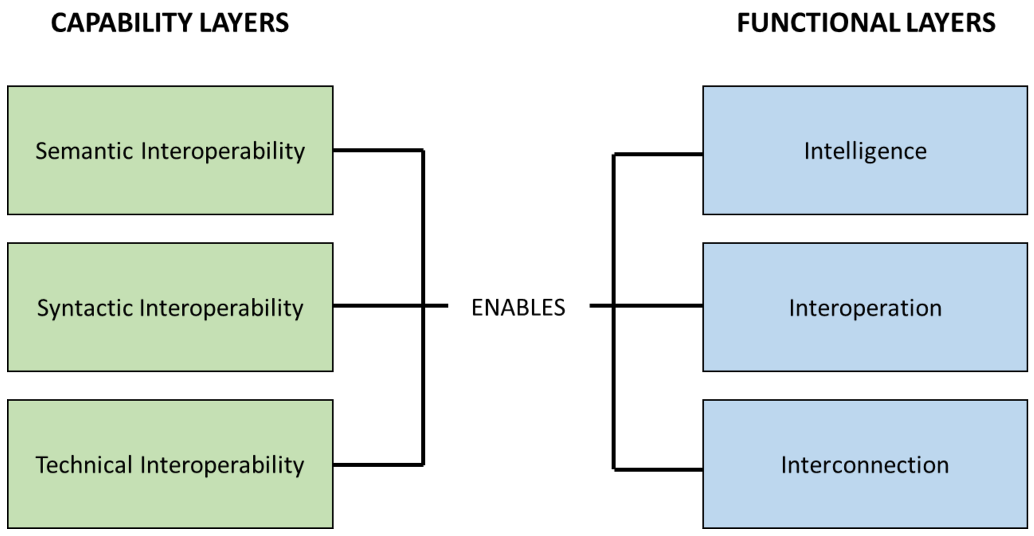


Figure 30: Interoperability layers: Capability layers enable Functional layers

These Capability Layers have been previously defined. Next, it is important to focus a little more on the semantic interoperability layer, as it critical to achieve interoperability in IoT ecosystems.

Semantic interoperability [78] is responsible for ensuring that the transmitted data is interpreted in a common way between the different systems and applications.

All actors involved in the transmission of information, both those who act as senders and receivers of data, must be under a standardized information model that will be used as a reference. The most common way to get systems to understand the same language is through data ontologies. These are a kind of maps which clearly explain what each thing means, so that it helps all devices and systems understand and share information correctly.

Through ontologies, the information that explains the data shared by IoT devices can be transformed into a structured sequence, which helps to make sense of the information that comes from devices and to understand it correctly, since a clear context is established for its subsequent interpretation.

Ontologies can also function as a channel for sending different types of instructions to devices. For this, ontologies provide guidelines to ensure that commands are executed properly. These guidelines include the following:

- Guidelines used to ensure security and access to information. Users can be registered, identities can be verified, and functions can activate or change access permissions to the Data Space.

- In device operation, to enable, deactivate, and change how virtualizations work for physical devices participating in the network.

Both CIM and SAREF are ontologies that seek to provide a structured and standardized semantic representation of information in their respective domains.

- **CIM** [79] is an ontology used in the management of electrical and energy systems. It was originally developed by the electrical industry and is maintained by IEC technical committee TC57. It focuses on representing common information about electrical and power systems, including generation, transmission, distribution, and consumption of electrical energy. With this representation it allows to describe equipment, topologies, measurements, events in electrical systems. This is essential for the efficient management and operation of the electricity grid.

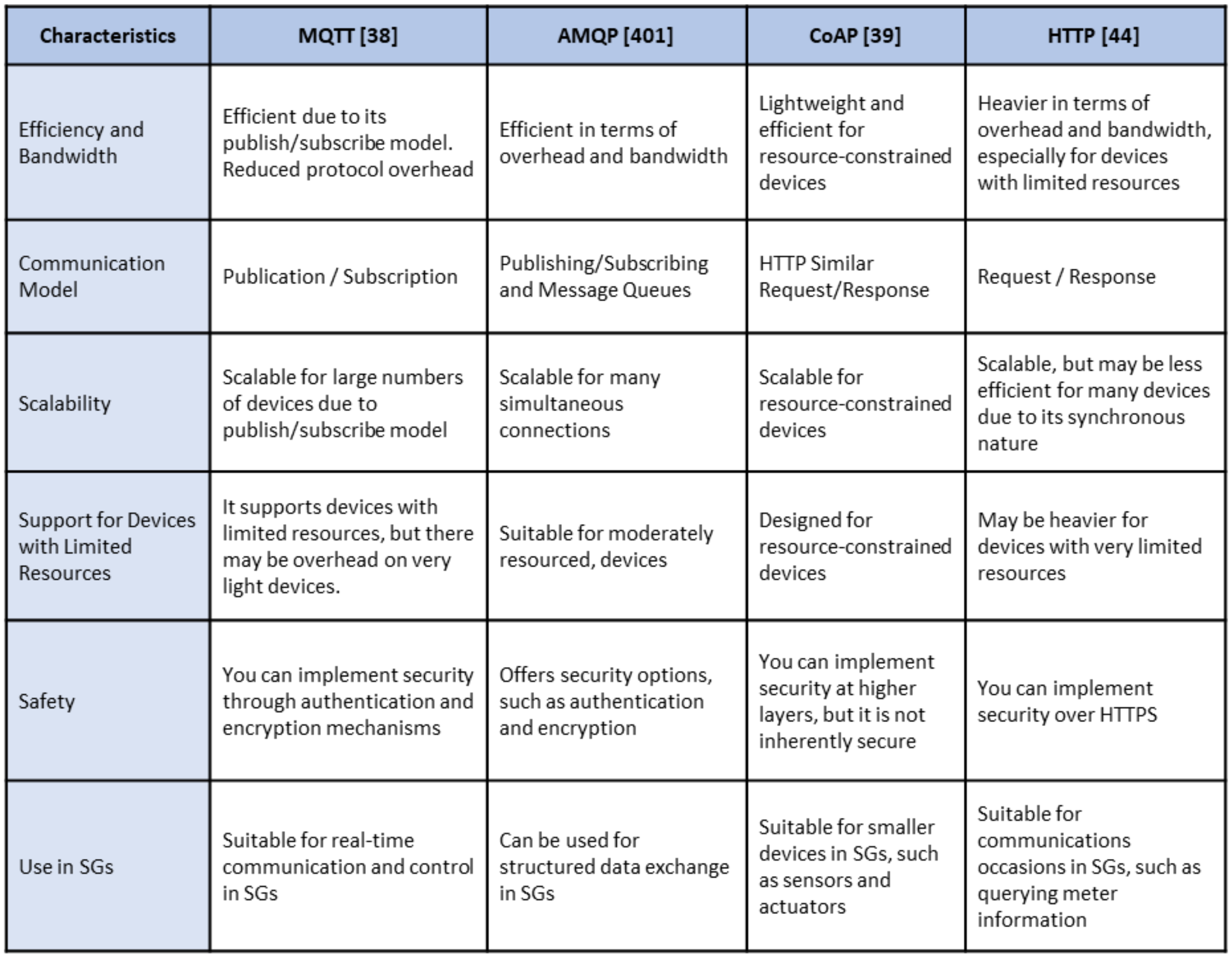
- **SAREF** [80] aims to describe smart devices and home appliances in the context of IoT. It was created by ETSI to facilitate interoperability between smart home devices and systems. This ontology allows you to describe the features, properties, and functionality of devices, which helps applications and services understand and communicate with these devices in a consistent and consistent manner.

IoT communication protocols applicable to SGs

In the context of IoT, it is essential to explore in depth the communication protocols that make possible the efficient and fluid integration between the different devices and systems.

These communication technologies have been mentioned in the **IoT** section. The following table, ***Table 5***, includes a summary of important characteristics regarding the application of the different protocols collected, in the field of SGs. This section will broaden the understanding of communication protocols in their use in the context of SGs.

Table 5: Summary of characteristics of communication protocols applicable to SGs



Characteristics	MQTT [38]	AMQP [401]	CoAP [39]	HTTP [44]
Efficiency and Bandwidth	Efficient due to its publish/subscribe model. Reduced protocol overhead	Efficient in terms of overhead and bandwidth	Lightweight and efficient for resource-constrained devices	Heavier in terms of overhead and bandwidth, especially for devices with limited resources
Communication Model	Publication / Subscription	Publishing/Subscribing and Message Queues	HTTP Similar Request/Response	Request / Response
Scalability	Scalable for large numbers of devices due to publish/subscribe model	Scalable for many simultaneous connections	Scalable for resource-constrained devices	Scalable, but may be less efficient for many devices due to its synchronous nature
Support for Devices with Limited Resources	It supports devices with limited resources, but there may be overhead on very light devices.	Suitable for moderately resourced, devices	Designed for resource-constrained devices	May be heavier for devices with very limited resources
Safety	You can implement security through authentication and encryption mechanisms	Offers security options, such as authentication and encryption	You can implement security at higher layers, but it is not inherently secure	You can implement security over HTTPS
Use in SGs	Suitable for real-time communication and control in SGs	Can be used for structured data exchange in SGs	Suitable for smaller devices in SGs, such as sensors and actuators	Suitable for communications occasions in SGs, such as querying meter information

Especially for this work, it is convenient to explain in greater depth the use of the MQTT protocol in the IoT landscape. [83]

This is mainly due to the following reasons:

- It is specially designed for IoT applications with low latency requirements and limited bandwidth, which is important in SGs where fast and efficient communication between IoT devices and the power grid infrastructure is important.

- It is ideal for scenarios of unstable connections and low power, ensuring that IoT devices can communicate reliably even in challenging conditions, which is also relevant in SGs that can include devices deployed in remote areas or in difficult environmental conditions.

- Its publish-and-subscribe topology is beneficial for SGs, where multiple devices need to monitor and receive data from multiple sources. It is based on the publish/subscribe model, which allows efficient distribution of messages to multiple subscribers.

- Finally, it is suitable for real-time monitoring and remote-control applications, aspects also important in SGs, where data from sensors and IoT devices must be transmitted and received quickly, allowing effective network management in real time.

- In terms of security and privacy, MQTT simplifies the process of encrypting messages using TLS and authenticating clients using modern protocols such as OAuth.

    - **TLS** (Transport Layer Security) [84] is a security protocol that encrypts and protects communication in networks, especially on the Internet. It provides an additional layer of security by establishing connections between devices or servers, ensuring that the transmitted information is encrypted and cannot be intercepted or altered by third parties.

    - **OAuth** (Open Authorization) [85] is an authorization protocol that allows users to give limited access to their information stored on an online platform or service to third-party applications. Instead of sharing your login credentials with the app, OAuth provides a secure, controlled process for granting temporary access to resources.

The operation of MQTT is simple, using a publish-subscribe pattern and involving the following components, as shown in ***Figure 31***.

- The **publisher** creates the message and publishes it to a topic created specifically for it.

- The **subscriber** receives messages relevant to the topic to which he has had to subscribe.

- A **broker** communicates with clients through a local network or an internet connection.

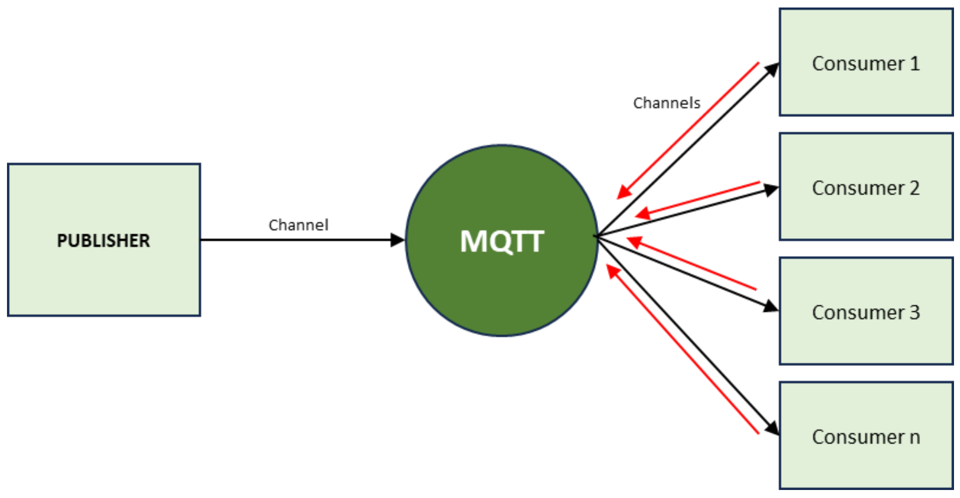


Figure 31: MQTT concept

Until now, these referred to protocols used when communicating IoT devices with the Internet. However, industrial communication and IIoT deployments also use protocols more focused on operations, leaving aside the sending of information. That is, they are protocols aimed at a controller device, such as a PLC, communicating with another machine that executes the orders. In the latter case, the most used protocol is Modbus [82].

Importance of "Edge" in Smart Grids

Accompanying IoT, it is common to hear the term "Edge". In IoT solutions, mainly in the industrial field, the Edge generates advantages when it comes to general commercial value.

The "edge" [86] refers to the level closest to the physical world, represented by devices. An edge node is located close to where the data is generated or collected, that is, at the point closest to the origin of the information. The main function of an edge node is to process, analyze and make decisions about data in real time, before it is transmitted to a centralized location, such as a data center or the cloud.

Applied to the Smart Grid, an Edge node in a smart grid is placed at strategic points where data related to it is generated, transmitted, or consumed in the different stages of generation, transmission, and distribution of electrical energy. A concrete example would be the installation of Edge nodes in electrical substations or secondary substations.

It is very important to emphasize the main challenge in IoT Edge, which is cybersecurity. Edge nodes act as the bridge between the power grid and industrial equipment. To isolate industrial equipment from possible attacks from the network, in addition to protecting the data that these computers capture and send to the nodes themselves, they must have specific protection mechanisms.

It is worth mentioning that the concept of edge contrasts with that of the cloud. In relation to this, the concepts of Cloud Computing and Edge Computing have been defined in the corresponding section of Relevant Theory of this work, **Cloud computing and Edge computing**.

Next, ***Table 6*** is incorporated with characteristics that are relevant in terms of the application of this computing in the Smart Grid, in which Edge Computing and Cloud Computing will be compared.

Table 6: Edge Computing vs Cloud Computing in Smart Grids

Feature	Edge Computing	Cloud Computing
Data Processing	Processes data at the point of origin of the power grid, reducing latency and enabling real-time decision-making for grid management and control	It processes data in remote centers or in the cloud, which can generate greater latency in decision making
Latency	Low, as data is processed and analyzed locally, enabling rapid response to changing events and conditions on the power grid	Higher, due to the need to send data over the network to the centralized processing center, which can lead to delays in making critical decisions
Connectivity	It is performed within the edge network, reducing dependence on connectivity across the network, ensuring continuous availability of data at the edge	It requires constant connectivity to the cloud, which can lead to problems if the network experiences outages or delays
Bandwidth Capacity	It requires less bandwidth, as data is processed locally on edge devices, decreasing the load on the network	It can require considerable bandwidth to send data to the cloud and receive responses, which can impact operational efficiency

Security	It offers greater control and local security, as sensitive data is kept within the edge network and is not sent over external networks	Security depends on measures implemented in the cloud, which can lead to concerns about privacy and data protection
Scalability	Scalability is limited to the local level, which may require an expansion of edge resources to handle a larger volume of data and devices	It offers greater scalability through the cloud, allowing it to handle large volumes of data and expanding devices
Costs	You can reduce data transmission costs because they are processed locally and do not need to send large amounts of information over the network	It may require upfront investments in cloud and storage infrastructure, although costs can spread over time
Flexibility	It allows customization and local adaptability to the specific needs of the electricity grid and real-time decision making	It offers less flexibility in customization because it relies on standardized services offered by cloud providers

Below are some examples of the application of Edge Computing in SGs.

In the case of remote control and monitoring devices in the MV, fault step detectors and transformer on-load tap changers can collect relevant data on the status and performance of equipment in real time. This data is processed and used to make decisions about the control and management of the electricity grid in the same transformation center, without the need to send all the data to a central location. Similarly, LV's measurement data concentrators and basic and advanced supervisors enable local collection and processing of energy consumption data and monitoring of power supply in the low-voltage grid. This makes it easy to detect and resolve issues in real time, without relying on the connectivity and latency associated with sending data to a centralized location.

After having dealt with the term Edge and Edge Computing in SGs and returning to the ISS defined in the **Intelligent Secondary Substation (ISS)** section, the ISS Edge Node is now incorporated.

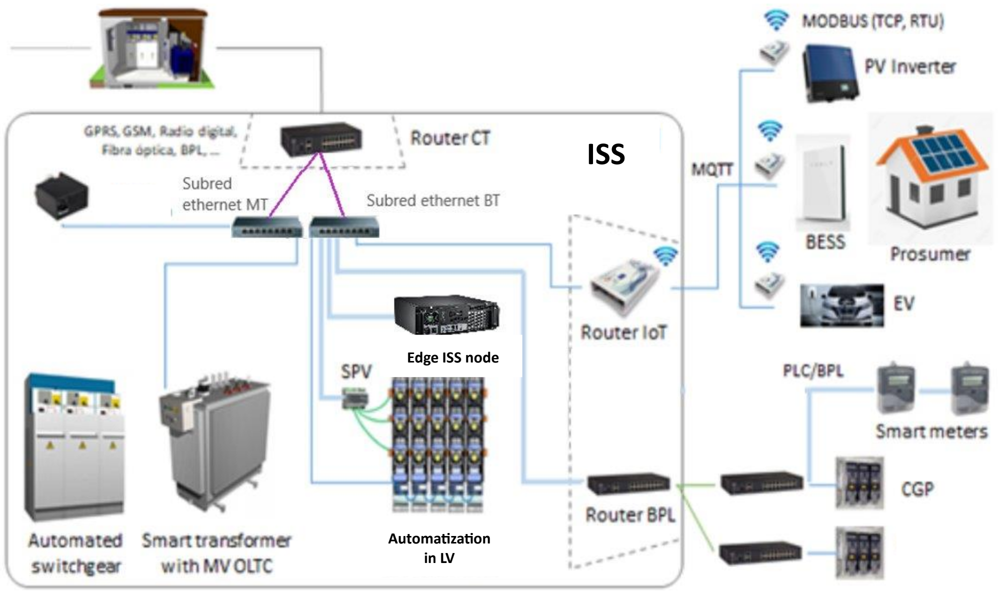


Figure 32: Edge Node incorporation in a ISS

In an advanced architecture, an "Edge ISS node" can be implemented as shown in ***Figure 32***, replacing the previously defined "Edge Computing Node". Instead of having a separate DCU, the ISS Edge node could combine advanced local processing, communication, and control functions within the same device. This node could take over many of the functions that were originally assigned to the DCU and edge computing node separately, simplifying the infrastructure and enabling more agile decision-making in place.

