PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EL IOT-PLC: UNA NUEVA GENERACIÓN DE CONTROLADORES LÓGICOS PROGRAMABLES PARA LA INDUSTRIA 4.0

JACOB ENRIQUE MELLADO ACEITÓN

Tesis presentada a la Dirección de Postgrado como parte de los requisitos para optar al grado de
Magíster en Ciencias de la Ingeniería

Profesor Supervisor:
FELIPE NÚÑEZ RETAMAL

Santiago de Chile, Noviembre, 2020

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EL IOT-PLC: UNA NUEVA GENERACIÓN DE CONTROLADORES LÓGICOS PROGRAMABLES PARA LA INDUSTRIA 4.0

JACOB ENRIQUE MELLADO ACEITÓN

Miembros del Comité:
FELIPE NÚÑEZ RETAMAL
ALDO CIPRIANO ZAMORANO
DIEGO DUJOVNE
CHRISTIAN LEDEZMA ARAYA

Tesis presentada a la Dirección de Investigación y Postgrado como parte de los requisitos para optar al grado de
Magíster en Ciencias de la Ingeniería

Santiago de Chile, Noviembre, 2020

“El jade nunca se convertirá en
una obra de arte sin ser tallado.”
Lao Zi

AGRADECIMIENTOS

Quiero agradecer especialmente a mi madre, a mi padre y a mi hermana por su apoyo y comprensión durante toda mi vida. También agradezco a mi profesor supervisor Felipe Núñez por guiarme durante estos años. A mis amigos Domingo, Felipe y Gabriel que me acompañaron durante los años que estuve en la universidad. Finalmente, quiero agradecer a todos los compañeros en la oficina por su ayuda y compañerismo.

Este trabajo ha sido apoyado por la Agencia Nacional de Investigación y Desarrollo (ANID) mediante el proyecto Anillo ACT192013.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El Controlador Lógico Programable (PLC) ha sido el elemento clave de los sistemas de control industrial a lo largo de toda la tercera revolución industrial, llamada revolución de la automatización, donde su papel ha sido principalmente comandar los lazos de control de bajo nivel regulatorio. Hoy en día, un nuevo paradigma conocido como Industria 4.0 está cambiando la forma en que las instalaciones industriales conciben sus sistemas de control industrial, promoviendo la conectividad generalizada entre sus componentes. A pesar de los grandes avances en las tecnologías de automatización, impulsados por el paradigma de la Industria 4.0 y su ecosistema hiperconectado, el PLC no ha visto todavía una versión modernizada que apunte a las funcionalidades que requiere un sistema de control orientado a la Industria 4.0. En este trabajo, un dispositivo llamado IoT-PLC es diseñado y prototipado, en un esfuerzo por generar un PLC a la medida de la revolución de la Industria 4.0. Para lograr un nuevo diseño, se realiza un estudio de los PLCs actuales y los protocolos industriales utilizados en el área de la automatización y el control. Además, se analizan las arquitecturas de referencia para la Industria 4.0 y las tecnologías emergentes. El IoT-PLC resultante es un dispositivo en el que cada funcionalidad trabaja dentro de un contenedor separado con recursos asignados para priorizar una tarea sobre otra. Este dispositivo IoT-PLC tiene capacidades de control, funcionalidades de computación de capa *fog* como filtrado y almacenamiento de datos de campo, y múltiples interfaces inalámbricas gestionadas independientemente por contenedores asignados a cada una de ellas. Finalmente, se presenta una evaluación del desempeño del prototipo.

Palabras Claves: Sistema Industriales Ciberfíssicos, Industria 4.0, Control Industrial, Internet de las Cosas, Computación fog.

ABSTRACT

The Programmable-logical-controller (PLC) has been the key building block of industrial control systems throughout the whole third industrial revolution, called automation revolution, where its role has been mainly to command low-level regulatory control loops. Nowadays, a new paradigm known as Industry 4.0 is changing the way industrial facilities conceive their industrial control systems by promoting pervasive connectivity between their components. Despite great advances in automation technologies, driven by the paradigm of the Industry 4.0 and its hyper-connected ecosystem, the PLC has not seen yet a modernized version targeting the functionalities an Industry 4.0-oriented control system requires. In this work, a device named IoT-PLC is designed and prototyped, in an effort to generate a PLC tailored for the Industry 4.0 revolution. In order to achieve a new design, a study of the current PLCs and industrial protocols used in the area of automation and control is carried out. In addition, the reference architectures for Industry 4.0 and emergent technologies are analyzed. The resulting IoT-PLC is a device in which each functionality works within a separate container with resources allocated to prioritize one task over another. This IoT-PLC device has control capabilities, fog-computing functionalities as filtering and storing field data, and multiple wireless interfaces managed independently by containers assigned to each of them. Finally, an evaluation of the prototype's performance is presented.

Keywords: Industrial Cyber-physical systems, Industry 4.0, Industrial control, Internet of Things, Fog computsing.

1. INTRODUCCIÓN

1.1. Motivación

En el último tiempo se ha visto un cambio tecnológico que sugiere que se está en proceso de transición hacia la cuarta revolución industrial o Industria 4.0. La Industria 4.0 no se basa en una sola tecnología, sino que en la idea de agrupar sistemas con tecnologías similares en torno al concepto de sistemas ciberfísicos (CPSs) y en particular al de sistemas ciberfísicos industriales (ICPSs), tal como se ve en la Figura 1.1, esta es la característica que diferencia a esta revolución tecnológica de las anteriores. La visión de la cuarta revolución industrial requiere que las unidades involucradas en los procesos sean más inteligentes, pequeñas y modularizadas en función de las tareas que desempeñan. Es por esto que se dice que los sistemas industriales del futuro van a ser moldeados para abordar la globalización, junto con la digitalización económica que demanda productos con mayor complejidad y personalización (Leitão et al., 2016).

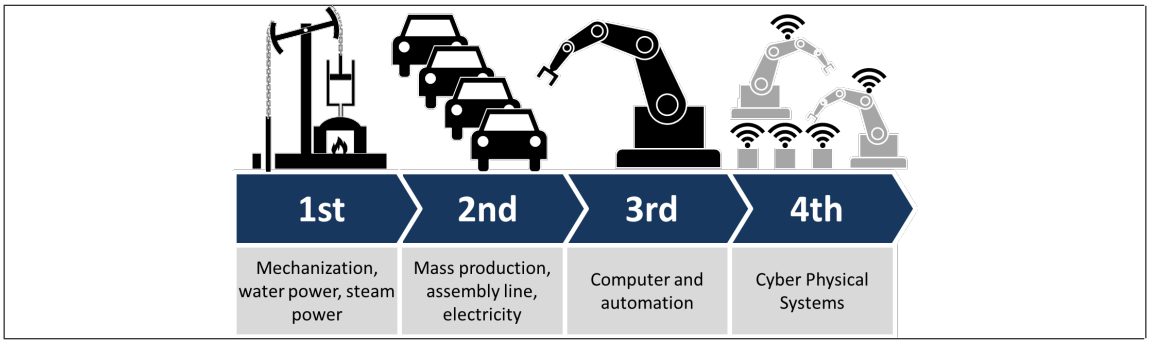


Figura 1.1. Revoluciones industriales y paso hacia la industria 4.0. Recuperado de: https://www.wikipedia.org/

Los actuales sistemas de control industrial (ICS) se desarrollaron en base al principio de separación de los problemas para poder manejar las complejidades de los sistemas, esto significó la obtención de una arquitectura con jerarquía vertical y con poca capacidad para la integración horizontal (Harrison, Vera, & Ahmad, 2016). En este tipo de arquitectura el flujo de información va desde los dispositivos de campo a los elementos de

control (*programmable logic controller* (PLC) , *supervisory control and data acquisition* (SCADA) y *distributed control system* (DCS)) y luego hacia los sistemas de administración (*manufacturing execution systems* (MESs), *production planning systems* (PPSs) o *enterprise resource planning* (ERP)). La integración de los sistemas de administración con los dispositivos de campo se ha utilizado para extraer datos de campo, transportarlos como información contextualizada hacia los niveles superiores para integrarla con la información administrativa y ayudar a la toma de decisiones de producción. Esta integración se ha hecho típicamente en cascada y dándole un grado de abstracción, según el estándar IEC 62264, para lograr esto se utiliza principalmente los protocolos OPC DA u OPC UA (Bangemann, Riedl, Thron, & Diedrich, 2016). Además, las funciones se implementan en redes separadas, en las que se emplean tecnologías alámbricas tradicionales basadas en los protocolos tipo Fieldbus y Ethernet (Galloway & Hancke, 2013) y se trabaja dentro de redes locales o con poco acceso desde el exterior.

Por otro lado, la visión de la Industria 4.0 está caracterizada por la inclusión de CPSs, los cuales combinan a los sistemas embebidos, con un comportamiento determinístico y a la nube que exhibe un comportamiento probabilístico y optimizado, con esto se pueden obtener sistemas colectivos descentralizados con la capacidad de autoorganizarse y tener mayor flexibilidad (Leitão et al., 2016). Por esto, la Industria 4.0 permite la combinación de tecnologías de automatización, tecnologías de la información (IT) y ciencias de la computación, así, la Industria 4.0 está relacionada con las ideas del Internet de las Cosas (IoT) (Bangemann et al., 2016).

En el caso particular de los ICPS, estos prometen revolucionar las plantas industriales de gran escala al permitir la adquisición a alta velocidad y el análisis eficiente de grandes cantidades de datos (Núñez, Langarica, Díaz, Torres, & Salas, 2020). Los ICPS son sistemas de control retroalimentado construidos sobre la aplicación de los principios de las tecnologías de la información y la comunicación (TIC) a los entornos industriales, y pueden considerarse como la próxima generación de ICSs (Harrison et al., 2016), con

los más avanzados protocolos de análisis y conectividad de datos en su núcleo (Leitão et al., 2016).

En base al concepto de CPSs, la integración es horizontal y vertical dentro de los sistemas de automatización, ya que involucran a un gran número de personas, sistemas IT y componentes de automatización (Harrison et al., 2016).

En la práctica los CPSs utilizan las redes para conectar dispositivos distribuidos y en la integración de los procesos la comunicación es una parte fundamental de la forma en que operan (Yao, Xu, and Liu (2016)), por esto los sistemas de comunicación industrial Ethernet o Fieldbus son mayormente utilizados, pero los sistemas inalámbricos industriales que se han desarrollado han empezado a utilizarse por las múltiples ventajas que conllevan, tales como menores costos de cableado, instalación en lugares de poco acceso y rápida integración de dispositivos (Galloway & Hancke, 2013).

En el proceso de transición hacia la industria 4.0 se requiere que las arquitecturas verticales utilizadas en los sistemas industriales convencionales deban ser revisadas y adaptadas para adecuarse al paradigma de la Industria 4.0, para que alcancen una integración tanto vertical como horizontal. De esta manera, los sistemas que ahora están en redes aisladas podrían interactuar directamente entre ellos y con agentes externos. Esto es importante, porque es necesario para intercambiar información entre distintos niveles durante la ejecución de procesos (Bangemann et al., 2016), los cuales se verían beneficiados por la interacción con otros procesos.

Dado que un sistema involucrado en la Industria 4.0 consiste de una gran variedad de componentes, se debe enfocar la ingeniería de los sistemas desde un nuevo punto de vista, para que los procesos de automatización puedan combinar de forma efectiva a todos los componentes, mientras se obtiene el comportamiento requerido. Debido a esto se han generado arquitecturas de referencia que involucran la estructura de desarrollo y operación de los sistemas, entre ellas está la Arquitectura de Referencia para la Industria 4.0 (RAMI) la que abarca un amplio rango de consideraciones útiles para la industria. De esta manera

los componentes que operan en el control industrial clásico podrían pasar de tener una gran cantidad de funciones a distribuir esas funciones a un conjunto de dispositivos de control, los que pueden ser dispositivos embebidos conectados a través de redes (Harrison et al., 2016).

En esta transición hacia la Industria 4.0 el PLC se encuentra entre los dispositivos potenciales a sufrir una transformación, este es una parte fundamental del área industrial, se encarga del control de los procesos, por medio de la recepción de señales que vienen de los sensores y así genera, mediante una ley de control, una señal de salida hacia los actuadores.

El PLC es un dispositivo que ha sido utilizado por mucho tiempo en el área industrial para controlar procesos, pero no ha visto un cambio en su modo fundamental de operar, ya que actualmente está principalmente diseñado para interactuar dentro del esquema de las tecnologías operacionales (OT), en el cual se utilizan diversos protocolos industriales ideados para el uso de actuadores y sensores. Por otra parte, un PLC orientado a la Industria 4.0 debe operar entre dos mundos: el OT y el de IT, porque debe utilizar redes de comunicación principalmente diseñadas para el área IT y, por otro lado, debe operar en el área industrial OT.

Los principales dominios del área IT son los ambientes corporativos y de hogares, en los que las funciones principales son las de transferir y procesar datos con dispositivos que utilizan los mismos protocolos. Además, no tienen restricciones de tiempo menores a los 50 ms y no necesitan que la información se reciba de forma determinística. En cambio, los dominios del área OT están relacionados con las industrias, en las que la tarea principal es la de controlar equipos físicos con restricciones de tiempo que pueden ser menores a los 10 ms y que necesitan que las redes sean determinísticas, junto con que la información que se envía es periódica y de unos pocos kilobytes (Galloway & Hancke, 2013).

Debido a estas diferencias en los dominios de trabajo, las áreas IT y OT se han desarrollado de manera paralela e independiente con poca integración entre ellas (Bangemann

et al., 2016) y han sido determinantes para que los PLCs aún no incluyan protocolos nuevos, porque las tecnologías más difundidas han sido diseñadas para los sectores IT, las cuales tienen características incompatibles con el área OT, por lo que al incorporar estas tecnologías IT, el PLC no podría cumplir con los requerimientos del área OT.

Como medida para solucionar el problema de la integración entre IT y OT, a la vez que se satisfacen sus requerimientos se ha presentado el paradigma *fog*, que en el ámbito de CPSs permite distribuir los sistemas de control, almacenaje y comunicación más cerca de los dispositivos finales. Con este paradigma se pueden disminuir los tiempos de respuesta, dado que se desplazan parte de las tareas que se realizan en el área IT como lo que se conoce ampliamente como nube, hacia zonas más cercanas a los sensores y actuadores. Las principales ventajas de *fog* son, además de los tiempos de respuesta, la descentralización de las tareas, la no existencia de un único punto de falla y el mejor aprovechamiento del ancho de banda para la transmisión de datos (Chiang & Zhang, 2016).

En este contexto, de migración hacia la Industria 4.0, se torna importante el flujo de la información entre los sensores y actuadores y los sistemas de administración, como el PLC se conecta con los primeros es factible reformularlo para que se incluya dentro de la capa *fog* para unir a los dispositivos de campo con la nube y sectores IT, así existiría un flujo eficiente de información entre la red de dispositivos interconectados.

Para que este nuevo PLC cumpla con las características necesarias para funcionar en este paradigma 4.0 debe cumplir con las restricciones que se requieren en el mundo OT, mientras se incluye a la nube y a las tareas de administración. De esta forma, el nuevo PLC debe ser capaz de interactuar con dispositivos que se comuniquen mediante protocolos diferentes y darles instrucciones de control, junto con procesar la información adquirida, extraer lo relevante y darle un formato estándar para transferirla hacia la nube, a la vez que se mantiene un funcionamiento robusto en caso de una falla en el funcionamiento de este nuevo PLC o externa a él.

1.2. Objetivo

El objetivo general de esta tesis es especificar, diseñar y fabricar un prototipo funcional de un IoT-PLC, el cual es una evolución del PLC. Este IoT-PLC estará basado en contenedores y trabajará como un nodo en la capa *fog* de forma que actúe como *gateway* entre los dispositivos que se comuniquen con protocolos diferentes y como controlador de los actuadores en base a la información que adquiere de los sensores y de lo que recibe de las capas más altas, como la nube. También, tendrá a cargo la tarea de extraer la información relevante que obtiene y enviarla como información procesada hacia capas superiores de forma estandarizada. Este IoT-PLC tendrá la capacidad de realizar las tareas clásicas de un PLC, junto con ser capaz de comunicar a través de conexiones inalámbricas (como Wi-Fi, Bluetooth, Zigbee, Thread e ISA100.11a) con dispositivos que pueden ser sensores, actuadores o sistemas de monitorización.

Para alcanzar el objetivo principal varios objetivos específicos deben ser abordados, estos se resumen en los siguientes puntos:

- Investigar arquitecturas de referencia en el ámbito de Industria 4.0 y arquitecturas basadas en el método de virtualización de sistemas operativos sin emular el *hardware*, denominado contenedor.
- Diseñar una arquitectura para el IoT-PLC utilizando contenedores.
- Obtener un modelo para la descripción de dispositivos y la forma en que se interactúa con ellos.
- Determinar el flujo de datos entre capas superiores e inferiores al IoT-PLC.
- Determinar el proceso de despliegue y migración de contenedores en los dispositivos de capa *fog*.
- Implementar y evaluar un prototipo de IoT-PLC.

1.3. Organización de la Tesis

El capítulo 2 presenta una introducción a los PLCs y los protocolos que utilizan los sistemas de control industriales en los que se involucran a los PLCs, tales como ISA-95, IEC 61131 y AutomationML. Luego, se continúa con los estándares de comunicación y los protocolos de comunicación que juntos son compatibles con la mayoría de los dispositivos industriales del área de la automatización.

El capítulo 3 trata a las arquitecturas de referencia que se han diseñado para ser utilizadas en la Industria 4.0, cada una con ejemplos de su aplicación y una proposición acerca de la inclusión de un PLC orientado a la Industria 4.0.

El capítulo 4 consiste en la introducción de tecnologías emergentes que podrían utilizarse en el ámbito de la Industria 4.0, como la virtualización de sistemas operativos mediante contenedores, protocolos de comunicación inalámbricos, bases de datos NoSQL y sus sistemas de administración más populares.

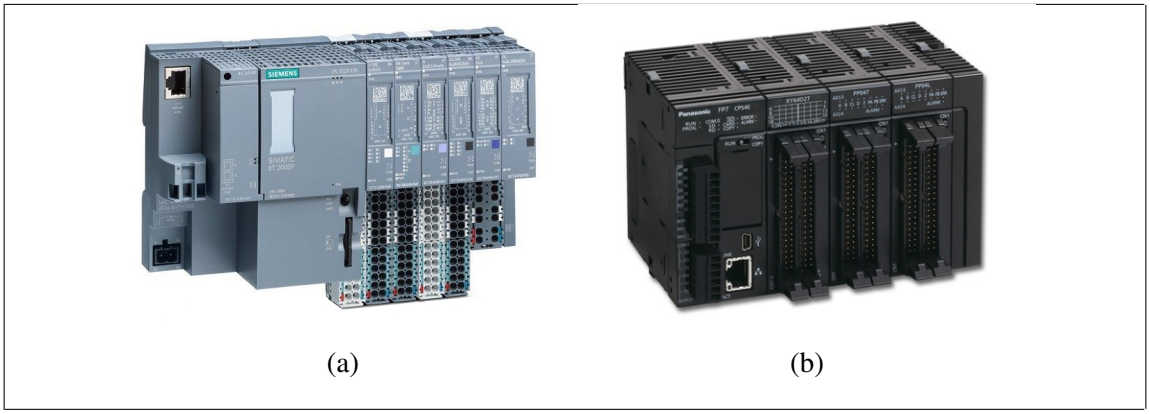
El capítulo 5 se presenta en detalle el diseño propuesto del IoT-PLC, con la descripción, objetivo y modo de actuar de las funcionalidades que incorpora. Finalmente, se realiza una comparación del IoT-PLC con los desarrollos propuestos en la literatura enfocados en modernizar al PLC y se finaliza con una comparación con los avances de los grandes fabricantes de PLCs en relación a la Industria 4.0.

El capítulo 6 consiste en la descripción del prototipo de IoT-PLC implementado, la planta en la que se evalúa el rendimiento de este nuevo PLC y la forma en que se realizaron las migraciones de los contenedores. Luego, se muestran los resultados y se realiza un análisis de ellos.

Finalmente, se presentan las conclusiones que se obtienen del trabajo realizado y las propuestas de trabajo futuro en esta linea de investigación.

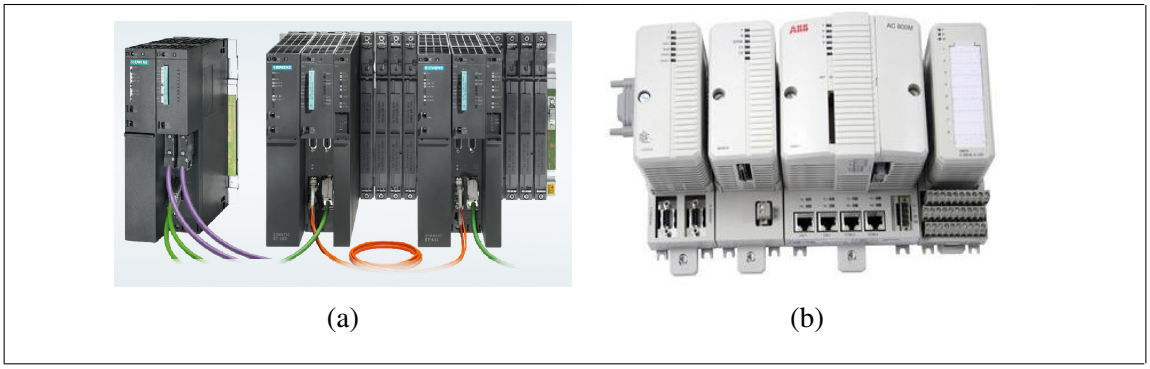
2. EL CONTROLADOR LÓGICO PROGRAMABLE EN EL CONTEXTO DEL CONTROL INDUSTRIAL

Un Controlador Lógico Programable (PLC) es esencialmente un computador digital que ha sido adaptado para funcionar dentro de entornos industriales como manufactura y líneas de ensamblaje, es un elemento que se utiliza por su alta confiabilidad en los procesos de control y facilidad de programación. Ejemplos de PLCs son el ET200 (Figura 2.1a) y el S7-400 (Figura 2.2a) de Siemens, el FP7 de panasonic (Figura 2.1b) y el PLC ABB serie 800 (Figura 2.2b).



(a) (b)

Figura 2.1. (a) PLC Siemens et200 y (b) PLC Panasonic serie FP7. Recuperados de: https://w3.siemens.com y https://www.panasonic-electric-works.com, respectivamente.



(a) (b)

Figura 2.2. (a) PLC Siemens S7-400 y (b) PLC ABB serie 800. Recuperados de: https://w3.siemens.com y https://new.abb.com, respectivamente.

Los ambientes industriales en los que opera el PLC requieren que, además de la robustez física que debiese tener este dispositivo, opere internamente de forma estable y sus medios de comunicación sean confiables en la transmisión de información. En la Tabla 2.1 se muestran varias características de una aplicación industrial dependiendo del contexto de operación, de esta tabla puede concluirse que el PLC debe funcionar en aplicaciones con amplios rangos en los requisitos de operación y confiabilidad, por ejemplo, en las fábricas y plantas de potencia el tiempo en que se requieren las señales de control es tan acotado ($<= 1$ ms) que debe tener la característica de ser determinístico y tener un pequeño rango de variación, lo que es fundamental para su funcionamiento y el buen rendimiento del control. Además, en (Gangakhedkar et al., 2018) se informa que el ancho de banda de los enlaces por los que se comunica un controlador con los dispositivos no debe ser menor a 1 Mbps en el caso de control de máquinas en movimiento y a 10 Mbps en caso de aplicaciones que utilicen robots móviles, con una variación en los tiempos de cada ciclo de operación menores a un 10%.

Tabla 2.1. Características de aplicaciones industriales en distintos contextos (Pang et al., 2017).

Aplicación	Período de cada ciclo de operación	Número de nodos	Confiabilidad
Construcción	10 segundos	10^3	Media
Procesos	100 milisegundos	10^4	Media
Fábricas	1 milisegundos	10^2	Alta
Sistemas de potencia	100 microsegundos	10^2	Alta
Electrónica de potencia	10 microsegundos	10^2	Muy alta

En este capítulo se describirá la estructura de un PLC industrial, en términos de *hardware*, las funciones que cumple el *software* incluido en cada una de sus partes y se explicará que cosas podrían cambiar de su actual estructura y funcionamiento para que pueda ser utilizado como un dispositivo de la Industria 4.0. Luego, se describirán los distintos estándares que actualmente se utilizan para definir las tareas y ubicaciones de los dispositivos en las plantas industriales, en términos de capas de una arquitectura y la interacción con otros elementos, estos son ISA-95, IEC 61131, AutomationML y IEC 61499. En específico, se mostrará como un PLC es definido en estos estándares y como estos restringen o fomentan a que el PLC se incluya en una planta que apunte a seguir los conceptos de la Industria 4.0 bajo estos estándares. Finalmente, se describirán los estándares y protocolos de comunicación más utilizados por los dispositivos industriales y los PLCs a los que se conectan ellos, estos son PROFIBUS, Modbus, PROFINET, OPC y OPC UA. Estos son importantes de estudiar, ya que son ampliamente usados en la industria, permiten la interoperatibilidad entre distintos fabricantes, permiten trabajar con las características de distintos ambientes industriales y un PLC orientado a la Industria 4.0 tendrá que ser compatible con estos estándares y protocolos para que pueda ser implementado y operado en un planta.

2.1. Estructura de un PLC

Los PLCs industriales tienen una estructura modular, en la que módulos con diferentes funciones se colocan en un chasis (también llamado *rack*). En este chasis la Unidad Central de Procesamiento (CPU) y otros módulos se personalizan para la aplicación concreta en que se quiera utilizar un PLC. Una sola CPU puede administrar más de un chasis y puede tener miles de entradas y salidas.

Los módulos de un PLC son por lo general intercambiables y separables, la principal ventaja de esto es que le da mayor flexibilidad de instalación y de funcionamiento según los requerimientos de los sistemas que se vayan a controlar. En la Figura 2.3 se

puede ver la estructura modular de un PLC, los módulos serán descritos en las siguientes subsecciones.

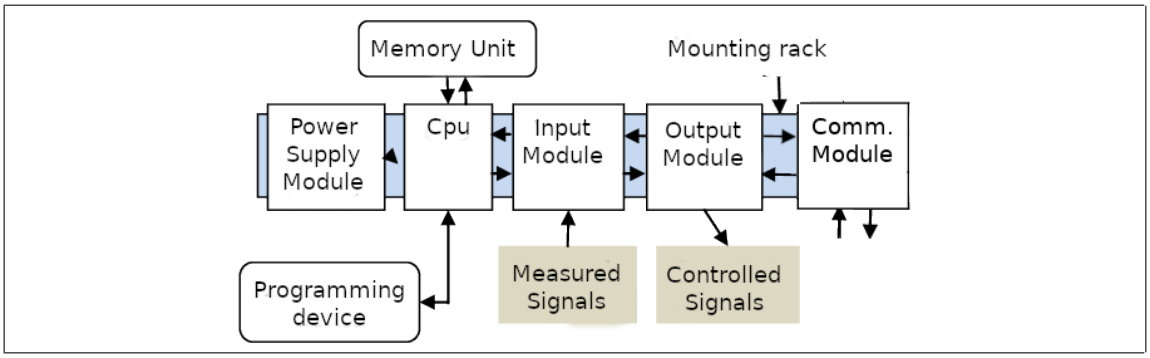


Figura 2.3. Módulos de un PLC industrial. (Varun & Thakur, 2015)

2.1.1. Módulo fuente de alimentación

La fuente de alimentación suministra energía a todos los módulos, convierte el voltaje de la línea AC en varios voltajes DC para que los circuitos electrónicos puedan operar. Además, filtra y regula los voltajes DC para asegurar el funcionamiento adecuado del PLC.

2.1.2. Módulo de señales de entrada y salida

Los módulos de señales de entrada y salida (I/O) establecen la interfaz entre los dispositivos físicos en el mundo real fuera del PLC y el plano digital dentro del PLC.

El módulo de entrada tiene un banco de terminales para conectar físicamente los dispositivos de entrada, como pulsadores e interruptores a un PLC. La función de un módulo de entrada es traducir las señales de los dispositivos de entrada a una forma que la CPU del PLC pueda entender.

El módulo de salida también tiene un banco de terminales para conectar físicamente los dispositivos de salida como solenoides y arrancadores de motor a un PLC. La función

de un módulo de salida es traducir las señales de la CPU del PLC a una forma que el dispositivo físico pueda utilizar.

2.1.3. Módulo Unidad Central de Procesamiento (CPU)

La CPU supervisa todas las operaciones del sistema y realiza todas las tareas necesarias para cumplir con las funciones del PLC. Es decir, realiza las siguientes tareas:

- Lee la información, es decir, el estado de los dispositivos de entrada conectados externamente con el módulo de entrada y la proveniente del módulo de comunicación.
- Almacena la información en la memoria para su uso posterior.
- Lleva a cabo operaciones matemáticas y lógicas como se especifica en el programa que se ejecuta.
- Después de ejecutar el programa, escribe los valores de los resultados en la memoria.
- Envía los datos a dispositivos externos como el módulo de salida y el de comunicación, para activar el *hardware* de campo.
- Realiza la comunicación entre los periféricos y los dispositivos externos.
- Realiza autodiagnósticos.

2.1.4. Unidad de Memoria

La unidad de memoria es el área de la CPU en la que se almacenan y recuperan los datos y la información. Esta unidad puede subdividirse en las siguientes cuatro partes:

- Memoria de imagen de I/O: La memoria de imagen de entrada consiste en las ubicaciones de memoria utilizadas para mantener los estados de cada dispositivo de campo de entrada. Mientras, que la memoria de imagen de salida consiste en ubicaciones de memoria que almacenan los estados de los dispositivos de salida. Como resultado de la resolución del programa del usuario los datos se almacenan

en el archivo de estado de salida y quedan a la espera de ser transferidos al módulo de salida.

- Memoria de datos: Se utiliza para almacenar los datos numéricos necesarios para el cálculo matemático.
- Memoria de usuario: Contiene el programa cargado por el usuario.
- Memoria de ejecución: Se utiliza para almacenar un programa de ejecución o *software* de sistema. Este es un programa especial que controla la acción de la CPU y, por consiguiente, la ejecución del programa del usuario. El *software* de sistema del PLC está diseñado para escanear la memoria de imágenes, interpretar las instrucciones del programa almacenado en la memoria y ejecutar los programas cargados. El *software* de sistema es suministrado por el fabricante del PLC y se mantiene permanentemente en esta memoria.

2.1.5. Dispositivo de programación

El dispositivo de programación se utiliza para comunicarse con los circuitos del PLC. Este dispositivo permite al usuario o programador entrar en la edición del programa a ejecutar, lo que permite escribir, ver y editar el programa y descargarlo en el PLC. También, permite al usuario monitorizar el PLC mientras está ejecutando el programa, con este sistema de monitoreo, cosas como las bobinas internas, registros, temporizadores y otros elementos no visibles externamente pueden ser monitoreados para determinar el funcionamiento adecuado. También, los datos de los registros internos pueden ser alterados, si es necesario para afinar el funcionamiento del programa mientras se depura.

Es importante mencionar que, aunque los conceptos fundamentales de la programación de PLCs son comunes a todos los fabricantes, las diferencias en el direccionamiento de I/O, la organización de la memoria y los conjuntos de instrucciones hacen que los programas de los PLCs nunca sean perfectamente interoperables entre los diferentes fabricantes. Incluso dentro de la misma línea de productos de un mismo fabricante, los diferentes modelos pueden no ser directamente compatibles.

2.1.6. Módulo de comunicación

Los PLCs tienen módulos de comunicación que pueden incorporar conectividad Ethernet, Modbus, DeviceNet, Profibus, entre otros. Por medio de este módulo la mayoría de los PLCs modernos pueden comunicarse a través de una red con algún otro sistema, como un dispositivo que ejecute un sistema SCADA o un navegador web.

Los PLCs utilizados en sistemas de I/O más grandes pueden tener comunicación *peer-to-peer* (P2P) entre CPUs. Esto permite que las partes separadas de un proceso complejo tengan un control individual, mientras que permite que los subsistemas se coordinen a través de un enlace de comunicación, estos enlaces de comunicación también se utilizan a menudo para conectar con dispositivos de interfaz hombre-máquina (HMI), empleadas para interactuar con las personas con el fin de configurar, informar sobre alarmas o controlar el funcionamiento de los sistemas, o estaciones de trabajo en un computador.

2.1.7. PLC en la Industria 4.0

El PLC con las características presentadas podría tener dificultades en su inclusión en un sistema de la Industria 4.0, porque las funciones de control sobre dispositivos de campo que se le asignan pueden cambiar rápidamente y deben interactuar con diversos tipos de equipos. En esto el PLC actual tiene falencias, ya que depende del dispositivo de programación para reprogramar sus funciones de control y tiene problemas de interoperabilidad entre fabricantes en el intercambio de información entre otros PLCs y dispositivos de capas superiores, tales como SCADAs o estaciones de ingeniería.

Estas falencias pueden ser mejoradas por un PLC operado por software (softPLC), ya que en él las funciones de control podrían ser automáticamente programadas y desplegadas y la interoperabilidad ser asegurada por estándares de programación del área IT que no dependan de los fabricantes, pero que aseguren confiabilidad en la operación en contextos del área OT. Además, con un softPLC el dispositivo de programación no sería

necesario, ya que el PLC podría configurarse mediante otro computador no especializado en la programación.

2.2. Estándares utilizados en sistemas de control industrial

En el área industrial se han creado distintos estándares que permitan mantener un grado de interoperabilidad entre los distintos elementos involucrados y definir las funciones de estos en un sistema, entre los más importantes, debido a su nivel de adopción, están ISA-95, IEC 61131 y AutomationML.

Además, debido al uso extendido de los PLCs en diferentes áreas se han diseñado estándares que tienen el objetivo de dar compatibilidad entre los lenguajes de programación que se utilizan, de estos estándares los más importantes son IEC 61131 parte 3 e IEC 61499. A continuación se presentan estos estándares.

2.2.1. ISA-95

La norma ISA-95 de Integración de Empresas/Sistemas de Control define los niveles de actividades en una organización, la pirámide que suele mostrar estos niveles se muestra en la Figura 2.4a, es el estándar de facto en cuanto a redes de intercambio de información industriales, dado que define específicamente las funciones que se realizan en cada nivel de su arquitectura, como deben transferirse los datos y al aplicar esta norma se logra definir eficientemente cada sistema de otro en un misma planta o una aparte. En general, los niveles de apoyo a la automatización y el control corresponden al 1 y 2, los sistemas de gestión de operaciones de fabricación (MOM) son de nivel 3 y los sistemas de planificación de recursos empresariales (ERP) son de nivel 4. ISA-95 define cuatro tipos principales de información que a menudo deben intercambiarse entre los sistemas MOM y entre los sistemas ERP y los sistemas MOM, estos tipos son: los materiales, el equipo, los activos físicos, el personal y sus funciones y calificaciones.

En este contexto, la norma ISA-95 se creó para permitir el intercambio y la coordinación eficientes de la información entre los diversos niveles, junto con posibilitar una manera estándar de describir la información para el intercambio entre ellos, incluidas las interrelaciones entre los diversos tipos de información.

El enfoque del modelo de información de ISA-95 se basa en un modelo de "Propiedad", los modelos de ISA-95 definen un conjunto mínimo de información independiente de la industria como atributos. La información específica de la industria, de la aplicación y de la empresa se representa como objetos de propiedad.

El PLC se encuentra en el nivel 1 de la norma ISA-95, por lo tanto, debe interactuar con los dispositivos de campo y con dispositivos de nivel 2 como los SCADAs. Un ejemplo de esto se puede apreciar en la Figura 2.4b, en donde se muestra un sistema de control en la que los dispositivos de campo se ubican en la parte inferior (nivel 0), estos dispositivos están conectados a PLCs (nivel 1) por medio de protocolos industriales y los controladores se comunican con sistemas de control que tienen una mayor amplitud (nivel 2).

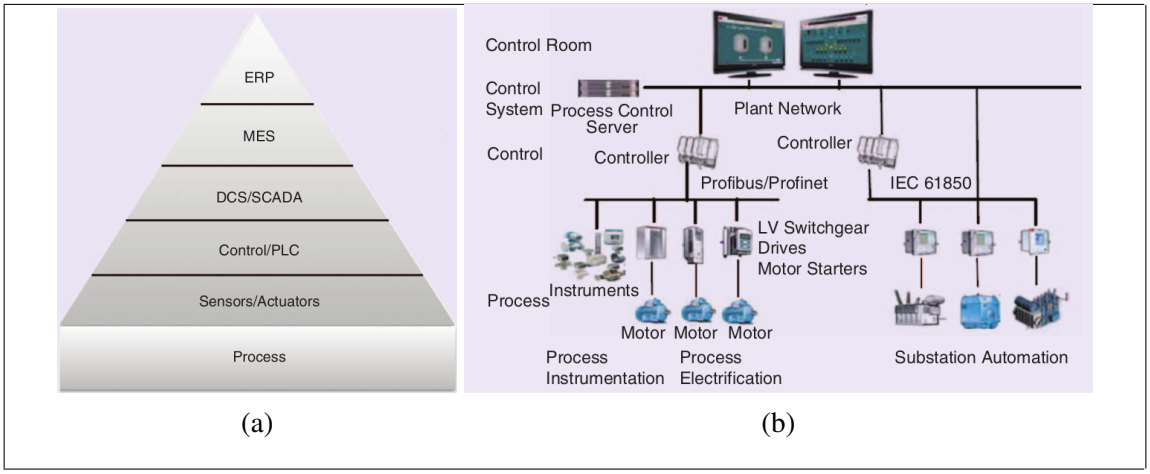


Figura 2.4. (a) Pirámide representativa del estándar ISA-95 y (b) Sistema de control con tres niveles del estándar ISA-95 (Delsing, 2017).

El PLC queda incluido en un sistema ISA-95 al definir los cuatro tipos principales de información. El primero, el de materiales, describe los materiales involucrados en el sistema de producción en el que está el PLC y los dispositivos de campo con que se comunica. El segundo, el de equipamiento, define el rol que tiene el PLC en el sistema de producción y se le asocia a los materiales y equipos. El tercero, el de activos físicos, identifica los recursos y características del PLC, se incluyen información del modelo, número de serial y sus propiedades. Por último, el cuarto tipo, el de personal, relaciona a los trabajadores involucrados y sus funciones con el rol del PLC en el sistema.

Para ejemplificar la inclusión de un PLC en un sistema ISA-95 se tomará como base al PLC de la izquierda de la Figura 2.4b. Aquí los cuatro tipos de información quedarían descritos por:

- Materiales: Los dispositivos de campo con los que se comunica el PLC serían los partidores para los motores y los *switches*, junto con los dispositivos de instrumentación.
- Equipamiento: El rol del PLC en este sistema podría ser la mantener la velocidad a la que giran los motores, por medio de los materiales mencionados en el punto anterior.
- Activos físicos: El modelo del PLC podría ser el FP7 de Panasonic, que utiliza sus capacidades de comunicación en los protocolos Profibus y PROFINET para la transferencia de datos con dispositivos de nivel 0, a la vez que utiliza la conectividad Ethernet para conectarse a la red de la planta (nivel 2).
- Personal: Los trabajadores de la planta podrían monitorizar el comportamiento del PLC y de los motores desde la sala de control. Además, podrían haber trabajadores que se encarguen de conectar los dispositivos de campo al PLC y operadores que regulen los parámetros del controlador.

Unas de las limitaciones que la norma ISA-95 impondría en un PLC enfocado en la Industria 4.0 sería la de la opción de interactuar directamente con otro PLC que está en el

mismo nivel, pero en un red aparte, ya que en vez de permitir una comunicación horizontal se tendría que seguir la estructura vertical de esta norma. De esta forma no puede haber una conexión directa entre PLCs, sino que entre PLCs y todos los dispositivos intermedios que pudiesen existir en la red, tales como SCADAs y *firewalls* de niveles superiores.

2.2.2. IEC 61131

IEC 61131 es un conjunto de normas e informes técnicos publicados con el objetivo de estandarizar a los PLCs y el es en el que actualmente se basan los fabricantes de PLCs para el desarrollo de sus dispositivos, el estándar está dividido en varias partes, estas se explican a continuación, en base a (John & Tiegelkamp, 2010).

- Parte 1: Información General: Esta parte contiene definiciones y características funcionales típicas que distinguen a un PLC de otros sistemas, se incluyen propiedades de este, tales como, procesamiento cíclico de las aplicaciones con almacenaje de los valores de entrada y salida o la división entre el trabajo de dispositivos de programación, PLC e interfaz humano-máquina (HMI).
- Parte 2: Requisitos y pruebas del equipamiento: Esta parte define las demandas eléctricas, mecánicas y funcionales de los dispositivos, así como las pruebas de calificación correspondientes. Aquí se enumeran las condiciones ambientales y las clases de estrés de los controladores y de los dispositivos de programación.
- Parte 3: Lenguajes de programación: Aquí los lenguajes de programación del PLC ampliamente utilizados en todo el mundo han sido organizados en una versión ajustada y orientada hacia el futuro. El modelo de software básico presentado y los lenguajes de programación se definen mediante definiciones formales, léxicas, sintácticas y descripciones semánticas. Los lenguajes de programación definidos son cinco: Diagrama de escalera, Diagrama de bloques de funciones (FBD), Texto estructurado, Lista de instrucciones y Bloques de función secuenciales.

- Parte 4: Guías del usuario: La cuarta parte está diseñada como una guía para ayudar al usuario del PLC en todas las fases de la automatización del proyecto. La información orientada a la práctica se da en temas que van desde el análisis de sistemas y la elección del equipamiento hasta la mantención.
- Parte 5: Comunicaciones: Esta parte respecta a la comunicación entre PLCs de diferentes fabricantes entre sí y con otros dispositivos. Se definen clases de conformidad para permitir que los PLCs se comuniquen, por ejemplo, a través de redes. Esta parte cubre las funciones de selección de dispositivos, intercambio de datos, procesamiento de alarmas, control de acceso y administración de red.
- Parte 6: Seguridad funcional: El objetivo de esta parte es la de adaptar los requisitos de la norma de seguridad IEC 61508 ("Seguridad funcional de sistemas eléctricos / electrónicos / electrónicos programables relacionados con la seguridad") y el requisito de la máquina IEC 62061 ("Seguridad de la maquinaria - Seguridad funcional de los sistemas de control eléctricos, electrónicos y programables relacionados con la seguridad") para los PLC.
- Parte 7: Programación de control difuso: El objetivo de esta parte de la norma es proporcionar a los fabricantes y usuarios un entendimiento común de la integración de las aplicaciones de control difuso basadas en IEC 61131-3 y facilitar la portabilidad de los programas difusos entre diferentes fabricantes.
- Parte 8: Directrices para la aplicación e implementación de lenguajes de programación: Esta parte ofrece interpretaciones para preguntas no respondidas por la norma, se incluyen pautas de implementación, instrucciones para el uso por parte del usuario final, así como asistencia en la programación.

Aunque este estándar no está enfocado en la Industria 4.0, no limita el actuar de los componentes en un sistema, en cuanto a las interacciones de jerarquía horizontal y vertical, además, se dan pautas de cómo implementar nuevas lenguajes. Sin embargo, no se tiene considerado que un PLC del futuro pueda incorporar más funciones que solo las de control y las relacionadas a una HMI (por ejemplo, base de datos y servidor OPC) y que estas

funcionen de manera paralela y con distintos lenguajes de programación, tal como se podría hacer en un computador de escritorio.

2.2.3. AutomationML

Esta especificación fue creada por el grupo de trabajo conformado por la Fundación OPC y AutomationML e.V.. El objetivo de AutomationML es interconectar las herramientas de ingeniería en sus diferentes disciplinas, por ejemplo, la planificación de plantas, la ingeniería mecánica, la ingeniería eléctrica y la programación de PLC.

El formato de datos AutomationML, normalizado en la norma IEC 62714, es un formato de intercambio de datos abierto, neutro, basado en XML y libre que permite a un dominio o una empresa englobar la transferencia de datos de ingeniería de los sistemas de producción en un panorama heterogéneo de herramientas de ingeniería. AutomationML permite modelar los componentes físicos y lógicos de las plantas como objetos de datos que implican diferentes aspectos. Un objeto puede estar formado por otros subobjetos y puede formar parte, a su vez, de una composición o agregación más amplia. Los objetos típicos en la automatización de plantas comprenden información sobre topología, geometría, cinemática, lógica y comportamiento. Lógicamente AutomationML está dividido en:

- Descripción de la estructura de la planta y los sistemas de comunicación expresados como una jerarquía de objetos AutomationML y descritos por medio del CAEX siguiendo la norma IEC 62424.
- Descripción de la geometría y la cinemática de los diferentes objetos AutomationML representados por medio de COLLADA 1.4.1 y 1.5.0 (ISO/PAS 17506:2012).
- Descripción de los datos lógicos relacionados con el control de los diferentes objetos AutomationML representados mediante PLCopen XML 2.0 y 2.0.1 (IEC61131-10).

- Descripción de las relaciones entre los objetos AutomationML y las referencias a la información que se almacena en documentos fuera del formato de nivel principal.

Los modelos AutomationML se dividen en diferentes árboles jerárquicos: InstanceHierarchy, RoleClassLibs, SystemUnit-ClassLib, e InterfaceClassLib. La organización en un árbol jerárquico implica que cada tipo de elemento puede tener elementos hijos del mismo tipo.

Un PLC para la Industria 4.0 podría ser perfectamente descrito en AutomationML, ya que las relaciones entre los objetos no tienen restricciones, sino que son diseñadas y especificadas por los desarrolladores del sistema de control. Por lo tanto, se tiene la flexibilidad de definir relaciones horizontales y verticales entre dispositivos, así se tienen la posibilidad de interconectar en AutomationML distintos sistemas que tengan un objetivo en común.

En la Figura 2.5 se muestra un fábrica de manufactura con diversas máquinas que depeden del trabajo de las otras para generar el producto final, la representación de este sistema en árboles jerárquicos de un modelo AutomationML se muestra en la Figura 2.6. En el árbol InstanceHierarchy la línea de producción está en un nivel más alto que las máquinas que componen al sistema y cada máquina tiene descripciones de sus relaciones con las actividades que debe realizar. En el árbol RoleClassLibs se definen los distintos roles que toman los dispositivos en el sistema y sus clases internas. En el árbol SystemUnit-ClassLib cada máquina de este sistema es definido, descrito y se le añaden las interfaces que tienen entre los dispositivos del sistema. Por último, en el árbol InterfaceClassLib se definen de forma individual las interfaces y enlaces que existen en el sistema, pero sin asociarlo directamente con los elementos de la fábrica.

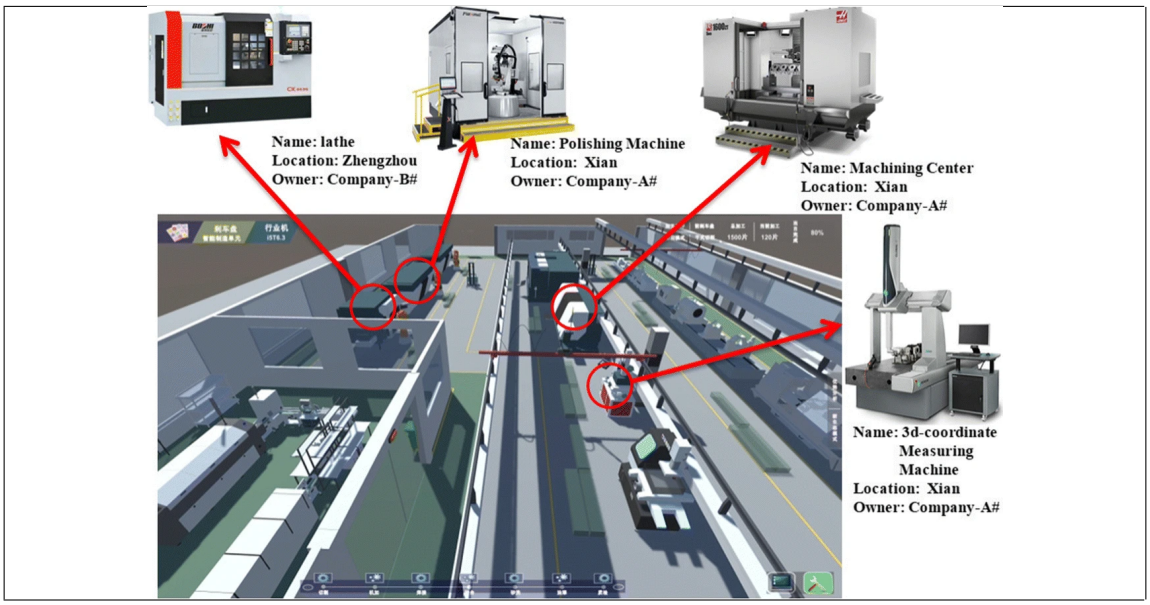


Figura 2.5. Fábrica de manufactura y sus máquinas principales (H. Zhang et al., 2020).

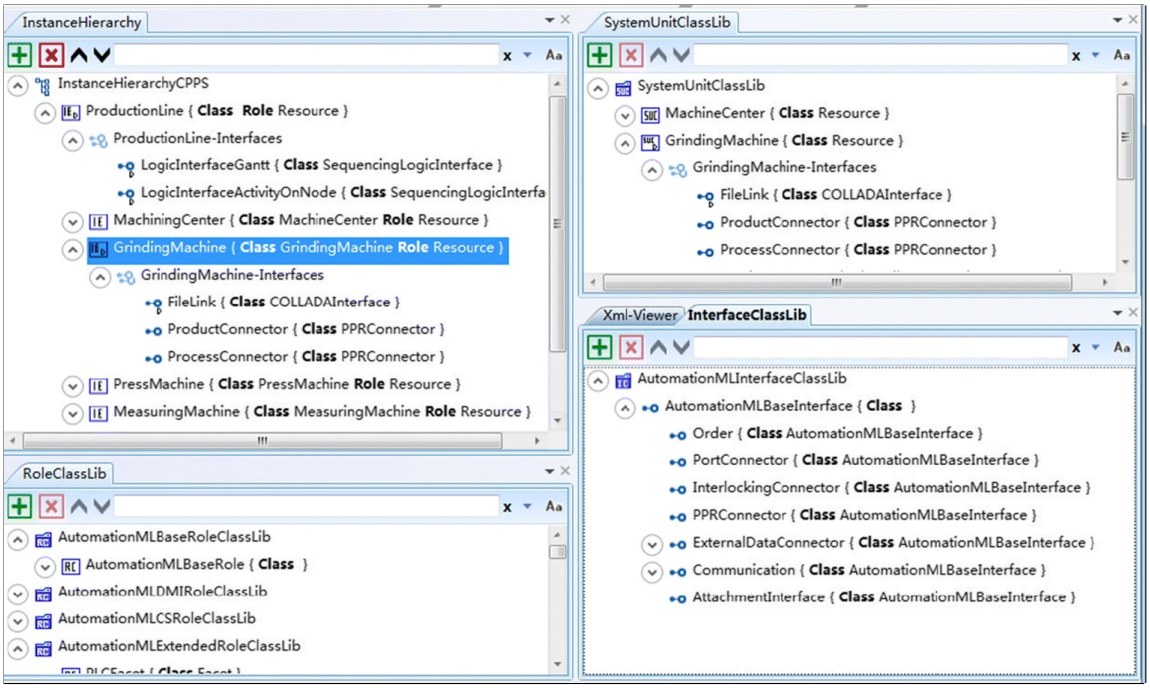


Figura 2.6. Árboles de jerarquía del modelo AutomationML de la fábrica de manufactura (H. Zhang et al., 2020).

2.2.4. IEC 61499

El estándar IEC 61499, que está basado en la parte 3 del estándar IEC 61131, está ideado para cumplir con la necesidad de crear software modular que pueda usarse en procesos de control industrial distribuido y se creó para implementarse a través de estructuras llamadas diagramas de bloques, las que muestran un formato independiente de la implementación. El estándar define a los bloques de función (FB) como su núcleo esencial, lo que permite procesos de producción implementados en funciones de control distribuido inteligente, modulares y flexibles (Salazar & Alvarado, 2014).

IEC 61499 define una arquitectura centrada en las aplicaciones, cada una de ellas es determinada por redes y bloques con funciones interconectados. Dentro del estándar se detalla cuáles deben ser los comportamientos de cada bloque según su función, los cuales pueden ser bloques con algoritmos de control, redes, adaptadores, entre otros. También, se definen los requisitos que deben cumplir los programas, sistemas y dispositivos, en términos de portabilidad, representación y configuración de los elementos.

Este nuevo estándar admite a los tipos de aplicaciones descentralizadas a través de la distribución de bloques funcionales, los que actúan como procesadores conectados a redes que se comunican entre sí de manera más eficiente que con el lenguaje de programación FBD de la tercera parte del estándar IEC 61131.

De las mejoras que presentan las funciones de bloque del IEC 61499, se destacan dos tipos específicos de mensajes: los mensajes de datos, estos están a cargo de operar los objetos tradicionales, y los mensajes de eventos, que se usan para programar y controlar la ejecución de los algoritmos. También, IEC 61499 permite que la estructura de control pueda ser dividida en múltiples controladores y se puedan evitar las desventajas de la planificación cíclica que se requieren en los bloques de IEC 61131 mediante la planificación por eventos (Salazar & Alvarado, 2014).

Para un PLC moderno orientado a la Industria 4.0 el estándar IEC 61499 podría ser un puente entre tecnologías IT que fuesen incluidas en el dispositivo y el de las funciones del

área OT, por ejemplo, incorporar un bloque de almacenaje de datos de control o un bloque para configurar al PLC como un cliente de un plataforma en la nube para obtener mejores parámetros para un controlador.

En la Figura 2.7 se muestra un ejemplo de un sistema de control basado en IEC 61499 implementado en una Raspberry Pi 3B por (Garcia, Salinas, Perez, Salazar L., & Garcia, 2019) , el sistema es un brazo robótico con la tarea de tomar objetos y ponerlos en una ubicación específica. Para lograr la tarea se utilizaron tres bloques IEC 61499: el de lectura, algoritmo de control y escritura. El bloque de lectura incorpora los *drivers* para que el bloque pueda leer los valores de los estados del sistema desde los recursos físicos (sensores) y los transmite de forma estandarizada hacia el bloque de control, este último bloque incorpora un algoritmo para la transmisión de acciones de control basadas en el modelo matemático del brazo robótico. Las acciones de control son leídas por el bloque de escritura, el que incorpora *drivers* para enviar información hacia los recursos físicos (actuadores) luego de interpretar los datos del controlador.

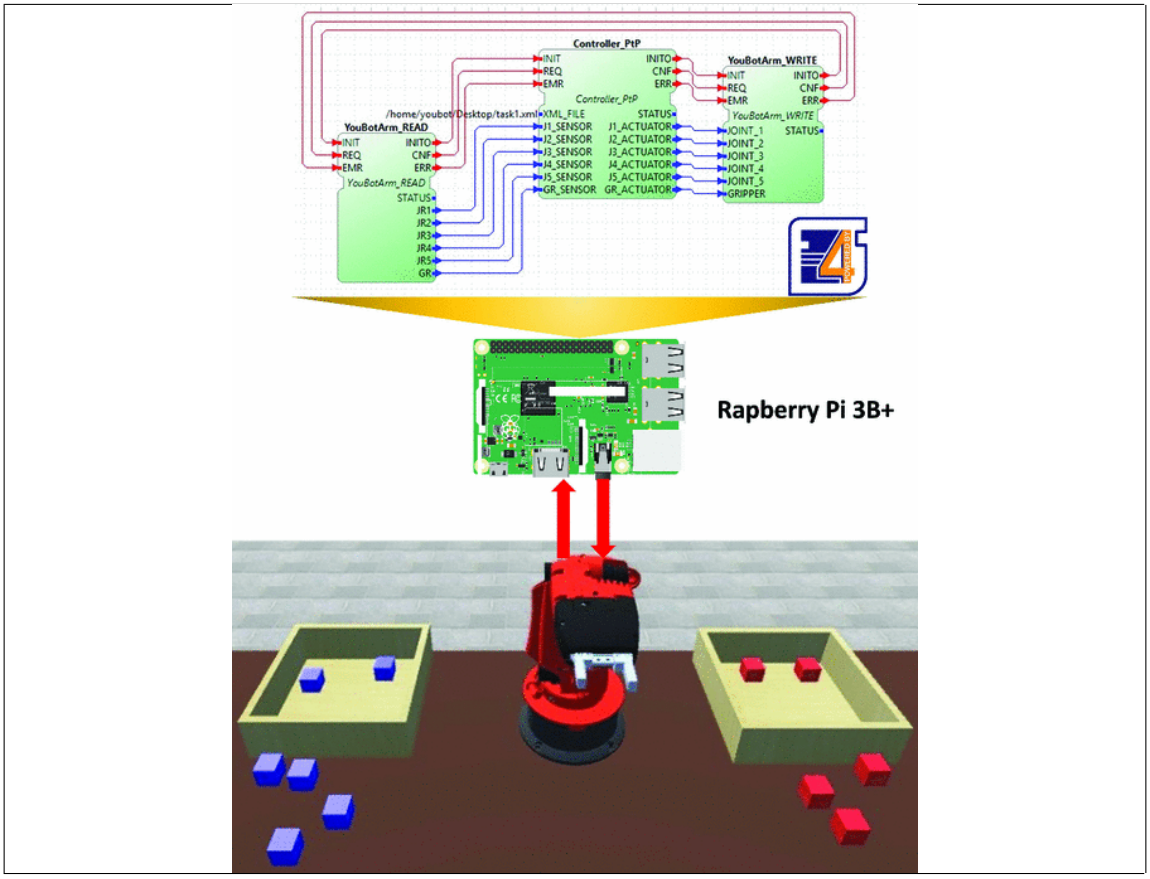


Figura 2.7. Brazo robótico controlado por bloques IEC 61499 (Garcia et al., 2019).

2.3. Estándares de comunicación industriales

En base a (Anybus, 2018), en el mercado de dispositivos con la capacidad de comunicarse en una red industrial 52% corresponde a los que se comunican mediante Ethernet Industrial, 42% a protocolos del tipo de bus de campo y un 6% a protocolos de comunicación inalámbrica. Entre los estándares de comunicación más populares y conocidos están PROFINET con 12%, PROFIBUS con 12%, Modbus con 6% y CANopen con 4%. Estos datos se basan en la compra de nuevos dispositivos en el año 2018. Por esto se describen a continuación los estándares de comunicación PROFIBUS, Modbus, PROFINET

y los protocolos OPC y OPC UA, protocolos que tienen como propósito facilitar la interacción entre las aplicaciones de *software* y el *hardware* de control de procesos en el que se incluyen a los PLCs. Además, en el Anexo A se tratan a los protocolos HART y CANOpen, los cuales son menos populares que los que se explicarán en este capítulo, pero aun son relevantes en la industria.

2.3.1. PROFIBUS

PROFIBUS (Process Field Bus) es un estándar para la comunicación por bus de campo en la tecnología de automatización. PROFIBUS se publica abiertamente como parte de la norma IEC 61158, es un estándar de red independiente del fabricante, cuya interfaz permite una amplia aplicación en la automatización de procesos, fabricación y construcción. Este estándar cumple con las normas EN 50170 y EN 50254. Desde enero de 2000, PROFIBUS está establecido en la norma IEC 61158, junto con otros siete sistemas de bus de campo. Los usuarios pueden utilizar como referencia este protocolo estándar internacional, cuyo desarrollo tiene como objetivo reducir los costos, ganar flexibilidad, confianza, orientación al futuro, adaptarse a las más variadas aplicaciones, interoperabilidad y múltiples proveedores.

Existen dos variantes de PROFIBUS:

- PROFIBUS DP (Periféricos Descentralizados): es la solución de alta velocidad para PROFIBUS, su uso se ha extendido en la industria por permitir el desarrollo de equipos de bajo costo, simples, de alta velocidad, además su utilización en la industria se centra principalmente en ambientes internos como gabinetes o pequeños espacios dentro de plantas o fábricas. Fue desarrollado específicamente para la comunicación entre los sistemas de automatización y los equipos descentralizados. Es aplicable en sistemas de control donde se enfatiza el acceso a dispositivos distribuidos de I/O y sustituye a los sistemas convencionales de 4 a 20 mA, HART o en transmisiones de 24 volts. Utiliza el medio físico

RS-485 o fibra óptica. La velocidad de transferencia de datos puede ajustar a velocidades ente 9,6 kb/s y 12Mbits/s y se utiliza principalmente en controles de tiempo crítico. Actualmente, el 90 por ciento de las aplicaciones que involucran esclavos PROFIBUS utilizan PROFIBUS DP. En la Figura 2.8 se muestra la pila OSI de este protocolo.

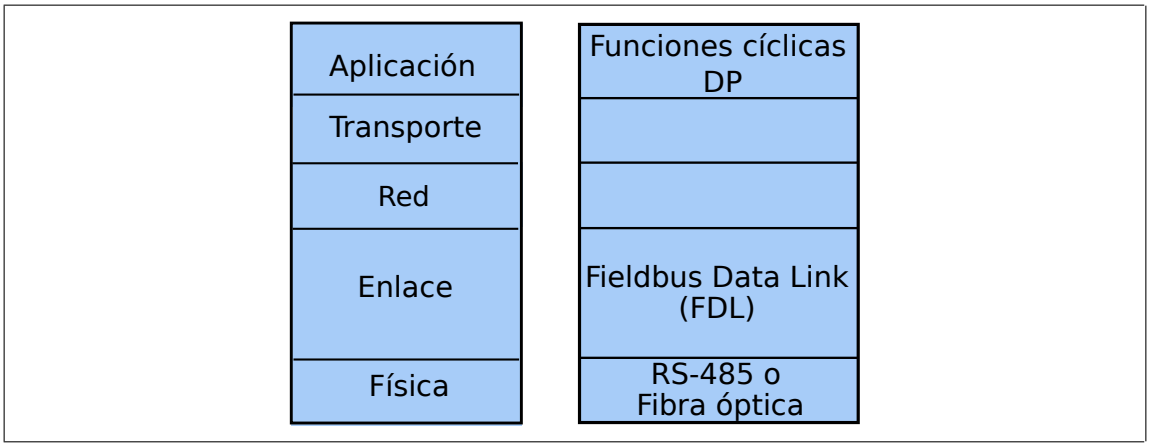


Figura 2.8. Pila OSI del protocolo PROFIBUS DP.

- PROFIBUS PA (Process Automation): El PROFIBUS PA es la solución PROFIBUS que atiende a los requisitos de la automatización de procesos, donde los sistemas de automatización y los sistemas de control de procesos se conectan con los equipos de campo, como transmisores de presión y temperatura, convertidores y posicionadores, fue específicamente diseñado para sustituir al estándar de transmisión 4 a 20 mA y ser implementado en dispositivos que se usen en ambientes exteriores, lugares en los que actualmente se prefiere este estándar por sobre los demás. El PROFIBUS PA permite la medición y el control a través de una línea y dos cables individuales. Además, alimenta al equipo de campo en áreas intrínsecamente seguras. PROFIBUS PA permite el mantenimiento y la conexión/desconexión de los equipos incluso durante el funcionamiento sin interferir en otras estaciones en áreas potencialmente explosivas. En la Figura 2.9 se muestra la pila OSI de este protocolo.

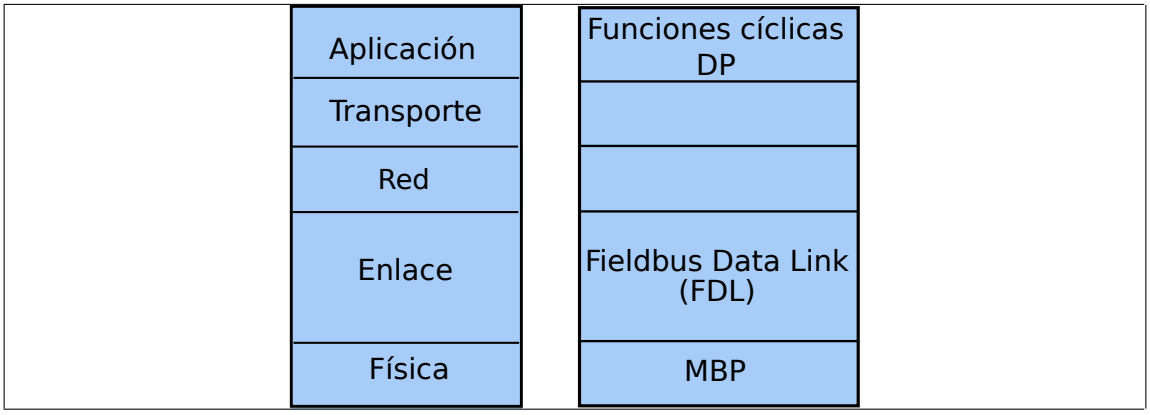


Figura 2.9. Pila OSI del protocolo PROFIBUS PA.

2.3.2. Modbus

Modbus es un protocolo de comunicaciones serial publicado originalmente para su uso con sus controladores lógicos programables (PLCs) y es ahora un medio comúnmente disponible para conectar dispositivos electrónicos industriales. Modbus es popular en los entornos industriales, porque se publica abiertamente y está libre de derechos. Fue desarrollado para aplicaciones industriales, es relativamente fácil de desplegar y mantener en comparación con otros estándares, y pone pocas restricciones, aparte del tamaño, en el formato de los datos a transmitir. Modbus utiliza el RS-485 como su capa física.

En este protocolo el dispositivo que solicita la información se llama maestro Modbus y los dispositivos que suministran la información son los esclavos Modbus. En una red Modbus estándar, hay un maestro y hasta 247 esclavos, cada uno con una dirección de esclavo única de 1 a 247. El maestro también puede escribir información a los esclavos. En la Figura 2.10 se muestra la pila OSI de este protocolo.