Application Virtualization

Application virtualization [87] is a technology that enables applications to run in isolated and virtualized environments, separate from the underlying physical infrastructure and other applications on the same system. This is achieved by creating containers or virtual instances that encapsulate the application or applications, and their dependencies, providing a consistent and isolated environment for the execution of the application.

The benefits of application virtualization in the context of IoT and SGs are discussed.

- **Resource efficiency**. It allows you to host multiple applications on a single physical server, thus maximizing resource utilization. Through virtualization, multiple applications are hosted in separate virtual machines or containers, each of these virtual environments operating with its own operating system on a single physical server.

- **Simplified management**. Transforming physical systems into virtual machines simplifies management with software-based policies and automated workflows. By using automated deployment and configuration tools, administrators can establish workflows to manage IT services efficiently.

- **Minimal downtime**. Multiple redundant virtual machines prevent downtime by migrating functions in the event of failures, eliminating the need for redundant physical servers.

- **Rapid provisioning**. Agile creation of virtual machines using existing infrastructure accelerates the deployment process and adapts to changing demands. To do this, instead of acquiring, installing, and configuring hardware for each application individually, virtual machines or containers can be created in an agile way using the infrastructure already established.

Next, the techniques of virtualization by Hypervisors and containers [88] aspects that are relevant for their application in the field of IoT and Smart Grids are compared in ***Table 7***.

Table 7: Summary of Hypervisors and Containers virtualization techniques

Aspect	Hypervisor Virtualization	Container Virtualization
Isolation	Complete isolation between virtual machines (VMs)	Limited isolation, they share the same OS kernel
Platform Compatibility	It allows the execution of different operating systems	You may have compatibility issues with OS and libs
Resource Consumption	Increased use of resources, due to operating systems	Efficient, share resources and disk space
Efficiency	Hypervisor overhead can affect performance	Less overhead, fast start-up and better performance

Portability	Less portability between different hypervisors	Highly portable between different environments
Fast Boot	Slower boot time due to full OS	Quick start al no need full OS
Scalability	Can require more resources for replication and scale	Easy scalability due to its lightness
Safety	Increased security and isolation between VMs.	Less insulation, potential for interference

The performance gap between containers and hypervisor-based virtualization has narrowed considerably. Containers excel at offering flexibility to adjust the number of applications or instances running on a node, as well as providing portability and simplified deployment. These qualities position containers as the most suitable option.

Going deeper now into the different container platforms, we highlight Docker, Cloud Foundry, Kubernetes, and CoreOS. All of them offer viable alternatives to virtualization. Next, a comparison is made between them in ***Table 8***.

Table 8: Comparison between container platforms

Characteristics	Docker	Cloud Foundry	Kubernetes	CoreOS
Ease of Use	Ease of use and rapid adoption	Offers a simple approach to deploying cloud applications	Moderate learning curve due to powerful orchestration	Easy to install and configure
Portability	Highly portable and can run in various environments seamlessly	Cloud portability, but your approach may be better suited for cloud-native applications	Ideal for highly distributed applications running in clusters. Offers portability and scalability	Portability, but may be less suitable for complex environments requiring high orchestration
Scalability	Scale-up and scale-out for applications, making it ideal for applications that need to expand quickly	Has scalability limitations compared to other solutions	Leader in cluster scalability	Effective scalability across server fleets, but may not be so much for complex clusters
Orchestration	Offers orchestration capabilities, but is simpler compared to Kubernetes	Offers robust application orchestration, which is beneficial for cloud deployments	Leader in cluster orchestration and ideal for applications requiring advanced, automated management	Offers less advanced orchestration, aimed at less complex applications
Adoption	Active community and wide adoption in the industry	Growing community and decent adoption	Large community and mass adoption	Growing community
Enterprise Support	Offers enterprise support options, offering security and technical assistance	It is backed by well-known companies, providing solid support options	Backed by major enterprises, ensuring high-quality business support options	Offers support available, although it may not be as comprehensive as in other solutions
Flexibility	Very flexible and allows customization according to the user's needs	It has some limitations in flexibility	Offers customization and flexible options for managing applications in complex clusters	Flexible and customizable, suitable if you are looking for a balance between flexibility and simplicity

The choice of Docker as an application virtualization tool in the context of this work is based on its unique combination offering of ease of use, portability and scalability, essential elements to deploy applications in a dynamic environment such as Smart Grids.

Using Docker in the IoT context and applying it to Edge Nodes offers several key advantages, where there are multiple different devices and hardware running applications.

In terms of Application Portability, Docker [89] places all application dependencies in a container that is portable across different platforms. Distributed applications can be built, moved, and run using containers. Developers and application administrators can run the same application on different computers, virtual machines, or the cloud.

In addition, Docker demonstrates lightweight and rapid performance in comparison with VMs. Unlike VMs, which entail booting an entire OS and consuming resources, Docker operates by running instances of individual applications without needing full OSs for each instance.

Docker allows to allocate and limit CPU, memory, network, and disk resources in all processes using Linux Control Groups. This ensures that one process does not need to use all the computer resources and leaves other processes without resources.

Docker images are read-only templates, and Docker logs store these images. Docker containers are created from Docker images, which contain everything needed to run an application.

Docker architecture, it is based on containers which have features such as cgroups and namespaces.

- **Cgroups** is a feature of the Linux kernel which allows the management and limitation of system resources for processes and groups of processes. It allows administrators to allocate and control the amount of CPU, memory, network bandwidth, and other resources that processes can use. In the case of having several containers running at the same time, this feature allows you to assign resource limits to each of the running containers, achieving greater efficiency in the use of resources.
- **Namespaces** are another key Linux kernel feature for effective interprocess isolation. It allows you to create isolated environments in which each process can operate. This is important to prevent processes from interfering with each other and to ensure that each process has its own namespace for things like the network, file systems, and other system resources.

Centralized node and application manager

A platform [90] is defined as a set of tools that allows more complex systems to be built. IoT platforms facilitate the development and deployment of IoT systems, allowing the end user to focus on what is most important to them, the operation of their business.

As IoT devices deployments grow in number and disperse geographically, the need for a centralized management system becomes evident, to provide basic information on the status of deployed devices and allow software updates to be performed remotely, including an integrated environment for application development, and debugging. All these applications running on the devices must be deployed remotely and securely.

Any IoT platform must be robust from an operational point of view, but also in terms of cybersecurity, especially in the industrial world. In fact, security is one of the most important barriers to the adoption of IT technologies by established companies with a strong OT heritage.

There are many types of platforms and when selecting an IoT platform, it is useful to distinguish between generic platforms and more specific products.

Generic platforms are those of providers such as Azure IoT Hub [17], AWS IoT Core [16], Google Cloud IoT [18] and Oracle IoT Cloud [19], which offer a wide ecosystem of tools. The options of these platforms are practically endless, but they usually have a considerable learning curve and are not adapted to the specific needs of certain customer segments.

On the other side of the spectrum, there are specific platforms, such as Barbara's Industrial Edge Platform, Minsait Onesite Phygital Edge, with more limited capabilities, but adapted to specific verticals. These platforms typically offer better user support and much shorter development and deployment times.

IoT Nodes

Based on the Edge nodes defined in the section Importance of "**Importance of "Edge" in Smart Grids**, IoT nodes [91] have in common their location near the place of data generation, allowing processing and analysis closer to the source of information. They act as intermediaries between data capture devices and centralized processing systems.

The key difference lies in their approach and functionality. Edge nodes are designed to carry out real-time preliminary processing and analysis of data before it is transmitted, while IoT nodes focus on efficiently collecting and transmitting data to core systems or the cloud for more in-depth analysis.

The deployment of IoT nodes consists of physically installing the nodes and configuring them using the logic and algorithms they require, for the use for which they have been designed.

The IoT node is the element on which the operating system and its applications run, either natively or in Docker Containers.

Onesite Platform

First of all, Minsait Onesait platform has to be introduced, and then focus on the Pygital Edge solution. Onesite Platform [92] is a broad platform that provides a number of modules and components that enable organizations to develop customized technology solutions for various purposes, such as data management, data analysis, building web and mobile applications, and implementing IoT solutions.

The platform offers a wide range of capabilities, including data and asset management, advanced analytics, data visualization, application creation, IoT device management, systems integration, process automation, and more.

This Platform will not be studied in depth in this work. However, [92] contains information about its architecture, modules, components, security and more in detail.

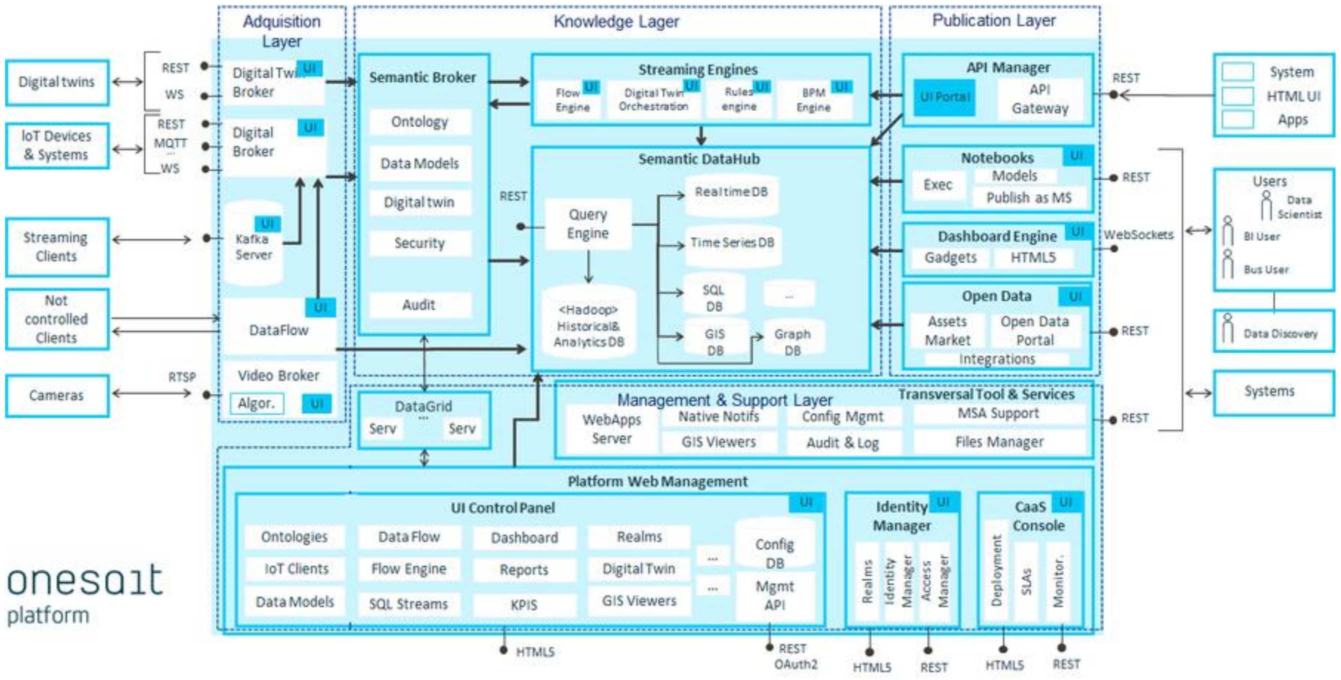


Figure 33: OneSite Platform Structure [92]

As can be seen in ***Figure 33***, this platform is structured in four layers of Acquisition, Knowledge, Publication, and Management and Support. This platform is designed to be scalable, robust and offer high performance in both processing and storage.

Phygital Edge Solution

Onesait Phygital Edge is a specific component within the Onesait Platform offering. It focuses on the management and administration of IoT devices and systems at the network edge, focusing on the management and control of field-deployed devices, enabling two-way communication with these devices, and providing functionalities to them.

It is an IoT/OT platform that can manage and distribute information throughout the energy value chain. By integrating customer data with network operational data, utilities can improve overall system performance.

This platform is based on a processing system close to the devices of the network, that is, the Edge nodes, which in this case refers to photovoltaic inverters and battery storage systems, where relevant information is generated and consumed. A virtualization and container architecture are implemented on these nodes to deliver microservices that efficiently distribute intelligence. Each virtualized edge node can include multiple Docker containers that host specialized algorithms and functions.

The implementation of the Edge Architecture is based on the collaboration of three key components. These are the field devices, the Edge Engine and the Edge Management System. The latter two work together to allow complete management of devices in the field, providing various functionalities through a specialized agent.

- First, there are the **devices in Field**. These devices are connected to the platform to allow their management remotely.

- The "**Applications and edges Management system" (AEMS)** is the distributed global management center. Its primary function is to commission, access, configure, update, and manage edge devices. *Figure 34* shows the interaction between the AEMS and SS.

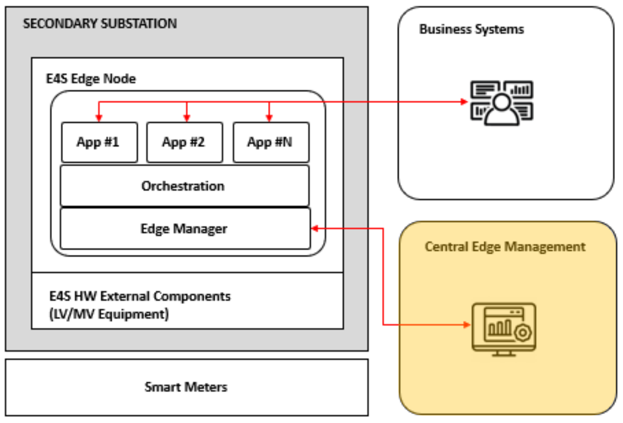


Figure 34: AEMS

The figure of the agent is important, which facilitates the interaction between the central manager and the edge node, ensuring the functionalities of the module. Agent management involves controlling and managing the software agents installed on the edge node, which can execute commands, collect data, and transmit it. In addition, AEMS should enable remote installation, uninstallation, upgrade, and downgrade of agents on the edge node, and monitor their status and performance. You must also provide an API for agent management, used by the graphical interface and command line.

- The **Edge Engine** is a collection of capabilities deployed on nodes through containers. These enable remote control of nodes in the field, local information acquisition and processing, as well as connection to the enterprise cloud for scalability and deeper analytics.

  The software architecture of the Edge Engine is container-based, as represented in ***Figure 35***, where each container can run independently of the rest and be deployed on any machine regardless of the host operating system. This allows the system to be independent of running containers and can quickly deploy new containers without the need to modify the host.

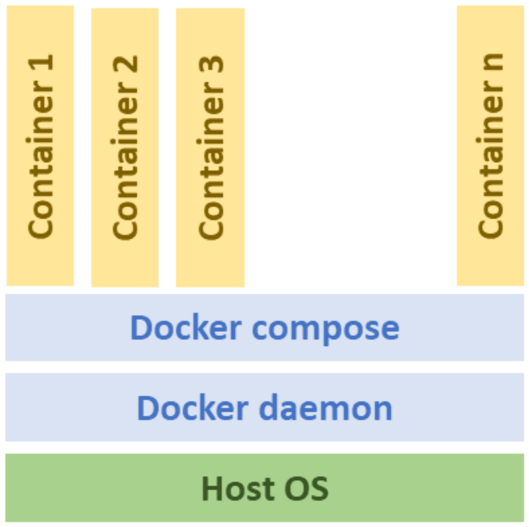


Figure 35: Edge containers architecture

The Edge container architecture is defined by three communication models represented in ***Figure 36***. There are **Information Consumers** that acquire MQTT topic information to process, transform or send it, **Information producers** who can send information to MQTT topics, and **Information Prosumers** that consume information from the topic, transform it, and generate new information for other nodes to use it.

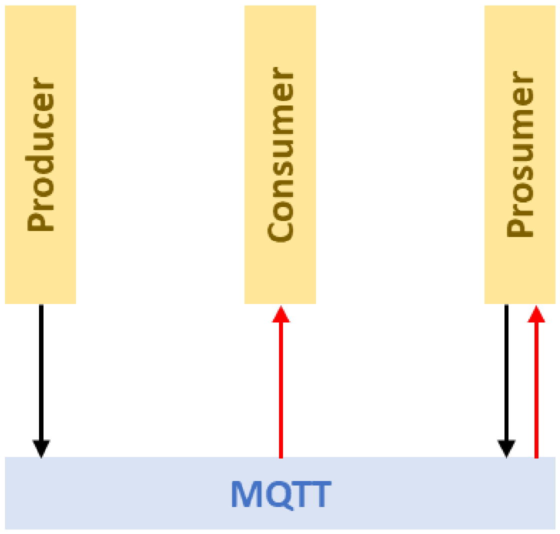


Figure 36: Communication models used in containers

To facilitate the exchange of information between producers and consumers, a MQTT broker is used as an intermediary. It receives the information generated by the producers and distributes it to other containers for consumption.

In addition, it consists of another component, the IoTa Agent or Edge Agent, which is essential for the operation and management of the system. Its key functions include:

- Start and manage the command-and-control information channel from the Edge Management System using MQTT/TLS.
- Communicate with edge nodes and internal IOT HUB modules.
- Process IOT HUB commands such as SSH reverse tunnels, configuration updates, and direct commands with OS scripts.

Security on the Phygital Edge Platform

In terms of safety, the Phygital Edge Platform has an identity record that stores information about devices and modules authorized to connect. Devices authenticate to the Edge Management System using credentials stored in the registry. Authentication is performed through an encrypted file associated with the MAC address of the device or through certificates generated by TPM2.0, as represented in ***Figure 37***. These measures ensure secure and reliable connections between devices and the management system.