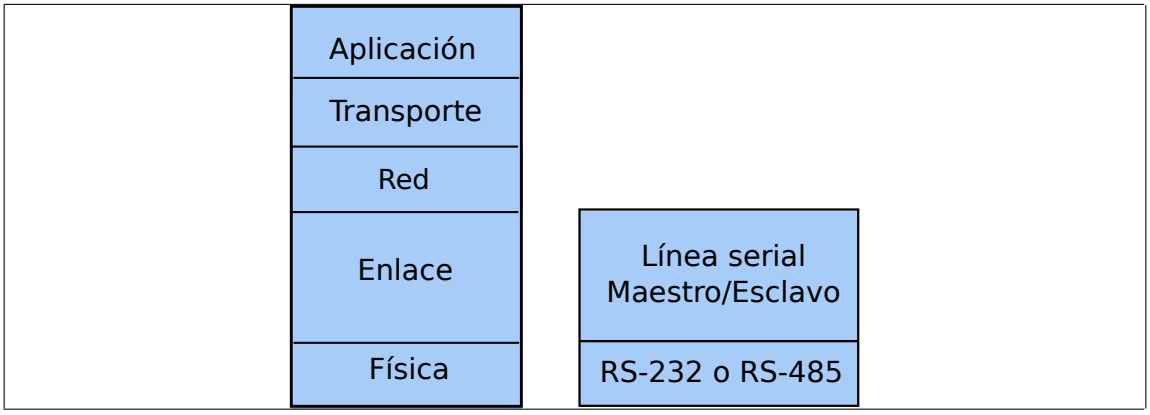


Figura 2.10. Pila OSI del protocolo Modbus.

2.3.3. PROFINET

Los controladores programables como los PLC y los PC se comunican entre sí y por lo tanto requieren que se transfiera una gran cantidad de paquetes de datos en varias y poderosas funciones de comunicación. Además, la integración eficiente a los sistemas de comunicación corporativos existentes, como Intranet, Internet y Ethernet es absolutamente obligatoria. Esta necesidad se satisface con el protocolo PROFINET, el cual se utiliza por esta razón como puente entre los dispositivos tipo SCADAs o estaciones de ingeniería con otros de más alto nivel y, debido a que basa parte de su protocolo en PROFIBUS, ha empezado a reemplazar en importancia a PROFIBUS DP.

PROFINET es un protocolo de comunicación para la tecnología de automatización industrial basado en Ethernet. PROFINET utiliza el modelo funcional básico de PROFIBUS, que permite a los usuarios de PROFIBUS pasar sin problemas a la nueva tecnología. PROFINET combina las ventajas del conocido sistema de bus de campo y del Ethernet industrial. Este estándar fue diseñado para entornos industriales en los que se requieren transferir datos dentro de restricciones de tiempo limitadas (menores a 1 ms), entre sus características están las de poder configurarse para operar de forma altamente determinística y la de alcanzar una gran velocidad de transferencia de datos (100 mbits). En la Figura 2.11 se muestra la pila OSI de este protocolo.

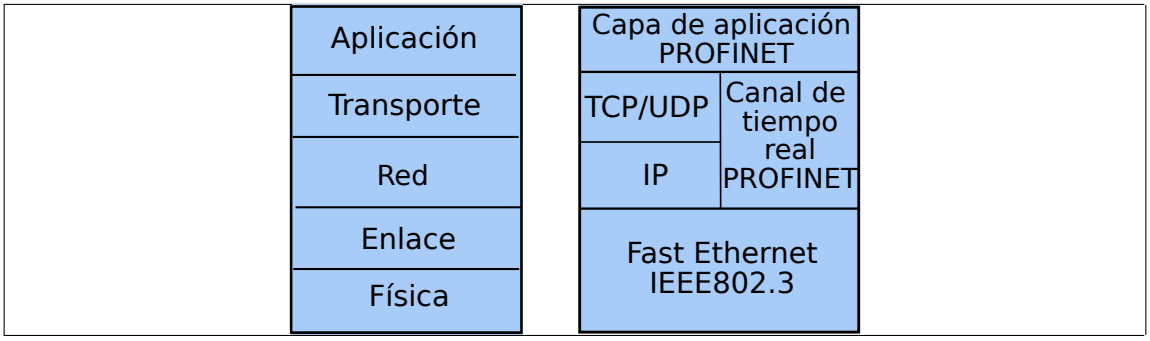


Figura 2.11. Pila OSI del protocolo PROFINET.

2.3.4. OPC

El estándar OPC es una serie de especificaciones desarrolladas por proveedores de la industria, los usuarios finales y los desarrolladores de *software*. Estas especificaciones definen la interfaz entre los clientes y los servidores, y viceversa, incluido el acceso a datos en tiempo real, la supervisión de alarmas y eventos, el acceso a datos históricos y otras aplicaciones. OPC fue diseñado para proporcionar un puente común para las aplicaciones de *software* basadas en Windows y el *hardware* de control de procesos, por lo que este protocolo solo funciona sobre aquella plataforma.

Entre sus propósitos está el de abstraer los protocolos específicos de los PLC (como Modbus, Profibus, etc.) en una interfaz normalizada que permita a los sistemas HMI/SCADA interactuar con un "intermediario" que convierta las peticiones de lectura/escritura de los PLCs genéricos en peticiones específicas de los dispositivos y viceversa.

La especificación define un conjunto estándar de objetos, interfaces, por ejemplo, interfaces de descripción de lenguajes (IDL) y métodos para su uso en aplicaciones de control de procesos y automatización para facilitar la interoperabilidad. La especificación OPC más común es la de acceso a datos OPC (OPC DA), que se utiliza para leer y escribir datos en tiempo real.

Además de la especificación OPC DA, se mantiene la especificación OPC de Acceso a Datos Históricos (HDA). A diferencia de los datos en tiempo real a los que se puede acceder con el OPC DA, el OPC HDA permite el acceso y la recuperación de los datos archivados.

También, se mantiene la especificación de alarmas y eventos OPC, y define el intercambio de información de mensajes de tipo de alarma y eventos, así como los estados variables y la gestión de estados.

2.3.5. OPC UA

OPC UA fue diseñado para mejorar las capacidades de las especificaciones de OPC e integrar más funcionalidades, pero manteniendo la equivalencia funcional con OPC, aunque ambos presentan marcadas diferencias, ya que OPC UA es una norma multiplataforma enfocada en ser escalable y segura que elimina las restricciones que OPC imponía debido a su diseño. OPC UA es una norma independiente de la plataforma en que se use, por lo que no depende del uso de Windows como OPC, mediante la cual se pueden comunicar diversos tipos de sistemas y dispositivos enviando mensajes de solicitud y respuesta entre clientes y servidores o mensajes de red entre publicadores y suscriptores a través de diversos tipos de redes. Es posible utilizar OPC UA en todos los dominios industriales, como sensores, actuadores, sistemas de control, MESs y ERPs, incluida el Internet industrial de las cosas y la Industria 4.0.

OPC UA se enfoca en los mismos temas industriales que OPC, mantiene las especificaciones OPC DA, OPC HDA, la de alarmas y la de eventos OPC, pero adapta su funcionamiento para mejorar las deficiencias de su predecesor.

La arquitectura del sistema OPC UA modela los clientes y los servidores como socios interactivos. Cada sistema puede contener múltiples clientes y servidores. Cada cliente puede interactuar simultáneamente con uno o más servidores, y viceversa. OPC UA define un modelo de infraestructura común, que especifica lo siguiente: el modelo de información

para representar la estructura, el comportamiento y la semántica, el modelo de mensaje para interactuar entre aplicaciones, el modelo de comunicación para transferir los datos entre puntos finales y el modelo de conformidad para garantizar la interoperabilidad entre sistemas.

En OPC UA los objetos se utilizan para representar sistemas, componentes, objetos del mundo real y objetos de software. El conjunto de objetos e información relacionada que un servidor pone a disposición de los clientes se denomina su *AddressSpace*. El objetivo principal del *AddressSpace* de OPC UA es proporcionar una forma estándar para que los servidores representen los objetos a los clientes. De esta forma, el *AddressSpace* de OPC UA representa sus contenidos como un conjunto estructurado jerárquicamente de nodos conectados por Referencias.

Otra característica de OPC UA es el modo de publicación y suscripción (PubSub), las aplicaciones de OPC UA que utilizan PubSub no intercambian directamente solicitudes y respuestas. En su lugar, los editores envían datos a un *Middleware* orientado a mensajes, sin saber qué suscriptores puede haber. Del mismo modo, los suscriptores expresan su interés en tipos específicos de datos, sin saber qué editores hay. El *Middleware* Orientado a Mensajes como se usa en esta especificación es cualquier infraestructura que soporte el envío y la recepción de Mensajes de Red entre las aplicaciones.

Una característica importante de OPC UA implementada con PubSub es el Modelo de Eventos, que define un sistema de eventos de propósito general que puede ser utilizado en muchos mercados verticales diversos. En este sistema los clientes se suscriben a los nodos para recibir notificaciones de eventos relacionados con los cambios de datos.

3. ARQUITECTURAS PARA SISTEMAS DE CONTROL INDUSTRIAL ENFOCADAS A LA INDUSTRIA 4.0

En este capítulo se presentan arquitecturas de referencia para la implementación de sistemas IoT orientados a la Industria 4.0 que involucran desde niveles en que están los dispositivos finales hasta los más altos que están relacionados a las estadísticas y tomas de decisiones. Tres son las arquitecturas que se presentan en este capítulo: *IoT world forum reference model*, RAMI 4.0 e *Industrial Internet Reference Architecture*, se dan ejemplos de implementación de cada una de ellas y se explica en que niveles un PLC para la Industria 4.0 podría emplazarse y que funciones podría cumplir.

Un PLC podría ubicarse cercano a los límites que separan a las áreas OT e IT en todas las arquitecturas que se presentarán, ya que así se mantiene la característica de control en tiempo real con baja latencia, mientras que se le da la flexibilidad de incorporar funciones de baja complejidad en el área IT. Esta ubicación queda bien definida en la concepción de capa *fog* de la arquitectura *IoT world forum reference model*, lo que le permitiría al PLC actuar como un puente entre las dos áreas que actualmente no tienen la capacidad de intercambiar información de forma directa y transparente, además de entregar información con mayor nivel de procesamiento hacia capas superiores. Por otro lado, las arquitecturas RAMI 4.0 e *Industrial Internet Reference Architecture* permiten definir las interacciones entre las entidades OT e IT y los objetivos que tendrán las tareas que se le asignan, tales como generar monitoreo y procesamiento para optimizar funciones a nivel local.

A continuación en este capítulo se presentan las tres arquitecturas mencionadas y en el Anexo B se muestran implementaciones industriales de la capa *fog*.

3.1. IoT world forum reference model

La arquitectura *IoT world forum reference model* (CISCO (2014)) considera siete capas que van desde los dispositivos físicos hasta los procesos que involucran los negocios y administración, tal como se puede apreciar en la Figura 3.1. En cada capa que se asciende

en este modelo se aumenta la abstracción de la información transmitida, pasando por etapas que llevan los datos de los dispositivos sin procesar en la primera capa hasta generar estadísticas, análisis y reportes en la sexta. A continuación se explicarán todas las capas de este modelo con mayor detalle.

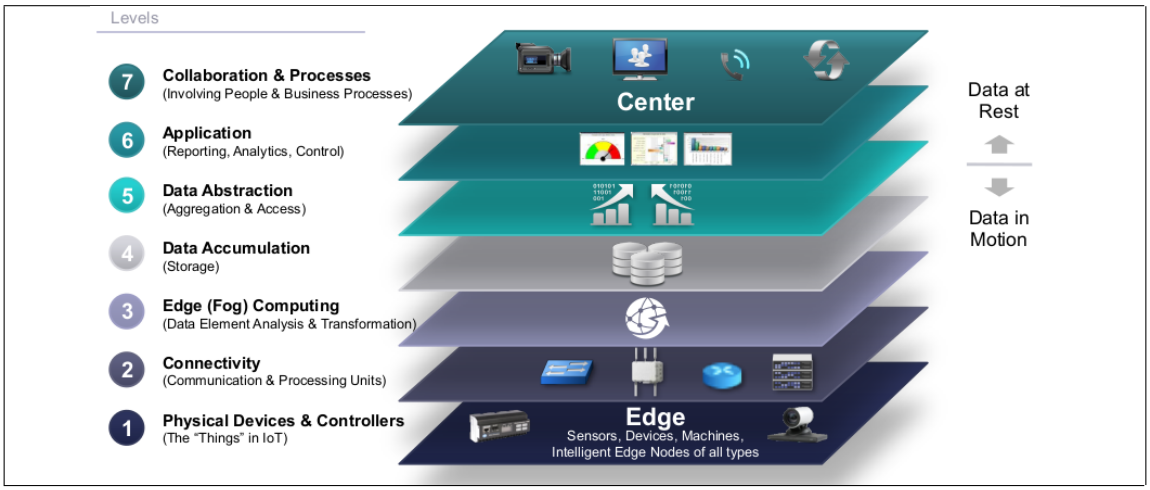


Figura 3.1. IoT world forum reference model. (CISCO, 2014)

3.1.1. Capa 1: Dispositivos físicos y controladores

En esta capa están los dispositivos de campo que envían y reciben información, estos son asociados a las "cosas" del paradigma. Los dispositivos son diversos tanto en características físicas como técnicas, pueden ser pequeños como microchip o del tamaño de automóviles.

3.1.2. Capa 2: Conectividad

Esta capa está centrada en la conectividad y las comunicaciones, su principal función es la de asegurar la transmisión confiable y oportuna. Las transmisiones comprenden a la comunicación de dispositivos en la capa 1 y la red, entre distintas redes y entre la capa 2 y 3 del modelo.

3.1.3. Capa 3: *Fog*

La capa *fog* se enfoca en procesar la información que viene del borde de la red (*network edge*) lo más cercana a él, para determinar si es que se envía a las capas superiores. Además, puede analizar los datos para generar información local que permita realizar instrucciones de operación hacia capas inferiores. Las principales funciones de esta son la evaluación de la información, reducción de datos y dar un formato consistente para las capas superiores. Esta capa permite que sus funciones se puedan realizar en tiempo real, gracias a que, al procesar información cerca de los dispositivos de campo, se evita la latencia que se obtendría con la transferencia de datos hacia capas más altas.

3.1.4. Capa 4: Acumulación de datos

En esta capa se decide si la información es útil para las capas superiores y, si lo fuese, convertirla en información útil para esas capas mediante la utilización de bases de datos u otra forma de sistematización del almacenaje de datos, también decide si la información se guarda para que sea utilizada en un corto plazo y si se combina con información previamente adquirida, todo lo que se genera en esta capa es para la utilización de aplicaciones que no necesitan responder en tiempo real.

3.1.5. Capa 5: Abstracción de datos

La capa de abstracción de datos se encarga de manejar los datos y su almacenamiento de forma que su utilización posterior sea simple y lo más rápida posible, para eso se acomodan los distintos formatos de datos que se le envían, se garantiza que los datos sean consistentes y asegura los datos mediante autentificación.

3.1.6. Capa 6: Aplicación

En la capa de aplicación es donde se produce la interpretación de la información. El *software* en este nivel interactúa con datos en reposo (*data at rest*), por lo que no

tiene que operar a velocidades de red. La función de esta capa depende de lo que se requiera hacer a gran escala según las necesidades que se tengan, por ejemplo la aplicación se puede enfocar en el control de dispositivos, monitorización de patrones de comportamiento, combinación de datos de dispositivos con los de otras fuentes con el objetivo de obtener estadísticas o información con sentido práctico.

3.1.7. Capa 7: Colaboración y procesos

Esta última capa está enfocada en dar a las personas las herramientas que necesitan para poder usar las aplicaciones y la información tratada anteriormente para cumplir con las tareas que deben realizar, principalmente estas herramientas hacen que el trabajo colaborativo entre personas sea más efectivo. Dado que las aplicaciones de la capa 6 permiten la entrega de la información necesaria y en el momento adecuado, en esta capa los trabajadores se enfocan en la toma de decisiones en base a la comunicación y la colaboración.

3.1.8. Emplazamiento de un PLC orientado a la Industria 4.0

Un PLC orientado a la Industria 4.0 podría estar ubicado en la tercera capa de esta arquitectura, la capa *fog*, ya que ahí se asegura que pueda mantener un flujo de información con los dispositivos de campo con una baja latencia, así podría realizar las tareas de control en tiempo real y permitir su uso de parte de los operadores de una planta de forma local. Además, en la capa *fog* un PLC moderno podría realizar tareas adicionales como análisis de datos de baja complejidad, para transmitir información procesada hacia capas superiores, y administrar la configuración de dispositivos de campo.

3.1.9. Implementación práctica de la arquitectura

En (Núñez et al., 2020) se utiliza a la arquitectura *IoT world forum reference model* como modelo base para diseñar e implementar una plataforma IIoT orientada al control de un espesador en el ámbito de la minería, las capas de esta arquitectura se presentan en la Figura 3.2. En la primera capa se encuentra el controlador del espesador y los dispositivos

físicos, en la segunda se implementa un servidor y cliente OPC UA para dar conectividad a los dispositivos antiguos, en la tercera capa se toman los datos y se procesan, en la cuarta capa se guardan los datos en una base de datos NoSQL, en la quinta capa se consolidan los datos y se les da un grado de abstracción mayor y en la sexta capa se generan modelos y controladores basados en redes neuronales.

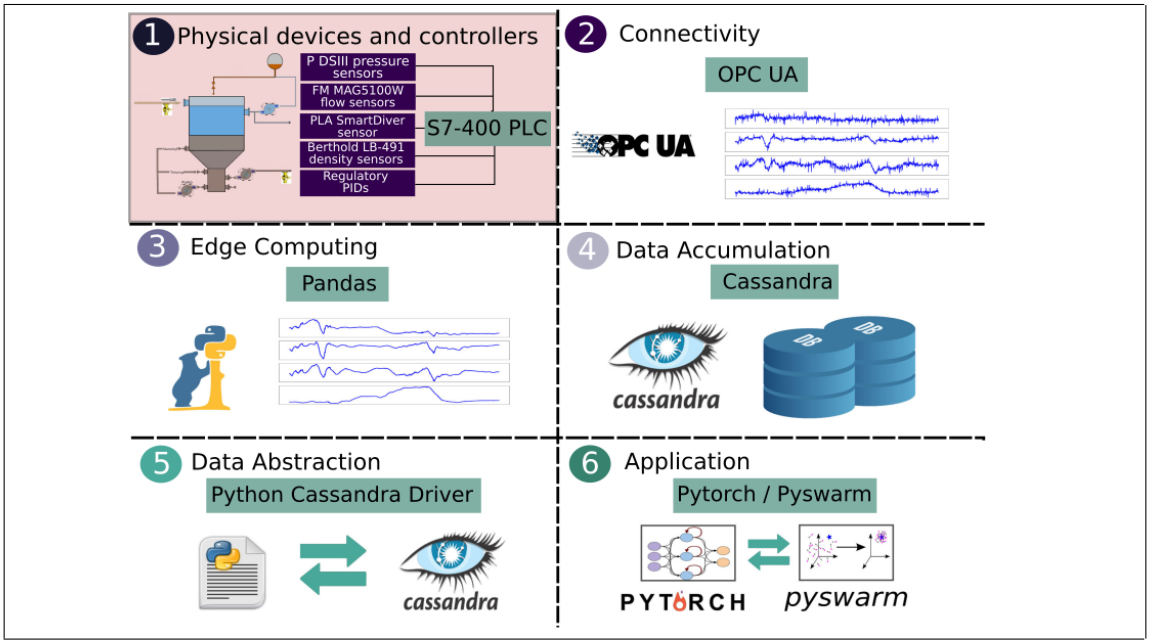


Figura 3.2. Plataforma IIoT orientada al control de un espesador (Núñez et al., 2020).

3.2. RAMI 4.0

La arquitectura *Reference Architecture Model for Industrie 4.0* (RAMI 4.0) fue creada con el fin de dar guía a los grupos interesados en migrar desde los sistemas industriales tradicionales hacia sistemas que respondan al paradigma de la Industria 4.0. Para esto la arquitectura se enfoca en dar respuesta a las siguientes preguntas: ¿qué tecnologías deben utilizarse para digitalizar y dar conectividad al sistema?, ¿cómo se deben estructurar y cómo deben funcionar los niveles de jerarquía según los estándares IEC 62264 e IEC

61512? y ¿en cuál fase del ciclo de vida se está según el estándar IEC 62890? (Schulte & Colombo, 2017).

La arquitectura RAMI 4.0 tiene un modelo en tres dimensiones, cada uno de los ejes se explica a continuación:

3.2.1. Niveles de jerarquía

Los niveles de jerarquías provienen del estándar IEC 62264, el cual es el estándar internacional para empresas IT y sistemas de control basado en ISA-95, estos niveles de jerarquía representan distintas funcionalidades dentro de una fábrica o instalación. Son siete de estos niveles en total, los que corresponden a:

(i) Producto: Es el nivel en que están los productos que se fabrican.

(ii) Dispositivo de campo: Aquí se encuentran los dispositivos electrónicos que recolectan y controlan el flujo de la información, como sensores, análisis de datos y alarmas.

(iii) Dispositivo de control: Son los dispositivos que se utilizan para dar instrucciones mediante señales de entrada y salida, se incluyen aquí a los PLCs, DCSs e interfaces de usuarios.

(iv) Estación: Este es el nivel donde se encuentran los operadores examinando los procesos y sus eventos, junto con realizar actividades administrativas que responden a ellos, por ejemplo, manejar la coordinación entre cadenas de ensamblaje.

(v) Centro de trabajo: En este nivel se encuentra la información de producción y los estados de fabricación, permite que se puedan tomar decisiones que ayuden a mejorar la productividad.

(vi) Empresa: Aquí se encuentran los procesos de planificación, servicios de envíos, finanzas y de marketing, también se utiliza información estadística de los productos que se fabrican.

(vii) Mundo conectado: Este nivel incluye la interacción entre los accionistas, proveedores y otros prestadores de servicio, aquí se logra compartir información de estrategias a mayor nivel, como estado del negocio y estrategias de marketing.

3.2.2. Ciclo de vida

Este eje involucra al ciclo de vida de una instalación y de los productos, basado en el estándar IEC 62890. Se divide en dos partes: tipo e instancia, la primera corresponde a la etapa de desarrollo de prototipos y sus validaciones, junto con la mantención por medio de actualizaciones de software o manuales de instrucciones. La segunda parte corresponde a la etapa en que se empieza a producir el producto, con lo que se generan datos prácticos de este, además, se incluye una fase de mantención que incluye actividades como el reciclaje de residuos y servicios para las instalaciones.

3.2.3. Capas de la arquitectura

El eje de las capas de la arquitectura permite describir a un sistema por medio de la separación de su propiedades en distintas capas, con esto se logra analizar y entender de mejor forma a los sistemas complejos. En el modelo se tienen seis capas, las que corresponden a:

(i) Activos: Describe las características de los componentes físicos como maquinaria, software, incluso proveedores de servicios y clientes.
(ii) Integración: Trata con el manejo de información como nexo entre el mundo digital y físico.
(iii) Comunicación: Provee la estandarización de las comunicaciones entre la capa de integración y de información, a través de protocolos de comunicación determinados.

(iv) Información: Esta capa toma todos los datos y los organiza con el fin de extraer información acerca de los productos y materiales fabricados, maquinaria y componentes utilizados.

(v) Funcionalidad: Incluye la coordinación de componentes a nivel general, también es responsable de generar reglas para las cadenas de producción y de los sistemas de control.

(vi) Negocios: En esta capa se incluyen las estrategias de mercado, como metas económicas y maniobras publicitarias, además de realizar análisis de costo y producción.

3.2.4. Límite entre IT y OT

Como se puede ver en la Figura 3.3, al utilizar la vista tradicional de la pirámide que representa a la parte de la arquitectura RAMI 4.0 conforme a la norma IEC 62264, es posible identificar claramente el borde donde el área OT se encuentra con el área IT, destacando que el área OT comprende todo lo que abarcan las primeras cinco partes de los niveles de jerarquía. El borde (*edge*) de control y gestión en tiempo real es donde ocurre (i) la recopilación de datos de las máquinas y la adquisición de datos de funcionamiento que posteriormente se transfieren de abajo hacia arriba desde el área OT a los componentes y sistemas de IT, y (ii) la transferencia de las órdenes de producción y mantenimiento de arriba hacia abajo desde el área IT a los componentes y sistemas OT.

Desde el punto de vista de la digitalización, tanto para los componentes y sistemas de IT como para los de OT, en la dimensión vertical de la arquitectura RAMI 4.0 es en la que se especifican y aplican los protocolos de comunicación, semánticas y los modelos operacionales para garantizar que los componentes y sistemas puedan conectarse sin problemas y puedan interoperar realizando sus propias funciones y cumpliendo los objetivos comerciales (Schulte & Colombo, 2017).

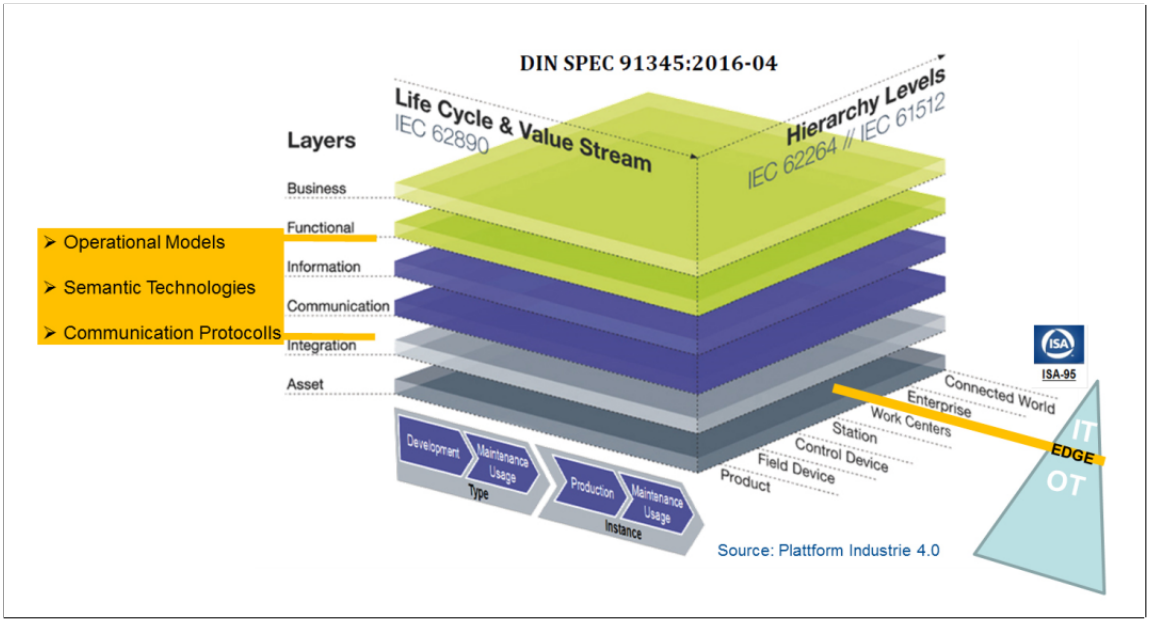


Figura 3.3. RAMI 4.0, límite entre las áreas IT y OT (Schulte & Colombo, 2017).

3.2.5. Emplazamiento de un PLC orientado a la Industria 4.0

En esta arquitectura un PLC orientado a la Industria 4.0 podría estar ubicado entre los niveles de jerarquía tercero y cuarto, ya que en el tercer nivel de jerarquía se mantendría la comunicación con los dispositivos de campo para realizar tareas de control y permitiría el uso del PLC por el personal de la planta, mientras que en el cuarto nivel de jerarquía el PLC podría realizar tareas de detección de eventos y supervisión de los procesos. Al estar ubicado entre estos niveles se le da flexibilidad de operación al PLC para incorporar nuevas funcionalidades relacionadas al control o al análisis de datos manteniendo el retardo en las comunicaciones con los dispositivos de campo dentro de los márgenes que requiere la industria. Además, el PLC estaría ubicado en el área OT del sistema de producción, pero con la posibilidad de interactuar con sistemas IT.

3.2.6. Implementación práctica de la arquitectura

En (Schulte & Colombo, 2017) se presentan los principales pasos de la migración de un sistema industrial de fabricación de láminas de plástico hacia un sistema que cumpla con los estándares de RAMI 4.0. Para esto utilizaron el sistema de ejecución de manufactura (MES) como administrador de los datos e información a nivel IT de todos los componentes del área OT y de niveles inferiores de la arquitectura del sistema, la cual se muestra en la figura 3.4. Esto lo lograron por medio de redes locales virtuales que conectaron a las plantas con los sistemas MES para la realización de tareas desde las oficinas, mientras que con el uso de OPC UA se dio conectividad a las plantas de producción hacia las redes de la compañía y se estandarizó la forma en que los sistemas de alto nivel obtenían la información de los sistemas de control y como la representaban de forma virtual. Por otro lado, los PLCs que se encontraban en los sistemas de control tuvieron que ser adaptados para que pudieran exponer la información a través de OPC UA. De tal forma, se llevó a adecuar al sistema según las dimensiones verticales de la arquitectura RAMI 4.0, dándole conectividad e interoperabilidad ente las dos áreas (IT y OT).

3.3. Industrial Internet Reference Architecture (IIRA)

La arquitectura IIRA es una arquitectura abierta para sistemas IIoT, está diseñada para que pueda ser utilizada en un amplio rango de tipos de industrias, junto con permitir interoperabilidad entre diversas tecnologías manteniendo un estándar de desarrollo. Es por esto que esta arquitectura es genérica y sus conceptos y modelos tienen descripciones con alto nivel de abstracción, lo que permite su amplia aplicabilidad.

La arquitectura identifica los puntos de mayor interés que comúnmente se encuentran en los sistemas IIoT y los clasifica en puntos de vista o *viewpoints* con las personas que están relacionadas a ellos (*stakeholders*), luego describe y analiza los puntos de interés para guiar en el manejo de estos, lo que lleva a generar representaciones abstractas en la arquitectura. Los *viewpoints* definidos son cuatro: negocios (*business*), uso (*usage*),

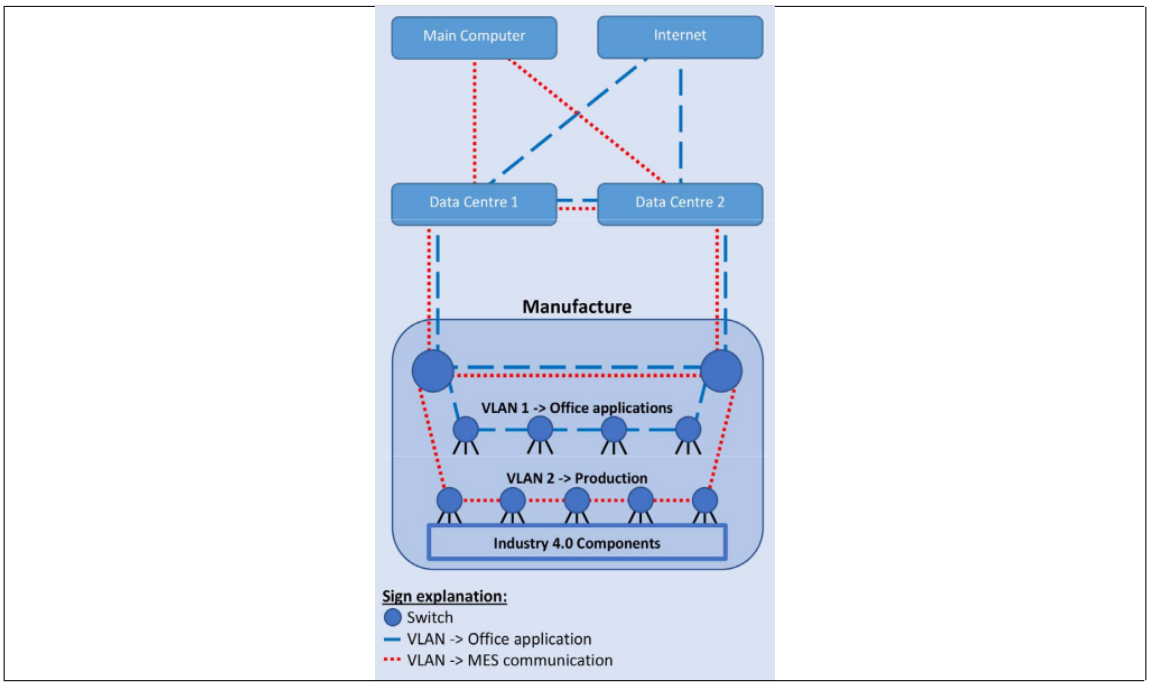


Figura 3.4. Arquitectura de sistema industrial de fabricación de láminas de plástico (Schulte & Colombo, 2017).

funcional (*functional*) e implementación (*implementation*), tal como se ve en la Figura 3.5. Estos cuatro puntos son la base de la arquitectura IIRA y pueden ser adaptados y extendidos según las necesidades propias de cada organización.

3.3.1. Puntos de vistas (*viewpoints*)

A continuación se describirán brevemente los cuatro puntos de vistas descritos en la arquitectura IIRA.