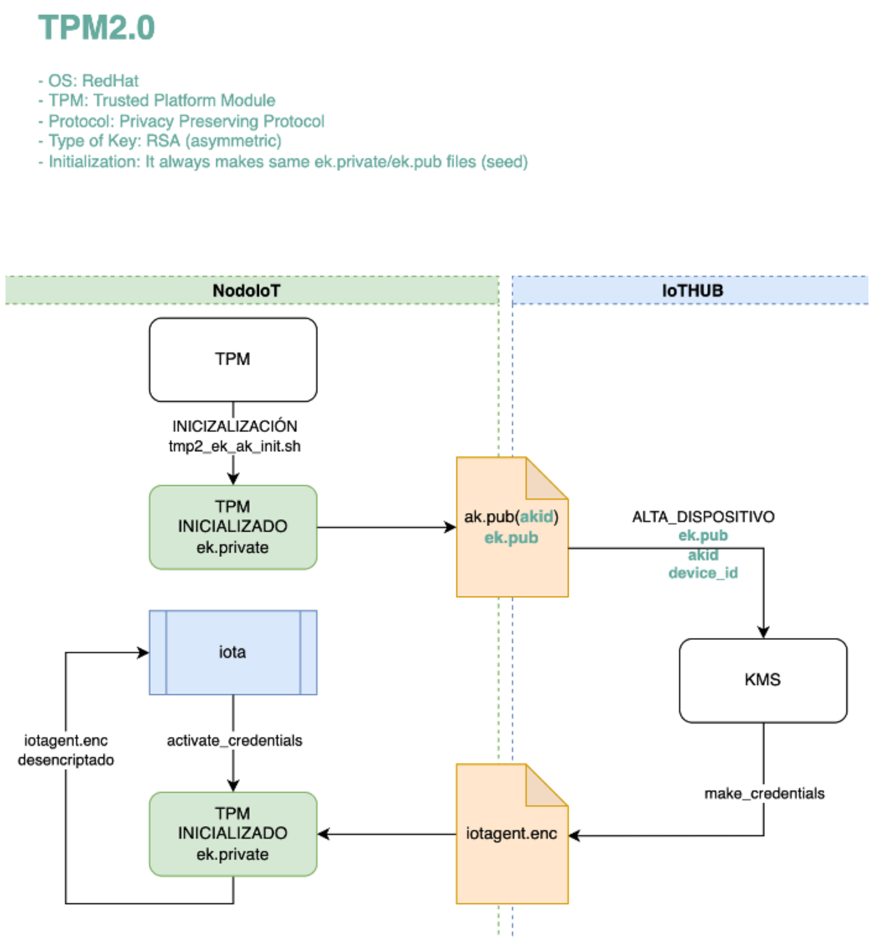

TPM2.0
- OS: RedHat
- TPM: Trusted Platform Module
- Protocol: Privacy Preserving Protocol
- Type of Key: RSA (asymmetric)
- Initialization: It always makes same ek.private/ek.pub files (seed)
NodoIoT
IoTHUB
TPM
INICIZALIZACIÓN
tmp2_ek_ak_init.sh
TPM INICIALIZADO ek.private
ak.pub(akid)
ek.pub
ALTA_DISPOSITIVO
ek.pub
akid
device_id
iota
KMS
iotagent.enc desencriptado
activate_credentials
make_credentials
TPM INICIALIZADO ek.private
iotagent.enc

Figure 37: Security on the Phygital Edge Platform: TPM 2.0. [92]

Application to the Secondary Substation

The Secondary Substation Platform (SSP) is built under an open-source philosophy, using free and open software technologies, which promotes collaboration, innovation, and adaptability to various situations. This platform is composed of Nodes that work creating a computing and analysis environment at the edge of the network.

Next, the following applications have been identified as possible SSP use cases, also identifying requirements to make them feasible. Those are summarized in ***Table 9*** below.

Table 9: Use cases of Phygical Edge Platform in a SSP

Use Case	Description	I/O Requirements
Data Concentrator for Smart Meters	This application collects data from smart meters using PLC PRIME technology. It then sends that data over the WAN connections available in the Data Hub	It requires a PRIME Base Node connected to the three phases and the neutral of the LV network in the secondary substations
Advanced LV supervision	This application monitors the LV panels on the SS. Communicates via DLMS with LV I/O cards	Requires the installation of I/O cards in each phase of the LV panel, to measure instantaneous values (V, I, P and Q) of each phase
Network Topology Identification	This application calculates the LV topology, identifying the line and phase for each smart meter connected to the low voltage panel. Uses information from Data Concentrator and LV Advanced Supervision	Does not require additional I/O as it uses data from previous applications
Interruptions Identification	This application has identified outages in the LV network, estimating the size of the outage and its impact on customers. Uses information from the three previous applications	Does not require additional I/O as it uses data from previous applications
Transformer Regulation Monitoring and Control	This application monitors the LV parameters of the transformer and regulates the output voltage to adapt the power quality to the state of the grid. It uses information from previous applications, and provides an interface with the transformer controller to modify the output of the transformer	Requires an I/O module that can operate the transformer controller
Video Analysis for visual monitoring	This application can cover several use cases based on the video signal coming from one or several digital cameras that capture different areas of the substation	Requires one or more video cameras
Substation Load Balancing and Load Profile	This application calculates secondary substation load statistics by phase, circuit and transformer to suggest better customer distribution	Does not require additional I/O as it uses data from other applications
Generation and Demand Prediction Analysis	This application calculates demand and generation predictions based on historical data from smart meters and low voltage panels	Does not require additional I/O as it uses data from other applications
Distributed Energy Resources Management (DERMS)	This application operates distributed energy resources to respond to specific grid situations.	Requires an I/O to communicate with the DERs

Container orchestration, an essential component of this platform, enables efficient management of applications and resources in complex, distributed environments. Within the framework of the SSP, this methodology is used to coordinate and manage applications and services in various clusters, ensuring their efficient, scalable, and highly available operation.

Within the communication model designed for the SSP, three types of information exchanges between the software components deployed in the substation are sought. The functional requirements of the applications identified so far establish three types of flows that must be compatible:

- **Message Type 1 - On demand**: One application needs information from another. This request must be answered immediately and in a synchronized manner.

- **Message Type 2 - Scheduled**: One application needs information from another. Due to the duration of some computational tasks required to generate a complete response, the task must be scheduled, and the requesting application receives a response with the Uniform Resource Identifier (URI) necessary to consume the information.

- **Type of Message 3 - Spontaneous**: An application wishes to subscribe to periodically receive certain information from another application, which provides the data spontaneously.

To achieve these types of communication, an essential communication protocol is used that has been explained above. This is **MQTT,** which as explained, is a lightweight messaging protocol used in the Publish/Subscribe scheme. Here is a quick overview of how MQTT solves these three types of flows.

- **Message Type 1**. One application can send an MQTT message to another to request information.

- **Message Type 2**. An application can send an MQTT message to schedule a task or request information, with details about when the task is expected to be completed. The recipient will process the task and send a response to the sender, including the URI needed to access the information once it is available.

- **Message Type 3**. To achieve this type of communication, an application can periodically publish data to a specific topic using MQTT. Applications interested in reading this information can subscribe to this topic and will automatically receive the published data spontaneously.

This protocol is used in conjunction with the **WoT**, a standard promoted by the W3C. This architecture provides the necessary framework for communication protocols to work effectively and in a standardized manner.

Using WoT will provide the SSP with an easy form of interoperability between applications deployed on different nodes. This section is crucial because it establishes how to achieve fluid and efficient communication between the different applications within the SSP.

An important part of WoT are TDs. The TDs define the IoT devices present in the SSP, thus allowing a clear understanding of their functionalities, and as a goal facilitating interoperability between devices from different providers, by ensuring that applications can communicate and take advantage of the specific characteristics of each device without compatibility problems. TDs simplify the integration of new devices into the platform, speeding up the development and deployment process. Finally, they offer a unified view of substation assets, improving efficient management and monitoring. When creating a TD in the context of SSP, this must include sections detailed in **ANNEX 2: TD**.

The following is the SSP-specific WOT-A architecture:

- **Applications as Things**. Each application deployed to the SSP will be considered a "Thing" as defined by the standard. This allows applications to be treated as identifiable and communicative entities in the architecture.

- **Functionalities of the Applications**. Each application will cover a full set of functionalities which include reading state information from the "Thing", updating its state with the ability to provoke actions, subscribing to receive notifications of changes in state information, invoking functions with input and output parameters that can cause actions or calculations, and subscribing to receive notifications of events that only report on state transitions.

  In the case of an electrical substation, each application has a kind of ID card that explains what it does and how it communicates. This is what has been defined as TD. Now, in order for all these applications to be understood and collaborated, they need a kind of central list where each one shows its card.

  This is what is known as the **WoT Directory**, where all applications hang their cards. This can be done in two different ways.

  - Visit to the online dashboard (link: "https://E4S-directory/tds") and view all application cards. It's like flipping through a directory to find out who is there and what they can do.

  - In addition, every time a new application arrives or leaves, it is advertised in a public online space (MQTT topic: "E4S-directory/tds"). When a new application arrives at the substation, it needs to "introduce itself" to the WOT Directory by showing its card. And if an application leaves, it notifies the Board so they know.

  To find out which applications are in the substation, there are two options:

  - Ask the WoT Directory to show you the list of applications and their cards.

  - The WoT Directory can warn you when there is a new application, or when one has been deleted.

- **Direct Communication Pattern.** Consumers (applications) and Things (applications) are connected without intermediaries, facilitating faster and more efficient communication between them.

  The Handshake is essential for communication between applications, in the context of the SSP and its relationship with WoT. It involves choosing protocols that ensure efficient communication, which for the specific case of SSP turns out to be JSON-MQTT.

  This combination uses JSON for structured data and MQTT for efficient communication, a combination that helps improve interoperability between applications and data in SS.

  The common use of JSON in this context and specially in TDs is due to its readability, simple structure of key-value pairs which facilitates the representation of information, recognition and support in multiple platforms and languages, flexibility to describe capabilities and details in a coherent way, fluid integration with web technologies and APIs, efficiency in data transmission in IoT networks with limited bandwidth, and a wide support of tools and libraries for manipulation in various programming languages. For more information about JSON text format, see **ANNEX 1: JSON**.

Now as for the security considerations of descriptions of things, there are specific risks and solutions, that are summarized in ***Table 10*** [93].

Table 10: TDs Security considerations

Risk	Description and Context	Detailed Solution
Interception and Manipulation of TDs	Modify TDs for man-in-the-middle attacks	Get TDs only through secure channels with mutual authentication
Limited duration Access	Need to restrict temporary and limited access	Use tokens to manage access and limit features
Vulnerability Audit	Attackers identify vulnerable systems using TDs	Audit and protect vulnerable systems. Do not hide TDs
Scripts Injection	Malicious scripts on TD values generate attacks	Sanitize TD strings before generating interfaces
JSON Analysis	TDs with JavaScript code compromise security	Do not pass JSON TDs through code execution mechanisms

Privacy in WoT is crucial to respect personal information and prevent unwanted data exposure. The proposed solutions focus on mitigating specific privacy risks, ensuring that users' data and identity are protected in WoT Application Descriptions. These proposals are set out in ***Table 11*** [93] below.

Table 11: TDs Privacy considerations

Risk	Description and Context	Solution
Context Leakage	Remote files can be unintentionally transferred while evaluating TDs, exposing private data	Implementations on limited-resource devices should use basic JSON processing and avoid following links to remote contexts
Immutable IDs	Fixed identifiers in TDs can track users and devices over time	Allow for identifiers to be updated as needed in the Descriptions, avoiding permanent links to hardware
Fingerprinting	Device and personal identification through fingerprints	Grant authorized users access to Application Descriptions with necessary info. Exclude unnecessary data
ID Metadata	Metadata in Application Description IDs can expose personal info	Avoid personal information
Global Uniqueness	Global identifiers can leak data if a central authority is compromised	Generate IDs with cryptographic techniques (like random UUIDs) without relying on a central registry
Inferred Personal Info	Device metadata might reveal personal details	Treat Descriptions for personal devices as sensitive. Obtain user consent for data use

Returning now to the platform, each application will deploy a web API and offer services through a RESTful architecture. Only query services that use the GET method are considered. The security of calls to these services will be set out in the appropriate section.

In terms of application virtualization, as discussed in **Application Virtualization** section, containers stand out for their flexibility, portability, and easy deployment, which makes them the optimal choice. In any case, certain benefits associated with VMs, such as complete isolation, cannot be ruled out to be widely employed within the framework of the SSP. Consequently, container-based virtualization, particularly with Docker, is becoming the de facto standard implementation for the SSP.

DEVELOPMENT OF SPECIFICATION

Context

As presented above, an application has been deployed with its corresponding TD, based on the WoT standard. The programmed TD defines the capabilities, properties, interactions, and other relevant information about the Thing, which in this case represents a hardware element, allowing interoperability and seamless integration with WoT applications. This TD facilitates integration and interaction with IoT devices or services by other applications or platforms.

The ultimate objective of the application is to receive signals from IoT devices deployed in a Minsait laboratory environment, fill a TD of a Transformer Temperature Sensor with the information of those messages that come from this type of device. After that, another application reads that TD and creates alerts when the temperature exceeds a previously determined value.

As mentioned in the previous paragraph, the hardware device chosen for which the proposed Thing Descriptor is filled, and measurements captured by it are subsequently monitored is a Wireless temperature sensor.

Wireless temperature sensors [94] are compact devices equipped with temperature measurement capabilities that can communicate without the need for physical wires.

Among the main features and advantages of this type of sensors, its wireless communication capacity stands out, which is based on protocols such as Bluetooth, Wi-Fi and Zigbee. This feature facilitates the fast and reliable transmission of temperature data, allowing agile responses to critical changes in it. In addition, this technology provides flexibility and scalability, since the sensors can be deployed and moved easily, making them a highly adaptable option to different monitoring scenarios.

From an economic point of view, wireless temperature sensors also offer significant advantages, compared to wired sensors. In large-scale deployments, wireless sensors can be more cost-effective as extensive wiring installations are avoided. These sensors are also ideal for applications in remote or hard-to-reach locations, where cable installation would be more complicated and expensive.

In the context of SGs, these sensors are very useful. Its strategic deployment allows monitoring temperature variations in electrical equipment, transmission lines, substations, and other critical components of the grid. Through continuous monitoring of temperature data, it allows operators to detect potential faults or problems in the network arising from overheating, allowing them to take preventive measures and improve network reliability and safety.

In addition, the wireless nature of these sensors allows for easy integration into smart grid communication and data analysis systems, allowing operators to make the most of the temperature data collected.

A specific application of these devices in SGs is temperature measurement in SS, which are critical components in electrical distribution systems. Monitoring temperature in these transformers is essential, as overheating can indicate problems that could lead to costly downtime and even equipment damage. Data collected by wireless temperature sensors in secondary transformers can be integrated into the overall network monitoring and control infrastructure.

The practical part of this work establishes several fundamental objectives, summarized in ***Table 12***:

- **Efficient Communication**. Establish fast and reliable communication between devices (Wireless temperature sensor) and systems (temperature control application) in a SG context, using the MQTT protocol.

- **Data Management**. Collect and process large volumes of data generated by IoT devices on the Smart Grid for more informed decision making.

- **Relevant Data Filtering**. Identify and extract relevant information from the received data for more effective management of devices in the electrical network.

- **Interoperability**. Facilitate integration and compatibility between devices and applications in the Smart Grid for an effective operation.

Among the strengths in interoperability, it is important to highlight the following:

- **MQTT protocol**. Using the MQTT protocol as a basis for communication facilitates greater interoperability, as it is a widely adopted standard and compatible with a variety of devices and systems.

- **JSON format**. The use of JSON format for data exchange facilitates the interpretation and processing of information on different platforms.

- **Customized TDs**. The use of customized TDs ensures that information about devices and their capabilities is structured and standardized, making it easier for other applications to understand and use.

Likewise, the strengths in the context of Smart Grids are the following:

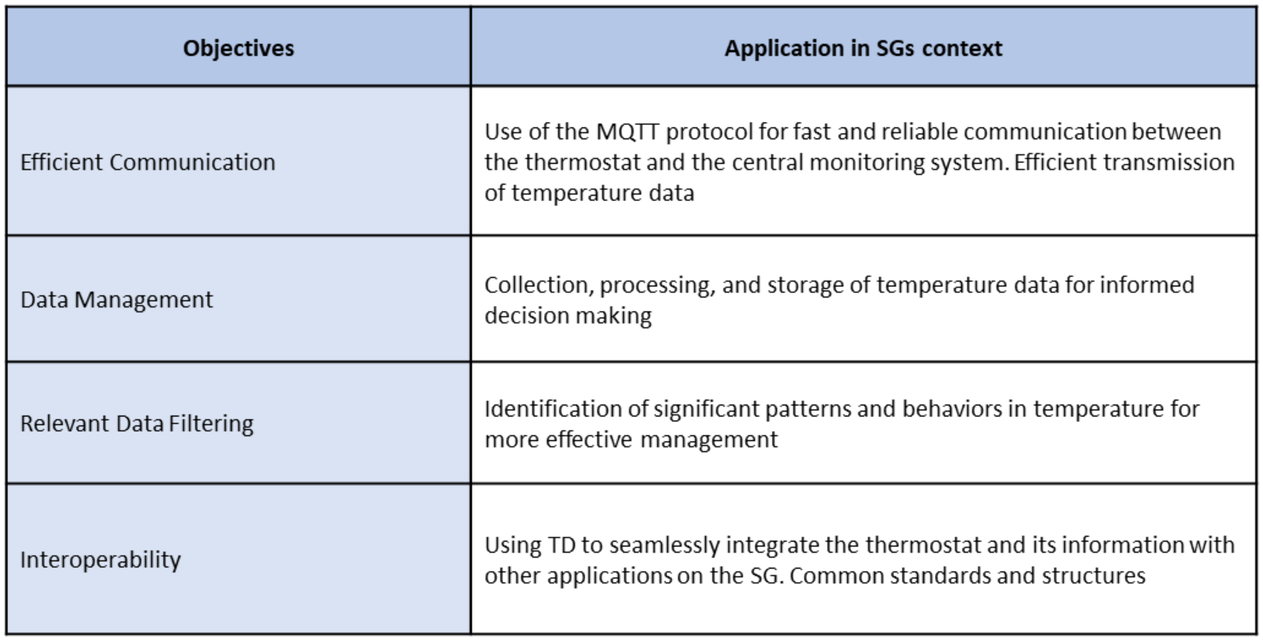
- **Energy Efficiency**. Efficient communication and data management enable better management of electrical power, leading to greater efficiency in the Smart Grid.

- **Cargo Management**. Control and monitoring applications can help in the management of electrical loads, optimizing power distribution and avoiding network overloads.

- **Real-time monitoring**. The ability to filter and analyze data in real time allows constant monitoring of the electrical grid, identifying problems and responding quickly to critical situations.

- **Integration of Renewable Energies**. MQTT applications facilitate the integration of renewable energy sources by providing a platform for data exchange between devices and systems on the Smart Grid.

Table 12: Specification development objectives and their application in SGs context



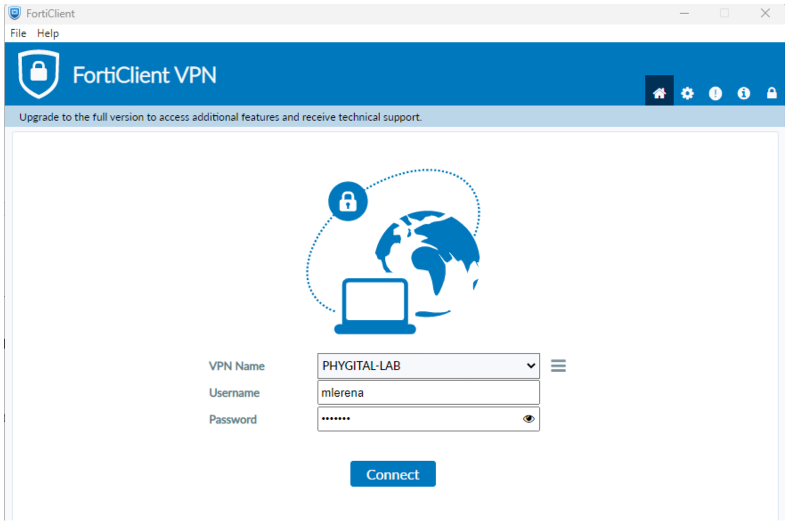
Objectives	Application in SGs context
Efficient Communication	Use of the MQTT protocol for fast and reliable communication between the thermostat and the central monitoring system. Efficient transmission of temperature data
Data Management	Collection, processing, and storage of temperature data for informed decision making
Relevant Data Filtering	Identification of significant patterns and behaviors in temperature for more effective management
Interoperability	Using TD to seamlessly integrate the thermostat and its information with other applications on the SG. Common standards and structures

Technologies, Tools, Programming Languages and Libraries

In the context of the application and the objective described above, various tools have been employed to carry out the work. Each is presented below.