3.3.1.1. Negocios

El punto de vista de los negocios apunta a atender los puntos de interés y preocupaciones de las personas y organizaciones en lo relativo a la visión de negocios y objetivos que se tienen al establecer un sistema IIoT. Además, identifica la forma en que el sistema alcanza los objetivos propuestos a través de sus capacidades fundamentales. Este punto

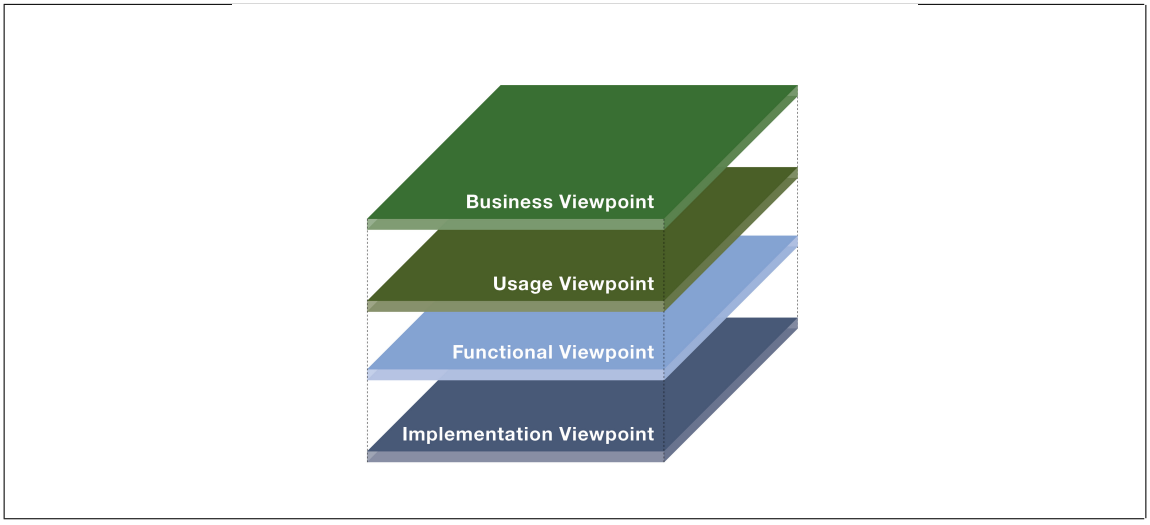


Figura 3.5. Industrial Internet Reference Architecture (IIRA) (Consortium, 2017).

está particularmente enfocado en las personas que toman las decisiones de negocios, administrador de productos e ingenieros de sistemas.

3.3.1.2. Uso

El punto de vista Uso se encarga de entender las principales capacidades del punto de vista Negocios, aquí se describen las actividades que coordinan varias unidades de trabajo en distintos componentes del sistema IIoT. Estas actividades describen como utilizar al sistema como el primer punto para satisfacer los requisitos de la organización, los puntos más importantes que se analizan son el diseño, implementación, despliegue, operaciones y evolución del sistema IIoT. Las personas más involucradas en este punto de vista son ingenieros de sistemas, administradores de producción y personas que están involucradas en las especificaciones del sistema IIoT.

3.3.1.3. Funcional

El punto de vista Funcional se enfoca en los componentes funcionales de un sistema IIoT, su estructura e interrelación, las interfaces e interacciones entre ellas y las interacciones que se realizan con elementos externos en el ambiente, para dar soporte a los usos y actividades del sistema en general. Las personas más relacionadas con este punto de vista son los arquitectos e integradores del sistema y desarrolladores.

En este punto de vista se puede dividir un sistema IIoT en distintos dominios funcionales que representan funcionalidades específicas, el flujo de interacciones entre ellos se puede ver en la Figura 3.6. A continuación se describen estos estos dominios,

- Dominio de control: Este dominio representa el conjunto de funciones que son realizadas por sistemas de control industriales. El núcleo de estas funciones comprende a los sistemas de control de lazo cerrado altamente ajustados, la lectura de datos de los sensores, aplicación de reglas y lógica y ejercer control sobre el sistema físico a través de actuadores. Tanto la precisión como la resolución en el tiempo suelen ser críticas.
- Dominio de operaciones: Este dominio representa el conjunto de funciones responsables del aprovisionamiento, gestión, monitorización y optimización de los sistemas en el dominio de control. Los sistemas de control industrial existentes se centran principalmente en la optimización de los activos en una única planta física, en tanto, en los sistemas de control IIoT deben subir un nivel y optimizar las operaciones entre tipos de activos, flotas y clientes. Esto abre oportunidades para añadir negocios y clientes, tal y como se establece en los dominios de nivel superior orientados a la empresa.
- Dominio de la información: El dominio de la información representa la recopilación de funciones para recopilar datos de varios dominios, más significativamente del dominio de control y transformar, persistir y modelar o analizar esos datos para adquirir inteligencia de alto nivel sobre el sistema en su conjunto.

Las funciones de recopilación y análisis de datos en este ámbito son complementarias a las que se llevan a cabo en el ámbito de control. En el ámbito del control, estas funciones participan directamente en el control inmediato de los sistemas físicos, mientras que en el ámbito de la información sirven para facilitar la toma de decisiones, la optimización de las operaciones en todo el sistema y la mejora de los modelos de sistemas a largo plazo.

- Dominio de aplicación: Aquí se representa al conjunto de funciones que implementan la lógica de aplicación que realiza funcionalidades de negocio específicas. Las funciones en este dominio utilizan la lógica de aplicación, reglas y modelos en un nivel alto y altamente ajustado para la optimización en un ámbito global. No mantienen operaciones continuas de bajo nivel, ya que éstas se delegan a funciones en el ámbito de control que deben mantener las normas y modelos locales en caso de pérdida de conectividad.
- Dominio de negocios: Las funciones en este dominio realizan las operaciones de extremo a extremo de los sistemas IIoT, integrándolas con los tipos tradicionales o nuevos de sistemas industriales, funciones de negocio específicas, incluyendo aquellas que soportan los procesos de negocio y las actividades de procedimientos. Ejemplos de estas funciones empresariales incluyen la planificación de recursos empresariales (ERP), la gestión de las relaciones con los clientes (CRM), la gestión del ciclo de vida del producto (PLM), el sistema de ejecución de fabricación, la gestión de recursos humanos (HRM), la gestión de activos, la gestión del ciclo de vida del servicio, la planificación del trabajo y los sistemas de programación.

3.3.1.4. Implementación

El punto de vista Implementación se centra en las tecnologías necesarias para implementar componentes funcionales (punto de vista funcional), sus esquemas de comunicación y sus procedimientos de ciclo de vida. Estos elementos son coordinados por

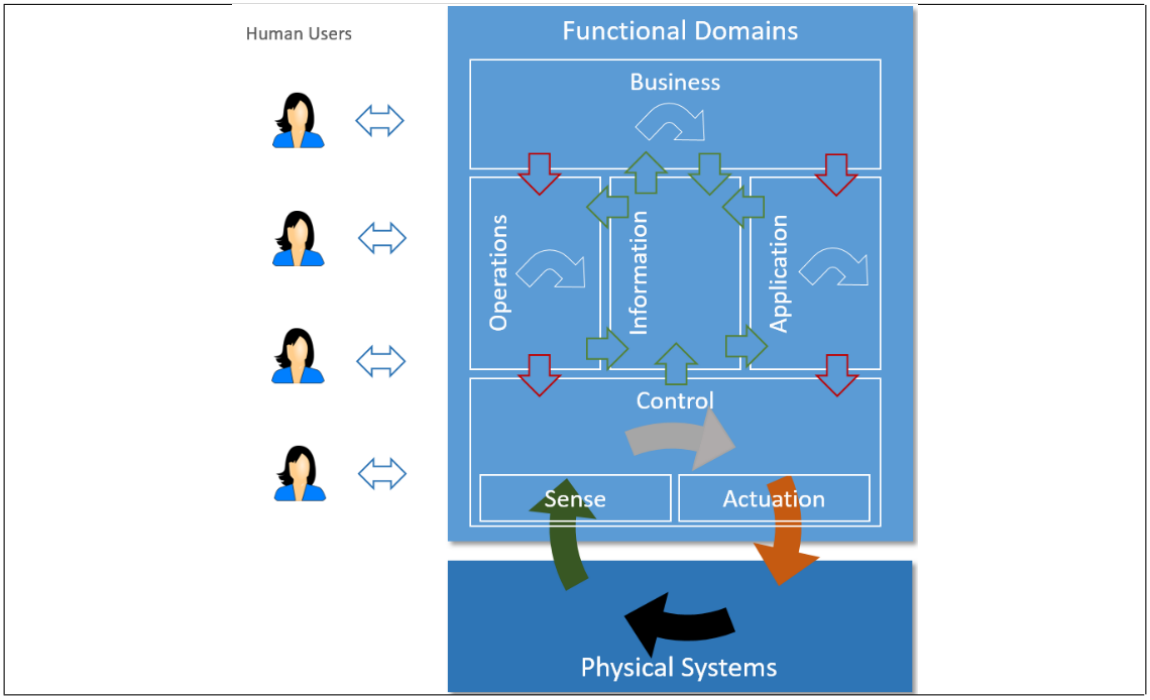


Figura 3.6. Arquitectura IIRA: dominios funcionales. (Consortium, 2017)

actividades (punto de vista de uso) y apoyan las capacidades del sistema (punto de vista de negocios).

El punto de vista de Implementación se refiere a la representación técnica de un sistema IIoT y las tecnologías y componentes del sistema necesarios para implementar las actividades y funciones prescritas por los puntos de vista de uso y funcional. La arquitectura de un sistema IIoT y la elección de las tecnologías utilizadas para su implementación también están guiadas por el punto de vista de negocios, incluidas las limitaciones de costes y tiempo de lanzamiento al mercado, la estrategia de negocios con respecto a los mercados objetivo, los requisitos pertinentes de regulación y la evolución prevista de las tecnologías. La implementación también debe cumplir los requisitos del sistema, incluyendo aquellos identificados como características clave del sistema que son comunes a todas las actividades y que deben aplicarse globalmente como propiedades de extremo a extremo del sistema IIoT. Este punto de vista es de particular interés para los arquitectos de sistemas y componentes, desarrolladores, integradores y operadores de sistemas.

3.3.2. Emplazamiento de un PLC orientado a la Industria 4.0

El PLC en su estado actual está representado en el dominio de control del punto de vista Funcional, debido a que en ese dominio se reúnen las funciones realizadas por un sistema de control. Mientras que un PLC orientado a la Industria 4.0 podría estar representado en el dominio de control, a la vez que en los dominios de operaciones y de la información del punto de vista Funcional, ya que un PLC moderno podría realizar optimizaciones de baja complejidad para mejorar el rendimiento del proceso industrial y, también, de recopilar datos de los dispositivos de campo para analizarlos de forma local o procesarlos para ser enviados hacia sistemas de toma de decisiones de alto nivel. Un PLC orientado a la Industria 4.0 no podría estar ubicado en un nivel mayor al del dominio de la información, porque no podría cumplir con la restricción de la latencia que necesitan los procesos industriales para mantener un buen rendimiento en el control de una planta.

3.3.3. Implementación práctica de la arquitectura

En (Luo et al., 2017) se implementa de manera real la arquitectura *three-tier Industrial Internet System* (IIS) para gestionar un sistema de manufactura inteligente heterogéneo, esta es una arquitectura de ejemplo mencionada en la arquitectura IIRA que pretende mostrar de manera generalizada los patrones que existen en la implementación de un sistema IIoT. Esta arquitectura implementada tiene tres niveles: *Edge*, *Platform* y *Enterprise*, en la figura 3.7 se muestran estos niveles y sus componentes, en la capa *edge* se recogen los datos de los dispositivos de campo, en esta capa se realizó el control de los dispositivos según las reglas de control de los niveles superiores. El nivel *Platform* se utilizó como medio para conectar los niveles *edge* y *Enterprise*, se desarrollaron interfaces de servicios y se implementó un servidor OPC UA, con esto este nivel proveía el servicio de recibir, procesar y enviar comandos de control desde el nivel *Enterprise* hacia el nivel *edge*, además, se usó para recolectar información para el nivel superior. El nivel *Enterprise* se implementaron las funcionalidades de alto nivel y los sistemas de decisiones enfocados en el manejo de la energía, esto permitió operaciones estratégicas para los usuarios finales.

Las tres principales funcionalidades fueron: aplicación web, cliente OPC UA y servicio de bases de datos.

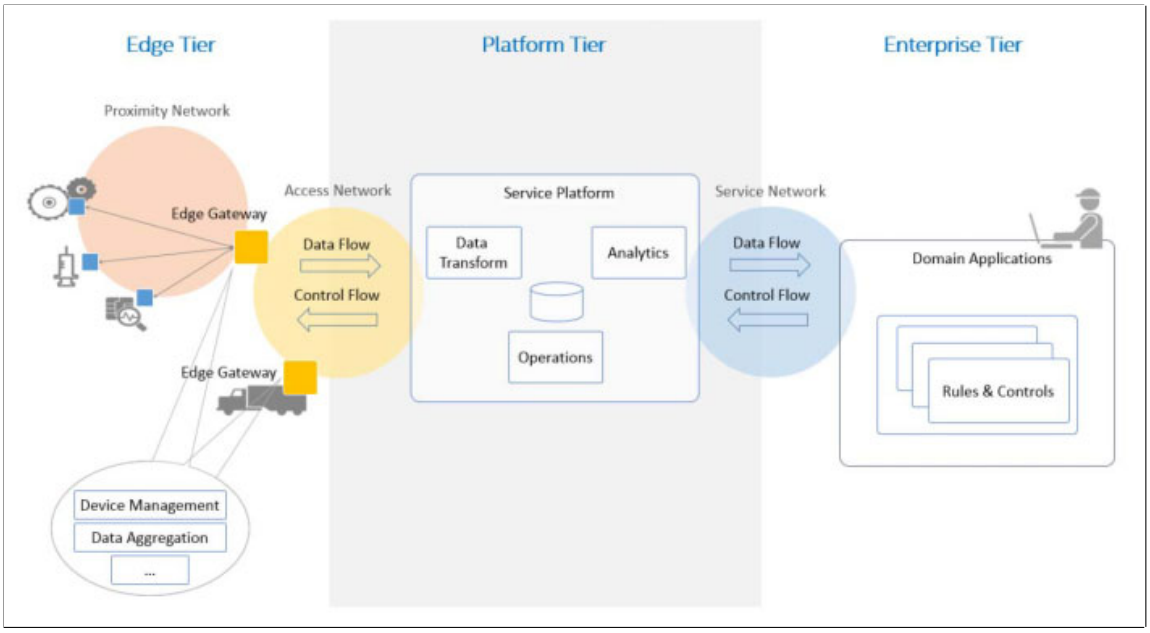


Figura 3.7. Arquitectura *three-tier Industrial Internet System* para gestión de un sistema de manufactura (Luo et al., 2017).

4. TECNOLOGÍAS PARA LA INDUSTRIA 4.0

Las infraestructuras de comunicación en los sistemas de automatización son complejas y se están volviendo aún más complejas y heterogéneas. En el contexto de la transferencia de información entre distintas entidades de un sistema de automatización industrial, es evidente que las redes de comunicación por sí solas no serán suficientes. El transporte real de la información es sólo un aspecto. Igualmente importante es la descripción y modelación de la información, así como las definiciones de cómo acceder a ella (Wollschlaeger, Sauter, & Jasperneite, 2017). Es por esto que el estudio y diseño de nuevas tecnologías de alto nivel para el manejo y almacenamiento de datos y su compatibilidad operativa con los protocolos de comunicación se vuelve importante en el desarrollo de ICPS.

Además, el papel de las interacciones orientadas a los servicios, así como la nube, están cambiando la forma en que se diseñan, despliegan y gestionan los CPSs. Utilizando las capacidades intrínsecas de la nube, como la virtualización, la escalabilidad, el rendimiento y la gestión del ciclo de vida, se pueden realizar mejores CPSs que pueden incluir dispositivos más ligeros, ya que las piezas más exigentes pueden ahora alojarse en la nube o en capas más cercanas al borde de la red. Así, los CPSs asistidos por la nube se consideran un factor clave para una multitud de escenarios tanto entre pares como en los de capas cruzadas (por ejemplo, entre SCADA y ERP) (Colombo, Karnouskos, Kaynak, Shi, & Yin, 2017).

Por estas razones, en este capítulo se estudiarán la virtualización de sistemas operativos en contenedores, protocolos de comunicación y bases de datos NoSQL que pueden complementarse para formar un ICPS orientado a la Industria 4.0. Estos temas son relevantes para un PLC de la Industria 4.0, porque con la virtualización en contenedores es posible desplegar aplicaciones que funcionen dentro de un PLC asistidas por la nube de forma automatizada en los dispositivos que se necesitan, sin la necesidad de tener un dispositivo y programa específico para la carga de programas de control, como sucede actualmente. Además, los contenedores permiten que las aplicaciones se ejecuten de forma

segura y aisladas entre ellas, por lo que las aplicaciones críticas para la estabilidad del sistema no se verían afectadas por las no críticas.

Por otro lado, los protocolos de comunicación inalámbricos (WirelessHART, ISA100.11a y ZigBee) que se explicarán le permitirían a un PLC orientado a la Industria 4.0 comunicarse con dispositivos de campo que no son posibles conectarlos mediante comunicaciones alámbricas a un mismo PLC, así sería posible monitorear y controlar sistemas más grandes con un solo algoritmo de control que pudiese obtener mejores resultados. También, se explicarán protocolos para dispositivos y redes con recursos restringidos (CoAP y MQTT), porque pueden ser aprovechados por dispositivos de capa *fog* que se comuniquen con otros dispositivos en esta misma capa, con el uso estos se minimizaría el uso de los recursos de la red y de los dispositivos en ella.

Por último, se describirán los distintos tipos de sistemas de bases de datos NoSQL y dos de las implementaciones más populares, Redis y MongoDB. Utilizar Redis, la cual guarda los datos en la memoria RAM, en un PLC que incluya sus aplicaciones en contenedores le permitiría a esta base de datos actuar como un enlace rápido de intercambio de información entre ellos con un rápido despliegue en el PLC. En el caso de MongoDB, base de datos orientada a los documentos, se permitiría que los diversos tipos de configuraciones de dispositivos, mensajes y datos de los controladores pudiesen guardarse en una misma base de datos sin tener que configurar específicamente cada tipo de documento, esto hace que el intercambio de información con capas superiores sea más flexible y simple.

4.1. Virtualización de sistemas operativos con contenedores

Un sistema operativo (SO) es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de *hardware* y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes (Tanenbaum & Renesse, 1992). Mientras que la virtualización de un SO significa instalar

un sistema operativo dentro de otro al que se le llama anfitrión (*host*), con esta forma de virtualización se crean particiones aisladas o entornos virtuales en un servidor físico.

Dada la diversidad de sistemas operativos, la virtualización de estos ha crecido en popularidad durante la última década, para permitir que el *software* se ejecute de manera predecible y correcta cuando se traslada de un entorno computacional a otro.

En este contexto es que los contenedores aparecen para proporcionar una forma de ejecutar sistemas operativos aislados en un solo dispositivo o sistema anfitrión. Los contenedores son una forma de virtualización pero, a diferencia de las máquinas virtuales (VMs), no virtualizan el *hardware* subyacente, sino que se sitúan sobre un dispositivo físico y de su SO anfitrión, por ejemplo, Linux o Windows. Al no virtualizar el *hardware*, cada contenedor debe compartir el núcleo del SO anfitrión para operar, pero los binarios, las bibliotecas y archivos necesarios son los únicos elementos que están contenidos dentro del contenedor.

En la literatura se hace hincapié en que los contenedores no son máquinas virtuales (Cook, 2017). De hecho, para un determinado sistema anfitrión, cualquier aplicación que se ejecuta dentro de un contenedor sigue utilizando el mismo núcleo que una aplicación que se ejecuta de forma nativa en ese anfitrión. Por el contrario, una aplicación que se ejecuta en una máquina virtual se ejecuta en un núcleo separado (y probablemente diferente) del anfitrión.

El despliegue de una aplicación en diversos sistemas resulta problemático, debido a que el árbol de dependencia de una aplicación puede ser bastante complejo, ya que incluso las dependencias inmediatas de la aplicación tienen dependencias en sí mismas y cada sistema puede tener diversas versiones de esas dependencias. Además, en el desarrollo de aplicaciones es importante mantener el entorno de desarrollo tan estable y fiable como sea posible, junto con que en un entorno de desarrollo distribuido, este necesita ser consistente en todas las máquinas. Estos son los principales temas en que los contenedores se enfocan, estos facilitan el desarrollo, el envío y la ejecución de las aplicaciones,

ya que estas se empaquetan en un objeto, llamado contenedor, junto con las utilidades y las bibliotecas compartidas de las que dependen, resolviendo así las complicaciones de mantener un entorno de desarrollo consistente y la de tener que verificar las dependencias en cada sistema en que se ejecuta la aplicación. Es por esto que la virtualización basada en contenedores es una gran opción para los microservicios, las operaciones de desarrollo y el despliegue continuo. (Holt & Huang, 2018)

En particular, el núcleo Linux soporta una serie de características de aislamiento que los contenedores utilizan al ejecutarse, a saber, espacios de nombres (*Namespaces*) y grupos de control (cgroups). Los espacios de nombres son una abstracción de algún recurso de todo el sistema. Un proceso que se ejecuta en un espacio de nombres específico parece tener su propia instancia de un recurso, aislándolo así del resto del sistema. Los grupos de control son un medio de limitar los recursos del sistema que los procesos (o grupos de procesos) pueden consumir.

Aunque los contenedores permitan trabajar en un entorno estable en el tiempo, para que los desarrolladores puedan utilizar versiones recién publicadas de las bibliotecas de *software*, versiones más antiguas que las versiones estables del sistema anfitrión o versiones reducidas de las librerías que existen en su anfitrión de escritorio, en algún momento puede ser necesario actualizar los componentes de los contenedores, ya que el soporte para ellos terminará finalmente y los errores y parches de seguridad de nuevas versiones se harán necesarios. Por lo tanto, cualquier plataforma informática, incluso si utiliza contenedores, rara vez se queda estática en cuanto a las versiones de su *software* (Holt & Huang, 2018).

4.1.1. Ventajas y desventajas de los contenedores frente a las máquinas virtuales

Entre las ventajas de los contenedores sobre las VMs está en que estas últimas ocupan más recursos del sistema, ya que cada una no sólo ejecuta una copia completa de un sistema operativo, sino que requiere una copia virtual de todo el *hardware* que el sistema operativo requiere para funcionar, esto se puede apreciar gráficamente en la Figura

4.1. Lo anterior se puede apreciar en el mayor uso de ciclos CPU y el uso de RAM en comparación con el uso de contenedores, en (Morabito, Kjällman, & Komu, 2015) se estudia el rendimiento de una máquina virtual (utilizando KVM) en comparación con las tecnologías de contenedores (Docker y LXC), los resultados de las pruebas muestran que la máquina virtual introduce una degradación de hasta un 30% en pruebas de CPU en un solo hilo, 35% en la escritura de información en un disco, ninguna degradación práctica del rendimiento en tareas que utilizan la memoria RAM y hasta un 28 % en la transferencia de información mediante TCP y UDP .

Por el contrario, todo lo que un contenedor requiere son las librerías mínimas necesarias para ejecutar un sistema operativo y las bibliotecas de apoyo y recursos del sistema para ejecutar un programa específico. Lo que esto significa en la práctica es que es posible correr muchas más aplicaciones en un solo dispositivo con contenedores que con una máquina virtual y el rendimiento se verá menos perjudicado al usar contenedores.

Además, los contenedores son excepcionalmente livianos: tienen un tamaño del orden de los megabytes y tardan sólo segundos en arrancar, a diferencia de los gigabytes y minutos que puede necesitar una VM. En (Kämäräinen, Shan, Siekkinen, & Ylä-Jääski, 2015) se analiza la diferencia en el rendimiento entre contenedores (Docker) y máquinas virtuales (QEMU) en los que se corren juegos en una nube, los resultados relacionados al tiempo que toma en inicar cada juego fueron que al contenedor le tomaba 0,3 segundos en iniciar y a la máquina virtual 24,9 segundos, la razón dada es que la diferencia en el tiempo se debe a que la máquina virtual debe iniciar un SO desde cero antes del juego, mientras que el contenedor solo inicia el juego.

Los contenedores ofrecen tanto eficiencia de recursos como flexibilidad de uso. La flexibilidad proviene de que el contenedor puede llevar todos los archivos que necesita con él. Al igual que una aplicación que se ejecuta en una máquina virtual, este puede tener sus propios archivos de configuración y bibliotecas dependientes, así como sus propias interfaces de red que son distintas de las configuradas en el anfitrión. Sin embargo, una

aplicación en contenedores es más fácil de mover que sus contrapartes directamente instaladas, y no tiene que competir por recursos tales como números de puertos, porque cada contenedor en el que se ejecutan tiene interfaces de red separadas.

Por otro lado, una de las principales desventajas de la virtualización basada en contenedores, en comparación con las máquinas virtuales tradicionales, es la seguridad. Los contenedores comparten el núcleo y otros componentes del sistema operativo del anfitrión. Esto significa que los contenedores están menos aislados entre sí que las máquinas virtuales, que tienen su propio sistema operativo. Si existe una vulnerabilidad en el núcleo, puede poner en peligro la seguridad de todos los contenedores. Las máquinas virtuales sólo comparten el hipervisor, lo que las hace menos propensas a los ataques que los núcleos compartidos de los contenedores.

También, las máquinas virtuales con cualquier tipo de sistema operativo pueden residir una junto a la otra en el mismo servidor, pero se debe iniciar un nuevo servidor, para poder ejecutar contenedores con diferentes sistemas operativos. Para aplicaciones empresariales complejas, esto puede ser una limitación seria. Además de eso, desplegar contenedores de una manera suficientemente aislada mientras se mantiene una conexión de red adecuada también puede ser desafiante (Jangla, 2018).

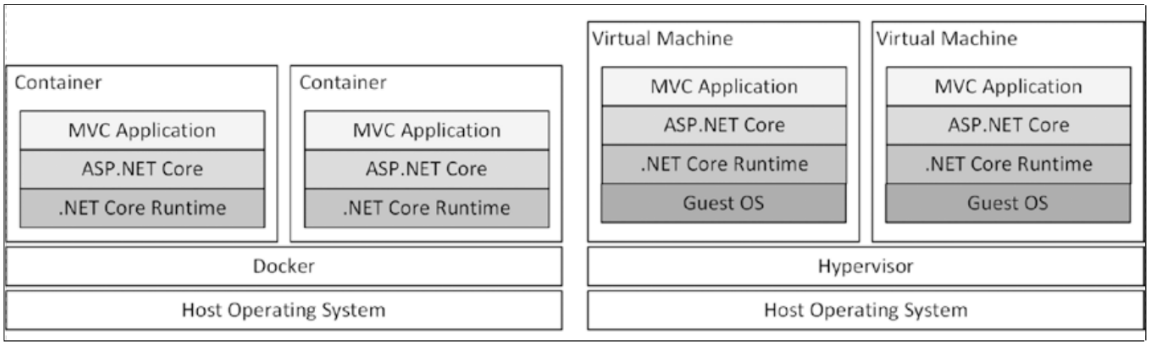


Figura 4.1. Diferencia entre utilizar un contenedor y máquina virtual para una aplicación (Freeman, 2017).

4.1.2. Plataforma para iniciar procesos en contenedores: Docker

Docker es una plataforma para iniciar procesos en contenedores que se ha convertido en la plataforma de facto para el transporte, manejo y funcionamiento de ellos. Con esta plataforma múltiples contenedores funcionan de manera aislada en el mismo núcleo del sistema operativo subyacente, y cada contenedor tiene su propia red y sistema de archivos. Cada contenedor Docker es un encapsulado del software y las dependencias necesarias para una aplicación y no incurre en los gastos generales de empaquetar un sistema operativo completo, que podría ser de varios gigabytes. En Docker las aplicaciones se ejecutan a partir de las imágenes asociadas a los contenedores, y cada imagen es específica para una aplicación o un software determinado. Una imagen se construye a partir de un Dockerfile, un Dockerfile es el archivo que define el conjunto de instrucciones que se utilizarán para descargar e instalar el software, establecer las variables de entorno y ejecutar los comandos (Vohra, 2017).

Una imagen de un contenedor Docker no es sólo un archivo, sino más bien un sistema de archivos. Este sistema de archivos está compuesto de múltiples capas, y cada capa contiene un archivo del contenido de esa capa que no puede ser cambiado. En otras palabras, es inmutable. Es esencialmente una instantánea (*snapshot*) de un contenedor Docker.

Las imágenes pueden llegar a ser bastante grandes muy rápidamente. Por lo tanto, están diseñadas para estar compuestas por capas de otras imágenes, permitiendo que se envíe una cantidad mínima de datos cuando se transfieran las imágenes a través de una red y que puedan guardarse en un registro en que las imágenes de varios contenedores puedan compartir sus capas.

La plataforma de administración de contenedores Kubernetes es una herramienta que permite la administración y despligue automatizado de contenedores en dispositivos distribuidos, esta plataforma se trata en el Anexo C.

4.2. Protocolos de comunicación inalámbricos

Ha habido un enorme interés en las tecnologías de redes de sensores inalámbricos (WSN) por parte de académicos, la industria y los desarrolladores de tecnología, debido a sus características. Entre las ventajas de la aplicación de tecnologías inalámbricas en los sistemas industriales para la vigilancia y el control de equipos y procesos figuran la posibilidad de flexibilizar la configuración, el apoyo a la movilidad y la eliminación del costoso cableado. Además, las tecnologías inalámbricas industriales permiten una expansión más fácil de la red para mejorar la productividad y la eficiencia (Candell, Kashef, Liu, Lee, & Foufou, 2018).

Para poder apoyar a distintas aplicaciones y atender a sus necesidades específicas, se han propuesto diferentes protocolos para las WSN diseñados para operar tanto en ambientes industriales como en no industriales, a continuación se presentan tres de ellos basados en IEEE 802.15.4 y que tienen dispositivos comerciales ya disponibles en el mercado.

4.2.1. WirelessHART

WirelessHART es el primer estándar industrial abierto de comunicación inalámbrica que fue diseñado como la extensión inalámbrica del protocolo HART. WirelessHART fue diseñado, desarrollado y estandarizado teniendo en cuenta los sistemas industriales y soporta los sistemas construidos sobre HART alámbrico. Comparado con el protocolo HART, wirelessHART tiene muchas ventajas en el campo del control industrial, este aporta simplicidad, robustez, menores costes de instalación y mantenimiento y una configuración más flexible a las aplicaciones de automatización y control industrial (Sahin & Ammari, 2014).

Además, WirelessHART soporta enrutamiento redundante para mejorar la fiabilidad. Por lo tanto, se considera que WirelessHART es robusto, energéticamente eficiente y confiable. Por otro lado, las topologías de red mantenidas por el administrador de la red en WirelessHART son tipo estrella y malla (*mesh*) (Kim & Tran-Dang, 2019).

WirelessHART utiliza tecnologías de salto de canal, acceso múltiple por división de tiempo (TDMA) y codificación de secuencia directa por ancho de espectro (DSSSC) para superar la interferencia con otras redes que se superponen; encuentra fácilmente caminos alternativos y ajusta los caminos de comunicación para proporcionar un rendimiento óptimo. En la Figura 4.2 se muestran las capas de la pila OSI utilizadas por WirelessHART, en ella se puede ver que en las dos primeras capas sigue el estándar IEEE 802.15.4 y en las tres restantes WirelessHART define sus propias capas.

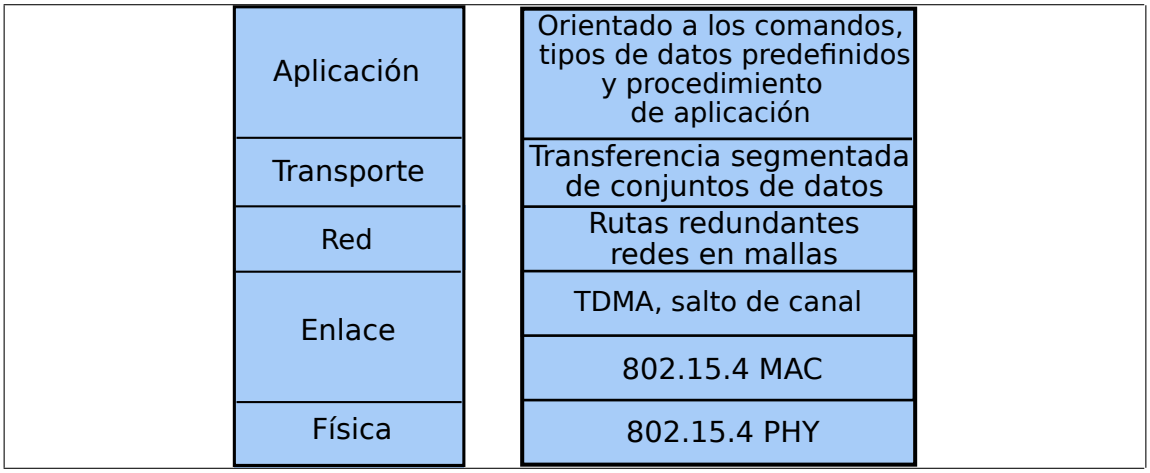


Figura 4.2. Pila OSI del protocolo WirelessHART.

En la Figura 4.3 se muestra una red WirelessHART y sus componentes, estos son descritos a continuación:

- Dispositivos de campo: Los dispositivos de campo son los sensores físicos distribuidos que están conectados al equipo de proceso o planta y que son capaces de enrutar y reenviar paquetes. Los dispositivos de campo pueden conectarse al equipo de la planta a través de otra red inalámbrica o pueden conectarse directamente a la red WirelessHART.
- Adaptadores: Los adaptadores son los habilitadores de la integración de los dispositivos HART cableados en la red WirelessHART. Uno o más dispositivos HART pueden conectarse a una red WirelessHART a través de adaptadores. En

el caso de las redes HART punto a punto, sólo se puede utilizar un adaptador para conectar un dispositivo HART a la red.