FortiClient VPN

Required to use Minsait Lab VPN. VPN Name, Username and Password is required, as seen below in ***Figure 38***.


FortiClient
File Help
FortiClient VPN
Upgrade to the full version to access additional features and receive technical support.
Connect

VPN Name: PHYGITAL-LAB
Username: mlerena
Password: •••••••

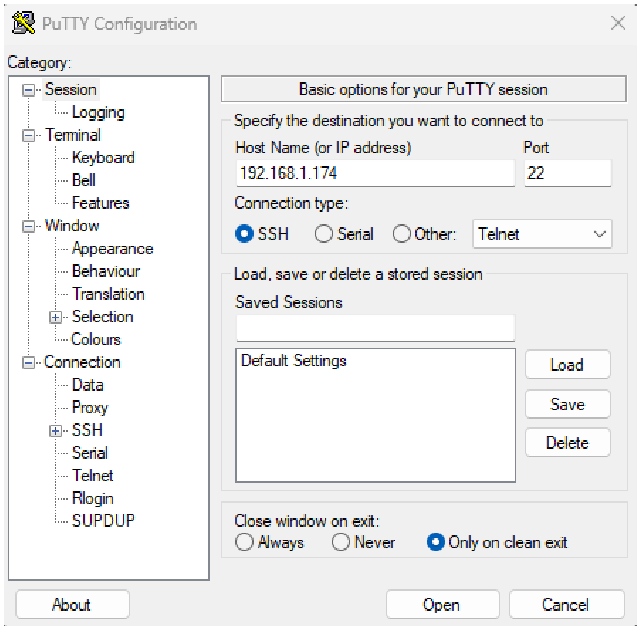
Figure 38: FortiClient VPN

Putty

Putty [97] is a free and open-source SSH client that can be used to connect to remote servers.

When connecting to an IP address using PuTTY, a connection is established to a remote device that has an assigned IP address. PuTTY allows to access that device command line remotely, allowing to manage, configure, and execute commands on that device as if you were interacting directly with it.

You log into an Ubuntu operating system on a server or a local machine, in this case with IP address 192.168.1.174, as shown in ***Figure 39*** . This makes it possible to use a Docker-compatible operating system (such as Ubuntu) to run the Python Docker image in a virtualized environment provided by the Windows VM.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.174
Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Close window on exit: Always Never [✓ Only on clean exit]

Figure 39: Putty Configuration Window

<u>Python</u>

In the work, Python was used because of its advantages in the context of IoT and applications in SGs. Some reasons why Python was chosen are:

- **Ease of development**. Python is an easy language to learn and use, which speeds up the process of application development.
- **Wide community and bookstores**. Python has a large community of developers and a vast collection of libraries that facilitate the implementation of various functionalities. These libraries allow you to connect and communicate devices, process data, and perform specific tasks more easily and efficiently.
- **Multiplatform**. Python is compatible with different operating systems, ensuring that the application works consistently across various platforms and devices.
- **Flexibility**. Python is a versatile language that allows you to develop a wide variety of applications, from small scripts to complex systems.

The Python libraries that have been necessary for the development of the applications have been json, paho mqtt client, and datetime.

- "**json**". The json library in Python provides tools for working with the JSON (JavaScript Object Notation) format. In this work, it was used to build and process the Thing Descriptor, which is a fundamental component for interoperability in the context of the Web of Things (WoT). JSON allows data to be structured in an organized manner, which facilitates the transmission and processing of information in IoT applications.

- "**paho.mqtt.client**". This library implements an MQTT client in Python, allowing communication using the MQTT protocol. At work, this library was used to establish communication between the Transformer Temperature Sensor and the central monitoring system, which allowed to transmit temperature data safely and reliably.

- "**datetime**". The datetime library in Python provides classes and functions for working with dates and times. At work, this library was used to record the date and time the temperature data was received. This is useful for analyzing data as a function of time and tracking temperature changes over time, which may be relevant for decision-making in the context of SGs.

Docker

Docker is an open-source platform that makes it easy to build, deploy, and manage containerized applications. It has been explained in previous sections in this document.

Some of the advantages of using Docker in this work are listed below:

- **Portability and consistency**. In the context of this work, Docker has been used to create an isolated development and execution environment. This ensures that the application runs consistently on any platform or operating system, avoiding compatibility issues and simplifying deployment across different environments.

- **Facilitates teamwork**. When using Docker, all members of the development team have the same development environment, ensuring that the application is developed and tested in a consistent environment. This has made it easier to help among team members.

- **Efficiency in implementation**. Docker allows you to pack your application and its dependencies into a single container, simplifying deployment and deployment of your application at different stages of development.

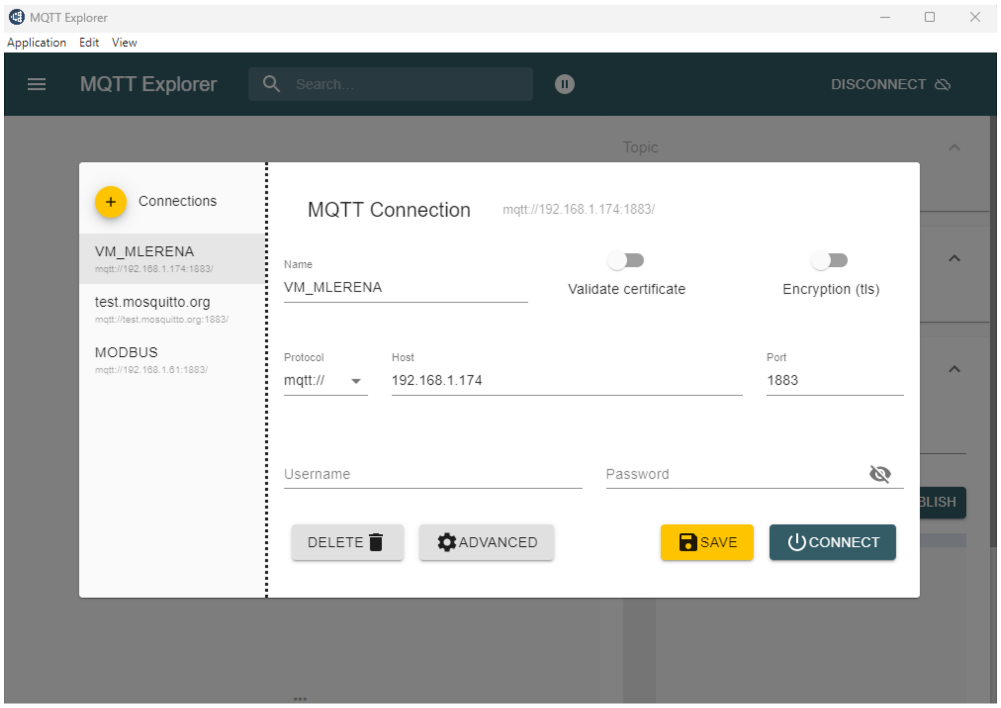
- **Isolation and security**. Running in isolated containers, the application and its dependencies do not affect the host operating system, improving security and minimizing the risks associated with running third-party applications.

In this development, "VirtualBox/Docker deployment installation Guide" from Minsait has been consulted, with the objective to guide through the entire process of been able to execute a python base image inside a Windows virtual machine.

MQTT Explorer

MQTT Explorer [97] is an open-source MQTT visualization and debugging tool that allows you to monitor and analyze MQTT messages that are exchanged between devices and systems. It provides an intuitive graphical interface that displays in real time the MQTT messages sent and received, as well as the topics to which the devices are subscribed.

This tool has been useful in development, as it allows you to verify MQTT communication and make sure that devices are communicating correctly. *Figure 40* illustrates the interface of the tool.



Connections:
VM_MLERENA mqtt://192.168.1.174:1883/
test.mosquitto.org mqtt://test.mosquitto.org:1883/
MODBUS mqtt://192.168.1.61:1883/

MQTT Connection mqtt://192.168.1.174:1883/
Name: VM_MLERENA
Validate certificate
Encryption (tls)
Protocol: mqtt://
Host: 192.168.1.174
Port: 1883
Username
Password

Figure 40: MQTT Explorer

Some of the advantages of using MQTT Explorer in this work are listed below:

- **Verification of MQTT communication**. In the context of this work, MQTT Explorer was employed to verify MQTT communication between the Transformer Temperature Sensor and the central monitoring system. By displaying in real time the MQTT messages sent and received, it was possible to ensure that the temperature data was being transmitted correctly between both devices.

- **Monitoring and debugging.** MQTT Explorer provides a complete view of MQTT topics and messages, making it easy to monitor and debug your system. This is especially relevant in IoT applications such as Smart Grids, where the reliability and efficiency of communication is very important.

A summary of the main tools used in this development, and their applications in the SGs context is summarized in ***Table 13***.

Table 13: Main tools used and their application in the SGs context

Tool	Description	SGs application
Python	Versatile and easy-to-use programming language	It facilitates the development of IoT applications in SGs
	Extensive community and libraries for specific tasks	It allows you to process data and communicate with devices in the context of IoT
	Multiplatform, compatible with various operating systems	It ensures that the application works in different environments
	Flexibility to develop a variety of applications	Adaptable to different requirements in the context of SGs
	json: Facilitates data processing in JSON format	Structure the TD and temperature data in an organized way
	paho.mqtt.client: Implements MQTT communication	Establishes efficient communication between the thermostat and the central system
	datetime: Provides tools for working with dates and times	Records the date and time of temperature data for analysis and tracking
Docker	Container platform that makes it easy to create, deploy, and manage	It provides an isolated and consistent environment for development and execution
	Portability and consistency in different environments	It ensures that the application works consistently across various platforms
	It facilitates teamwork and avoids configuration conflicts	It streamlines the development and deployment of the application in different stages
	Isolation and security in the execution of the application	Improves security by running your application in isolated containers
MQTT Explorer	Open source MQTT visualization and debugging tool	Verifies MQTT communication between the thermostat and the central system
	Provides monitoring and debugging of MQTT messages	Facilitates the identification and resolution of communication problems
	Intuitive graphical interface to represent MQTT communication	It allows to visualize in real time the messages and topics sent and received

Model

This section focuses on the conceptual model of the solution, boxing in red the part in which the proposed development is being intervened, as shown in ***Figure 41***.

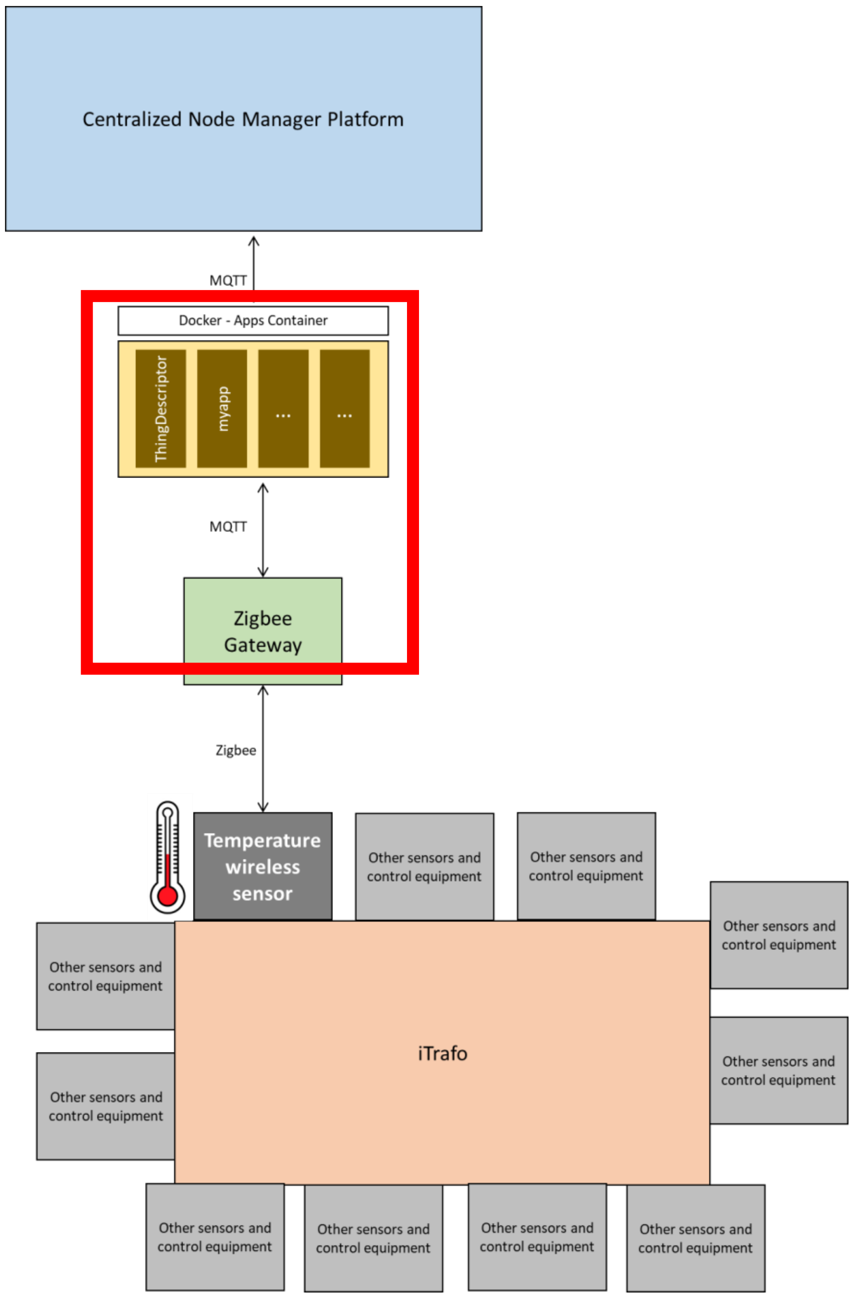

Centralized Node Manager Platform
MQTT
Docker - Apps Container
ThingDescriptor
myapp
...
...
MQTT
Zigbee Gateway
Zigbee
Temperature wireless sensor
Other sensors and control equipment
iTrafo

Figure 41: Development of Specification - Model

As has been advanced in the introduction of the development of the specification, an application will be developed that consists of reading signals by MQTT, fill a TD of a Transformer Temperature Sensor with the information of those messages that come from this type of device, and also publish it by MQTT, so that it can be read by other applications. This is the case of another developed application, which reads that TD and creates alerts when the temperature exceeds a previously determined value.

In the context of an ISS, a wireless temperature sensor is used to capture the outside temperature of the transformer. This device communicates via Zigbee, with the Zigbee Gateway.

The Zigbee gateway is a device that acts as an intermediary between Zigbee devices and other communication systems. In essence, the Zigbee gateway allows Zigbee devices to communicate with systems and applications that use other communication protocols, such as MQTT. Information collected from Zigbee devices, such as sensors or actuators, can be transmitted via MQTT to the central system.

Once IoT applications and devices have performed their calculations or carried out their actions, it is critical that the results and data generated are shared and communicated effectively. To achieve this, a "node and application management platform" is used. This platform acts as a control and coordination center for IoT infrastructure. This part of communication with the platform does not fall within the practical development of this work.

Design

To achieve the objective proposed in this project, two applications have been developed in Python: **ThingDescriptor.py** and **myapp.py.** Both will be explained below.

App 1: ThingDescriptor

This application reads by MQTT of the broker address = "192.168.1.174", and is subscribed to the topic "hola", from which it reads the information coming from the measuring devices. Then, publish the TD filled with that information, in the topic "adios", to which other control and monitoring applications will be subscribed.

The messages read are in the form below. This is a piece of data in JSON format. The beginning and end by [] represents that it is an array, within which different elements are presented in order. These elements have a key-value form, which allows information to be stored and accessed in a structured way. This is a way of associating related data, where each "key" acts as a unique identifier and each "value" represents the information or data associated with that specific key. There are a number of elements that will be key when reading the message by the application.

- First of all, the **profile** is Zigbee. This is because the information comes from a Zigbee device, which uses Zigbee wireless technology for communication and interaction with other devices within a personal area network.
- Second, the **device ID**. It is a unique code that identifies each device.
- In relation to the device ID, it also includes the **device type**. Among them are Transformer Temperature Sensors and other types of sensors. You will receive messages from all kinds of devices, and the application will be the one that filters those that interest you.
- Next, the **signal ID** and **signal type**. It is a unique code that identifies the signal sent.
- In relation to the previous one, the **type of signal**. Each device is configured to send a series of signals, such as ambient temperature, voltage, intensity, and other parameters. As for the type of device, messages from different types of signals will be received, and it will be the application that filters that or those signals that are important for the proposed application.
- Finally, the **number** element is important. Send the captured value.

```
[{
  "profile" : "zigbee",
  "deviceId" : "000D6F00040DA3D6",
  "signalId" : "0201A0000",
  "signal" : "LocalTemperature",
  "description" : "Report",
  "value" : "2420",
  "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
     "deviceType" : "THERMOSTAT",
     "number" : 21
   }]
```

The application reads such messages and fills in a TD model with that information.

When the TD is filled, it is published by MQTT in the same broker address, in the topic "adios".

Using MQTT Explorer, these messages that are published in "adios" can be read.