- Enrutadores: Los enrutadores son tipos especiales de dispositivos de campo, sin embargo, no interfieren con el proceso hasta que son necesarios para la mejora de la conectividad inalámbrica.
- *Gateway*: Los *gateways* se reconocen como los puentes que proporcionan la conexión entre la red WirelessHART y la planta.
- Puntos de acceso: La conexión física real a la red WirelessHART es proporcionada por los puntos de acceso. Cada punto de acceso tiene una identificación única.
- Administrador de la red: El administrador de la red es responsable de la configuración y mantenimiento general de la red WirelessHART. Recoge información de los dispositivos de campo a través del *gateway* para determinar el estado de la red, las rutas que se deben establecer, gestionar los recursos dedicados y compartidos. El administrador de la red actualiza la información de enrutamiento y el programa de comunicación cuando nuevos nodos se unen a la red WirelessHART.
- Administrador de seguridad: El Administrador de seguridad es responsable de supervisar el estado de seguridad de la red, prevenir ataques, generar claves de sesión, de unión y de red con la incorporación del gestor de red.

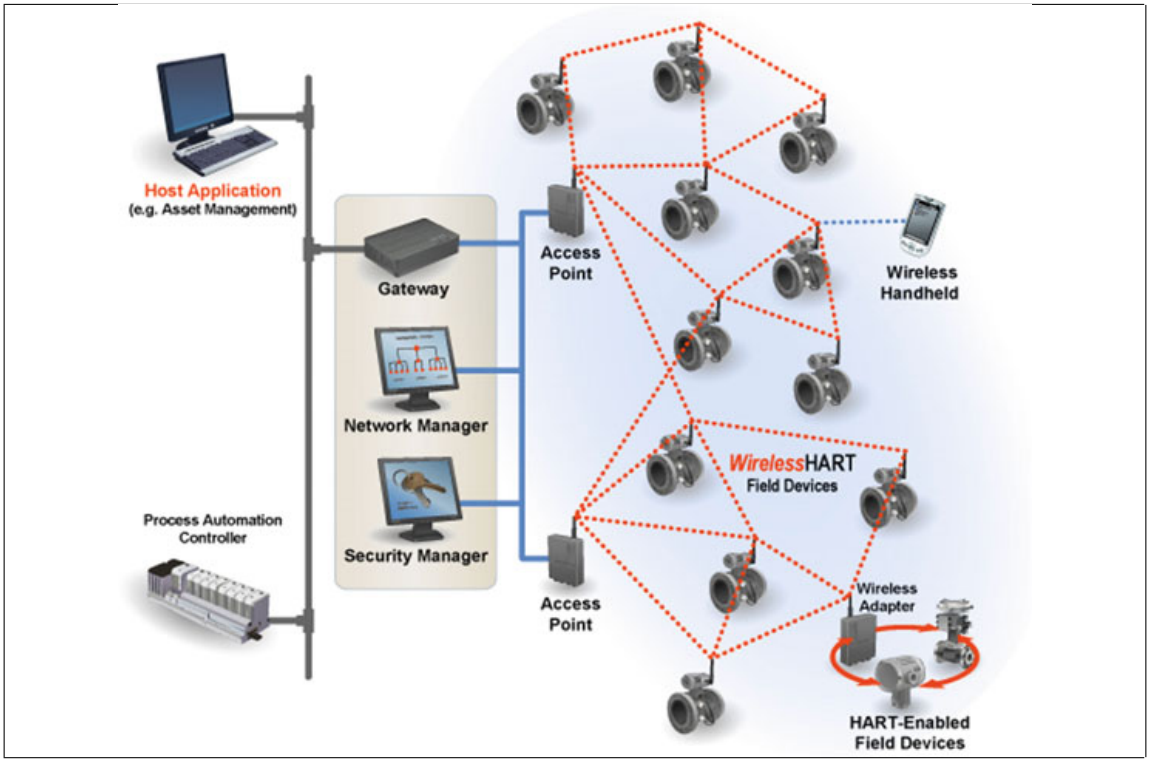


Figura 4.3. Red WirelessHART y sus componentes (Kim & Tran-Dang, 2019).

4.2.2. ISA100.11a

En 2009, el comité ISA.100 aprobó la norma ISA100.11a, que es una norma de tecnología de redes inalámbricas abiertas y multifuncionales para proporcionar comunicaciones de datos fiables, robustas y seguras para aplicaciones industriales no críticas, de alerta, de control de supervisión, de control de lazo abierto y de control de lazo cerrado. La norma ISA100.11a tiene una amplia cobertura de aplicaciones y se conecta fácilmente a diferentes tipos de redes de comunicación, la conectividad inalámbrica de baja velocidad de datos se apoya con mayores niveles de seguridad y gestión del sistema y las aplicaciones del orden de los 100ms son totalmente compatibles. Además, ISA100.11a proporciona una comunicación fluida con los protocolos de aplicación existentes, por ejemplo, HART, Modbus, Profibus, etc.

Similar a wirelessHART, la capa física se basa en el IEEE 802.15.4. ISA100.11a también utiliza salto de canal y lista negra de canales para reducir los efectos de la interferencia. El protocolo ISA100.11a es muy robusto, ya que utiliza tres tipos de diversidad diferentes, la diversidad espacial para utilizar trayectos múltiples para el reenvío de datos con capacidad de red en malla, la diversidad de frecuencia para permitir el salto de frecuencia y la diversidad temporal para manejar un mecanismo de reintento en las transmisiones.

La red ISA100.11a es capaz de manejar miles de dispositivos, sin embargo, existen limitaciones prácticas para no aumentar negativamente el tráfico de datos de la red y el consumo de energía de los nodos de los sensores.

ISA100.11a emplea tres tipos de red: topologías de estrella, malla y malla-estrella, según las funciones de los dispositivos en la red. Por ejemplo, en una red ISA100.11a, la función de enrutamiento está separada de las funciones de los sensores y los actuadores. Con esta separación, los dispositivos de campo de ISA100.11a se denominan dispositivos finales sin capacidad de enrutamiento o nodos de enrutamiento con capacidad de enrutamiento.

En la Figura 4.4 se muestran las capas de la pila OSI utilizadas por ISA100.11a, en ella se puede ver que en las dos primeras capas sigue el estándar IEEE 802.15.4, luego agrega una subcapa propia de este protocolo, en la siguientes dos capas (Red y transporte) ISA100.11a utiliza 6LoWPAN y UDP los cuales están estandarizadas y ampliamente difundidos en la literatura y en productos comerciales.

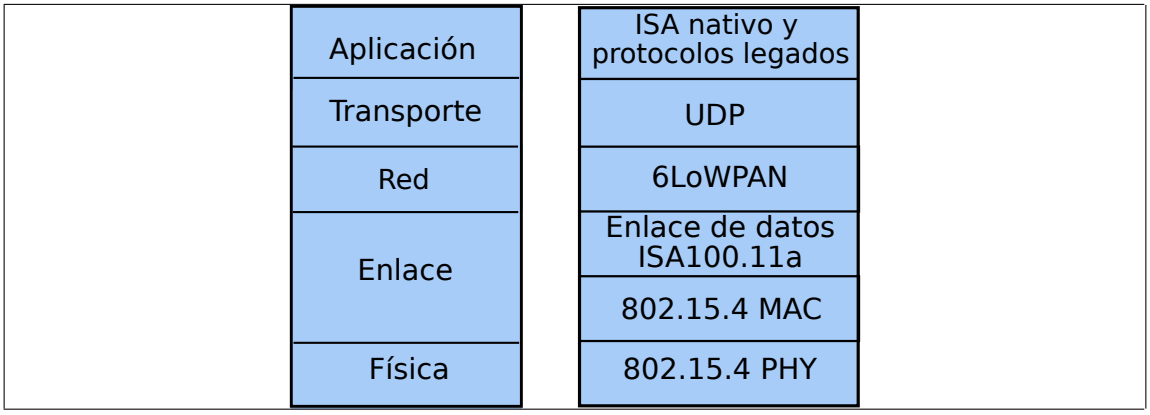


Figura 4.4. Pila OSI del protocolo ISA100.11a.

En la Figura 4.5 se muestra una red ISA100.11a y sus componentes, estos son descritos a continuación:

- *Gateway*: El *gateway* es uno de los dispositivos de campo más importantes de la red. Actúa como una interfaz entre la red inalámbrica y la red de la planta.
- Administrador del sistema: El administrador del sistema es responsable de controlar la red, los dispositivos de red, los recursos de red y las comunicaciones. Después de unirse a la red, para poder optimizar la topología de la red, comienza a asignar recursos y proporciona una lista de los nodos vecinos apropiados. Para ello, el administrador del sistema debe conocer el nivel de conectividad de la red con las mediciones reales de las calidades de los enlaces.
- Administrador de seguridad: El administrador de seguridad proporciona servicios de gestión clave para permitir comunicaciones de datos seguras con la cooperación del administrador del sistema, autentifica los dispositivos. Para poder establecer una comunicación con un nodo vecino, el nodo correspondiente debe requerir una nueva clave de sesión del gestor de seguridad. Por lo tanto, puede garantizar que sólo los dispositivos autentificados pueden comunicarse entre sí.
- Dispositivo de enrutamiento: El dispositivo de enrutamiento es capaz de enrutar datos a los otros dispositivos de la red ISA.100.11a.

- Dispositivo no enrutador: El dispositivo no enrutador no es capaz de enrutar datos, proporciona datos de sensores a otros dispositivos o utiliza datos de otros dispositivos.
- *Backbone Router*: El *Backbone Router* es responsable de establecer la conexión con otras redes mediante el enrutamiento de datos a la red principal o desde la red principal y encapsula los datos que llegan a los dispositivos ISA100.11a.

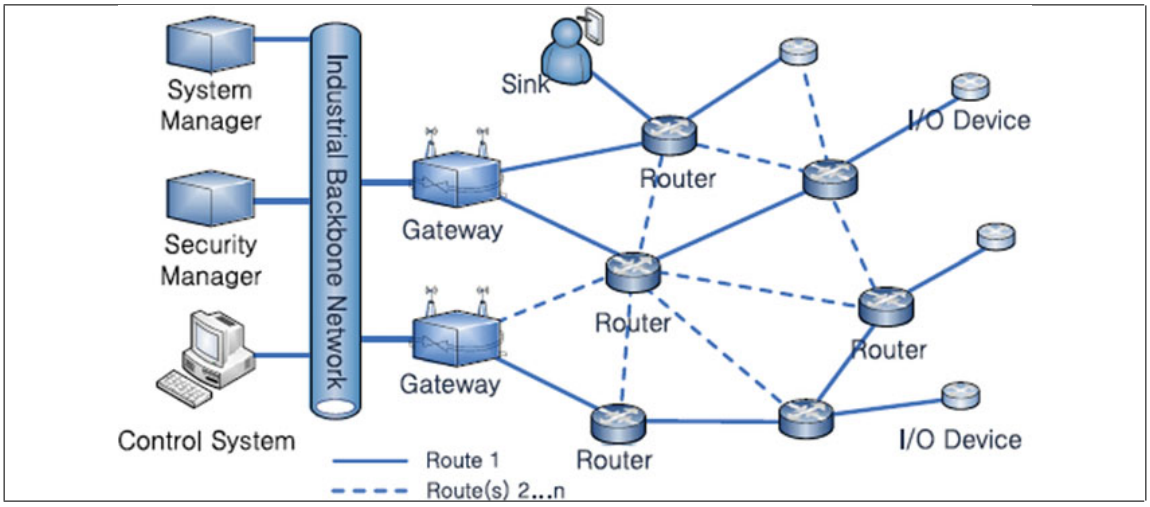


Figura 4.5. Red ISA100.11a y sus componentes (Kim & Tran-Dang, 2019).

4.2.3. Zigbee

ZigBee es una tecnología inalámbrica estandarizada, de corta distancia, bajo costo y bajo consumo, basada en el protocolo IEEE 802.15.4, diseñada para satisfacer la creciente demanda de redes inalámbricas de bajo consumo y eficientes entre aplicaciones de redes de sensores y de control. ZigBee es propuesta por la ZigBee Alliance, que es un grupo de compañías que proporciona estándares ZigBee innovadores, fiables y fáciles de usar. ZigBee es una tecnología de red de malla inalámbrica auto-organizada que soporta más de 64.000 dispositivos en una sola red. La topología en estrella de ZigBee es la topología menos compleja que puede conducir a la simplicidad y a una mayor fiabilidad, por otra

parte, las topologías de P2P pueden aumentar la fiabilidad, ya que incluyen múltiples caminos hacia el coordinador de ZigBee y los dispositivos finales de ZigBee.

ZigBee utiliza diferentes técnicas de espectro ensanchado para protegerse de la interferencia de multitrayecto y de banda estrecha. De hecho, la norma IEEE 802.15.4 sugiere varios canales en las bandas de 915 MHz y 2,4 GHz para ZigBee. ZigBee puede seleccionar el canal menos interferido para luchar contra la interferencia. Además, el coordinador de ZigBee tiene la capacidad de reunir la red ZigBee en un canal diferente en caso de cualquier situación de interferencia. Además, la capa IEEE 802.15.4 MAC se basa en CSMA/CA (Carrier sense multiple access with collision avoidance) que el nodo sensor escuchará el canal antes de transmitir el paquete.

En la Figura 4.6 se muestran las capas de la pila OSI utilizadas por ZigBee, en ella se puede ver que en las dos primeras capas sigue el estándar IEEE 802.15.4, las capas restantes son definidas por ZigBee para su funcionamiento en específico.

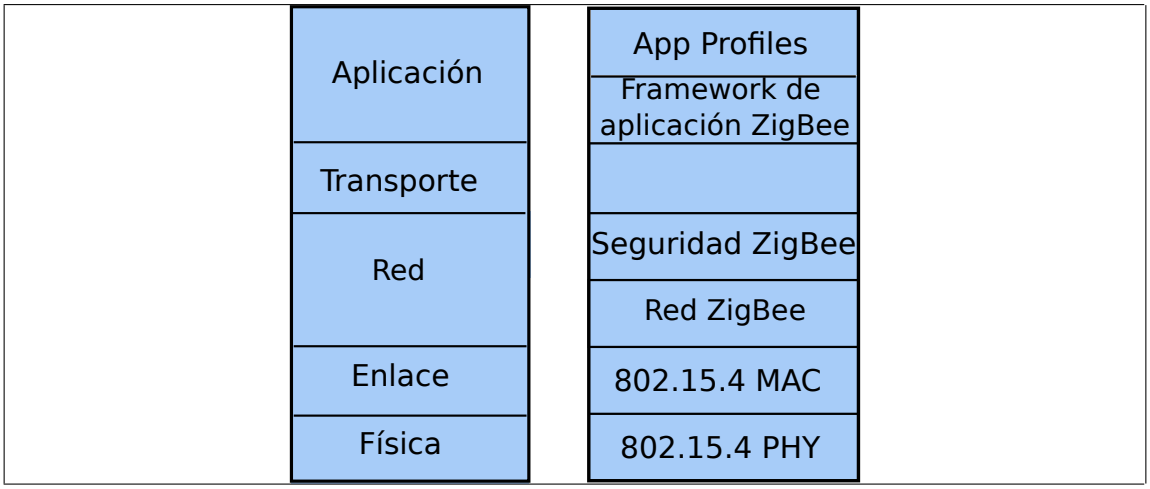


Figura 4.6. Pila OSI del protocolo Zigbee.

En la Figura 4.7 se muestra una red ZigBee y sus componentes, estos son descritos a continuación:

- Coordinador ZigBee (ZC): El coordinador forma la raíz del árbol de la red y puede tender un puente a otras redes. Hay exactamente un coordinador ZigBee en cada red, ya que es el dispositivo que inicia la red originalmente (la especificación ZigBee Light Link también permite el funcionamiento sin un ZigBee Coordinator, lo que lo hace más utilizable para los productos domésticos de venta al público). También, almacena información sobre la red, incluyendo el actuar como el centro de confianza y el depósito de las claves de seguridad.
- Enrutador ZigBee (ZR): Además de ejecutar una función de una aplicación, un enrutador ZigBee puede actuar como un enrutador intermedio, transmitiendo datos de otros dispositivos.
- Dispositivo final ZigBee (ZED): Contiene la funcionalidad suficiente para hablar con el nodo padre (ya sea el coordinador o un enrutador); no puede transmitir datos de otros dispositivos. Esta relación permite que el nodo esté dormido una cantidad significativa del tiempo, lo que alarga la vida de la batería que pueda estar utilizando. Un ZED requiere la menor cantidad de memoria, y por lo tanto su fabricación puede ser menos costosa que la de un ZR o un ZC (Kim & Tran-Dang, 2019).

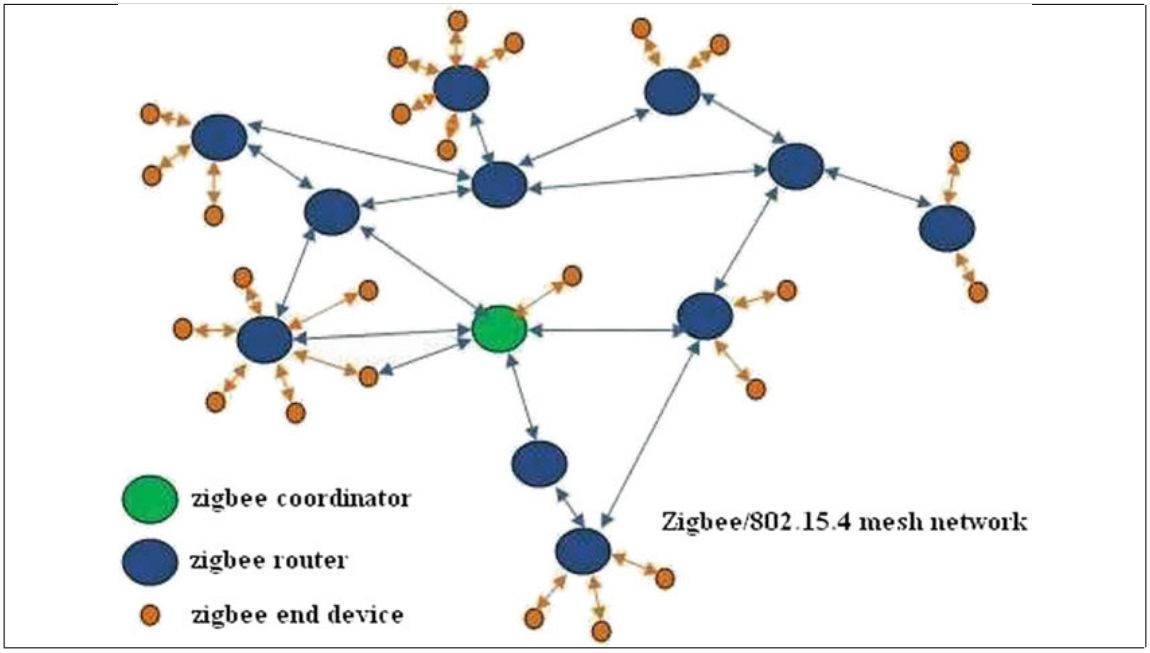


Figura 4.7. Red ZigBee y sus componentes (Kim & Tran-Dang, 2019).

ZigBee no es un protocolo diseñado para operar en aplicaciones industriales, pero en la literatura se han realizado estudios acerca de la aplicabilidad de este protocolo en ambientes industriales. En (Lennvall, Svensson, & Hekland, 2008) se analizó la diferencia entre ZigBee y WireslessHART de forma teórica, las conclusiones de la comparación sugieren que WirelessHART es más apto para el control de procesos porque es más robusto, sufre de menos interferencia en el canal inalámbrico, el enrutador consume menos energía y es más seguro, ya que la información va cifrada y hay autentificación de los mensajes.

En (Ramos et al., 2018) se compara el rendimiento de una red WirelessHART con una ZigBee ubicadas en una planta refinadora de petróleo, cada red estaba compuesta por un *gateway* (WirelessHART) o coordinador (ZigBee) y cuatro nodos, cada uno de ellos tenía una separación con otro de entre 20 a 40 metros, había una configuración en la que algunos dispositivos estaban en la línea de visión y otros que no lo estaban. La conclusión del estudio es que para configuraciones de red pequeñas, como la estudiada, el indicador

de la intensidad de la señal recibida (RSSI) y la tasa de transmisión exitosa (STR) son comparables entre ZigBee y WirelessHART.

Por otro lado, en (Habib, Haddad, & Khoury, 2015) se analizan la diferencias en la tasa de pérdida de paquetes y el retardo punto a punto entre una red WirelessHART y ZigBee con 20 nodos, incluyendo al *gateway* (WirelessHART) y coordinador (Zigbee). Los resultados muestran que el retardo es menor en Zigbee (cercano a 50 ms versus 400 ms de WirelessHART), mientras que la pérdida de paquetes es aproximadamente cero para WirelessHART y de 20% a 60% para ZigBee.

Con estos resultados puede concluirse que ZigBee no es apto para el control de procesos, debido a la pérdida de paquetes y su consecuente retardo en el envío de información, pero sí es factible utilizarlo en ambientes industriales para la medición de variables.

En la Tabla 4.1 se muestran cinco características relevantes para las aplicaciones industriales de los protocolos que se describieron.

Tabla 4.1. Características relevantes para las aplicaciones industriales de los protocolos ISA100.11a, WirelessHART y ZigBee (Vilajosana et al., 2020).

	ISA100.11a	WirelessHART	ZigBee
IP *compliant*	Sí	Sí	No
Acceso	TSCH	TSCH	CSMA/CA
Planificación	Centralizada	Centralizada	Sin planificación
Capa de aplicación	Objetos de *software*, ISA nativo	Comandos HART	Objetos y perfiles ZigBee
Transporte	Sin conexión (*connectionaless*) UDP	Servicio sin conexión (*connectionaless*)	Sin capa de transporte

4.3. Protocolos enfocados en dispositivos con recursos restringidos

Dada la gran cantidad y diversidad de dispositivos que pueden llegar a conectarse a una red en un sistema IoT se han desarrollado protocolos de comunicación ligeros para extender la longevidad, la escalabilidad y la fiabilidad de la red, por lo que podrían ser aprovechados para conectar PLCs entre ellos y, de esta manera, permitir el intercambio de información de dispositivos o variables de un proceso. Dos de estos protocolos son CoAP y MQTT, los cuales serán descritos a continuación.

4.3.1. CoAP

CoAP es un protocolo de transferencia de datos optimizado para las comunicaciones máquina-máquina entre dispositivos con bajos recursos computacionales. CoAP proporciona un modelo de interacción de solicitud/respuesta, intercambiando pequeños mensajes en la capa de transporte UDP. Es adecuado para nodos restringidos y redes restringidas (por ejemplo, de baja potencia, con pérdidas). En particular, CoAP permite añadir soporte UDP para peticiones unidifusión y multidifusión, intercambiar datagramas asíncronos sin conexión, baja sobrecarga de cabecera en los paquetes transmitidos, baja complejidad de las operaciones y soporta mecanismos simples para acciones de *proxy* y de caché. En cuanto a la fiabilidad, CoAP admite cuatro opciones en cuanto a los diferentes tipos de mensajes, a saber, Confirmable, No confirmable, Acuse de recibo y Reinicio (Bellavista & Zanni, 2016).

El modelo interactivo de CoAP es similar al modelo cliente/servidor de HTTP. CoAP emplea una estructura de dos capas, una capa inferior, la capa de mensajes, que ha sido diseñada para tratar con UDP y la capa de solicitud/respuesta, que se refiere al método de comunicación y se ocupa del mensaje de solicitud/respuesta.

Así, el protocolo CoAP está diseñado para ser utilizado y considerado como un sustituto del protocolo HTTP por ser un protocolo ligero de capa de aplicación para sistemas IoT.

4.3.2. MQTT

MQTT es un protocolo ligero de conectividad máquina-máquina para IoT, está basado en un modelo de publicación/suscripción. Permite lograr una buena escalabilidad y puede soportar dinámicamente una amplia gama de aplicaciones, especialmente en los dominios de IoT y máquina-máquina. Cada recurso MQTT está modelado como un cliente y puede conectarse a un *broker* MQTT sobre el protocolo TCP. MQTT tiene características intrínsecas que lo convierten en una opción valiosa en los entornos de IoT con requisitos de baja latencia, bajo ancho de banda y eficiencia energética, por ejemplo, un pequeño encabezamiento de cabecera y reenvío automático de mensajes cuando los clientes se reconectan (Bellavista & Zanni, 2016). Además, proporciona fiabilidad con la posibilidad de seleccionar dinámicamente una de las tres opciones de nivel de calidad de servicio (QoS) para el envío de mensajes:

- QoS0: A lo sumo el mensaje es enviado una vez, con el modo de mejor esfuerzo de la red. La recepción de un mensaje no requiere un respuesta para informar el recibo y el mensaje no se almacena. El mensaje podría perderse o duplicarse. Es el modo más rápido de transferir mensajes.
- QoS1: El mensaje se envía al menos un vez y podría ser entregado varias veces, con posibles duplicaciones. Con esta opción, los mensajes deben ser almacenados localmente en el remitente, hasta que hayan sido entregados a su receptor, para permitir posibles retransmisiones.
- QoS2: El mensaje se envía exactamente una vez, con la garantía de que no se produzca ninguna duplicación de los mensajes. Amplía la QoS1, almacenando los mensajes también en los receptores para evitar cualquier duplicación.

4.4. Bases de datos NoSQL

Las bases de datos SQL siguen un modelo de datos relacional para almacenar los datos. En este modelo, los datos se almacenan en filas y columnas en forma tabular. Las

tablas relacionadas se pueden interrelacionar entre sí. Las bases de datos relacionales son capaces de manejar un enorme volumen de datos, pero tienen grandes inconvenientes, por ejemplo, la escalabilidad es mucho más complicada que la simple adición de nuevo *hardware* y el costo asociado a la escalabilidad es alto, por esto las bases de datos relacionales han mostrado su debilidad en el manejo de grandes cantidades de datos, especialmente en la obtención de escalabilidad horizontal. Para hacer frente a estas limitaciones, las bases de datos NoSQL se han empezado a utilizar cada vez más con el propósito de manejar grandes cantidades de datos, ya que pueden proporcionar escalamiento automático, mejor rendimiento y alta disponibilidad (Amghar, Cherdal, & Mouline, 2018). Las bases de datos NoSQL siguen un modelo de datos no relacionales. El modelo no relacional admite el almacenamiento de datos sin esquema en diversas formas, como documentos y grafos. Con características como la escalabilidad horizontal, el almacenamiento sin esquema, el apoyo a los datos no estructurados, NoSQL se vuelve competente para el almacenamiento de datos de sistemas IoT.

El sistema que maneja los datos, las transacciones, los problemas o cualquier otro aspecto de la base de datos es el Sistema de Gestión de Bases de Datos (DBMS). Actualmente hay un gran número de bases de datos NoSQL, con sus respectivos DBMS, que se agrupan en cuatro categorías:

- Basadas en clave-valor: Los datos se almacenan como pares de clave-valor y se manejan mediante claves que son únicas. Los datos constan de dos partes, una clave de texto, y un valor que es el dato real, y puede ser un tipo de datos numéricos, texto, un objeto o un documento. El mecanismo de la base de datos clave-valor es similar al de un diccionario o un mapa. Al ser muy sencillas, las bases de datos de valores clave son consideradas como las más rápidas. Ejemplos en esta categoría son Redis, Memcached, Berkeley DB y Amazon DynamoDB.
- Orientadas a la columna: En esta categoría los datos se almacenan como filas y columnas, en la que cada clave está asociada a una o más columnas. Aunque las columnas similares se almacenan juntas en una familia de columnas, y las filas

pueden contener columnas diferentes. Existen varias DBMSs orientadas a las columnas, entre ellas, Cassandra, Hbase y Bigtable.

- Orientadas a los documentos: Los datos se almacenan como colecciones de documentos, estos documentos se guardan en forma serializada como XML (*Extensible Markup Language*), JSON (*JavaScript Object Notation*), y BSON (*binary encoded* JSON). Las estructuras de los documentos no tienen que ser similares, esto significa que cada documento puede contener un conjunto diferente de datos. DBMSs orientadas a documentos incluyen a MongoDB, Couchbase y Ravendb.
- Estructurada en grafos: Los datos se almacenan en forma de grafos, cada grafo consiste en un conjunto de nodos y enlaces, donde los nodos actúan como los objetos y los enlaces actúan como la relación entre los objetos. Estas bases de datos se utilizan para almacenar datos relacionados entre ellos, como datos bioinformáticos o datos de redes sociales. Por sus características, las bases de datos estructuradas en grafos son muy difíciles de fragmentar. Los ejemplos de DBMSs estructuradas en grafos incluyen a Neo4j y GrapDB.

A continuación se describirán dos de las DBMSs más populares, una perteneciente a la categoría de bases de datos clave-calor y la otra a bases de datos orientada a los documentos, específicamente escogidas por su rapidez y flexibilidad en cuanto a la manipulación de los datos que un PLC orientado a la Industria 4.0 podría necesitar en su funcionamiento. Otras dos DMBS, del tipo estructurada en grafos y orientada a las columnas, se tratan en el Anexo D.

4.4.1. Redis

Redis es una de las bases de datos clave-calor más populares, tiene una estructura de almacenamiento de datos en memoria, lo que significa que cuando se ejecuta Redis los datos se cargan por completo en la memoria, por lo que todas las operaciones se ejecutan en ella y periódicamente Redis guarda los datos en el disco duro. Los datos se almacenan

como pares de valor clave-valor. Los valores en Redis no se limitan a una simple cadena, sino que también pueden contener estructuras de datos más complejas como listas, conjuntos, matrices de bits y otros.

Entre las principales características de Redis están la escalabilidad a través de la distribución de los valores clave en los nodos, la disponibilidad de una Interfaz de Programación de Aplicaciones (API) para soportar el procesamiento en tiempo real. Además, para asegurar la seguridad, Redis proporciona seguridad de red y características de autenticación, y también puede deshabilitar algunos comandos.

4.4.2. MongoDB

MongoDB es una base de datos distribuida de código abierto orientada a los documentos, creada en 2007 en la empresa de publicidad en *Internet DoubleClick* (ahora propiedad de Google). MongoDB almacena datos en formato BSON (*binary json*) y cada documento BSON contiene pares clave-valor. Los datos pueden ser replicados usando el modelo maestro-esclavo. Aunque MongoDB soporta la transferencia automática de datos a través de múltiples nodos. MongoDB tiene su propio lenguaje de consulta que implementa todas las operaciones GRUD (Crear , Leer, Actualizar, Eliminar).

MongoDB utiliza la escalabilidad horizontal añadiendo más servidores para aumentar la capacidad de almacenamiento del sistema, soporta el procesamiento de datos en tiempo real, proporciona muchas características para asegurar la base de datos, incluyendo la autenticación y el cifrado. MongoDB proporciona operaciones de agregación que agrupan valores de múltiples documentos y realizan diversas operaciones sobre los datos agrupados para obtener un único resultado.

5. EL IOT-PLC

En las secciones anteriores se han descrito a las arquitecturas orientadas a la Industria 4.0 y nuevas tecnologías que podrían formar parte de ellas, como consecuencia de la adopción de ellas se podría dejar atrás a la pirámide propuesta por el estándar ISA-95, ya que la pirámide de automatización tradicional y jerárquica se suele considerar una estructura física de los sistemas de producción. Mientras que mediante la introducción de nuevas tecnologías de comunicación, específicamente los protocolos IT y los desarrollados en el contexto IoT, esta estructura (física) puede ser obviada, por ejemplo, por sensores conectados directamente a una nube. De esta manera, los componentes de automatización se considerarán como CPSs, en su mayoría interconectados entre ellos (Jasperneite, Sauter, & Wollschlaeger, 2020). Como resultado de esto, el PLC, en su concepción actual, tiene dificultades en su inclusión directa en estas arquitecturas y la inclusión de nuevas tecnologías resulta engorrosa de realizar.

Por otro lado, impulsados por la idea de la Industria 4.0, los servicios y modelos de información relacionados están definidos para ser utilizados en diferentes dominios y contextos, según sea la aplicación. Además, los dispositivos industriales serán computacionalmente más rápidos y completos, por lo que elementos como *gateways* podrían ser algo más que simples unidades de protocolo y conversión de datos, tendrán que actuar como entidades inteligentes que controlen y representen el (sub)sistema de automatización subyacente. Por último, pero no por ello menos importante, también pueden asegurar el acceso a partes del sistema (Wollschlaeger et al., 2017).

En este capítulo se presentará la contribución de esta tesis, un PLC de propósito general diseñado para operar en los paradigmas de la Industria 4.0 como un dispositivo ubicado cerca del borde de la red que apunta a transformar el actual PLC para que tenga cabida en las nuevas arquitecturas de los sistemas industriales y sus funcionalidades sean una opción a los temas tratados anteriormente. Se explicará cómo este PLC está estructurado, cómo se comunica mediante distintas interfaces, la forma en que permite que los dispositivos de

campo con que se comunica puedan ser vistos como CPSs desde capas superiores y otorga una capa de abstracción para el despliegue de tareas de control y manejo de datos que no dependen de los diferentes protocolos de comunicación utilizados por los dispositivos.

5.1. Diseño del modelo desarrollado

El IoT-PLC es un PLC para la Industria 4.0. El esquema del PLC que se propone, y se explica en detalle en las secciones siguientes, cumple con las funcionalidades requeridas para operar dentro del paradigma de la Industria 4.0, es decir, cumple con las restricciones típicas del mundo OT, a la vez que incluye la conectividad con la nube moderna y las herramientas de administración típicas del mundo IT.

El IoT-PLC fue diseñado para interactuar con dispositivos que se comunican a través de diferentes protocolos, para extraer la información pertinente para una mayor consolidación de la nube y para dar un formato de datos estándar para la integración transparente a la nube, además de realizar las tareas de control que al PLC actual se le asigna.

Es por esto que el IoT-PLC se concibe como un nodo en la capa *fog* de un ICPS, de esta manera actúa como i) un *gateway* para dispositivos de campo que se comunican utilizando una variedad de protocolos orientados a sensores y actuadores, particularmente inalámbricos, tal como se muestra en la Figura 5.1 en las dos capas inferiores; y ii) un controlador de primer nivel, manipulando los actuadores en base a la información que adquiere de los sensores y comandos que recibe de la nube, haciendo uso de las dos capas superiores de la Figura 5.1.

Por otro lado, el retardo propio de las comunicaciones con la nube no afectan al rendimiento del proceso de control, ya que los algoritmos de control funcionan de forma autosuficiente en el IoT-PLC.

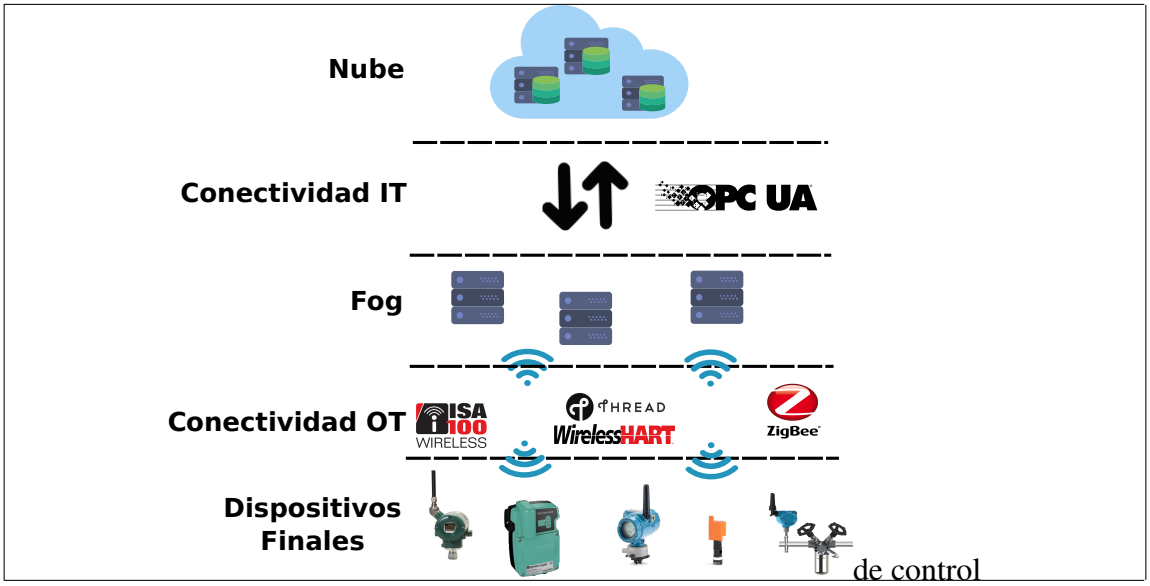


Figura 5.1. Arquitectura en la que se encuentra el IoT-PLC.

El diseño del IoT-PLC cuenta con las siguientes funcionalidades y características principales:

- Las aplicaciones operan dentro de sistemas operativos virtualizados (contenedores).
- Incluye módulos de filtrado de datos, base de datos y HMI.
- Incluye un módulo de conectividad CoAP para la comunicación con otros dispositivos de capa *fog*.
- Incluye un módulo de conectividad OPC UA para la comunicación con la nube.
- Incluye un módulo de conectividad para las interfaces inalámbricas que se comunican con los dispositivos de campo.
- Múltiples controladores pueden ser desplegados en un mismo dispositivo, estos pueden ser desarrollados en el lenguaje de programación que se prefiera.
- Asignación automatizada de dispositivos de campo con controladores recibidos desde la nube, junto con el posterior inicio del lazo de control asociado a ellos.

- Se representan a los dispositivos de campo y controladores como dispositivos virtuales y se permite exposición de estos como CPSs a la nube a través de OPC UA.
- Se implementa un sistema de arranque para las aplicaciones y un sistema de recuperación en caso de fallas en las aplicaciones.
- Migración en vivo de los contenedores que representan a los procesos desde un IoT-PLC a otro para continuar el proceso de control en el último.

5.2. Arquitectura diseñada

El IoT-PLC está estructurado internamente de acuerdo a la Figura 5.2, donde cada bloque representa un *script* o proceso que se ejecuta dentro de un contenedor que funciona de forma independiente e intercambia información con otros bloques. Además, la formulación en bloques de la arquitectura interna del IoT-PLC apunta a que este opere como dispositivo de propósito general, de esta manera los bloques pueden realizar las tareas asignadas en la forma que se estime conveniente según las necesidades que se tengan en el sistema en el que se encuentre funcionando el IoT-PLC. La arquitectura implementa una variedad de interfaces inalámbricas para la conectividad a nivel de campo con sensores y actuadores. Además, incluye, internamente, un módulo de filtrado para el preprocesamiento de los flujos de datos y una base de datos local ligera para el análisis de tendencias a corto plazo y la visualización a través de su interfaz hombre-máquina (HMI). También, incluye un módulo de administración que se encarga de generar lazos de control utilizando sensores y actuadores de campo, de instalar controladores en forma de contenedores para el control regulatorio de bajo nivel y de orquestar el proceso de migración de los procesos hacia otro IoT-PLC.

Los bloques de interfase, en la parte inferior de la Figura 5.2, proporcionan conectividad con los dispositivos de campo y permiten elevar el nivel de los datos para que los bloques superiores puedan trabajar de forma transparente, independientemente de la naturaleza particular de los dispositivos de campo. Los bloques superiores de la Figura 5.2

trabajan con datos abstractos y ofrecen funcionalidades relacionadas con el algoritmo de control y con la conectividad con usuarios humanos, otros IoT-PLCs y la nube.

Para la conectividad de alto nivel, contiene dos interfaces, a saber, una interfaz OPC UA para la conectividad con la nube y una interfaz CoAP para la comunicación con otros dispositivos IoT-PLC desplegados en la red de dispositivos de campo y la capa *fog*, esta interfaz es útil cuando un determinado IoT-PLC necesita formar un lazo cerrado con un sensor o actuador inalámbrico fuera de su rango inalámbrico y utiliza un segundo IoT-PLC como nodo de relé. Esta solución de conectividad horizontal reduce la latencia con respecto a una solución de integración vertical en la que la nube actúa como puente y la estabilidad de las conexiones no está asegurada.

La característica de incluir los procesos en contenedores permite la asignación de recursos de los dispositivos a cada uno de ellos en función de la prioridad que tengan en el sistema de control. Además, para trasladar el sistema de control a otro IoT-PLC se incluye una funcionalidad de migración en vivo con mantención de estado de memoria (*stateful live migration*). Una migración en vivo es el proceso en el que el estado en memoria de un sistema operativo huésped en funcionamiento se registra, transfiere y luego se vuelve a iniciar en el lugar de destino. El estado en memoria incluye a las aplicaciones, los procesos del sistema y los recursos actualmente cargados en la memoria para un acceso rápido, este contiene el progreso (estado) de las aplicaciones en ejecución, incluyendo cualquier dato en el que la aplicación esté trabajando actualmente (Machen, Wang, Leung, Ko, & Salonidis, 2018). Con esto no es necesario detener un sistema de control si el PLC tiene que ser apagado, porque la migración hace posible restaurar el progreso del contenedor en otro PLC exactamente donde estaba el PLC original. Por otro lado, al ejecutar cada bloque en un contenedor es posible manejarlos desde la nube o migrarlos para continuar el proceso de control en otro dispositivo. Esto le da al IoT-PLC la flexibilidad de cambiar el tipo de controlador en diferentes momentos sin tener que detenerlo completamente, sino sólo el bloque controlador.

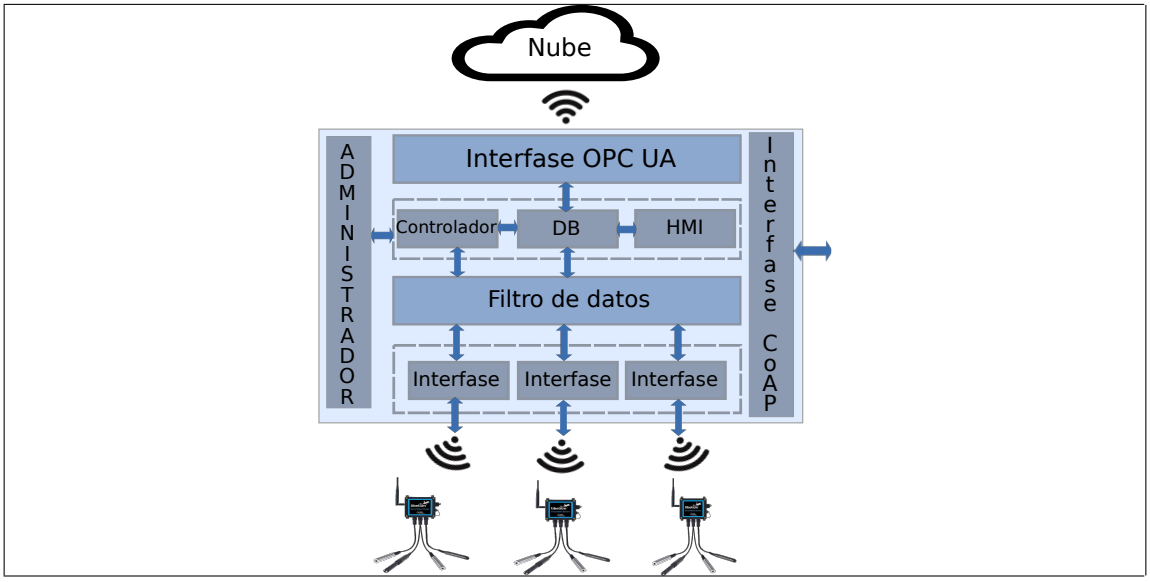


Figura 5.2. Estructura de bloques funcionales diseñada para el IoT-PLC.

5.3. Dispositivos y controladores virtuales

En el IoT-PLC se representan a los dispositivos y controladores como dispositivos virtuales (un objeto de programación). Estos dispositivos virtuales están pensados para ser utilizados como un método de abstracción para trabajar con entidades reales, de modo que las aplicaciones del IoT-PLC puedan interactuar con los datos desde una perspectiva de alto nivel, sin la necesidad de interactuar con los dispositivos físicos.

Los dispositivos virtuales se basan en el modelo de objetos de recursos presentado en (Lan, Shi, Wang, & Zhang, 2019), en el que el modelo pretende reflejar la integración del mundo físico y el espacio de información para los objetos IoT. El modelo de recursos utilizado para el IoT-PLC se muestra en la Figura 5.3. En este modelo cada IoT-PLC tiene un objeto de tipo carpeta en el servidor OPC UA con sus controladores asignados, dentro de esa carpeta cada controlador incluye las propiedades y variables necesarias y los dispositivos virtuales que representan a los sensores y actuadores (dispositivos de campo).

Para los dispositivos de campo hay dos tipos de descripciones, una es usada internamente por el IoT-PLC y la otra es usada por la nube a través de OPC UA. El primer tipo es el recurso del dispositivo con tres cuadrados cian, cuyo significado es el siguiente: Los "atributos" son un valor que normalmente no cambia, como el fabricante, número de serie y las especificaciones del dispositivo, el recurso estado "estado" se refiere a los valores de algunos parámetros en un momento determinado, tales como, ubicación, veces que ha sido utilizado y estados de sus componentes (por ejemplo, porcentaje apertura de una válvula o si un sensor está midiendo) y el "control" se refiere a la información de la interfaz de control proporcionada por el dispositivo, tal como, tipo de controlador, parámetros y las interfaces utilizadas para los lazos de control. El segundo tipo de descripción incluye a los tres cuadrados cian y añade el cuadrado verde, donde "historia" se refiere a la operación o datos cargados por el dispositivo en el pasado.

Para guardar y gestionar las configuraciones y propiedades de los dispositivos virtuales se utiliza el formato *json*, que da flexibilidad a los usuarios y desarrolladores para almacenar tipos de datos complejos sin enfrentarse a las restricciones de las bases de datos relacionales, a la vez que es un formato que al ser utilizado no sobrecarga a los recursos computacionales del IoT-PLC.

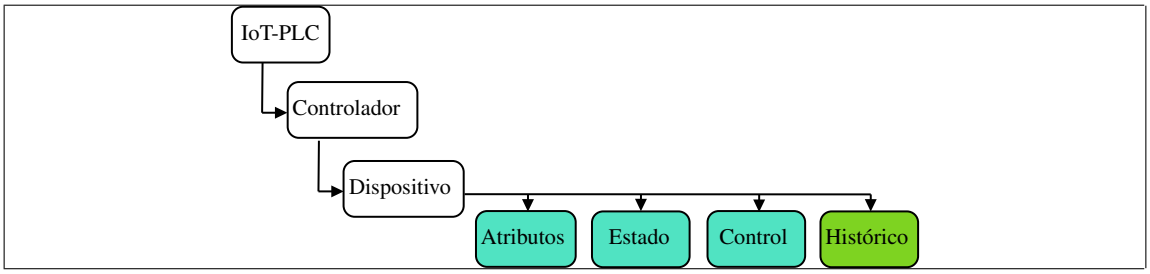


Figura 5.3. Modelo de objetos para los dispositivos virtuales.

5.4. Arranque y sistema de recuperación

El proceso de arranque del IoT-PLC comienza con la nube o un operador local dando la señal a un programa (*script*) en un dispositivo para iniciar los contenedores que incluyen a

los procesos de cada bloque del IoT-PLC, en caso de que no los tengan, pueden ser creados localmente, ser obtenidos de un IoT-PLC cercano o directamente de la nube. El orden de inicio de los contenedores sigue el que se muestra en la Figura 5.4, luego se llama periódicamente al mismo programa para comprobar si cada bloque está funcionando, en caso de que un bloque se haya detenido se reinicia inmediatamente para mantener la operación lo más limpia posible.

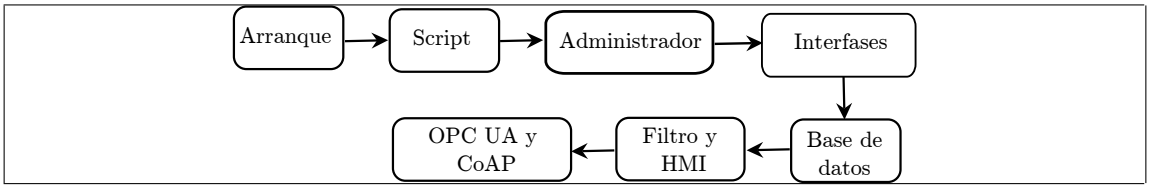


Figura 5.4. Orden en que se inicial los bloques funcionales.

5.5. Interfaces

La arquitectura del IoT-PLC está compuesta por tres tipos de interfaces: la relacionada con los dispositivos de campo, la interfaz CoAP que permite la conectividad con otros dispositivos IoT-PLCs y la interfaz OPC UA para dar conectividad hacia la nube. A continuación se explicará en mayor detalle su funcionamiento.

5.5.1. Interfaces con dispositivos de campo

El IoT-PLC basa su conectividad con los dispositivos de campo en bloques de interfaz bidireccionales que traducen los datos de los bloques de IoT-PLC de alto nivel a los dispositivos de campo y viceversa, cada bloque de interfaz es capaz de comunicarse a través de un protocolo, por ejemplo, Zigbee, Bluetooth o Wi-Fi. Los bloques de interfaz se comunican con los bloques superiores, los bloques Administrador y Filtro, mediante mensajes de eventos de publicación/suscripción. De este modo, los bloques superiores pueden obtener datos siempre que lo necesiten, sin depender directamente de la velocidad de transmisión de los dispositivos de campo o de los demás bloques que componen al IoT-PLC.

Los bloques de interfaz están diseñados de tal manera que cualquier bloque superior puede suscribirse a las interfaces, de modo que los datos pueden transferirse hacia o desde nuevos bloques funcionales simplemente suscribiéndose a las interfaces. Por lo tanto, los bloques de interfaz proporcionan un mecanismo normalizado para interactuar con los dispositivos de campo, independientemente del protocolo de comunicación y otras particularidades tecnológicas de estos últimos.

5.5.2. CoAP

El bloque de interfaz CoAP está destinado a proporcionar conectividad con otros IoT-PLC o dispositivos de capa *fog*. Usando esta interfaz, es posible intercambiar datos de dispositivos y controladores, y dar comandos de control a otros IoT-PLCs, como una señal de terminación del lazo de control. Esto se logra mediante el uso de la característica de descubrimiento *Multicast* del servidor CoAP, lo que da como resultado un proceso de descubrimiento descentralizado entre los dispositivos de una red.

Con esta interfaz, también, es posible formar un lazo de control con más de un IoT-PLC, en el caso que los dispositivos de campo estén fuera del alcance de alguno de ellos, un segundo IoT-PLC actuaría como nodo de relé para uno o varios dispositivos.

El protocolo CoAP, una solución madura orientada al área IoT, fue elegido por sus propiedades probadas en los dispositivos *gateway*, tales como baja sobrecarga, latencia reducida e intercambio descentralizado de datos (Bellavista & Zanni, 2016). Estas características aseguran que la comunicación con otros IoT-PLC no reducirá el rendimiento de los procesos de control local.

5.5.3. OPC UA

A través de este protocolo los datos recogidos se envían a los servidores en la nube. Además, desde la nube se pueden enviar descripciones de dispositivos, controladores y lazos para ser añadidos o modificados durante el proceso de control, también, la interacción entre la nube y el IoT-PLC permite solicitar información desde ambas direcciones.

A través de este bloque OPC-UA también es posible enviar los comandos necesarios desde la nube a un IoT-PLC para proceder a la migración de los contenedores a otro dispositivo.

Mediante el uso de OPC UA, es posible mapear las funciones expuestas por el IoT-PLC a los estándares ISA-95, IEC 61131 y AutomationML, descritos en la Sección 2.2. Por ejemplo, los controladores, los dispositivos asignados, sus configuraciones y las formas de acceder a sus entradas y salidas pueden implementarse con el estándar IEC 61131 para OPC UA. Mientras que las descripciones de los dispositivos y el propio IoT-PLC, junto con la información relativa al personal implicado y a los tipos de procesos con los que interactúan, pueden insertarse en un sistema compatible con la norma ISA-95 para intercambiar información con los sistemas MOM. Por último, el IoT-PLC y su descripción pueden incluirse en un modelo AutomationML de una planta y un sistema de comunicaciones, además, de incluir una descripción de los sistemas de control implementados.

5.6. Administrador

El bloque Administrador se encarga de tomar datos de las descripciones de los dispositivos y controladores de la base de datos para instanciar los dispositivos y controladores virtuales, que tienen la capacidad de agruparse para formar un lazo de control. Los dispositivos virtuales, el controlador y los lazos de control se representan y utilizan como objeto de programación en el bloque Administrador. Cuando se crea un lazo de control, el Administrador inicia un proceso dentro de un contenedor dedicado a la tarea de control, al que se transfieren los dispositivos virtuales y el lazo de control, de modo que el objeto que

representa al lazo de control pueda funcionar y enviar los datos generados al bloque de la base de datos independientemente del gestor. De esta manera, el gestor puede modificar el lazo de control en cualquier momento si fuese necesario cambiar uno o más parámetros, o puede detener el proceso de control, sin interferir con otros lazos de control o bloques del IoT-PLC.

5.7. Filtro de datos

El bloque de filtro de datos funciona como una capa de abstracción entre los bloques de interfaz y los elementos superiores. El bloque de filtro de datos recoge la información de las interfaces y la preprocesa para su posterior uso por los bloques superiores. El flujo de datos entrante de los bloques de interfaz se filtra para reducir el ruido y enmascarar los valores atípicos mediante el uso de filtros. Además, los filtros aplicados en este bloque se instancian de acuerdo con la configuración enviada desde la nube, por lo que se pueden aplicar diferentes filtros según las necesidades de los sensores y procesos. Los datos validados se publican luego hacia los bloques superiores que requieran los datos.

Para llevar a cabo esto el bloque tiene tareas adicionales al filtrado que son necesarias para el buen funcionamiento del bloque y del IoT-PLC en general, tales como, la asignación interna de las interfaces que se comunican con los dispositivos de campo con los bloques Controlador y Base de datos, mapeo de los datos filtrados en el bloque con los datos que entran al bloque y manejo de metadatos que contengan el tiempo en que se reciben los datos, que filtro debe aplicárseles y el origen y destino que tienen dentro del IoT-PLC.

5.8. Base de datos

El bloque de la base de datos actúa como una unidad de consolidación. La información se transforma para hacerla más significativa cuando se envía a la nube o se toma de la HMI. Los datos consolidados en este bloque corresponden a la información filtrada de los

sensores, los datos generados por el algoritmo de control dentro del IoT-PLC, y los datos enviados desde el bloque HMI. En particular, el bloque de la base de datos se encarga de enviar datos al bloque OPC UA para ser transmitidos a la nube. También recibe del bloque OPC UA las descripciones y configuraciones de los dispositivos a los que se conectará el IoT-PLC y las guarda. El bloque de la base de datos proporciona una capa de adaptación que hace que los datos sean independientes de las escalas de tiempo del dispositivo de campo.

5.9. Interfaz hombre-máquina

Se añade una interfaz hombre-máquina (HMI) al IoT-PLC para dar la opción a un usuario local o externo del sistema de configurar todos los parámetros que intervienen en el algoritmo de control, y poder supervisar el estado de las variables controladas y manipuladas desde cualquier lugar, siempre que se dé la autorización correspondiente. La HMI sólo tiene un flujo de datos con el bloque de la base de datos, de modo que en caso de modificaciones de ese bloque el Administrador y los procesos de control pueden interactuar este último. Cuando la HMI requiere datos, emite una solicitud al bloque de la base de datos, luego la base de datos recupera la información, que finalmente es presentada al usuario por la HMI.

5.10. Flujo de operación

El funcionamiento del IoT-PLC, luego de iniciar todos los bloques funcionales que lo componen, sigue el flujo de secuencia que se muestra en la Figura 5.5. El proceso de funcionamiento comienza cuando la nube envía la configuración de los dispositivos de campo a ser utilizados. Entonces, el IoT-PLC comenzará a buscar los dispositivos, si están disponibles recibe el indicador de disponibilidad y los datos extra del dispositivo. Luego, se procede a instanciar un dispositivo virtual en el IoT-PLC y se establece una conexión con el dispositivo físico, para su posterior uso en un lazo de control.

La siguiente parte del proceso comienza cuando, desde la nube o localmente, se añaden los parámetros de configuración de un controlador y opcionalmente la nube se suscribe a eventos relacionados con el funcionamiento del lazo de control, como mediciones nuevas o una variable supera un cierto valor. Posteriormente, el IoT-PLC instancia un controlador virtual con los parámetros obtenidos y, si los sensores y actuadores necesarios están presentes, se instancia un lazo de control virtual y se envía a un contenedor para que empiece el proceso de control.

Entonces, se inicia el proceso de muestreo de los sensores y, con esto, el controlador comienza a enviar señales de control a los actuadores. Cuando los sensores envían las mediciones, también transmiten su estado, para informar constantemente de su funcionamiento. Para el caso de los actuadores, estos transmiten su estado después de recibir la señal de control. Luego, los datos del proceso se envían a la base de datos y a la nube, según las suscripciones correspondientes.

Otro caso del proceso de funcionamiento es cuando la configuración del dispositivo ya está almacenada en el IoT-PLC, pero el dispositivo no está conectado, el IoT-PLC esperará hasta que el dispositivo de campo esté disponible, luego procederá a iniciar la creación de su dispositivo virtual.

La forma en que un dispositivo comunica que está disponible es a través de un mensaje de difusión (*broadcast*) que indica su disponibilidad. Si el IoT-PLC tiene su configuración, indica que se le ha asignado el estado de disponible. En caso contrario, cuando el IoT-PLC responde que no tiene una configuración para el dispositivo, éste envía la configuración necesaria para formar un dispositivo virtual. Con esa configuración, el bloque Administrador procede a guardar la información enviada por el dispositivo en la base de datos y desde allí se envía al bloque OPC UA para guardar su modelo en la nube. Después de esto, el dispositivo es informado de que está disponible para ser utilizado y ser asignado a un proceso de control.

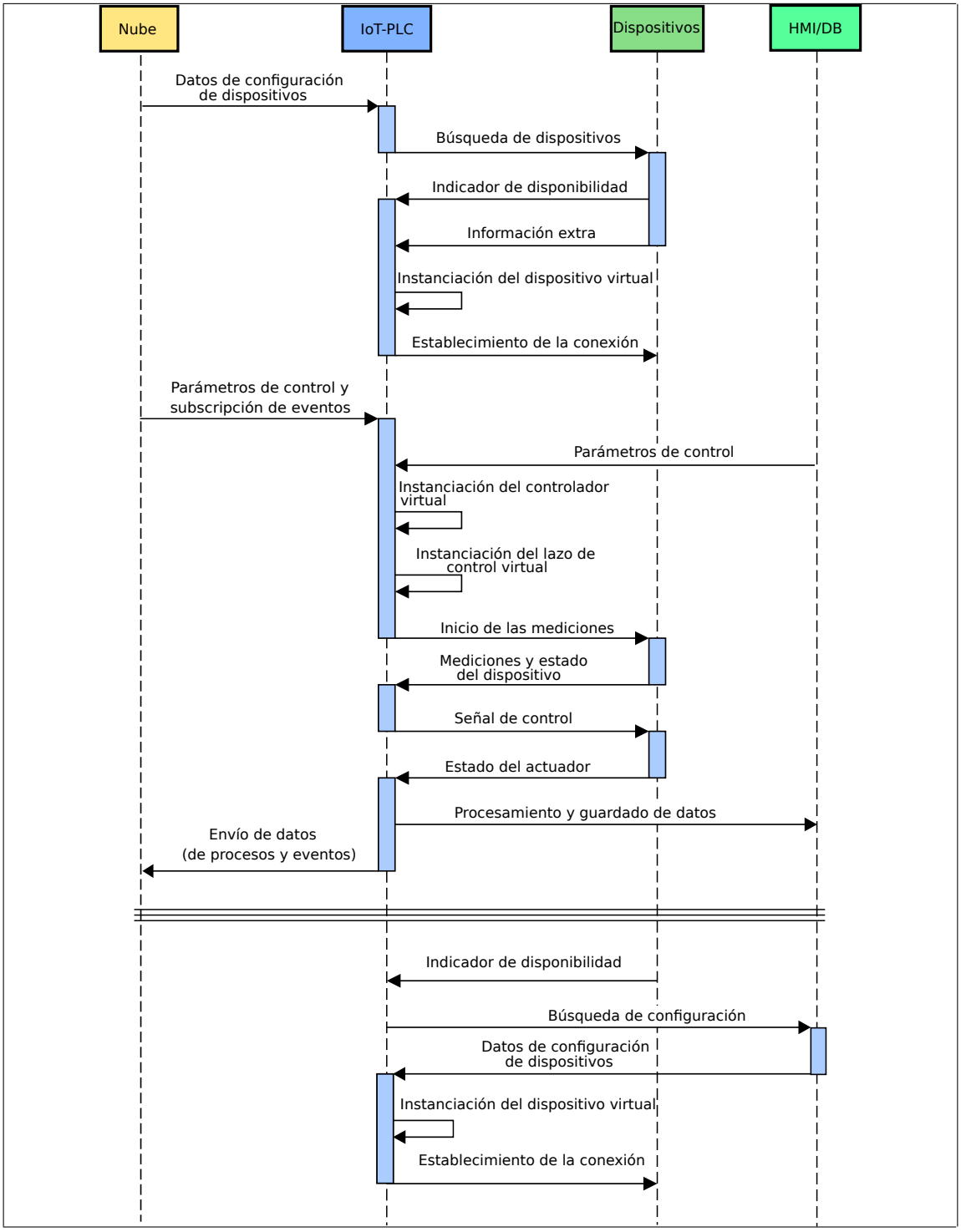


Figura 5.5. Flujo de operación que sigue el IoT-PLC.

5.11. Despliegue y migración de contenedores

El proceso de carga de los contenedores hacia un IoT-PLC tiene dos variantes: i) despliegue de contenedores para empezar el uso de contenedores sin información pasada y ii) la migración de contenedores desde un IoT-PLC hacia otro con los datos que estaban en la memoria de los contenedores.

5.11.1. Despliegue de contenedores

El despliegue de contenedores empieza cuando desde la nube o un operador local inicia un IoT-PLC, como se mencionó anterioremente estos contenedores pueden ser creados localmente, ser obtenidos de un IoT-PLC cercano o directamente de la nube. Los métodos para obtener los contenedores pueden utilizarse en conjunto, ya que, las capas que dan origen a las imágenes necesarias para iniciar cada contenedor pueden ser adquiridas desde cualquier punto.

Para formar las imágenes, que necesitan los contenedores que conforman a los bloques funcionales del IoT-PLC, se optó por la creación de una gran capa base que tuviera incluida todos los componentes mínimos necesarios comunes a todos los contenedores y por pequeñas capas que incluyen el programa que representa a un bloque funcional del IoT-PLC o las librerías que solo un contenedor necesita. Así, se evita la redundancia en la incorporación de librerías en distintos contenedores, por lo que se ahorra espacio en las unidades de almacenamiento y se disminuye el uso de la red cuando se transfieren las imágenes de los contenedores. En la Figura 5.6 se muestra el orden en que se obtienen las capas de cada contenedor, este orden empieza por la capa más grande y sigue por las capas más pequeñas.

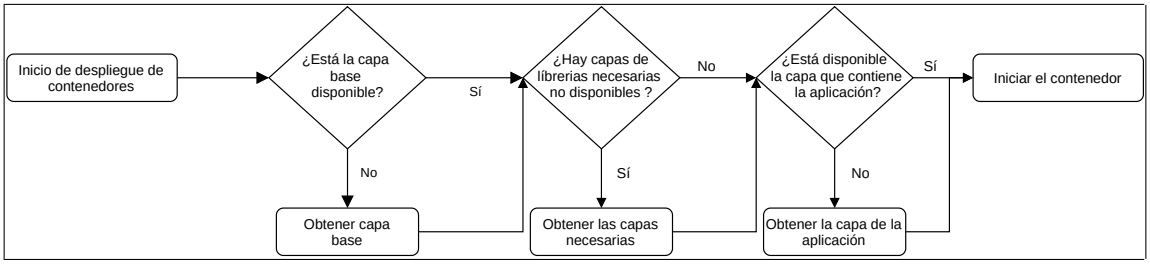


Figura 5.6. Orden de despliegue de las capas de cada contenedor.

A modo de ejemplo, en la imagen 5.7 se muestra una capa base para todos los contenedores (cuadro en color verde), una capa para cada una de las aplicaciones (cuadros en color amarillo) y una capa para las librerías que solo necesita el controlador (cuadro en color rosa). En caso del despliegue de los contenedores, la capa base podría ser obtenida desde un IoT-PLC cercano, la capa de las librerías del controlador podría crearse en el IoT-PLC y las capas de las aplicaciones descargarse desde la nube.

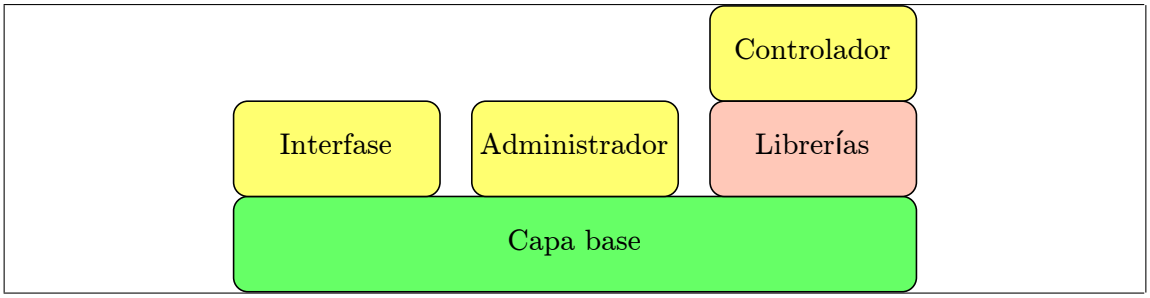


Figura 5.7. Ejemplificación del uso de capas para formar las imágenes de los contenedores.

5.11.2. Migración de contenedores

El diseño del IoT-PLC permite que los contenedores que lo componen puedan migrar entre dos de ellos utilizando un procedimiento de migración en vivo con mantención de estado de memoria. Para ello, se asigna a la nube como la encargada de dirigir el proceso de migración. Este proceso de migración requiere de dos dispositivos: uno con los contenedores que dirigen el proceso de control con sus dispositivos asignados y otro sin

los contenedores con los recursos necesarios para poder comunicarse con los dispositivos utilizados por el primero. En la Figura 5.8 Se muestra el estado inicial de este proceso.