Next, the developed application, along with some comments to follow the code (#).

```
import json
import paho.mqtt.client as mqtt

# MQTT configuration
broker_address = "192.168.1.174"
topic = "hola"
topic_publish = "adios” # Topic for TD publications

# Creates a list to store filtered data
datos_filtrados = []

# Personalized TD
thing_descriptor = {
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:wot:com:ZIV:DCU001",
    "title": "Wireless temperature sensor TD”
    "@type": "dcu",
    "descriptions": {
           "en": "This is a Wireless Temperature Sensor virtualized instance.",
           "es": "Esta es una instancia virtualizada del termostato."
    },
    "version": {
        "instance": "0.5"
    },
    "created": "2023-07-15T18:00:00Z",
    "modified": "2023-08-10T14:05:00Z",
    "timezone": "CET",
    "securityDefinitions": {
        "nosc": {
            "scheme": "nosec",
            "in": "header"
        },
        "basc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": ["nosc"],
    "properties": {
        "state": {
            "name": "state",
            "title": "Application environment state",
              "description": "Returns whether the internal services of the application are running properly or not. Return values: Running, Idle, Error...",
            "type": "string",
            "enum": ["OK", "ERROR"],
            "writeOnly": False,
            "readOnly": False,
            "observable": True,
            "forms": [
                {
                    "op": "readproperty",
                     "href": "https://$IdService/api/v1/properties/state",
                    "htv:methodName": "GET",
                    "contentType": "application/json"
                }
            ]
        },
        "inventory": {
            "name": "inventory",
            "title": "TempInventory",
           "description": "Inventory of all elements (Q: are these just reports, or also events/alarms? Is there an E4S schema for inventory?)",
            "type": "object",
            "properties": {
                "idDevice": {
                    "type": "string",
                    "writeOnly": True,
                    "readOnly": False,
                    "observable": True
                },
                "profile": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "deviceId": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "signalId": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "signal": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "description": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "value": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "timeStamp": {
                    "type": "integer",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "timeStampInNanos": {
                    "type": "integer",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "deviceType": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "number": {
                    "type": "number",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                }
            },
            "writeOnly": False,
            "readOnly": False,
            "observable": True,
            "forms": [
                {
                    "op": "readproperty",
                     "href": "https://$IdService/api/v1/prop
erties/inventory",
                    "htv:methodName": "GET",
                    "contentType": "application/json"
                }
            ]
        },
        "currentProgrammedTasks": {
            "name": "currentProgrammedTasks",
            "title": "dcuProgrammedTasks",
            "description": "Provides all Programmed Tasks in
DCU in JSON object format",
            "type": "object",
            "properties": {
                "idDevice": {
                    "type": "string",
                    "writeOnly": True,
                    "readOnly": False,
                    "observable": True
                },
                "profile": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "deviceId": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "signalId": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "signal": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "description": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "value": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "timeStamp": {
                    "type": "integer",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "timeStampInNanos": {
                    "type": "integer",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "deviceType": {
                    "type": "string",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                },
                "number": {
                    "type": "number",
                    "writeOnly": False,
                    "readOnly": True,
                    "observable": True
                }
            },
            "writeOnly": False,
            "readOnly": False,
            "observable": True,
            "forms": [
                {
                    "op": "readproperty",
                     "href": "https://$IdService/api/v1/properties/currentProgrammedTasks",
                    "htv:methodName": "GET",
                    "contentType": "application/json"
                }
            ]
        }
    }
}

# Creates a function to fill the TD
def fill_thing_descriptor(data):
    thing_descriptor_filled = thing_descriptor.copy()
     thing_descriptor_filled["id"] = f"urn:dev:wot:com:ZIV:{data['deviceId']}"
     thing_descriptor_filled["properties"]["inventory"]["properties"]["profile"]["value"] = data.get("profile")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["deviceId"]["value"] = data.get("deviceId")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["signalId"]["value"] = data.get("signalId")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["signal"]["value"] = data.get("signal")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["description"]["value"] = data.get("description")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["value"]["value"] = data.get("value")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["timeStamp"]["value"] = data.get("timeStamp")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["timeStampInNanos"]["value"] = data.get("timeStampInNanos")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["deviceType"]["value"] = data.get("deviceType")
     thing_descriptor_filled["properties"]["inventory"]["properties"]["number"]["value"] = data.get("number")
    return thing_descriptor_filled

def on_connect(client, userdata, flags, rc):
    print("Conectado al broker MQTT")
    client.subscribe(topic)

def on_message(client, userdata, msg):
    m_decode = str(msg.payload.decode("utf-8", "ignore"))
    print("data Received type", type(m_decode))
    print("data Received", m_decode)
    print("Converting from Json to Object")
    data_list = json.loads(m_decode)  # decode json data

    for data in data_list:
        # Filter messages by deviceType and signal
        if data.get("deviceType") == "THERMOSTAT" and data.get("signal") == "LocalTemperature":
            datos_filtrados.append(data)

            # Constructio of TF with filtered data
            thing_descriptor_filled = fill_thing_descriptor(data)

            # Print the entire thing descriptor to the console
            print(json.dumps(thing_descriptor_filled, indent=4))
            # Publish the thing descriptor in the new topic
            client.publish(topic_publish, json.dumps(thing_descriptor_filled))

# Configuration of MQTT client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# Connection to MQTT broker
client.connect(broker_address, 1883, 60)
 # Keep the MQTT client running
client.loop_forever()
```

The proposed TD describes a virtualized instance of a wireless temperature sensor and the properties associated with it, such as its status, inventory, and current scheduled tasks. This descriptor provides a framework for understanding and accessing device properties and characteristics in a standardized manner, which is essential for interoperability in an IoT environment.

Again, its basic elements are remembered.

- **@context**: Defines the context of the document, in this case, version 1 of the W3C Thing Description standard.
- **id**: A unique identifier used to uniquely identify the described entity.
- **title**: A descriptive title of the entity.
- **@type**: Defines the type of the entity.
- **descriptions**: Provides descriptions of the entity in different languages.
- **version**: Indicates the version of the instance.

- **created**: The date the descriptor was created.
- **modified**: The date of the last modification of the descriptor.
- **securityDefinitions**: Defines the security schemes used to access the entity.
- **security**: Specifies which security schemes apply to this entity.

  To maximize security in the Thing Descriptor, multiple layers of security can be implemented. If you want to achieve a high level of security, you would opt for the following option:

```
"securityDefinitions": {
    "apiKey": {
        "scheme": "apiKey",
        "in": "header",
        "name": "Authorization"
    },
    "basicAuth": {
        "scheme": "basic",
        "in": "header"
    },
    "jwt": {
        "scheme": "bearer",
        "bearerFormat": "JWT",
        "in": "header",
        "alg": "RS256"
    }
},
"security": ["apiKey", "jwt"],
```

  Three security schemes have been defined:

  - **apiKey**: Uses an API token to authenticate requests. The token is passed in the HTTP request header under the name "Authorization".
  - **basicAuth:** Employs basic authentication, where a username and password are required in requests. The information is passed in the HTTP request header.
  - **jwt:** Uses JSON Web Tokens (JWT) to authenticate requests. JWTs are passed in the HTTP request header under the "bearer" scheme. In addition, the RS256 signature algorithm is used to ensure the integrity of the token.

- **properties**: Defines entity properties, such as "state", "inventory", and "currentProgrammedTasks". Each property has details such as name, type, whether it is write-only, read-only, whether it is observable, etc.
- **forms:** Defines how properties are accessed, in this case, through HTTP GET requests.

App 2: MyApp

This second application reads by MQTT from the broker address = "192.168.1.174", and from the topic = "goodbye", in which the previous application publishes the messages.

Of the whole message, it makes a filter of only taking into account those where the type of device is a Transformer Temperature Sensor, and also that send the Local Temperature signal. Extracting the value of it, it compares it with a maximum value established (in this case 20ºC) to do the tests.

It has been configured to send by MQTT to the same broker, but to the topic events, the corresponding message depending on the value of the temperature captured.

```
import paho.mqtt.client as mqtt
import json
from datetime import datetime

# Configuration of the MQTT connection
broker_address = "192.168.1.174"
topic_subscribe = "adios"
topic_publish = "events"  # New topic to publish messages

def on_connect(client, userdata, flags, rc):
    print("Conectado al broker MQTT")
    client.subscribe(topic_subscribe)

def on_message(client, userdata, msg):
    m_decode = str(msg.payload.decode("utf-8", "ignore"))
    print("Mensaje recibido:", m_decode)
    data = json.loads(m_decode)   # Decodes the JSON message
     number = data["properties"]["inventory"]["properties"]["number"]["value"]
     signal = data["properties"]["inventory"]["properties"]["signal"]["value"]

    t_max = 20  # Maximum value of temperature (limit)

            current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    if number > t_max:
        message = current_time + " - The measured value of the signal " + signal + " has reached its limit and has been recorded. Value captured: " + str(number) + " °C"
    else:
        message = current_time + " - The measured value of the signal " + signal + " is normal"

    # Post the message in the new topic
    client.publish(topic_publish, message)
```

```

# MQTT client configuration
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# MQTT broker connection
client.connect(broker_address, 1883, 60)

# Keep the MQTT client running
client.loop_forever()
```

Both applications have then been containerized using Docker, using Putty to connect to a remote server.

```
login as: minsait
minsait@192.168.1.174's password:*****
```

- A welcome message is displayed with information about the version and operating system, links to documentation, Ubuntu management tools and support, among others.

```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-45-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
         Receive updates to over 25,000 software packages with your
         Ubuntu Pro subscription. Free for personal use.

           https://ubuntu.com/pro

      El mantenimiento de seguridad expandido para Applications está desactivado

      Se pueden aplicar 102 actualizaciones de forma inmediata.
      Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

      Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
      Vea https://ubuntu.com/esm o ejecute «sudo pro status»


      The list of available updates is more than a week old.
      To check for new updates run: sudo apt update
      *** System restart required ***
      Last login: Thu Jul 27 10:31:00 2023 from 192.168.1.99
```

- In Unix-based operating systems, such as Ubuntu, "sudo su" is used to switch from the current user to the superuser account or root user, to obtain administrator privileges to perform tasks that require elevated permissions.

```
      minsait@VM-MLERENA:~$ sudo su
      [sudo] contraseña para minsait:****
```

- Next, a directory called "python_project_maria" is created, which is subsequently accessed, and where the ThingDescriptor.py file is created, that is, the first application previously. Then, the same is done with the other application, myapp.py.

```
      root@VM-MLERENA:/home/minsait# mkdir python_project_maria
      root@VM-MLERENA:/home/minsait# cd python_project_maria
```

```
32.      root@VM-MLERENA:/home/minsait/python_project_maria# nano
   ThingDescriptor.py
```

- (At this point the Code of the myapp app is entered)

```
33.      root@VM-MLERENA:/home/minsait/python_project_maria# nano myapp.py
```

- (At this point the Code of the myapp app is entered)
- A "requirements_thing_descriptor.txt" file is created, which, as its name suggests, includes requirements for the deployment of the application thing_descriptor, where the text "paho-mqtt" is entered, as well as the other necessary libraries for the functionality of said application. Same is done with "requirements_myapp.txt".

```
34.      root@VM-MLERENA:/home/minsait/python_project_maria# nano
   requirements_thing_descriptor.txt
35.      root@VM-MLERENA:/home/minsait/python_project_maria# nano
   requirements_myapp.txt
36.      root@VM-MLERENA:/home/minsait/python_project_maria# nano
   Dockerfile_thing_descriptor

   # Use a base Python image

   FROM python:3

   # Install libraries

   RUN pip install --no-cache-dir paho-mqtt datetime mqtt

   # Copy the application file to the container

   COPY ThingDescriptor.py requirements_thing_descriptor.txt /app/

   # Set the working directory

   WORKDIR /app

   # Run the app

   CMD ["python", "ThingDescriptor.py"]
```

- Now the "nano" command is used to create a file named "Dockerfile_myapp".

```
37.      root@VM-MLERENA:/home/minsait/python_project_maria# nano
   Dockerfile_myapp

   # Usa una imagen base de Python

   FROM python:3
```

```
    # Install libraries

    RUN pip install --no-cache-dir paho-mqtt datetime mqtt

    # Copy the application file to the container

    COPY myapp.py requirements_myapp.txt /app/

    # Set the working directory

    WORKDIR /app

    # Run the app

    CMD ["python", "myapp.py"]
```

- Next, the following steps of the process are performed:
  - Create images using "docker build", naming them "thing_descriptor_app_image:latest" and "myapp_image:latest".
  - Run containers based on those images with "docker run". The containers are named "thing_descriptor_container" and "myapp_container".

```
root@VM-MLERENA:/home/minsait/python_project_maria# docker build -t thing_descriptor_app_image:latest -f Dockerfile_thing_descriptor .
[+] Building 0.6s (9/9) FINISHED
root@VM-MLERENA:/home/minsait/python_project_maria# docker run -d --name thing_descriptor_container thing_descriptor_app_image:latest
6e72ccb33867881a8ced71e5e55e2100efb73d47d4bb083411e129f7e55362d0
root@VM-MLERENA:/home/minsait/python_project_maria# docker build -t myapp_image:latest -f Dockerfile_myapp .
[+] Building 0.5s (9/9) FINISHED

root@VM-MLERENA:/home/minsait/python_project_maria# docker run -d --name myapp_container myapp_image:latest
dd2473d82c997779afe0bcff55a76f92bf0e3e603f92e60cb3bc3dc69db47ae6
```

- Finally, it will be verified that the containers are running with "docker ps".

```
root@VM-MLERENA:/home/minsait/python_project_maria# docker ps
```

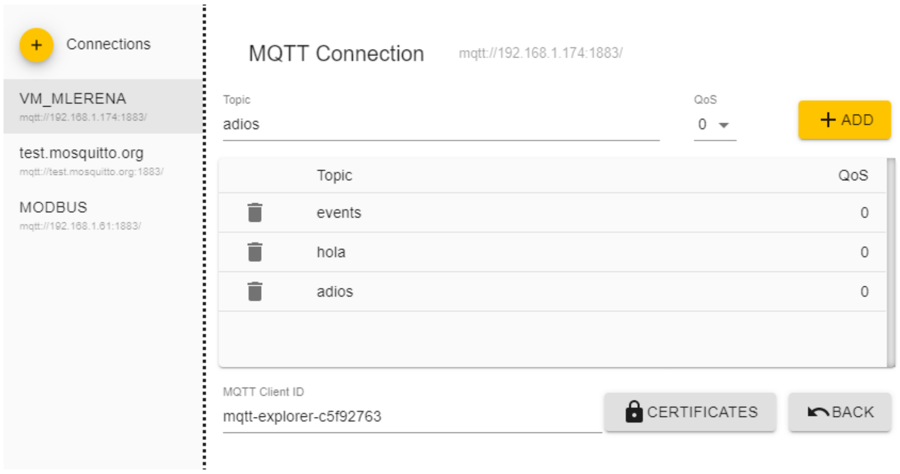
- Here it can be seen the containers, their status, image, and image id.

```
CONTAINER ID   IMAGE                COMMAND             CREATED          STATUS          PORTS     NAMES
dd2473d82c99   myapp_image:latest   "python myapp.py"   13 seconds ago   Up 12 seconds             myapp_container
```

```
50.      6e72ccb33867   thing_descriptor_app_image:latest
             "python ThingDescrip…"   4 minutes ago    Up 4
   minutes                                                thing_descriptor
   _container
51.      fb35bfe64c8a   emslab.onesaitplatform.com/onesait-things/edge-
   mqtt:2.0.0   "/docker-entrypoint.…"   2 weeks ago      Up 2
   weeks      0.0.0.0:1883->1883/tcp, :::1883->1883/tcp   edge.mqtt
```

Results

As previously mentioned, to visualize results and see that the applications work correctly, MQTT Explorer has been used. An MQTT connection to the server is established at the IP address 192.168.1.174, through port 1883, which is the standard port for MQTT. Within the advanced options, you configure the MQTT topics on which you want to view information or publish messages. These steps have been done and captured in *Figure 42*.


Connections
VM_MLERENA
mqtt://192.168.1.174:1883/
test.mosquitto.org
mqtt://test.mosquitto.org:1883/
MODBUS
mqtt://192.168.1.61:1883/
MQTT Connection
mqtt://192.168.1.174:1883/
ADD
CERTIFICATES
BACK

Topic: adios
QoS: 0

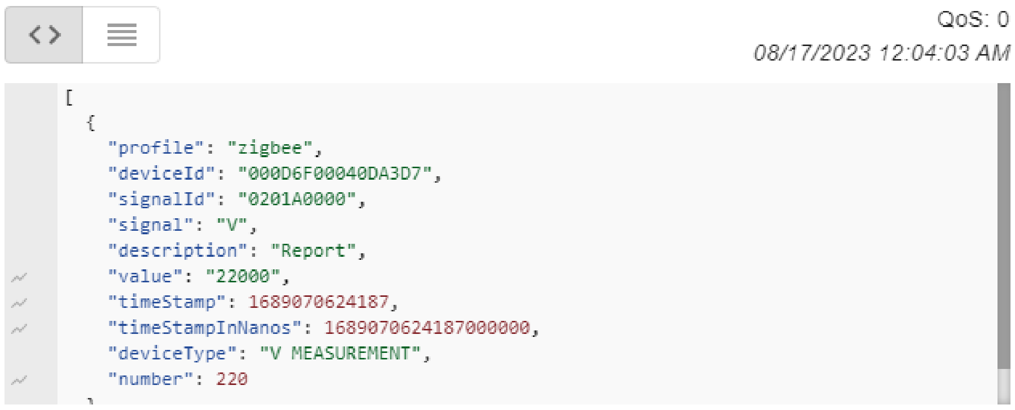
Topic	QoS
events	0
hola	0
adios	0

MQTT Client ID: mqtt-explorer-c5f92763

Figure 42: MQTT Explorer – Testing

To evaluate the work that has been carried out, a series of tests have been carried out.

- **Test #1**: A message in topic "hola" is read from a device other than a Transformer Temperature Sensor, shown in *Figure 43*.



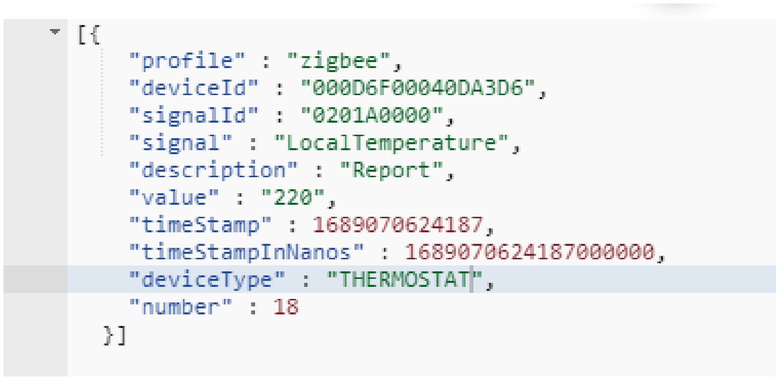
QoS: 0
08/17/2023 12:04:03 AM

```
[
  {
    "profile": "zigbee",
    "deviceId": "000D6F00040DA3D7",
    "signalId": "0201A0000",
    "signal": "V",
    "description": "Report",
    "value": "22000",
    "timeStamp": 1689070624187,
    "timeStampInNanos": 1689070624187000000,
    "deviceType": "V MEASUREMENT",
    "number": 220
```

Figure 43: Test #1 topic "hola"

As expected, in the topics "adios" and "events" there is no message, since, performing the filtering of the data, that information that does not come from Transformer Temperature Sensors, does not want to be taken into account for this application.

- **Test #2**: A message published in topic "hola" is read. The deviceType is now a Transformer Temperature Sensor, and whose temperature does not exceed the limit value of 20ºC, as shown in ***Figure 44***.