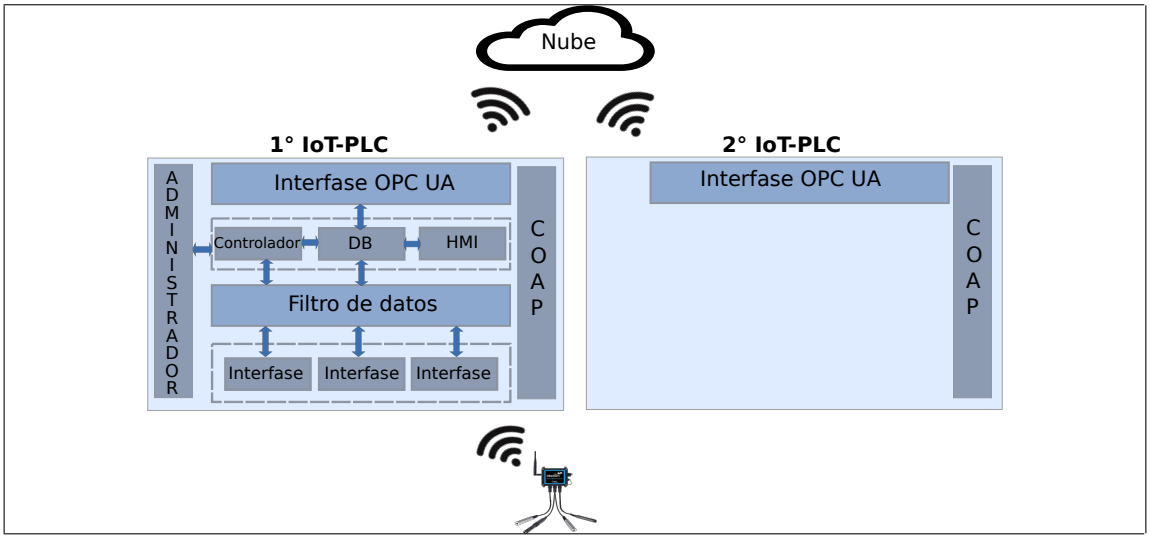


Figura 5.8. Proceso de migración de contenedores: Estado inicial.

El flujo de operaciones de la migración comienza enviando instrucciones desde la nube al primer dispositivo IoT-PLC a través de OPC UA. Estas instrucciones indican el comienzo del proceso y el primer IoT-PLC se encarga de verificar mediante el uso de la característica de descubrimiento de CoAP que el segundo IoT-PLC está disponible y de generar un repositorio con los contenedores originales, sin el estado en memoria de los contenedores. Luego, la nube le dice al segundo IoT-PLC que obtenga los contenedores del primero, cuando esta tarea se completa con éxito se informa a la nube. En la Figura 5.9 se muestra el estado de esta etapa intermedia.

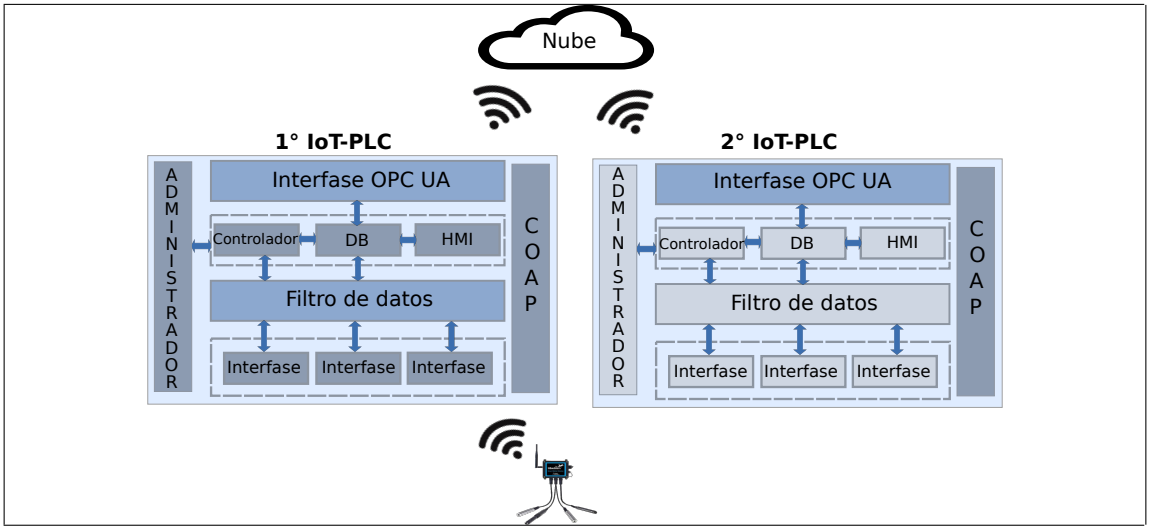


Figura 5.9. Proceso de migración de contenedores: Estado intermedio, contenedores en el segundo IoT-PLC sin iniciar.

A continuación, desde la nube, para guardar el progreso de los contenedores desde que se iniciaron, se inicia el proceso de guardado estado en memoria, esto se llama punto de control de contenedores. Luego, estos estados se transfieren al segundo IoT-PLC y los contenedores se inician inmediatamente. Durante el punto de control, cada contenedor se pausa para obtener el estado en memoria y después de eso se despausa para mantener el sistema en funcionamiento el mayor tiempo posible. Durante esta pausa el programa que obtiene el estado en memoria realiza los siguientes pasos: empieza por congelar el proceso y sus subprocesos, luego toma la información de la memoria, los descriptores de archivos y párametros del proceso en el sistema y la copia hacia la unidad de almacenamiento. Por último, elimina todos los cambios en el sistema que se tuvieron que realizar para obtener el estado de memoria y reinicia el proceso en su estado original (previo a la pausa). Por lo tanto, el tiempo de esta pausa depende de la cantidad de archivos que cada contenedor manipule y la cantidad de memoria que utilice del sistema.

Finalmente, la nube le indica al primer IoT-PLC que se comunique con sus dispositivos asignados para cambiar el destino de los datos enviados hacia las interfaces del segundo

IoT-PLC, luego los contenedores se detienen. En este momento, el proceso de control continúa en el segundo IoT-PLC. El estado final del proceso se muestra en la Figura 5.10.

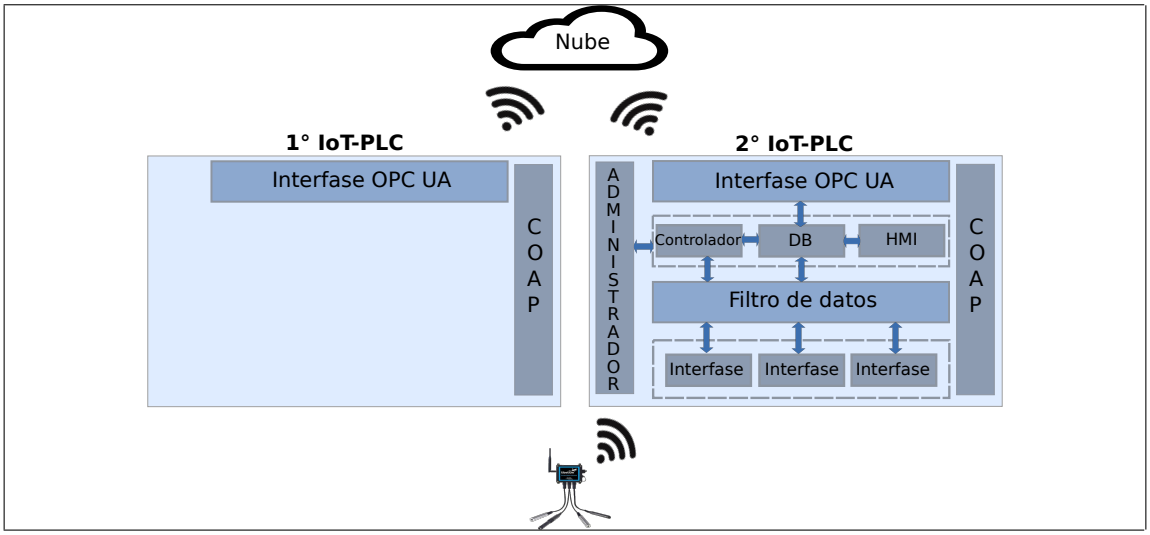


Figura 5.10. Proceso de migración de contenedores: Estado final.

Para interactuar con el repositorio generado se utiliza una API HTTP, lo que permite utilizar los métodos de solicitud GET, HEAD, POST, PUT y DELETE para obtener los contenedores necesarios. El repositorio de contenedores fue diseñado para ser creado en el IoT-PLC, porque de esta manera no depende de la calidad de la conexión entre la nube y el IoT-PLC para la transmisión de los contenedores y la transmisión entre los IoT-PLC permite un uso más eficiente de la red local, aunque la opción de utilizar a la nube como repositorio es posible si las circunstancias la hacen necesaria.

5.12. Comparación con desarrollos existentes en la literatura

En la literatura reciente, existen desarrollos similares al IoT-PLC. Un soft-PLC dispuesto en la nube fue diseñado en (Goldschmidt et al., 2015), buscando modernizar al PLC y aumentar su escalabilidad mientras se hace la operación rentable, aquí los datos del sensor son enviados a la nube usando OPC UA y el resto del PLC está corriendo completamente en la nube como una pieza de *software*, este controlador incluye base de datos,

caché y la posibilidad de que un ingeniero de control modifique los parámetros del sistema. Las ventajas de esta estructura sobre el actual PLC se basan en su escalabilidad horizontal y en la diversificación de las tareas que realiza el controlador, pero el principal problema es la latencia entre los dispositivos de campo y la nube. En comparación con este desarrollo, IoT-PLC presenta ventajas en términos de latencia, ya que opera como nodo de capa *fog* y no en la nube. Además, el IoT-PLC puede operar en la red local sin depender de la conectividad hacia la nube, por lo que en caso de falla en la conectividad hacia el exterior podría seguir operando. Por otro lado, el soft-PLC en la nube no fue diseñado para migrar el *software* del PLC hacia otro soft-PLC, mientras que el IoT-PLC está estructurado para permitir tal migración.

La arquitectura propuesta por (Popović & Rakić, 2018) da un paso adelante en esta dirección al construir una infraestructura para la integración de sistemas de control IoT basados en el paradigma *fog*. La estrategia de transmisión de datos está diseñada para aplicaciones de control en tiempo real y añade la comunicación *fog-fog* sobre un modelo de transmisión simple, donde los nodos de capa *fog* juegan el papel de un PLC. El sistema incluye aplicaciones que se ejecutan dentro de los nodos y está adaptado para un procesamiento en tiempo real dentro de un *middleware* distribuido orientado al manejo de mensajes en forma de componentes de agente de servicio configurables. Al igual que el IoT-PLC, esta concepción de PLC da la posibilidad de interactuar con diferentes dispositivos de campo mediante diferentes interfaces, pero, a diferencia del IoT-PLC, no define una forma de describir a los dispositivos para interactuar con ellos en capas superiores (como lo hacen dispositivos virtuales), lo que impide la integración con plataformas de nubes sin el conocimiento de cada formato de datos para cada dispositivo. Además, las aplicaciones comparten recursos sin haber una distribución que priorice una aplicación sobre otra, lo que reduce la fiabilidad e impide el rendimiento en tiempo real, mientras que en el IoT-PLC las aplicaciones se ejecutan en contenedores que limitan el uso de recursos y las aislan de los posibles fallos de otras aplicaciones.

En tanto, para aislar cada aplicación en el nodo *fog* se presenta en (Dolui & Kiraly, 2018) una arquitectura de código abierto basada en micro servicios y multicontenedores para un *gateway* IoT similar a la del IoT-PLC, aquí los servicios en contenedores se utilizan para realizar funciones asociadas a un *gateway* como el descubrimiento de dispositivos, la gestión de datos y la integración en la nube. La arquitectura propuesta puede adaptarse para realizar un control en tiempo real mientras se dispone de conectividad con la nube, pero el diseño no permite un despliegue rápido y ligero de los lazos de control con sus dispositivos e interfaces virtuales, porque antes de que se inicie el controlador tiene que comunicarse con otros contenedores para saber cómo interactuar con los dispositivos de campo y obtener los datos del proceso. En comparación, el IoT-PLC es más eficaz al momento de desplegar lazos de control, ya que el método de suscripción del controlador con las interfaces permite un flujo estandarizado y rápido de información con los dispositivos de campo.

En (Badar, Lou, Graf, Barth, & Stich, 2019) se propone un dispositivo de computación ubicado en el borde de la red para proporcionar control usando Comunicación Industrial basada en IP deterministíca, este dispositivo de computación de borde proporciona diferentes aplicaciones y servicios a los dispositivos de campo como funciones de control virtual, análisis de datos y visualización. En este trabajo el PLC basado en *hardware* es reemplazado por un controlador virtual capaz de comunicarse con dispositivos de campo usando tecnologías de redes alámbricas o inalámbricas tales como los buses de campo existentes, Ethernet industrial, redes deterministícas emergentes y redes celulares. Además, en este trabajo se menciona el valor añadido de la contenedorización del control virtual, ya que puede desplegarse y ejecutarse en paralelo en diferentes máquinas virtuales o contenedores en una única plataforma de *hardware*, lo que permite la flexibilidad del sistema, en relación a esto, el IoT-PLC desarrollado utiliza las ventajas mencionadas de los contenedores para facilitar el despliegue de las funciones y sus nuevas versiones. Por otro lado, a diferencia del controlador virtual propuesto, en el IoT-PLC se incluye la comunicación entre PLCs ubicados en el borde de la red, mediante el protocolo CoAP, para

el envío de comandos e información. En cuanto a la comunicación con la nube, el IoT-PLC no solo envía los datos del proceso a la nube, como lo hace el controlador propuesto, sino que se incluyen configuraciones de dispositivos, suscripción de eventos y la descarga de contenedores con aplicaciones.

Una arquitectura para los dispositivos de control se introduce en (Azarmipour, Elfaham, Gries, & Epple, 2019), denominada PLC 4.0, que funciona como un puente entre las aplicaciones de automatización industrial y las tecnologías de la información y proporciona una plataforma virtual para pruebas y simulaciones que se ejecutan en paralelo al procedimiento de control. La obra trata de una nueva arquitectura para el PLC que aplica tanto tecnologías de contenedores como soluciones de hipervisor para garantizar el despliegue dinámico y la conectividad, así como la seguridad, la separación de las aplicaciones críticas, SOs en tiempo real y la comunicación. Además, proporciona una capa de abstracción que aumenta la agilidad del PLC y mejora la calidad del servicio (QoS) y la escalabilidad de la comunicación. La arquitectura propuesta permite cambiar la lógica de control sin afectar al rendimiento, pero ninguna de las demás aplicaciones de los PLC puede modificarse mientras están activas y la migración de contenedores entre los PLC no está incluida en las características, por lo que hacer un cambio en el sistema implicaría detener una o más aplicaciones de los PLC. En cambio, el IoT-PLC permite el manejo selectivo de las aplicaciones que se ejecutan y, al estar aisladas unas de otras, el detener una no implica la detención de otra. En cuanto a la migración, el IoT-PLC puede realizar tareas de control y de migración hacia otro de forma simultánea para evitar que el proceso físico esté fuera de funcionamiento durante un largo tiempo.

En (Ferrer, Mohammed, Chen, & Lastra, 2017) los dispositivos utilizados en la fabricación con capacidades RESTful se conectan a la nube a través de un *gateway* que desempeña el papel de un PLC, que crea instancias virtuales de los dispositivos reales y envía los datos a niveles superiores utilizando mensajes MQTT. Los dispositivos conectados a la arquitectura propuesta se describen en base al modelo FiWare, de esta manera cada dispositivo tiene sus características más relevantes asociadas a su instancia virtual. Esta

arquitectura está pensada para ser utilizada simplemente para conectar dispositivos, pero si se utilizara para el control, el problema sería la falta de capacidad de procesamiento en el PLC como nodo en el borde de la red, que sólo se utiliza como nodo de retransmisión para la integración de datos y carece de una lógica de control. Las funciones presentadas en este trabajo se pueden comparar con el bloque Administrador del IoT-PLC, debido a que este incluye la creación de objetos virtuales y su asociación con las interfaces de comunicación, y la comunicación por medio de MQTT con la conectividad que da el bloque OPC UA como forma de comunicación con niveles superiores. Sin embargo, el IoT-PLC desarrolla, mediante contenedores, la inclusión de tareas de control, procesamiento y guardado de datos, conectividad con otros nodos *fog* y dispone de una HMI, todo esto para realizar la tareas que permite un PLC actual y dar mayor capacidad de procesamiento en el borde de la red.

5.13. Comparación con desarrollos orientados a la Industria 4.0 de fabricantes de PLCs

Los grandes fabricantes de dispositivos de control han lanzado productos listos para su uso en las industrias, estos apuntan a tener un dispositivo de control que puedan interactuar con los dispositios IoT e integrar este dispositivo de control en la Industria 4.0. Estos productos difieren en su concepción, unos son mejoras que modifican directamente al PLC, otros son dispositivos aparte que le permiten utilizar nuevos tipos de comunicaciones y otros son computadores industriales que son distintos a los PLCs, pero permiten realizar las tareas de éste mediante plataformas proporcionadas por los fabricantes. Los desarrollos de los grandes fabricantes y su comparación con el IoT-PLC se presentan a continuación:

5.13.1. Omron

El fabricante Omron se enfoca en dos productos que permiten que un controlador se integre en un sistema de la Industria 4.0: computadores industriales y un PLC con conectividad mejorada, en las siguientes subsecciones se detallan estos productos.

5.13.1.1. Computador Industrial

Un computador industrial (IPC) es un computador de oficina modificado para operar en ambientes industriales al que, en general, se le incorporan funcionalidades de conectividad industrial del tipo Ethernet y, a través de éste, se realiza la comunicación con módulos que permiten el flujo de datos con dispositivos de campo.

En el caso particular de la gama de IPCs de Omron, estos funcionan con el sistema operativo Windows, dentro del dispositivo físico no se incluyen cables y sus componentes pueden ser rápidamente reemplazados para disminuir el riesgo de fallas. En cuanto al *software*, los IPCs pueden ejecutar dos sistemas operativos simultáneamente: Windows y Sysmac, ambos se pueden ejecutar de forma independiente. Sysmac permite la programación y ejecución de aplicaciones según el estándar IEC 61131, mientras que el uso Windows se enfoca en aplicaciones de análisis de datos y programas de alto nivel.

Con esto se logran realizar las funciones de un PLC convencional, mientras se incluyen aplicaciones no relacionadas al control de las variables de un sistema. La ubicación de un IPC en un sistema de control industrial se muestra en la Figura 5.11.

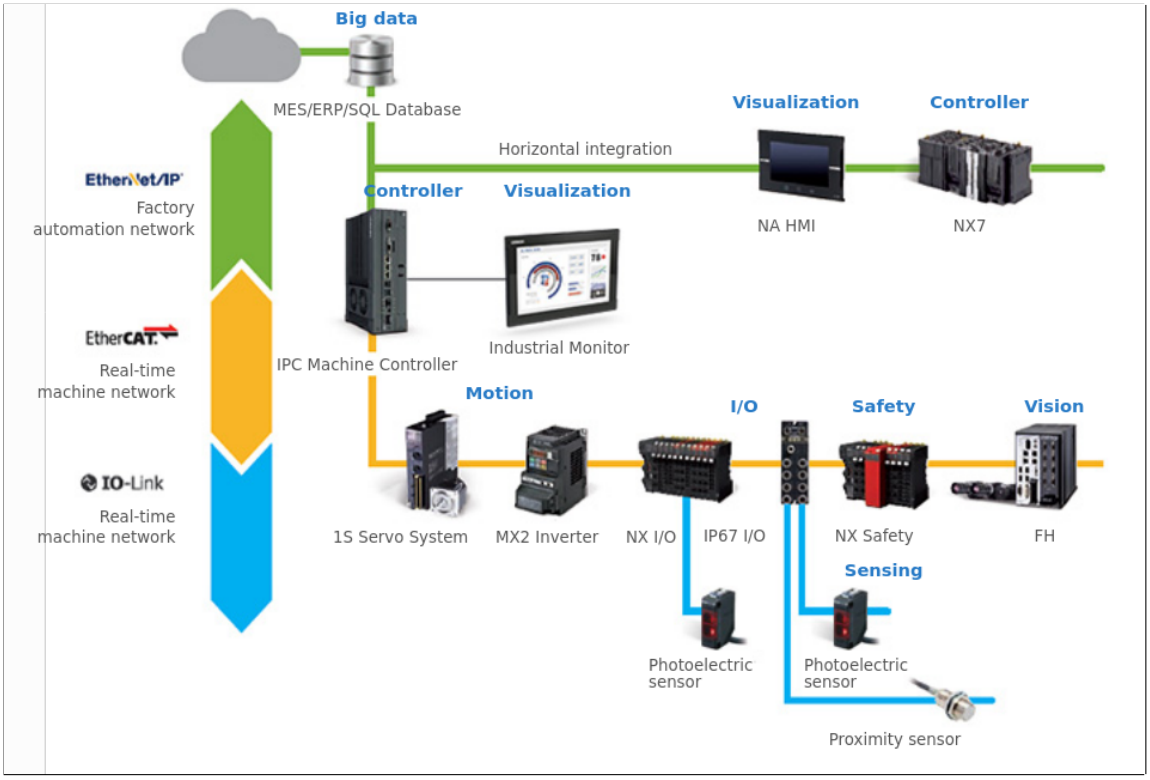


Figura 5.11. Ubicación de un IPC Omron en un sistema de control. Recuperados de: https://www.ia.omron.com.

5.13.1.2. Micro PLC

El Micro PLC desarrollado por Omron es el CP2E (Figura 5.12), el cual es un PLC compacto que incluye conectividad máquina-máquina e IIoT. El módulo central incorpora un *switch* Ethernet y dos puertos Ethernet, con esto se elimina la necesidad de un *switch* externo al PLC y se permite la comunicación con los módulos de expansión. La comunicación con los dispositivos de campo se permite a través de protocolos seriales y Modbus.

Además, es posible incorporar bloques de funciones para el desarrollo mas expedito de aplicaciones de control basado en IEC 61131, tales com o controladores PID autoajustables, bloques Ethernet para el intercambio de información entre dispositivos y Modbus para la comunicación con dispositivos serial.

Figura 5.12. Micro PLC CP2E de Omron. Recuperados de: https://automation.omron.com.

5.13.2. ABB

El fabricante ABB se enfoca en una línea de PLCs con conectividad mejorada mediante un módulo de comunicación que le permite incluir las funcionalidades para operar como *gateway*. Esta línea es denominada AC500 (Figura 5.13), el cual puede funcionar como *gateway* en el borde de la red y puede ser conectado directamente a la nube mediante comunicaciones cifradas. La conectividad hacia la nube se logra a través del uso de estándares de comunicación del tipo Fieldbus y el uso de protocolos OPC UA y MQTT, de esta forma la comunicación entre el PLC y la nube es independiente de la plataforma de uso. Además, el AC500 incluye un servidor web HTML5 para el desarollo de una HMI directamente en el PLC.

Figura 5.13. PLC ABB AC500. Recuperados de: https://www.abb.com/plc.

5.13.3. Siemens

Siemens ha trabajado en *gateways* a los que se conectan sus PLCs para habilitar la comunicación con capas superiores, esto tiene la ventaja de no necesitar un cambio de los equipos, sino que solo se necesita conectar un *gateway* como si fuese un módulo de comunicación.

SIMATIC IOT2000 y SIMATIC CloudConnect 7 son las líneas de *gateways* que Siemens tiene a disposición, el primero está diseñado para dispositivos industriales de cualquier tipo, mientras que el segundo se enfoca en los PLCs de la línea S7, la ubicación de un *gateway* Siemens se muestra en la Figura 5.14 . Ambos son compatibles con los protocolos MQTT, Modbus, Ethernet y plataformas en la nube como Siemens MindSphere, IBM Cloud y Microsoft Azure IOT Hub. Con estos productos se permite la incorporación de uno o varios PLCs en un sistema IIoT sin cambiar la arquitectura interna del PLC, esto significa que la incorporación de aplicaciones que utilicen tecnologías nuevas, como base de datos NoSQL o virtualización de OSs, deban ser ejecutadas de forma externa al PLC y la información resultante tenga que ser transmitida a través del *gateway*.

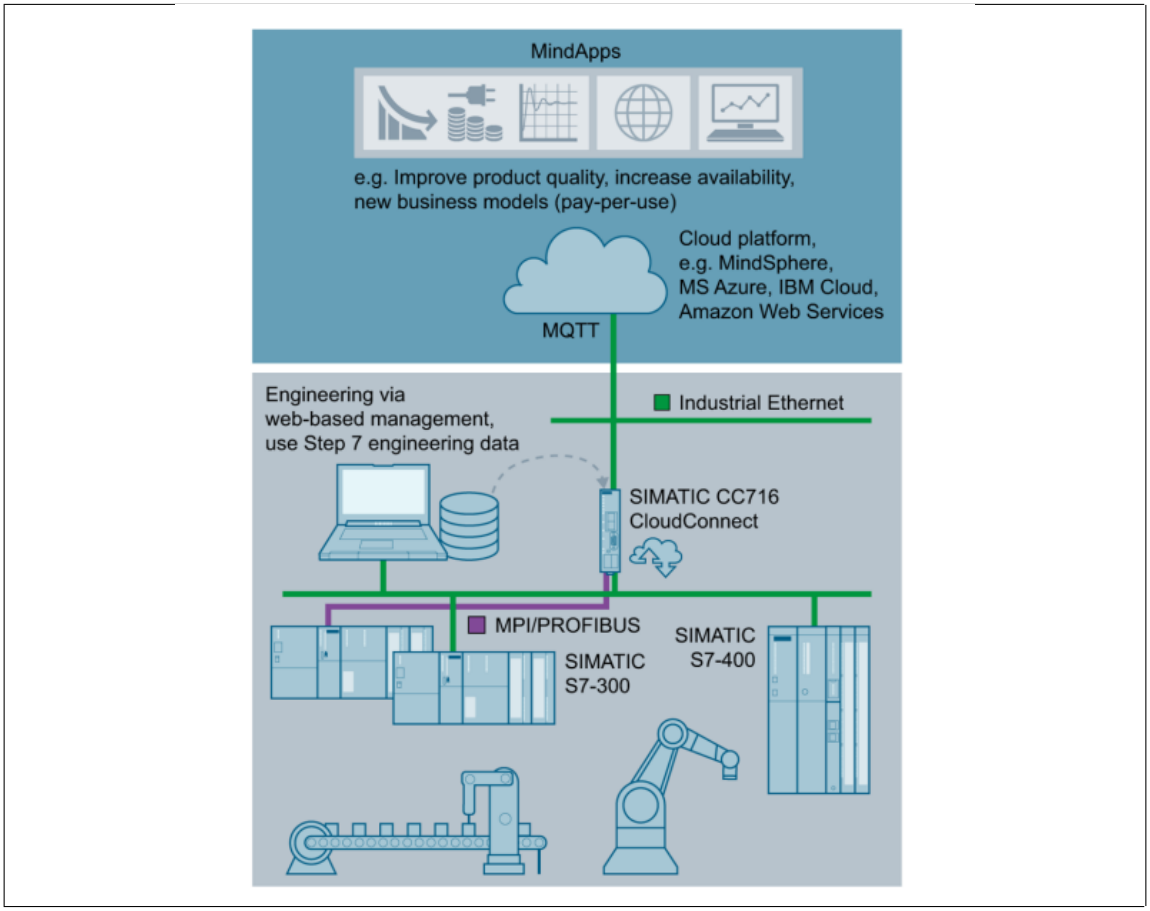


Figura 5.14. Ubicación de un gateway SIMATIC CloudConnect 7 en un sistema de control. Recuperados de: https://new.siemens.com.

5.13.4. Rexroth

Rexroth centra sus desarrollos, relacionados a la industria del futuro, en la plataforma ctrlX AUTOMATION lanzada en junio de 2020. Esta plataforma utiliza un sistema operativo abierto basado en Linux y opera en la línea de *hardware* ctrlX, la cual tiene como núcleo fundamental al ctrlX CORE, un IPC de alto rendimiento que a través de Ethernet de tiempo real se conecta con módulos de físicos de conectividad, I/O y control. ctrlX AUTOMATION provee de todos los elementos requeridos por una solución de control industrial, incluye programas que permiten un rápido desarrollo de HMIs, *gateways*, controladores basados en IEC 61131 y C/C++, servidores web y cortafuegos. Al ser una

plataforma abierta los usuarios de la plataforma pueden incluir desarrollos del área IT o incluir programas de control escritos en otros lenguajes de programación. De esta manera ctrlX AUTOMATION permite implementar todas las funcionalidades de un PLC y otras relacionadas con el área IT dentro de un mismo dispositivo.

La arquitectura de ctrlX AUTOMATION se muestra en la Figura 5.15, posee tres capas: la más interna es la del *hardware*, la capa intermedia es la del *software* que se ejecuta en él y la externa es la que permite la conectivdad con los sistemas disponibles y sistemas de administración.

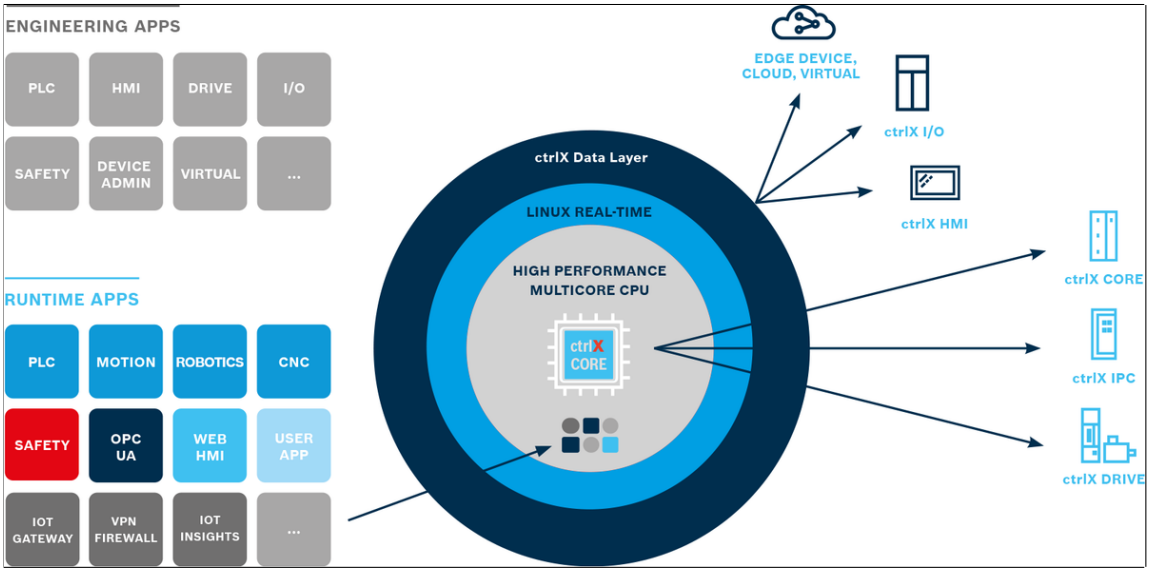


Figura 5.15. Arquitectura de plataforma ctrlX AUTOMATION de Rexroth. Recuperados de: https://apps.boschrexroth.com/microsites/ctrlx-automation/.

5.13.5. Comparación de los desarrollados con el IoT-PLC

Al comparar al IoT-PLC con los desarrollos de los grandes fabricantes Siemens, Omron y ABB se puede ver que ninguno de ellos poseen la característica de incorporar a la tecnología de contenedores para la ejecución de procesos, por esta misma razón la migración de los contenedores para migrar el proceso de control hacia otro dispositivo no ha sido incluida en los PLCs comerciales. Además, la incorporación de tecnologías de

base de datos presentes en el IoT-PLC solo tiene cabida en los IPCs, como los de Omron, pero estos no son dispotivos multiplataforma, por lo podrían haber limitaciones en los programas que se puedan ejecutar. En cuanto a la conectividad, los estándares que el IoT-PLC incluye en su estructura (CoAP, OPC UA, interfaces inalámbricas) pueden ser incorporados en un PLC de cualquier fabricante por medio de módulos de conectividad o el uso *gateways*, pero esto último significa que haya un dispositivo compartido con otros externos al PLC del que se dependa, lo que podría afectar al rendimiento. Además, las funciones que realiza el bloque Administrador difícilmente puedan ser incorporadas en un PLC convencional, por lo que habría que incorporar un dispositivo externo al que el PLC pueda comunicarse para obtener la información que resulta del bloque Administrador.

Por otro lado, al comparar la plataforma ctrlX AUTOMATION de Rexroth con el IoT-PLC se puede notar que la primera no es una arquitectura específica para operar como PLC, mientras que el segundo sí lo es, pero permite que los desarrolladores implementen aplicaciones de control. Por lo tanto, la estructura de bloques, la estructura de descripción de los dispositivos virtuales, las interfaces de comunicación y el sistema de administración de dispositivos podrían ser programados y ejecutados en la plataforma ctrlX AUTOMATION. Sin embargo, la incorporación de contenedores y la migración de ellos en vivo no es mencionado por Rexroth y probablemente no sea viable incluirlos en la práctica, debido a que la plataforma es extensa y depende profundamente de las librerías del SO, por lo que el contenedor de una aplicación tendría tantas dependencias que cualquier cambio en las librerías del SO o de la plataforma haría necesario la creación de un nuevo contendor actualizado, así se perdería la flexibilidad a la que apunta la virtualización mediante contenedores.

6. CONTROL DE PROCESOS EN TIEMPO REAL USANDO UN IOT-PLC

En esta sección se describirán la implementación de un prototipo del IoT-PLC, la forma en que fueron desarrollados sus bloques internos y las razones por las que se escogieron los distintos tipos de *software*. Luego, se describirá la planta que se utilizó para la evaluación del prototipo y la forma en que se implementó. Después, se detallarán los pasos que se siguieron en la evaluación, tanto en la parte en que no aún no se inicia una migración de contenedores como durante el proceso de migración en vivo. El capítulo finaliza con la presentación de los resultados y el análisis de ellos.