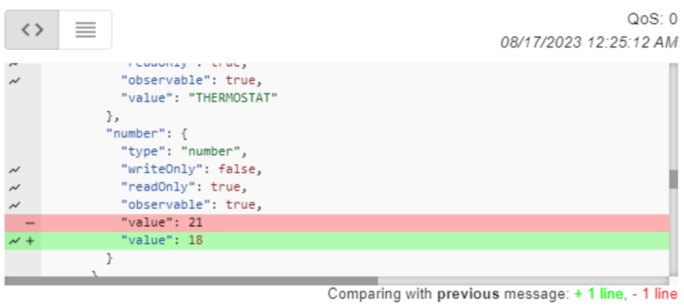
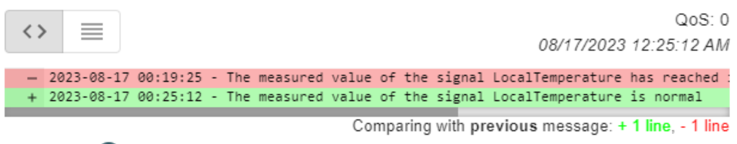


```
[{
    "profile" : "zigbee",
    "deviceId" : "000D6F00040DA3D6",
    "signalId" : "0201A0000",
    "signal" : "LocalTemperature",
    "description" : "Report",
    "value" : "220",
    "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
    "deviceType" : "THERMOSTAT",
    "number" : 18
}]
```

Figure 44: Test #2 topic "hola"

We see now how in topic "adios", a message has been published containing the TD filled with the information read from "hola", and in the topic "events" the corresponding message has been received for temperature measurement within the limits. Those are shown in ***Table 14***.

Table 14: Figures Test#2 topics adios (left) and "evets" (right)

Topic "adios"	Topic "events"
	

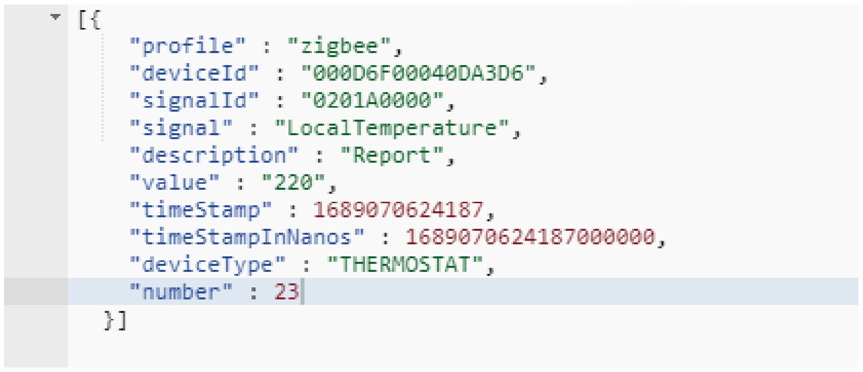
Topic "adios":

```
QoS: 0
08/17/2023 12:25:12 AM
        "observable": true,
        "value": "THERMOSTAT"
      },
      "number": {
        "type": "number",
        "writeOnly": false,
        "readOnly": true,
        "observable": true,
-       "value": 21
+       "value": 18
      }
Comparing with previous message: + 1 line, - 1 line
```

Topic "events":

```
QoS: 0
08/17/2023 12:25:12 AM
- 2023-08-17 00:19:25 - The measured value of the signal LocalTemperature has reached :
+ 2023-08-17 00:25:12 - The measured value of the signal LocalTemperature is normal
Comparing with previous message: + 1 line, - 1 line
```

- **Test #3**: A message published in topic "hola" is read. The deviceType is a Transformer Temperature Sensor, and whose temperature exceeds the limit value of 20ºC, as shown in ***Figure 45***.

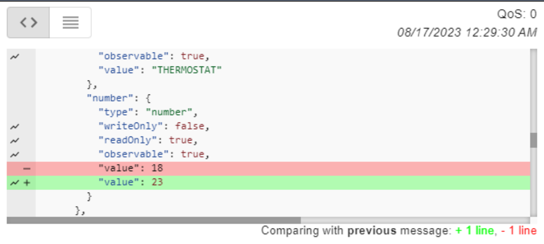
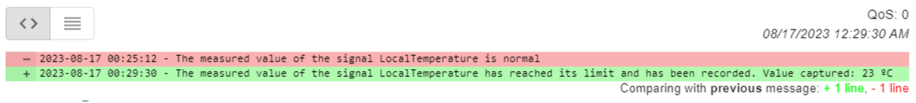


```
[{
    "profile" : "zigbee",
    "deviceId" : "000D6F00040DA3D6",
    "signalId" : "0201A0000",
    "signal" : "LocalTemperature",
    "description" : "Report",
    "value" : "220",
    "timeStamp" : 1689070624187,
    "timeStampInNanos" : 1689070624187000000,
    "deviceType" : "THERMOSTAT",
    "number" : 23
}]
```

Figure 45: Test #3 topic "hola"

In topic "adios" the TD now filled with the information read is published. In topic "events", the message that corresponds to a measure captured out of limits is published. These are shown in ***Table 15***.

Table 15: Figures Test#3 topics adios (left) and "evets" (right)

Topic "adios"	Topic "events"
	

```
QoS: 0
08/17/2023 12:29:30 AM
          "observable": true,
          "value": "THERMOSTAT"
        },
        "number": {
          "type": "number",
          "writeOnly": false,
          "readOnly": true,
          "observable": true,
 -        "value": 18
 +        "value": 23
        }
      },
Comparing with previous message: + 1 line, - 1 line
```

```
QoS: 0
08/17/2023 12:29:30 AM
 - 2023-08-17 00:25:12 - The measured value of the signal LocalTemperature is normal
 + 2023-08-17 00:29:30 - The measured value of the signal LocalTemperature has reached its limit and has been recorded. Value captured: 23 ºC
Comparing with previous message: + 1 line, - 1 line
```

- **Test #4**: Regarding the result of the containerization of the applications, as discussed in the previous design section, the results below are interpreted.

  As described, three containers have been created in the system, and all of them are running correctly.

  - A container named **"myapp_container"** from the image "myapp_image:latest". This container executes the command "python myapp.py" and has been running for 12 seconds.

- A container named **"thing_descriptor_container"** from the image "thing_descriptor_app_image:latest". This container executes the command "python ThingDescriptor.py" and has been running for 4 minutes.

- A container named **"edge.mqtt"** from the image "emslab.onesaitplatform.com/onesait-things/edge-mqtt:2.0.0". This container executes the "/docker-entrypoint.sh" command and has been running for 2 weeks. In addition, this container maps host port 1883 to internal port 1883, enabling MQTT communication.

```
7. CONTAINER
   ID   IMAGE                                                        COMMAN
   D                  CREATED          STATUS          PORTS
                           NAMES
8. dd2473d82c99   myapp_image:latest
      "python myapp.py"        13 seconds ago   Up 12
   seconds                                               myapp_container
9. 6e72ccb33867   thing_descriptor_app_image:latest
       "python ThingDescrip…"   4 minutes ago    Up 4
   minutes                                                thing_descriptor
   _container
10.      fb35bfe64c8a   emslab.onesaitplatform.com/onesait-things/edge-
   mqtt:2.0.0   "/docker-entrypoint.…"   2 weeks ago      Up 2
   weeks      0.0.0.0:1883->1883/tcp, :::1883->1883/tcp   edge.mqtt
```

The developed specification marks a milestone in the integration of IoT devices within the framework of WoT. With the addition of TDs, interoperability with a diverse range of IoT devices is ensured. Although initially conceived for a Transformer Temperature Sensor, its inherent adaptability makes it a versatile solution for different devices. This flexibility enables agile integration into the WoT ecosystem, which, in turn, streamlines deployment and management when containerizing applications. Ultimately, this approach enriches the usefulness of IoT devices by encouraging the exchange of data and alerts, promoting interoperability across diverse applications and platforms, regardless of vendor.

ECONOMIC IMPACT

It is important to focus the economic impact analysis in relation to distributed computing, since the approach of this project is similar in this aspect. Another more specific point is the savings due to the implementation of applications under an interoperability environment that will provide savings in maintenance costs, costs associated with the implementation of applications due to competition between suppliers and faster commissioning.

Economic impact related to distributed computing

Within the framework of technological evolution and the constant search for improvements in the efficiency of electricity networks, the proposal has arisen to incorporate Edge Node equipment to the LV network, with the aim of revolutionizing its operation and turning it into an intelligent infrastructure. This initiative not only represents a step towards the modernization of the electricity system, but also a challenge that entails a significant economic impact in different areas.

The incorporation of Edge Node equipment involves a series of strategic investments that are distributed in several crucial areas, as described below. These investments are summarized in ***Table 16***.

1. Installing Edge Node Equipment

The first investment focuses on the acquisition and installation of this equipment at strategic points of the low voltage network. These devices, which act as intermediate nodes between the end devices and the central infrastructure, enable real-time data processing and decentralized decision-making. This investment is categorized as CAPEX, as it is an initial investment for infrastructure.

2. Management Platform Deployment

To effectively operate and manage the new Edge Node equipment, the implementation of a management platform is required. The choice between a cloud-based or on-premise platform will affect acquisition, configuration and maintenance costs.

- In the cloud model, applications and services are hosted on servers and computing resources provided by cloud service providers. Choosing a cloud approach involves hosting the platform on the cloud provider's servers, which can deliver agility, rapid deployment, and automated updates.
- In the on-premises approach, organizations deploy and maintain their applications and services on their own on-premises infrastructure, owning and managing the physical servers, storage, and other resources in their physical location. Opting for an on-premises approach involves installing and configuring the platform on your own servers and computing resources. This provides greater control and customization but requires a larger initial investment in hardware and human resources for maintenance and upgrades.

3. Interoperability Certifications

Ensuring interoperability between new equipment and the existing environment involves testing and certification. This validation stage is essential to ensure smooth and secure operation of the smart infrastructure.

4. Licensing and Application Costs

These costs involve acquiring software licenses, including those that could be dockerized to facilitate deployment. OS and antivirus costs must be considered to support the new model. These costs can generate recurring and non-recurring expenses.

5. Required Components

Intelligent infrastructure is not limited to deploying Edge Node equipment, but also other necessary components, such as add-on nodes and switches. These elements are essential to establish the LV network as an intelligent and adaptable entity, and their costs are non-recurring.

6. Operation and Maintenance

Once the intelligent infrastructure is established, the operation and maintenance phase begin, involving recurring costs associated with the ongoing operation of the infrastructure, including the energy required for the equipment, preventive and corrective maintenance, as well as the management of the control and monitoring platform.

Table 16: Investments related to distributed computing

Aspect	Type of cost	Description
Installing Edge Node Equipment	CAPEX	Initial investment in the acquisition and strategic location of Edge Node equipment in the low voltage network
Management Platform Deployment	CAPEX	Implementation of a management platform for remote control and monitoring of equipment
Interoperability Certifications	CAPEX	Testing and certification to ensure interoperability between new equipment and existing environment
Licenses and Applications Costs	CAPEX	Acquisition of software licenses, including dockerized applications
Other necessary Components	CAPEX	Investment in additional nodes, switches and other components for intelligent infrastructure
Operation and Maintenance	OPEX	Recurring costs for continuous operation of the infrastructure. It includes energy, preventive/corrective maintenance and control platform management

In the process of transforming the low-voltage (LV) grid into a smart infrastructure through the integration of Edge Node equipment, a number of significant opportunities arise to generate savings and additional sources of income. These possibilities are the direct result of technological modernization, and of the operational efficiency and improvement in the quality of service that intelligent infrastructure can offer. Numerical estimations related to savings come from documentation and knowledge from Minsait. Opportunities are summarized in ***Table 17***.

1. Operation and Maintenance Efficiency

The addition of Edge Node equipment enables remote data monitoring, reducing costs associated with physical inspections and maintenance labor. By enabling remote maintenance and automatic deployment of solutions, recurring savings in OPEX are anticipated. It is estimated that this efficiency can result in an approximate saving of 20%, by reducing the resources and time dedicated to inspections and repairs in situ.

2. Reduction of Compensation for Breakdowns

Real-time control and management of the LV network through smart infrastructure minimizes the chances of overload and damage. This real-time monitoring also allows for the implementation of more effective preventive measures. As a result, a reduction in compensation for breakdowns is anticipated, which translates into recurring savings in OPEX by decreasing the costs associated with compensation for interruptions and damages to customers.

3. Claims Agent Costs

The ability to more effectively manage LV network reduces the need to involve third parties such as insurers and lawyers in the resolution of claims and disputes. Efficient management and the ability to provide real-time information can prevent problems and mitigate damage, leading to recurring savings in operating costs by reducing the need for legal and insurance services.

4. Reduction of Technical and Non-Technical Losses

The implementation of efficiency algorithms through Edge Node equipment allows a more precise control of load balancing in the LV network. This results in a decrease in technical and non-technical losses. The smart node facilitates efficient load balancing management, which in turn leads to an estimated 10% reduction in recurring operating expenses. This decrease in losses contributes directly to efficiency and savings in operating costs.

5. Savings in Central Computing, Storage and Communications

Efficient management of Edge Node equipment reduces the amount of data sent and processed centrally. This results in a decrease in compute, storage, and communications resource requirements. Tasks can be processed in parallel, significantly reducing the time needed to complete complex calculations. Faster processing times translate into faster task completion, which can have a positive impact on operational efficiency and resource utilization. Distributed computing systems can efficiently distribute tasks based on available resources, ensuring that each node is used effectively. This maximization of resource use can lead to cost savings and improved performance, decreasing recurring expenses in OPEX.

6. Investment Reduction

The transformation to an intelligent infrastructure offers the advantage of acquiring functionalities through a market open to various providers. Instead of investing in expensive individual high-performance servers, distributed computing allows organizations to use a low-cost commodity hardware cluster, resulting in savings in hardware acquisition and maintenance costs. This openness to other options lowers investment costs, as competition and a diversified supply can generate direct savings. In addition, the agility in the development of new applications demonstrated in the project, implies a substantial saving estimated at 80% in CAPEX, by accelerating the process of implementation and adaptation to changing demands.

7. Greater Market Agility

The rapid implementation of new applications and functionalities, which has been developed and so demonstrated in the project, has a positive impact on CAPEX. The agility to develop and deploy solutions responds to changing market needs effectively, reducing development times and associated costs, resulting in an estimated 80% CAPEX savings.

8. Integration of New Functionalities and Lower Consumption

The ability to quickly incorporate new functionality through intelligent infrastructure provides flexibility and efficiency. Utilities can scale their computing resources based on demand. During peak periods, additional compute nodes can be added, and during off-peak periods, excess nodes can be shut down. This dynamic scaling ensures efficient use of resources, avoiding unnecessary energy consumption and utility expenses. Agile integration and architecture optimization result in an estimated 25% recurring savings in OPEX.

9. Quality Improvement

The increase in network availability and reliability, achieved through intelligent infrastructure, translates into real-time operations and reduced downtime. This improves service quality and lowers OPEX by avoiding lengthy outages and costly repairs.

10. Network infrastructure cost savings

Distributed computing can optimize network bandwidth by bringing computation closer to the data source or by reducing data transfer between nodes, which can help save on network infrastructure costs.

Table 17: Savings related to Distributed Computing

Aspect	Type of cost	Description
Operation and Maintenance Efficiency	OPEX	Remote monitoring reduces inspection and labor costs. Estimated savings of 20% in OPEX by optimizing maintenance and repair processes
Reduction of Compensation for Breakdowns	OPEX	Real-time control minimizes damage and compensation for outages. Recurring savings in OPEX are anticipated
Agent costs for claims	OPEX	Efficient management reduces the need for third parties in claims resolution. Decrease in legal and insurance costs
Reduction of Technical and Non-Technical Losses	OPEX	Efficiency and load balancing algorithms reduce losses and improve efficiency. Estimated savings of 10% in OPEX
Savings in Central Computing, Storage and Communications	OPEX	Less centralized data optimizes compute, storage, and communications resources. Parallel processing saves time and improves efficiency
Reduction of Investments	CAPEX	Acquiring functionalities through an open marketplace reduces costs. Using basic clusters saves on hardware acquisition and maintenance. Estimated 20% CAPEX savings
Greater Market Agility	CAPEX	Rapid application deployment translates into 80% CAPEX savings. Effective adaptation to changing demands
Integration of new functionalities and lower consumption	OPEX	Agile incorporation of functionalities and efficient scaling of resources. Estimated savings of 25% in OPEX
Quality improvement	OPEX	Increased availability and reliability reduce downtime and repair costs
Network infrastructure cost savings	CAPEX	Distributed computing optimizes bandwidth and reduces network infrastructure costs

Economic impact related to interoperability

The development of applications under an interoperability standard can have several advantages for companies, understanding interoperability as the ability of different systems, applications, or devices to communicate, exchange data and work together seamlessly.

By focusing on interoperability, DSOs can achieve numerous benefits that will be detailed below and summarized in ***Table 18.***

One of the benefits of interoperability is to reduce development costs by allowing you to leverage existing software components, protocols, and standards, as opposed to implementing expensive custom solutions from scratch. Interoperable applications streamline data exchange, improving efficiency, productivity, and downtime. Due to its adaptability to technological changes and demands of utilities, it promotes scalability and operational expansion. By prioritizing interoperability, lock-in with suppliers is avoided and competition between them.

Table 18: Benefits of interoperability

Benefit	Description
Reduction of Development Costs	Leverages existing components and standards to reduce development costs compared to custom solutions
Faster data Exchange and communication	Interoperable applications streamline communication between systems, increasing efficiency, improving productivity, and reducing downtime
Flexibility and Scalability	Adaptable to technological changes and requirements, allowing to expand operations and add functionalities without significant interruptions
Facilitate collaboration	It fosters collaboration across teams, departments, and organizations by enabling seamless communication between systems
Improve customer experience	Integration of services and products improves the customer experience, increasing their satisfaction and loyalty
Avoid Vendor-lock-in	It avoids dependence on a specific supplier or technology, promoting competition between suppliers
Compatibility with technological advances	It ensures compatibility with emerging technologies and standards, minimizing obsolescence
Compliance	It can help meet regulatory regulations that require interoperability standards for data security and privacy
Easy integration with third parties	Easy integration with third-party services, APIs and platforms, leveraging specialized services
Greater data accuracy	Reduced errors and consistency of data by transferring it between disparate systems, resulting in greater accuracy and reliability
Long term benefits	Investing in interoperability benefits in the long run, keeping companies agile, competitive and prepared for future technological changes

ALIGNMENT WITH SUSTAINABLE DEVELOPMENT GOALS (SDGs)

Currently, there is a growing awareness of the need to address global challenges and promote sustainable development in society. In this context, the Sustainable Development Goals (SDGs) established by the United Nations in 2015, are presented as a comprehensive guide to address the most critical problems facing our planet, with the aim of achieving a more just, equitable and sustainable future.

Figure 46 highlights 5 SDGs with which this project is aligned. They are later summarized in ***Table 19***.

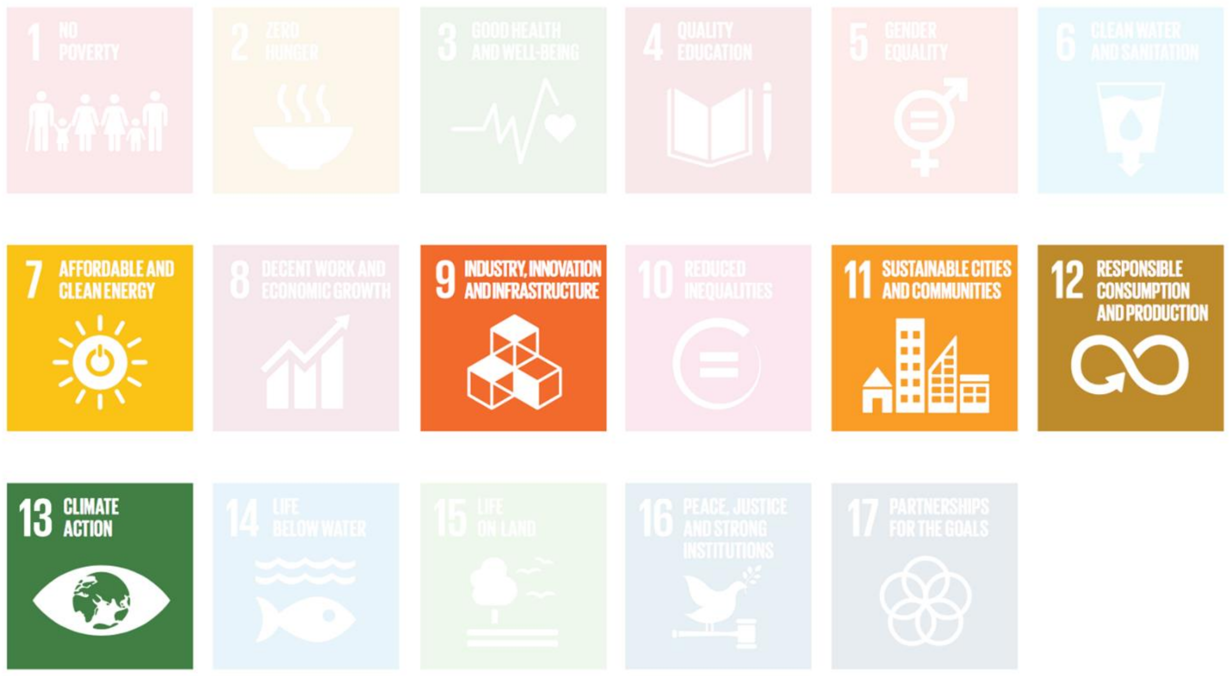

1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

Figure 46: Project alignment with SDGs

This project "Interoperability of applications in the context of Smart Grids" is aligned with goal number 7 of the global agenda: "Affordable and Non-Polluting Energy". [4]

This objective is focused on ensuring in the short-term universal access to affordable, secure, sustainable and modern energy. Sustainable Development Goal (SDG) number 7, for the year 2030, seeks to achieve several targets. These include increasing renewable energy generation, doubling the global rate of energy efficiency, promoting international cooperation to facilitate access to clean energy research and technology, promoting investments in energy infrastructure and clean technologies, and expanding infrastructure and improving technology to provide modern and sustainable energy services to all, especially in developing countries, least developed countries, small island developing States and landlocked developing countries.

Smart Grids are a promising solution to improve energy management and distribution. Not only allowing greater efficiency and reliability in the electricity supply, but facilitating the integration of renewable energy sources, and promoting the adoption of clean technologies in the energy sector. Effective interoperability between the different applications and devices involved in the management of the electricity grid is essential to optimize energy management and achieve a transition towards a more sustainable energy system.

This project is also aligned with SDG number 9, entitled "Industry, Innovation and Infrastructure". [5]

This objective seeks to encourage the development of sustainable infrastructure, promote sustainable industrialization, and encourage technological innovation.

It is closely related to the project "interoperability of applications in the context of smart grids". The project promotes the integration of advanced technologies in the energy sector, allowing to improve more efficient energy management, as well as better planning and management of energy infrastructure. In addition, it fosters collaboration between different industry players and innovation in the energy sector. This involves the development and adoption of common standards, communication protocols and data exchange systems, which facilitates the integration of new technologies and solutions in the field of smart grids.

In addition to the above two goals, the project also has important links with SDGs 11, 12 and 13.

SDG 11, "Sustainable Cities and Communities", aims to make cities and human settlements inclusive, safe, resilient and sustainable. [6]

By integrating smart technologies and renewable energy systems, it is possible to improve energy efficiency in urban environments and reduce emissions, which contributes to more sustainable cities.

SDG 12, "Responsible Consumption and Production", focuses on ensuring sustainable consumption and production patterns. [7]

The project is in line with this objective by promoting energy efficiency, the use of renewable energy and the optimization of energy production and distribution. With greater and better control and management of energy, more responsible and sustainable consumption is encouraged.

SDG 13 "Climate Action" seeks to take urgent action to combat climate change and its effects. [8]

The project directly contributes to this objective by facilitating the integration and use of renewable energy, reducing greenhouse gas emissions, and improving resilience to the impacts of climate change.

Table 19 summarizes the SDGs aligned with this project.

Table 19: Summary of project alignment with SDGs

ODS number	ODS Title	Project alignment with ODS
7	Affordable and clean energy	The project seeks to improve the management and distribution of energy, facilitating through new technologies the integration of renewable sources and clean technologies to achieve universal access to sustainable and affordable energy
9	Industry, Innovation and Infrastructure	The project promotes technological innovation in the energy sector, fosters collaboration between industry actors and improves the efficiency and planning of energy infrastructure
11	Sustainable cities and countries	By integrating smart technologies and renewable energy systems, the project improves energy efficiency in urban areas and contributes to more sustainable and resilient cities
12	Responsible Consumption and Production	The project promotes energy efficiency, the use of renewable energy and responsible energy management, promoting sustainable and responsible consumption
13	Climate Action	By integrating renewable energies and reducing emissions, the project contributes directly to the fight against climate change, improving resilience to its effects

CONCLUSIONS

This section will proceed to evaluate the objectives initially proposed for this work, in addition to compiling in a coherent way the conclusions derived from the segments dedicated to the **ANALYSIS** and **DEVELOPMENT OF SPECIFICATION** sections.

Regarding the objective of "**Study the role of application management platforms and Edge nodes to fulfill the vision of Smart grids**". The analysis reveals that application management platforms and edge nodes are essential for managing and monitoring smart grids. Specifically, edge nodes enable preliminary processing and analysis at the data generation site. Centralized management becomes crucial as IoT device deployments expand, enabling remote updates and an integrated environment for application development and debugging. The Onesait Phygital Edge platform, based on a virtualization and container architecture, proves to be an optimal solution for managing and distributing information across the energy value chain.

On the next objective, "**Study Application Virtualization as an interoperability driver**", it has been shown that application virtualization technology, as evidenced in the comparison between hypervisors and containers, promotes resource efficiency, simplified management, and portability in the context of IoT and Smart Grids. The choice of Docker as a virtualization tool is supported by its ease of use, portability, and scalability. These features align perfectly with the changing and diverse demands of smart grids.

On the third objective proposed, "**Research on the standards defined by WoT to achieve interoperability of applications**", it has been concluded that the implementation of WoT concepts allows to establish a framework for a fluid and efficient communication between various deployed applications. TDs facilitate a unified understanding of IoT devices in the SSP by promoting interoperability, and the WoT Directory serves as a central repository for application TDs, allowing them to be presented and collaborated effectively. The chosen communication pattern, together with the JSON-MQTT combination, improves interoperability and data exchange.

Finally, the objective "**To develop a specification of a Thing Description to enable interoperability of applications in the Secondary substation domain**". Through the development of the specification, the seamless integration of applications with hardware in secondary substations is verified. The creation of an application to fill the TD of a Transformer Temperature Sensor with data from IoT devices, and another that generates alerts according to temperature, underscores its practical usefulness. This flexibility fosters the adaptability of the solution to various IoT devices, as well as enriching data exchange and cooperation between applications and platforms, regardless of the provider.

RECOMMENDATIONS FOR FUTURE WORKS

The integration of IoT in Smart Grids, just with the research in the standards defined by WoT to achieve interoperability of applications in this context, has marked a milestone in the evolution of electrical infrastructures. Now, by exploring the possibilities of AI, new doors are opening to further drive efficiency, sustainability, and automation in the power grids of the future. In the following publication [100], information of great interest is presented to make some recommendations for future work in the field of SGs and AI.

Considering the increasing complexity of power grids, future work could focus on the implementation and optimization of AI techniques, such as artificial neural networks, genetic algorithms, and reinforcement learning, to improve the efficiency and automation of distributed resource management in SGs.

Future work could also address how to implement fully automated and self-learning systems in SGs, using machine learning algorithms. That would allow the network to adapt and learn from past operations for planning and failure prevention.

As SGs become more susceptible to cyberattacks, therefore, another idea would be to explore how artificial intelligence techniques, such as machine learning and anomaly detection, can strengthen cybersecurity and privacy in these networks, ensuring the confidentiality and integrity of critical data and operations.

Based on the analysis of WoT as an essential part for interoperability in SGs, it could be explored how AI techniques can improve communication and collaboration between IoT devices from different vendors, investigating how to facilitate the translation and adaptation of data between communication formats and different protocols.

Exploring how AI can be employed in optimizing resource management, security, automation in the context of SGs could lead to significant advances in the efficiency and reliability of the power grids of the future.

ANNEX 1: JSON

JavaScript Object Notation (JSON) [98-99] is a standard text format used to represent structured data in JavaScript object syntax. It is mainly used to transmit data in web applications.

This is a string formatted reminiscent of JavaScript literal objects. The basic data types that can be found within a JSON are strings, numbers, arrays, Booleans, and other object literals, which allows you to build a hierarchy of data, making it easier to find and access the data it contains.

As for the JSON syntax, it is based on:

- **Information arrangements**. It is done using keys.
- **Information**. As for the information it stores, it must be separated by commas.
- **Objects**. Keys are used to group objects.
- **Data fixes**. Brackets are used to group data arrays.

As mentioned, the basic data types that a JSON can contain are strings, numbers, arrays, Booleans, and other object literals.

- **Strings**. They must be enclosed in double quotation marks and the key must be a sequence of characters.

  {"Name":"Mary"} , or {"Age":"25"}

- **Objects**. When assigning objects to a value, key signs are used to contain the assembly information. About the information it contains, it must follow the format designated for Strings.

  {"Person":{"Name":"María", "Age":"25", "City":"Madrid"}}

- **Arrangements**. Arrays should be integrated in a way that is quite similar to the format and syntax of strings and numbers. In this case, square brackets are used to contain the data.

  {"Person":["Maria", 25, "Cristina"]}

- **Boolean**. These types of values are used in cases where the information can be either true or false.

{"Authorized":true}

- **Null**. This is a unique expression used to identify a key that is not yet assigned a value, but is still not empty. It is usually used to recognize a lack of value. As for Booleans, the result of this is null if there is no content, and data if there is.

  {"Name":null}

ANNEX 2: TD

When creating a valid TD the following sections must be included:

- **Root object**

  It is the starting point in a TD. Within this section, the unique identifier attribute provided by the system plays a key role in establishing connections between the application and other infrastructure components. It is important to emphasize that each application registered as a device that offers services must have its own identifier.

```
{
"@context": "https://www.w3.org/2019/wot/td/v1",
"type": "[topo|outage|DCU|AdvLVS|LVS]",
"id": "$IdService",
"description": "Información legible por humanos.",
"version": {"instance": "1.2.1"},
"created": "YYYY-MM-DDTHH:MM:SS",
"modified": "YYYY-MM-DDTHH:MM:SS",
"securityDefinitions": {
"no_sc": {
"scheme": "no-sec"
}
},
"security": ["no_sc"],
"properties": {...},
"actions": {...},
"events": {...},
"links": [...]
}
```

  - **@context**: It is the reference to the context of the Description of Things (TD) in the URL provided. It also indicates which version is being used.
  - **type**: Indicates the type of the application. It can be one of the following: topo (Topology), outage (Interrupts), DCU (Distribution Control Unit), AdvLVS (Advanced System for Reading Secondary Values), LVS (System for Reading Secondary Values), among others.
  - **id**: Represents the unique identifier of the application ($IdService), provided by the system.
  - **description**: Provides human-readable information about the app, such as its purpose or functionality or other relevant features.
  - **version**: Indicates the version of the application instance, which can be useful for tracking updates.

- **created y modified**: Estas fechas indican cuándo se creó y modificó por última vez la descripción de la aplicación.

- **securityDefinitions**: Defines the security definitions for the application. There are different types of security that can be used:

    - **Basic-Auth**. This scheme involves using a user name and password to authenticate users. The server verifies the credentials provided before allowing access to resources.

    - **OAuth (Open Authorization).** OAuth is a protocol that allows applications to gain limited access to user accounts in web services. Allows users to authorize an application to access their resources without sharing their full credentials.

    - **JWT (JSON Web Token).** JWT is an open standard that defines a compact, self-contained method for securely transmitting information between parties as a JSON object. It can be used for authentication and secure information transfer.

    - **TLS/SSL (Transport Layer Security/Secure Sockets Layer).** These are security protocols that ensure secure communications on the Internet. They use digital certificates to authenticate parties and encrypt data to keep it confidential.

    - **API Key.** API keys are unique codes used to authenticate requests to an API. Applications must provide the correct key to access services.

    - **Bearer Token.** It is a type of token used in OAuth. It is an access token that allows an application to access resources on behalf of the token holder without sharing the holder's credentials.

- **security**: Indicates the security measures applied according to the chosen scheme.

- **Properties**

They define key attributes that characterize the application. Two main properties are required for all applications:

- **Inventory**. This property provides a complete list of items that the application manages or controls. For example, you could include details about the devices or computers that the app handles.

- **currentGlobalParameters**. This property contains the current configuration parameters for the application. It can include tweaks or settings that affect how the app operates.

- Other Specific properties will be defined individually for each application..

```
"properties": {
    "temperature": {
        "type": "number",
        "description": "Current temperature reading",
        "unit": "Celsius",
        "readOnly": true
    },
    "humidity": {
        "type": "number",
        "description": "Current humidity level",
        "unit": "Percentage",
        "readOnly": true
    }
}
```

Each of the properties has a data type that it will store. The description property provides a brief description of the property, unit specifies the unit of measurement for the values it contains, and readOnly sets whether property can be read, read and modified by users.

- **Actions**

These are the operations that an application can perform or respond to based on the requests it receives.

- "instantaneousReport": This action is used to obtain real-time reports, in those cases that require quick updates on changing data, such as sensor readings. To generate this request, parameters such as the report code, start date, and end date of the query must be provided. The response is obtained in the body of the response and provides the requested report.

- "batchReport": This action is used when longer reports are required. Instead of receiving an immediate response, the response is scheduled via MQTT, including parameters such as the requester's IdService, report code, and query dates in the request. The response provides an idpet, or identification of the request that will be used to trace the response. Once the "idPet" is obtained, the requested report is received through MQTT.

```
"actions": [
    "instantReport" : {
        "description" : "",
        "input":{
            "cod" : "string",
            "from":"dateTime",
            "to":"dateTime",
            "required" : ["cod", "from","to"]
        },
        "forms": [
            {
                "op": "invokeaction",
                "href" : "https://$IdService/api/v1/instantReport",
                "htv:methodName": "GET",
                "contentType": "application/json"
            }
        ]
    }
]
```

Of the previous two explained, an "instantReport" action is defined.

- **description**. This property allows you to add a description of the action.
- **Input**. Defines the input parameters that the requestor must provide when invoking the action.
- **Forms**. This defines the ways in which the action can be invoked. For example, an "href" attribute contains the URL where the corresponding API to perform an action is located.

- **Events**

These are application-specific and so are defined individually.

```
"events": {
    "powerOutage": {
        "description": "Notifies when a power outage is detected",
        "data": {
            "type": "boolean",
            "description": "True if power outage detected, false otherwise"
        }
    },
    "deviceFailure": {
        "description": "Notifies when a device failure is detected",
        "data": {
            "type": "string",
            "description": "Details about the device failure"
        }
    }
}
```

The description of each event provides information about when this event will be reported. The "data" section describes the details of the event.

BIBLIOGRAPHY

[1] Y. Fan, 'Smart grids: From renewable sources to machine learning', Edu.au, 03-2021. [Online]. Available: https://www.eit.edu.au/wp-content/uploads/2021/04/310321_TechTopicWebinar_SmartGrids_Recording.pdf.

[2] I. Corporativa, '"Smart grids", un salto tecnológico para un mundo descarbonizado', Iberdrola, 22-Apr-2021. [Online]. Available: https://www.iberdrola.com/conocenos/nuestra-actividad/smart-grids.

[3] I. Colak, 'Introduction to smart grid', in 2016 International Smart Grid Workshop and Certificate Program (ISGWCP), 2016, pp. 1–5.

[4] 'ODS 7 Energía asequible y no contaminante', Pacto Mundial, 11-Oct-2021. [Online]. Available: https://www.pactomundial.org/ods/7-energia-asequible-y-no-contaminante/.

[5] 'ODS 9 Industria, innovación e infraestructura', Pacto Mundial, 11-Oct-2021. [Online]. Available: https://www.pactomundial.org/ods/9-industria-innovacion-e-infraestructura/.

[6] 'ODS 11 Ciudades y comunidades sostenibles', Pacto Mundial, 11-Oct-2021. [Online]. Available: https://www.pactomundial.org/ods/11-ciudades-y-comunidades-sostenibles/.

[7] 'ODS 12 Producción y consumo responsables', Pacto Mundial, 11-Oct-2021. [Online]. Available: https://www.pactomundial.org/ods/12-produccion-y-consumo-responsables/.

[8] 'ODS 13 Acción por el Clima', Pacto Mundial, 11-Oct-2021. [Online]. Available: https://www.pactomundial.org/ods/13-accion-por-el-clima/.

[9] Papaioannou I., Tarantola S., Lucas A., Kotsakis E., Marinopoulos A., Ginocchi M., Olariaga Guardiola M., Masera M., 'Smart grid interoperability testing methodology. JRC Technical Reports. Ioulia Papaioannou', 2018.

[10] Radatz J, Geraci A, Katki F (1990) IEEE standard glossary of software engineering terminology. IEEE Std 610121990(121990):3, Available: https://ieeexplore.ieee.org/document/159342

[11] A. Bröring, S. Schmid, C.-K. Schindhelm and A. Khelil, „Enabling IoT Ecosystems through Platform Interoperability," IEEE Software, 34(1), pp. 54- 61, 2017.

[12] O. Vermesan, Advancing IoT Platforms Interoperability. New York: River Publishers, 2022.

[13] A. Goudarzi, F. Ghayoor, M. Waseem, S. Fahad, y I. Traore, "A survey on IoT-enabled smart grids: Emerging, applications, challenges, and outlook", Energies, vol. 15, núm. 19, p. 6984, 2022.

[14] I. Corporativa, '¿Qué es el IIoT? Descubre el Internet Industrial de las Cosas', Iberdrola, 22-Apr-2021. [Online]. Available: https://www.iberdrola.com/innovacion/que-es-iiot.

[15] M. Zabaleta, 'Redes inteligentes e IoT industrial - Barbara', Barbaraiot.com. [Online]. Available: https://barbaraiot.com/es/blog/las-smart-grids-y-el-iot-industrial.

[16] 'AWS IoT', Amazon.com. [Online]. Available: https://docs.aws.amazon.com/es_es/iot/latest/developerguide/what-is-aws-iot.html.

[17] 'IoT Hub', Microsoft.com. [Online]. Available: https://azure.microsoft.com/es-es/products/iot-hub/.

[18] 'Google Cloud IoT Core documentation', Google Cloud. [Online]. Available: https://cloud.google.com/iot/docs.

[19] 'Oracle Internet of Things Cloud Service', Oracle Help Center, 31-Jan-2017. [Online]. Available: https://docs.oracle.com/en/cloud/paas/iot-cloud/index.html.

[20] 'Barbara - the cybersecure Industrial Edge Platform', Barbaraiot.com. [Online]. Available: https://barbaraiot.com/.

[21] 'IoT solutions: Connected & intelligent systems', Bosch Global. [Online]. Available: https://www.bosch.com/research/fields-of-innovation/connected-and-intelligent-systems/.

[22] 'Phygital Minsait', Minsait.com. [Online]. Available: https://www.minsait.com/es/aceleradores/phygital.

[23] "Grupo de trabajo de CT Inteligente - FutuRed. Plataforma española de redes eléctricas", FutuRed. Plataforma española de redes eléctricas, 24-mar-2021. [oniline]. Available in: https://www.futured.es/grupo-trabajo-ct-inteligente/.

[24] '¿Qué son las Smart Grids o redes inteligentes?', REPSOL, 17-Jul-2023. [Online]. Available: https://www.repsol.com/es/energia-futuro/tecnologia-innovacion/smart-grids/index.cshtml.

[25] Tokio School, 'Historia y evolución del Internet de las Cosas (IoT)', Tokio School, 19-Apr-2022. [Online]. Available: https://www.tokioschool.com/noticias/internet-de-las-cosas-evolucion/.

[26] Mahda Noura, Mohammed Atiquzzaman, Martin Gaedke: Interoperability in Internet of Things: Taxonomies and Open Challenges, Published online: 21 July 2018 in Mobile Networks and Applications (2019) 24:796 – 809. Available: https://doi.org/10.1007/s11036-018-1089-9

[27] B ISO/IEC 2382-1:1993 Information Technology – Vocabulary Part 1: Fundamental terms. International Organization for Standardization (ISO). Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=7229

[28] Radatz J, Geraci A, Katki F (1990) IEEE standard glossary of software engineering terminology. IEEE Std 610121990(121990):3, Available: https://ieeexplore.ieee.org/document/159342

[29] Kiljander J, D'Elia A, Morandi F, Hyttinen P, Takalo-Mattila J, Ylisaukko-Oja A, Soininen JP, Cinotti TS (2014) Semantic interoperability architecture for pervasive computing and internet of things. IEEE Access 2:856 – 873

[30] InterFuture – Future of Interoperability, for "ICT of the Future" programme by the Austrian Ministry for Transport, Innovation and Technology and the Austrian Research Promotion Agency (FFG). Authored by Mag. Kaltenbrunner Rainer – IDC, Mag. Neuschmid Julia – IDC, Dr. Bieber Ronald – OCG, DI Baumann Wilfried – OCG, Mag. Meir-Huber Mario.

[31] J. P. Vivanco, "El Centro de Transformación Inteligente, la clave para una red más sostenible, flexible y eficiente", Indracompany.com.

[32] Hannu Järvinen, Web Technology based Smart Home Interoperability, doctoral dissertation, Aalto University, School of Science, Department of Computer Science, Web Services Group. ISBN 978-952-60-6510-6 (pdf).

[33] Smart Grid Coordination Group, Smart Grid Reference Architecture, CEN-CENELEC-ETSI, Tech. Rep., 2012.

[34] Hugo Lueders Director para Europa de Asuntos Institucionales. Iniciativa Para La Elección De Software Software Choice. ISC, "Marco europeo de interoperabilidad", Bing..

[35] H. van der Veer and A. Wiles, „Achieving Technical Interoperability – the ETSI Approach," ETSI White Paper No.3, 3rd edition, April 2008.

[36] "Qué es la Web de las Cosas: WoT vs IoT", El Negocio Digital, 08-oct-2019. [online]. Available in: https://elnegociodigital.com/post/que-es-la-web-de-las-cosas-wot-vs-iot.

[37] R. Zafar, A. Mahmood, S. Razzaq, W. Ali, U. Naeem, y K. Shehzad, "Prosumer based energy management and sharing in smart grid", Renew. Sustain. Energy Rev., vol. 82, pp. 1675–1684, 2018.

[38] "MQTT - the standard for IoT messaging", Mqtt.org. [online] online in: https://mqtt.org/.

[39] "CoAP Protocol", Coap.space. [Online]. Available in: https://coap.space/.

[40] "AMQP Protocol", Amqp.org. [Online]. Available in: https://www.amqp.org/.

[41] "IEC 61850 – home", Iec.ch. [Online]. Available in: https://iec61850.dvl.iec.ch/.

[42] "Overview of DNP3 protocol," Dnp.org. [Online]. Available in: https://www.dnp.org/About/Overview-of-DNP3-Protocol.

[43] "IEC 60870-5-104", Iec.ch. [Online]. Available in: https://webstore.iec.ch/preview/info_iec60870-5-104%7Bed2.0%7Den_d.pdf.

[44] "HTTP - hypertext transfer protocol", Www.w3.org. [Online]. Available in: https://www.w3.org/Protocols/.

[45] "WHAT IS IOT," AMAZON.COM. [ONLINE]. AVAILABLE IN: HTTPS://AWS.AMAZON.COM/ES/WHAT-IS/IOT/.

[46] J. JOHNSON, "WHAT IS AN IOT ROUTER?", EMNIFY.COM, 09-NOV-2022.

[47] M. SAGAR, D. ENG, Y P. DIAMANDIS, "THE ROLE OF VIRTUALIZATION IN A SMART-GRID ENABLED SUBSTATION AUTOMATION SYSTEM", CONCOGRP.COM. [ONLINE]. AVAILABLE IN: HTTPS://WWW.CONCOGRP.COM/DOWNLOADS/WHITE-PAPERS/THE-ROLE-OF-VIRTUALIZATION-IN-A-SMART-GRID-ENABLED-SUBSTATION-AUTOMATION.PDF.

[48] M. S. V. KOTAK, "A REVIEW PAPER ON INTERNET OF THINGS(IOT) AND ITS APPLICATIONS", RESEARCHGATE.NET, JUN-2019. [ONLINE]. AVAILABLE IN: HTTPS://WWW.RESEARCHGATE.NET/PUBLICATION/341353106_A_REVIEW_PAPER_ON_INTERNET_OF_THINGSIOT_AND_ITS_APPLICATIONS.

[49] "BLUETOOTH® TECHNOLOGY WEBSITE – THE OFFICIAL WEBSITE FOR THE BLUETOOTH WIRELESS TECHNOLOGY. GET UP TO DATE SPECIFICATIONS, NEWS, AND DEVELOPMENT INFO", BLUETOOTH.COM. [ONLINE]. AVAILABLE IN: HTTPS://WWW.BLUETOOTH.COM/.

[50] I. AMIN Y A. SAEED, "5.10 WIRELESS TECHNOLOGIES IN ENERGY MANAGEMENT", EN COMPREHENSIVE ENERGY SYSTEMS, ELSEVIER, 2018, PP. 389–422.

[51] "WI-FI", 05-AGO-2019. [ONLINE]. AVAILABLE IN: HTTPS://WWW.CISCO.COM/C/ES_MX/PRODUCTS/WIRELESS/WHAT-IS-WIFI.HTML.

[52] "NFC FORUM", NFC-FORUM.ORG, NOV-2010. [ONLINE]. AVAILABLE IN: HTTPS://MEMBERS.NFC-FORUM.ORG/APPS/GROUP_PUBLIC/DOWNLOAD.PHP/14778/NFC8_DIGITALPROTOCOL.PDF.

[53] "RED DE DATOS CON NOMBRE", GOBETECH.COM. [ONLINE]. AVAILABLE IN: HTTPS://TECH.GOBETECH.COM/52432/QUE-ES-LA-RED-DE-DATOS-CON-NOMBRE.HTML.

[54] "MOBILEFIST SERVER. IBM DOCUMENTATION", IBM.COM, 05-MAR-2021. [ONLINE]. AVAILABLE IN: HTTPS://WWW.IBM.COM/DOCS/EN/MPF/8.0.0?TOPIC=OVERVIEW-MOBILEFIRST-SERVER.

[55] W. SHI, J. CAO, Q. ZHANG, Y. LI, Y L. XU, "EDGE COMPUTING: VISION AND CHALLENGES", IEEE INTERNET THINGS J., VOL. 3, NÚM. 5, PP. 637–646, 2016.

[56] S. SINGH Y N. SINGH, "CONTAINERS & DOCKER: EMERGING ROLES & FUTURE OF CLOUD TECHNOLOGY", EN 2016 2ND INTERNATIONAL CONFERENCE ON APPLIED AND THEORETICAL COMPUTING AND COMMUNICATION TECHNOLOGY (ICATCCT), 2016, PP. 804–807.

[57] "DOCUMENTATION - WEB OF THINGS (WOT)", WWW.W3.ORG. [ONLINE]. AVAILABLE IN: HTTPS://WWW.W3.ORG/WOT/DOCUMENTATION/.

[58] "WEB OF THINGS (WOT) ARCHITECTURE 1.1", WWW.W3.ORG. [ONLINE]. AVAILABLE IN: HTTPS://WWW.W3.ORG/TR/2023/PR-WOT-ARCHITECTURE11-20230711/.

[59] "MAKING THE WEB WORK", W3C. [ONLINE]. AVAILABLE IN: HTTPS://WWW.W3.ORG/.

[60] G. POLIZZI, "CÓMO PUEDE AYUDAR EL IOT A LAS UTILITIES ANTE EL RETO QUE SUPONEN LAS FUENTES DE GENERACIÓN DOMÉSTICAS", INDRACOMPANY.COM. [ONLINE]. AVAILABLE IN: HTTPS://WWW.INDRACOMPANY.COM/ES/BLOGNEO/AYUDAR-IOT-UTILITIES-RETO-SUPONEN-FUENTES-GENERACION-DOMESTICAS.

[61] "WEBINAR - INTEROPERABILITY: CONNECTING THE DOTS IN A FRAGMENTED DIGITAL ENERGY LANDSCAPE", 4E ENERGY EFFICIENT END-USE EQUIPMENT, 16-NOV-2022. [ONLINE]. AVAILABLE IN: HTTPS://WWW.IEA-4E.ORG/EDNA/NEWS/INTEROPERABILITY-CONNECTING-THE-DOTS-IN-A-FRAGMENTED-DIGITAL-ENERGY-LANDSCAPE-29-NOV-2022-1400-1500-PARIS-TIME-EDNA-AND-THE-IEAS-DIGITAL-DEMAND-DRIVEN-ELECTRICITY-NETWORKS-3DEN/.

[62] EUROPEAN PLATFORM INITIATIVE, "ADVANCING IOT PLATFORMS INTEROPERABILITY", IOT-EPI.EU, 2018. [ONLINE]. AVAILABLE IN: HTTPS://IOT-EPI.EU/WP-CONTENT/UPLOADS/2018/07/ADVANCING-IOT-PLATFORM-INTEROPERABILITY-2018-IOT-EPI.PDF.

[63] "BRIDGE THE GAP WITH IOT GATEWAYS", THALES GROUP. [ONLINE]. AVAILABLE IN: HTTPS://WWW.THALESGROUP.COM/EN/MARKETS/DIGITAL-IDENTITY-AND-SECURITY/IOT/INSPIRED/IOT-GATEWAY.

[64] "INTRODUCCIÓN A INTERNET DE LAS COSAS (IOT) DE AZURE - AZURE IOT", MICROSOFT.COM. [ONLINE]. AVAILABLE IN: HTTPS://LEARN.MICROSOFT.COM/ES-ES/AZURE/IOT/IOT-INTRODUCTION.

[65] "IOT EDGE STREAM ANALYTICS", CROSSER. [ONLINE]. AVAILABLE IN: HTTPS://CROSSER.IO/SOLUTIONS/IOT-EDGE-ANALYTICS/.

[66] "IEEE 802.3 ETHERNET", IEEE802.ORG. [ONLINE]. AVAILABLE IN: HTTPS://IEEE802.ORG/3/.

[67] "FILE TRANSFER PROTOCOL DOCUMENTATION", WWW.W3.ORG. [ONLINE]. AVAILABLE IN: HTTPS://WWW.W3.ORG/PROTOCOLS/RFC959/.

[68] W. M. EDDY, "RFC 9293: TRANSMISSION CONTROL PROTOCOL (TCP)", IETF.ORG. [ONLINE]. AVAILABLE IN: HTTPS://WWW.IETF.ORG/RFC/RFC9293.HTML.

[69] J. POSTEL, "USER DATAGRAM PROTOCOL", RFC EDITOR, 1980.

[70] "IPV6 - PROTOCOLO DE INTERNET VERSIÓN 6 - ¿QUÉ ES IPV6?", GOB.ES. [ONLINE]. AVAILABLE IN: HTTPS://IPV6.MINECO.GOB.ES/IPV6/PAGINAS/QUE-ES-IPV6.ASPX.

[71] R. ATKINSON Y S. N. BHATTI, "RFC 6747: ADDRESS RESOLUTION PROTOCOL (ARP) FOR THE IDENTIFIER-LOCATOR NETWORK PROTOCOL FOR IPV4 (ILNPV4)", IETF DATATRACKER, 10-NOV-2012. [ONLINE]. AVAILABLE IN: HTTPS://DATATRACKER.IETF.ORG/DOC/RFC6747/.

[72] "HTML", WHATWG.ORG. [EN LÍNEA]. DISPONIBLE EN: HTTPS://HTML.SPEC.WHATWG.ORG/MULTIPAGE/.

[73] "EXTENSIBLE MARKUP LANGUAGE (XML)", WWW.W3.ORG. [ONLINE]. AVAILABLE IN: HTTPS://WWW.W3.ORG/XML/.

[74] "INTRODUCTION TO ASN.1", ITU. [ONLINE]. AVAILABLE IN: HTTPS://WWW.ITU.INT/EN/ITU-T/ASN1/PAGES/INTRODUCTION.ASPX.

[75] "SOAP VERSION 1.2 PART 1: MESSAGING FRAMEWORK (SECOND EDITION)", WWW.W3.ORG. [ONLINE]. AVAILABLE IN: HTTPS://WWW.W3.ORG/TR/SOAP12-PART1/.

[76] "SNMP protocol introduction, definition, ports, MIBs, OIDs - SNMPcenter", the SNMP center, 08-feb-2020. [Online]. Available in: https://www.snmpcenter.com/what-is-snmp/.

[77] "Automatización de subestaciones eléctricas con IEC 61850", ETAP. [Online]. Available in: https://etap.com/es/product/iec-61850-substation-automation.

[78] PT7. Inteligencia del Dato ia4t", IA4TES. [Online]. Available in: https://www.bing.com/search?q=PT7.+Inteligencia+del+Dato+ia4t&qs=n&form=QBRE&sp=-1&lq=0&pq=pt7.+inteligencia+del+dato+ia4t&sc=1-31&sk=&cvid=21AEBF2676E14BAFA1A20AD80382C9A8&ghsh=0&ghacc=0&ghpl=.

[79] "IBM documentation - Common Information Model", Ibm.com, 31-ago-2021. [Online]. Available in: https://www.ibm.com/docs/en/i/7.1?topic=management-common-information-model.

[80] M. Poveda-Villalon, "SAREF: the Smart Applications REFerence ontology", Etsi.org. [Online]. Available in: https://saref.etsi.org/core/v3.1.1/.

[81] por:Barbara, "Protocolos de comunicación IoT que debes conocer", Barbaraiot.com. [Online]. Available in: https://barbaraiot.com/es/blog/protocolos-iot-que-deberias-conocer.

[82] "Modbus Protocol", Www.ni.com. [Online]. Available in: https://www.ni.com/es-es/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html.

[83] R. Kamal, "MQTT vs. AMQP: A head-to-head comparison of IoT protocols", Intuz.com. [Online]. Available in: https://www.intuz.com/blog/mqtt-vs-amqp-iot-protocols-you-must-know.

[84] "Transport Layer Security Protocol", Cloudflare.com. [Online]. Available in: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/.

[85] "OAuth 2.0 — OAuth", Oauth.net. [Online]. Available in: https://oauth.net/2/.

[86] M. Zabaleta, "El auge del Edge Computing en el IoT Industrial - Barbara", Barbaraiot.com. [Online]. Available in: https://barbaraiot.com/es/blog/el-auge-del-edge-computing-en-el-iot-industrial.

[87] "What is Application Virtualization?", VMware, 02-mar-2021. [Online]. Available in: https://www.vmware.com/es/topics/glossary/content/application-virtualization.html.

[88] "¿Qué es la virtualización?", Ibm.com. [Online]. Available in: https://www.ibm.com/es-es/topics/virtualization.

[89] S. Singh y N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology", en 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804–807.

[90] A. Cantos, "Plataformas IoT: qué son y cuál es la mejor para su negocio", Barbaraiot.com. [online]. Available in: https://barbaraiot.com/es/blog/plataformas-iot-que-son-y-como-pueden-beneficiar-a-tu-empresa.

[91] M. Zabaleta, "¿Qué son los nodos IoT y para qué sirven?", Barbaraiot.com. [online]. Available in: https://barbaraiot.com/es/blog/nodos-iot.

[92] "Platform Architecture at a Glance - platform-doc-en - Developer Portal", Atlassian.net. [online]. Available in: https://onesaitplatform.atlassian.net/wiki/spaces/DOCT/pages/2220813314/Platform+Architecture+at+a+Glance.

[93] Web of Things (WoT) Thing Description 1.1", Www.w3.org. [online]. Available in: https://www.w3.org/TR/2023/PR-wot-thing-description11-20230711/.

[94] "Wireless temperature sensor", Disruptive-technologies.com. [online]. Available in: https://www.disruptive-technologies.com/products/wireless-sensors/wireless-temperature-sensor.

[95] "Product downloads", Fortinet. [online]. Available in: https://www.fortinet.com/support/product-downloads.

[96] "Download PuTTY - a free SSH and telnet client for Windows", Putty.org. [online]. Available in: https://putty.org/.

[97] T. Nordquist, "MQTT explorer", MQTT Explorer. [online]. Available in: https://mqtt-explorer.com/.

[98] "Trabajando con JSON", Mozilla.org. [online]. Available in: https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON.

[99] M. Coppola, "JSON para principiantes: qué es, para qué sirve y ejemplos", Hubspot.es, 26-dic-2022. [online]. Available in: https://blog.hubspot.es/website/que-es-json.

[100] S. S. Ali y B. J. Choi, "State-of-the-art artificial intelligence techniques for distributed smart grids: A review", Electronics (Basel), vol. 9, núm. 6, p. 1030, 2020.

[101] Néstor Rodríguez Pérez, "Analysis of an edge-computing-based solution for local data processing at secondary substations", universidad pontificia icai, august 2020