6.1. Implementación del prototipo

Para la implementación del IoT-PLC se utilizó una tarjeta de desarrollo Raspberry Pi 4B en su versión de 4GB de memoria RAM, su interfaz Ethernet fue usada para conectar al prototipo a internet, para transferir datos con un computador que simulaba a una nube y para la comunicación mediante CoAP entre otro prototipo del IoT-PLC. En cuanto a la conectividad con los sensores y actuadores, se utilizó la interfaz Bluetooth incluida en la tarjeta Raspberry Pi 4B, además, fue utilizado un módulo XBee S2D ZigBee por medio de la comunicación serial de la tarjeta, este módulo provee una interfaz de comunicación Zigbee (IEEE 802.15.4).

Para el intercambio de datos con el módulo XBee se programó un *script* en el lenguaje Python que utilizaba la librería xbee, la cual implementa el *software* para usar las APIs XBee mediante comunicación serial.

El sistema de intercambio de información, basado en suscripción/publicación, entre los bloques que componen al IoT-PLC fue implementado usando la DMBS Redis, la cual tiene dentro de sus características ser liviana, de alto rendimiento y estable.

El bloque de base de datos fue implementado con dos sistemas de base de datos. El primero, SQLite fue utilizado para guardar los valores de las mediciones de los sensores

y las señales de control dirigidas hacia los actuadores, SQLite es una base datos rápida y que añade una baja carga computacional. El segundo fue MongoDB, el que se utilizó para guardar información de las configuraciones de los controladores, dispositivos de campo y el tipo de filtro que debía usarse.

La interfaz OPC UA fue desarrollada mediante la librería python-opcua, esta librería permite da la opción de iniciar tanto servidores como clientes, por lo que el servidor OPC UA fue ubicado en el computador que simulaba la nube y los IoT-PLCs fueron conectados como clientes.

El bloque HMI se desarrolló con el *framework* Flask y la visualización interactiva con la librería Bokeh, ambas escritas principalmente en Python. Con estas librerías se logró que la HMI fuese accesible mediante cualquier navegador web, ya que usan el protocolo HTTP.

La plataforma de *software* que se utilizó para el uso de los contenedores fue Docker, el cual destaca por ser ligero, flexible y permitir la división de los recursos entre distintos contenedores. Docker Registry 2.0 fue utilizado para la creación y manejo del repositorio que se necesita en la migración de contenedores, su ventaja es que está diseñado para integrarse con Docker y sus APIs.

Finalmente, la migración en vivo de los contenedores se desarrolló con el *software* CRIU, este permite pausar un contenedor, guardar su estado en memoria de forma persistente y volver a iniciarlo en el mismo dispositivo u otro distinto. Mientras que la automatización del proceso de migración se desarrolló utilizando OPC UA.

6.2. Planta: Sistema de cuatro tanques

Para evaluar al prototipo del IoT-PLC en un sistema de control y a la migración en vivo de contenedores se utilizaron dos prototipos de IoT-PLC conectados en la misma red local. Se escogió un sistema de cuatro tanques (Figura 6.1) para ser controlado, este

sistema fue tomado de la formulación propuesta en (Johansson, 2000) y simulado dentro de un ambiente Python.

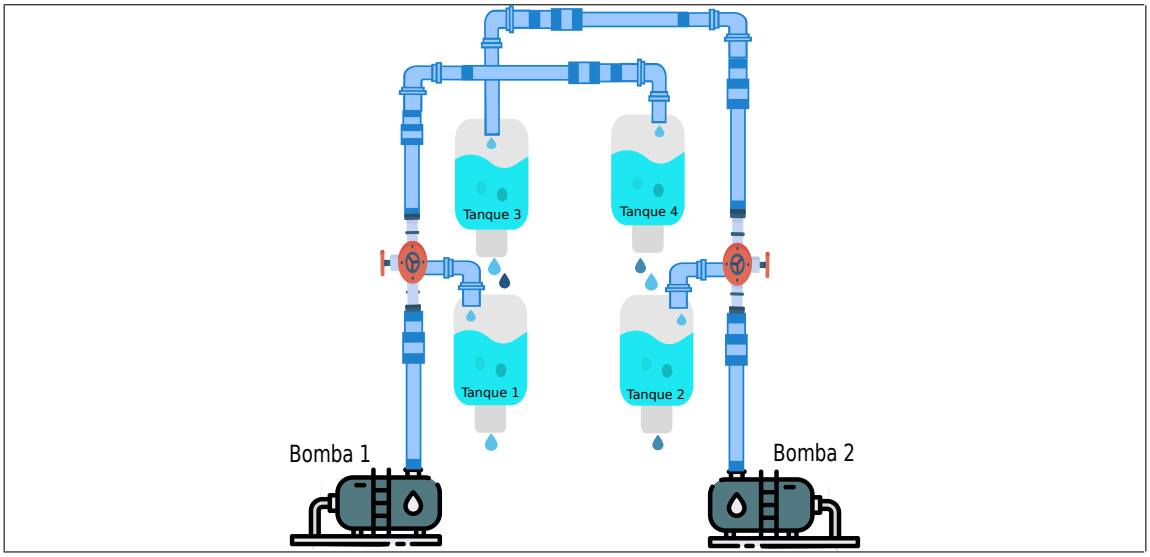


Figura 6.1. Sistema de cuatro tanques simulado.

Las ecuaciones que modelan el comportamiento del sistema de cuatro tanques son las siguientes:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1 \tag{6.1}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2 \tag{6.2}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 \tag{6.3}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 \tag{6.4}$$

En las ecuaciones A_i corresponde al área de la sección transversal del tanque i, a_i corresponde al área de la sección transversal del agujero de salida, h_i es el nivel de agua, el voltaje aplicado a la bomba i es v_i y el parámetro γ_i depende de la apertura de las válvulas. Para la evaluación se utilizaron los valores mostrados en la Tabla 6.1 para los parámetros anteriores:

Tabla 6.1. Parámetros del sistema de cuatro tanques.

Parámetros	Valor
A_1, A_3	28 cm^2
A_2, A_4	32 cm^2
a_1, a_3	0,071 cm^2
a_2, a_4	0,057 cm^2
g	981 cm/s^2
γ_1	0,7
γ_2	0,6
k_1	3,33 cm^3/Vs
k_2	3,33 cm^3/Vs

El sistema simulado fue implementado en una tarjeta de desarrollo Beaglebone Black Wireless (BBBW), en el cual los datos generados de los sensores fueron transferidos mediante un módulo XBee S2D ZigBee conectado a la BBBW mediante UART. Mientras que los datos de los actuadores se recibían por medio de la interfaz Bluetooth incorporada en la tarjeta. En la Figura 6.2 se muestran las partes del prototipo, las interfaces de comunicación utilizadas y el sistema simulado.

Este sistema simulado en base al modelo mostrado tiene la limitación, con respecto a un sistema de tanques real, de no incluir a los tiempos de respuesta de los actuadores y de los sensores en la dinámica del sistema. Además, los tiempos de procesamiento de las señales digitales de los sensores y actuadores no están definidos, aunque pueden asumirse cercanos a la realidad, ya que la tarjeta BBBW debe procesar las señales y simular la planta al mismo tiempo, de esta forma se obtienen tiempos de procesamiento promedios parecidos a los que tendrían los dispositivos reales. Por otro lado, para el caso del procesamiento de señales análogas no se tienen incluidos sus tiempos de procesamiento en sensores y actuadores. Sin embargo, la simulación se considera adecuada para probar al prototipo del IoT-PLC, porque incluir tales tiempos de respuestas y de procesamiento influiría en el rendimiento del bloque Controlador y solo requeriría una adaptación o cambio de los controladores que se probarán (PID y MPC), pero esto no cambiaría el funcionamiento de los demás bloques, por lo que las conclusiones que se obtengan de las pruebas con el sistema simulado serán comparables a si se utilizase un modelo más realista o uno real.

Es importante mencionar que este prototipo no está diseñado para operar en ambientes industriales, sino que es una prueba de concepto del IoT-PLC ubicado en un ambiente controlado, siempre en las mismas condiciones y sin elementos que interfieran en la comunicación inalámbrica de forma aleatoria. Además, es conocido que ZigBee y Bluetooth no tienen las características ni confiabilidad necesarias para una red industrial, pero estos protocolos fueron utilizados porque en la red de prueba creada hay solo tres nodos ZigBee separados por menos de un metro y, como se mostró en la Sección 4.2.3, el rendimiento sería comparable a protocolos industriales inalámbricos como WirelessHART en estas condiciones. Mientras que en el caso del uso Bluetooth, en la red hay tres nodos separados por menos de un metro y en estas condiciones el rendimiento de este protocolo debiese presentar una baja pérdida de paquetes y un retardo menor a 50 ms, esto en base a las pruebas que se hicieron en (Meier et al., 2007), en el que dispositivos Bluetooth fueron ubicados en un ambiente industrial separados por 3 metros y el resultado mostró un error relativo de $1,4 \cdot 10^{-4}$.

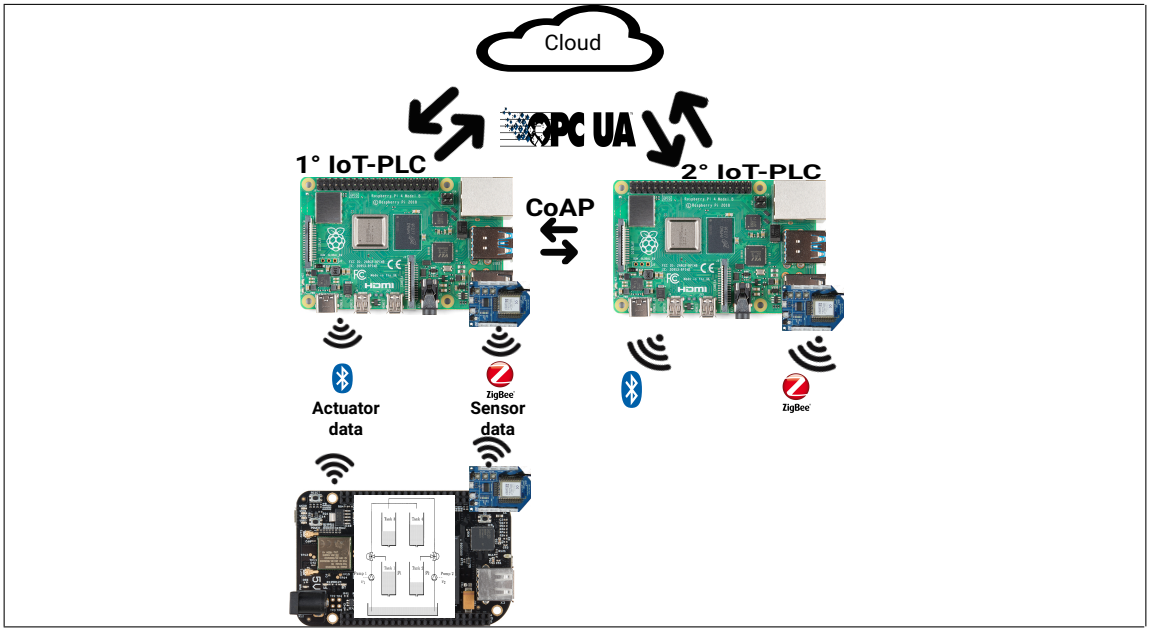


Figura 6.2. Partes del prototipo del IoT-PLC.

La evaluación consistió en controlar el nivel de agua de los tanques inferiores (tanque 1 y 2) manipulando el voltaje de las bombas 1 y 2. El nivel inicial, en el cual empieza la evaluación, se ajustó en 12,4 cm para el primer tanque y 12,7 cm para el segundo tanque, luego se cambiaron las referencias para llevar el nivel de agua de los tanques a 32 cm y 28 cm, respectivamente. Luego, se avisa desde la nube que el primer IoT-PLC (el que controla los niveles de agua) debe iniciar un proceso de migración hacia el segundo IoT-PLC, este aviso ocurre dos minutos después de que se cambian las referencias de los niveles de agua. Durante la migración se realizan otros dos cambios de referencias adicionales, el primero ocurre justo después de terminar de crear el repositorio con las imágenes de los contenedores, en este las referencias se ajustan en 25 cm para el primer tanque y 39 cm para el segundo tanque. El último cambio de referencia se inicia justo después de que el segundo IoT-PLC empieza a controlar el proceso, aquí las referencias se ajustan a 35 cm para el primer tanque y 20 cm para el segundo tanque. Para controlar los niveles de agua se experimentó con dos tipos de controladores, un PID y un MPC. En las siguientes subsecciones se presentarán los resultados de la evaluación.

6.2.1. Controlador PID

En esta prueba se utilizaron dos controladores PID desacoplados para manipular los voltajes de las bombas, uno por cada tanque al que se le controla el nivel de agua. Como se puede ver en la Figura 6.3, el IoT-PLC pudo llevar y mantener el nivel de agua de cada tanque según las referencias ajustadas y luego de la migración de los contenedores el sistema sigue estable en la nueva referencia. Sin embargo, durante el guardado de estado en memoria el nivel de agua tiene un repentino aumento de nivel, esto ocurre durante la pausa de los contenedores que se necesitan para el control (interfaces XBee y Bluetooth, Filtro y controlador PID), este comportamiento será analizado más adelante en esta sección.

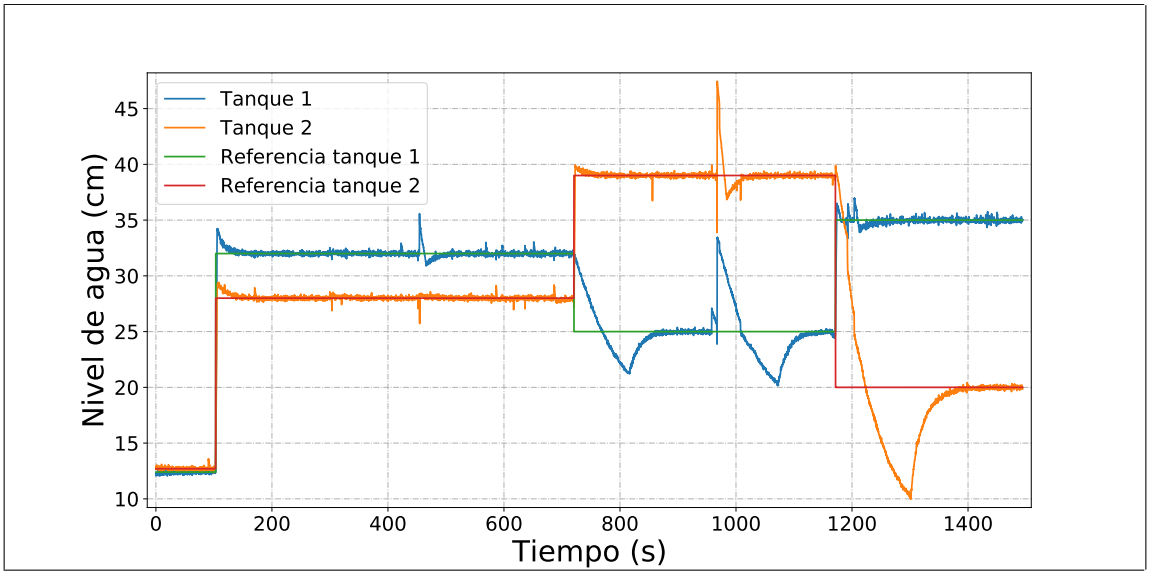


Figura 6.3. Resultado de evaluación con controlador PID.

En relación al tamaño de las imágenes de los contenedores, se puede ver en la tabla 6.2 que la imagen base para el lenguaje de programación Python y las librerías que son comunes a todas las otras imágenes es la más grande, mientras que casi todas las otras no superan los cinco megabytes. La excepción es la imagen del bloque de la base de datos, porque en ella se debieron instalar los programas Redis y MongoDB.

Tabla 6.2. Tamaño de los contenedores utilizados en la prueba con controladores PID.

Imagen	Tamaño (kilobytes)
Base Python	554.711
Xbee	8
OPC UA	15,6
Administrador	16,9
HMI	4.949
Filtro	6,1
Base de datos	504.493
Controlador PID	11
CoAP	6,1
Bluetooth	5,4

En la migración seis fueron los contenedores a los que se les realizó el guardado de su estado en memoria, correspondientes a los bloques Controlador, Administrador, interfase Bluetooth, interfase XBee, HMI y Filtro. Los bloques a los que no se les guardó el estado en memoria fueron interfase OPC UA, interfase CoAP y Base de datos, ya que los bloques de interfaces ya estaban iniciados en el segundo IoT-PLC antes de empezar la migración. Mientras que para el bloque Base datos no es necesario guardar sus estados en memoria, porque su funcionamiento no varía si es que empiezan desde cero o sigue desde el punto en que fue pausado luego de ser migrado, dado que solo recibe peticiones externas al bloque y guarda información en bases de datos que se transfieren durante la creación de los repositorios en el primer IoT-PLC.

En cuanto a la duración total de la migración, en esta prueba fue de dieciséis minutos y veintisiete segundos. De este tiempo ocho minutos y veintiún segundos fueron requeridos para generar el repositorio con las imágenes de los contenedores en el primer IoT-PLC, cuatro minutos y once segundos fueron requeridos para la transferencia de las imágenes entre IoT-PLCs y su posterior carga en el sistema. El tiempo total que se utilizó para la obtención de los estados en memoria y su transferencia fue de tres minutos y cuarenta y siete segundos, de ese tiempo la duración de la pausa por la que cada contenedor tuvo que pasar se muestra en la Tabla 6.3.

Dado que el tiempo de pausa de los contenedores que se necesitan para el control (interfaces XBee y Bluetooth, Filtro y controlador PID) son críticos para la estabilidad de la planta se presenta una mejora al proceso de migración en la Sección 6.2.4, el cual involucra el inicio de un contenedor temporal que supla el funcionamiento del contenedor pausado.

Tabla 6.3. Tiempo que se pausa cada contenedor al guardar su estado en memoria con el controlador PID.

Bloque	Controlador	Administrador	Bluetooth	XBee	Filtro	HMI
Pausa (segundos)	3,95	7,9	3,1	6,2	4,3	72,1

6.2.1.1. Comparación de latencias entre funcionamiento con y sin contenedores

Para comparar la carga extra que añaden los contenedores, durante el proceso de control, se realizó una evaluación con los *scripts* Python que representan a los bloques del IoT-PLC ejecutándose directamente en el sistema operativo y otra con los *scripts* dentro de contenedores. La evaluación constó de los tres cambios de referencia descritos anteriormente separados por cuatro minutos y el último cambio se mantuvo por tres minutos, todo esto sin iniciar el proceso de migración, debido a que, al correr los *scripts* sin contenedores, la migración no puede llevarse a cabo. Los datos registrados para determinar la

carga extra fueron: la latencia total, determinada tiempo transcurrido entre que el sensor envía un dato y el actuador recibe una señal de control, y el tiempo transcurrido desde justo después de que el IoT-PLC recibe una medición hasta justo antes de enviar una señal de control. Los resultados del tiempo promedio y la desviación estándar de las mediciones se pueden ver en la Tabla 6.4, estos fueron obtenidos con tres mil quinientas mediciones. Además, en las Figuras 6.4, 6.5, 6.6 y 6.7 se presentan histogramas donde se concentran más del $99,9\%$ de las mediciones.

Tabla 6.4. Comparación de latencias entre funcionamiento con y sin contenedores.

Tiempo (ms)	Promedio	Desviación estándar
Latencia total (sin contenedores)	273	71,6
Tiempo ocupado por IoT-PLC (sin contenedores)	30	39,2
Latencia total (con contenedores)	260	70,1
Tiempo ocupado por IoT-PLC (con contenedores)	25	33,2

Los resultados muestran que el tiempo promedio utilizado por el IoT-PLC para procesar la información es menor cuando se utilizan contenedores, lo mismo ocurre con la dispersión de las muestras. El hecho de que el desempeño haya sido mayor al utilizar contenedores podría explicarse por las siguientes razones: la primera es que el efecto de la carga computacional añadida por los contenedores sea menor a los efectos derivados de la prioridad asignada por el sistema operativo a los programas ejecutándose, así el efecto no sería apreciable en esta prueba. La segunda razón es que los contenedores, al asignar y limitar los recursos computacionales a los que cada contenedor puede acceder, permiten que los bloques del IoT-PLC no se disputen los mismos recursos entre ellos y los que tienen mayor importancia obtienen mayor prioridad y recursos. Al analizar los histogramas se puede llegar a la conclusión de que la segunda razón es la que tendría mayor influencia en

el funcionamiento del IoT-PLC, ya que al existir una organización de los recursos computaciones, las tareas de los procesos deberían tender a realizarse en un período tiempo más estable.

El hecho de que la asignación de recursos permita obtener tiempos de procesamiento más estables posibilita a que el IoT-PLC pueda asegurar tiempos de procesamiento acotados, por lo tanto, según los requerimientos del proceso a controlar, un operador podría ajustar el nivel de recursos que cada contenedor puede utilizar para obtener tiempos de procesamiento estables, esto siempre y cuando el *hardware* sobre el que opera el IoT-PLC lo permita.

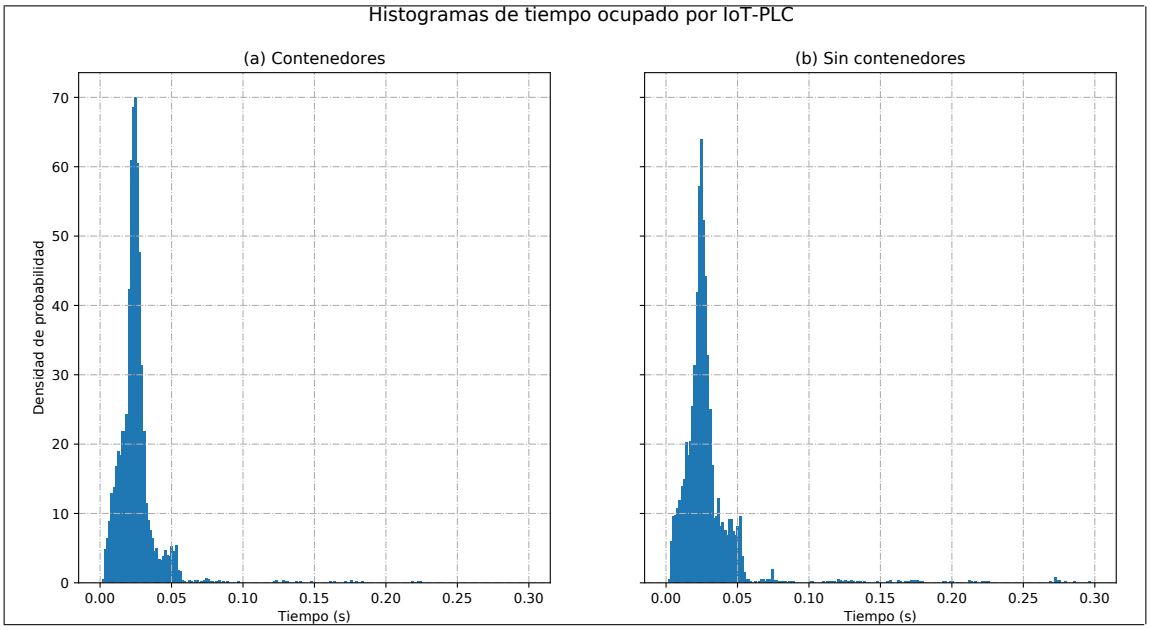


Figura 6.4. Histogramas de tiempo ocupado por IoT-PLC con controlador PID.

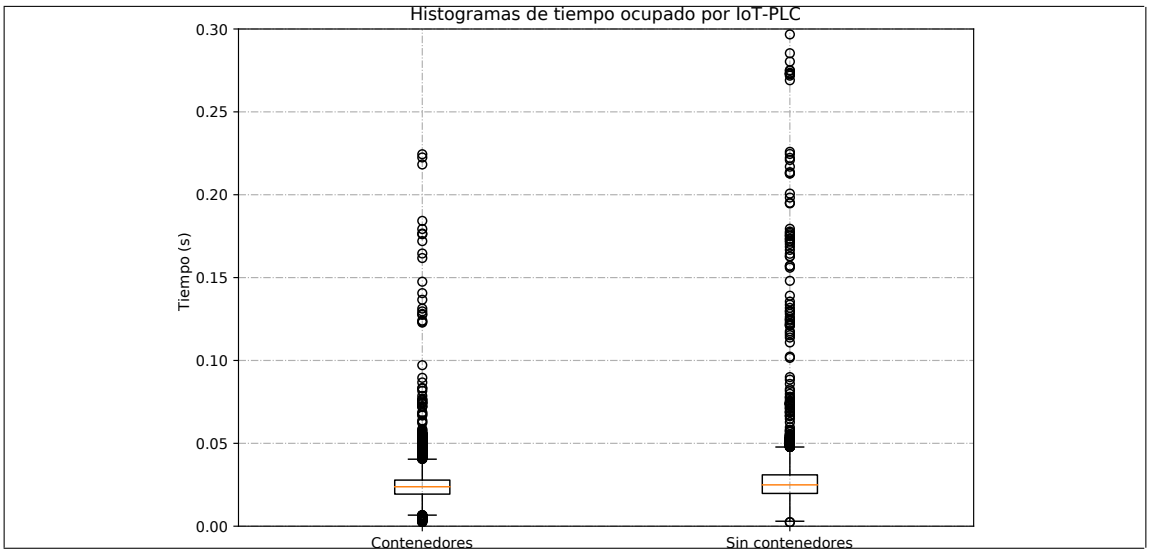


Figura 6.5. Boxplot de tiempo ocupado por IoT-PLC con controlador PID.

Por el lado de la latencia total, los resultados muestran que el tiempo utilizado es ligeramente menor cuando se utilizan contenedores y que los resultados de las mediciones son menos dispersas con el uso de ellos, pero la diferencia puede haber ocurrido por la variación en los tiempos de transmisión propios de los canales inalámbricos utilizados y no por el uso de contenedores .

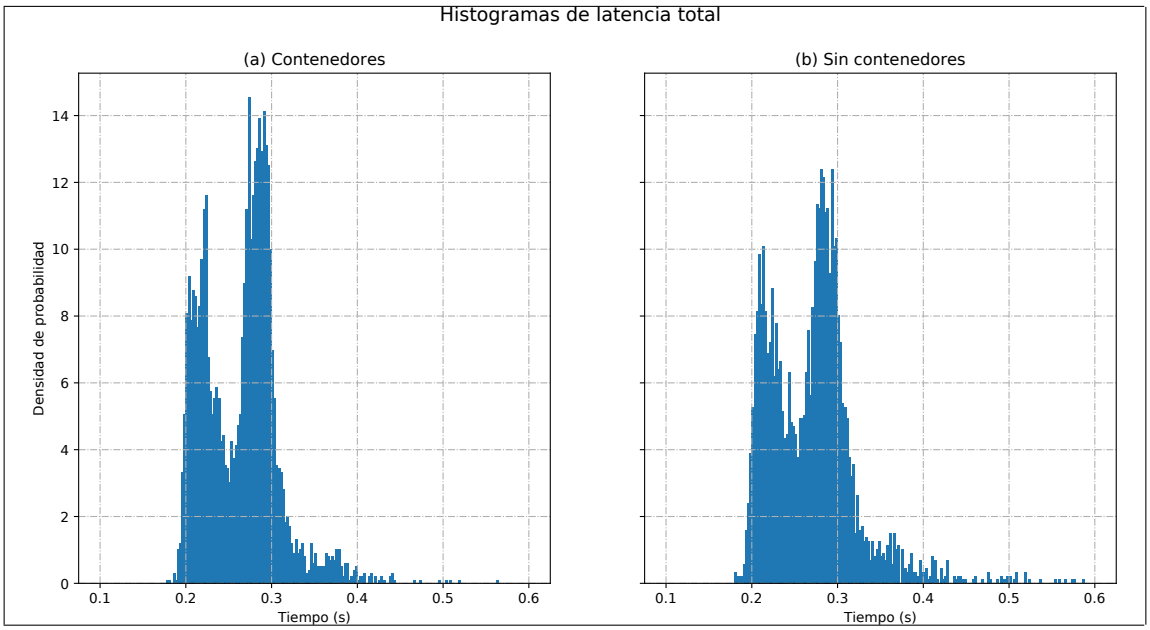


Figura 6.6. Histogramas de latencia total con controlador PID.

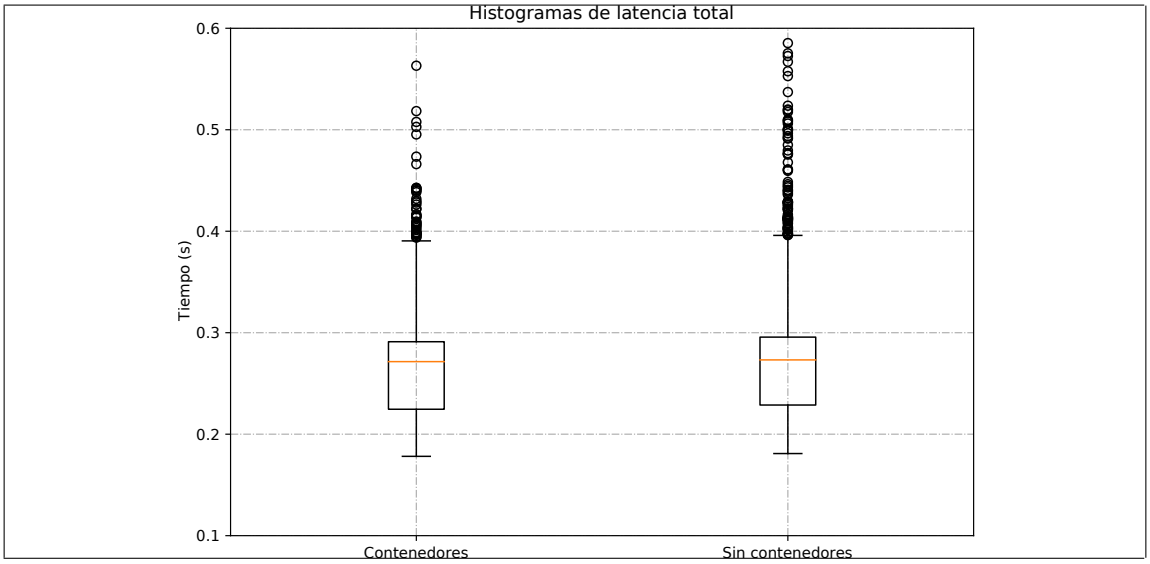


Figura 6.7. Boxplot de latencia total con controlador PID.

6.2.2. Controlador MPC

En la siguiente prueba se utilizó un controlador MPC que utilizaba un modelo lineal del sistema de cuatro tanques para calcular las señales de control futura, el modelo lineal fue obtenido del modelo linealizado del sistema presentado en (Johansson, 2000) y se muestra a continuación:

$$\frac{dx}{dt} = \begin{bmatrix} \frac{-1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & \frac{-1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & \frac{-1}{T_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_4} \end{bmatrix} x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2 k_2)}{A_3} \\ \frac{(1-\gamma_1 k_1)}{A_4} & 0 \end{bmatrix} u \tag{6.5}$$

Modelo en que T_i equivale a $\frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}}$, x_i corresponde a $h_i - h_i^0$ y u_i es igual a $v_i - v_i^0$, el superíndice 0 corresponde al valor de la variable en el punto de linealización. La función de costo del controlador MPC se presenta a continuación:

$$\begin{aligned} \min_{u_o,\ldots,u_{N_p},x_0,\ldots,x_{N_p}} \quad & \sum_{k=0}^{N_p-1} \left[(x_k - x_{ref})^T Q_x (x_k - x_{ref}) + (u_k - u_{ref})^T Q_u (u_k - u x_{ref})\right] \\ & + \Delta u_k^T Q_\Delta \Delta u_k + (x_{N_p} - x_{ref})^T Q_{x_{N_p}} (x_{N_p} - x_{ref}) \end{aligned} \tag{6.6}$$

Las restricciones a las que está sujeto el controlador son las siguientes:

$$\begin{aligned} u_{min} \leq \quad & u_k \quad \leq u_{max} \\ x_{min} \leq \quad & x_k \quad \leq x_{max} \\ \Delta u_{min} \leq \quad & \Delta u_k \quad \leq \Delta u_{max} \\ u_{-1} \quad = \quad & \overline{u} \\ x_0 \quad = \quad & \overline{x} \\ \text{donde} \qquad \Delta u_k \quad = \quad & u_k - u_{k-1} \end{aligned} \tag{6.7}$$

Para las pruebas se ajustaron los horizontes de predicción y control en 120 con un período de muestreo de $0,25$ segundos, los pesos de las matrices Q_x, $Q_{x_{N_p}}$, Q_u y Q_Δ fueron ajustados para que se tomaran en cuenta solo a los niveles de agua de los tanques 1 y 2, estas matrices se muestran a continuación:

$$Q_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10^{-6} & 0 \\ 0 & 0 & 0 & 10^{-6} \end{bmatrix}, Q_{x_{N_p}} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 10^{-6} & 0 \\ 0 & 0 & 0 & 10^{-6} \end{bmatrix} \tag{6.8}$$

$$Q_u = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix}, Q_\Delta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.9}$$

Las imágenes de los contenedores que se usaron en esta prueba fueron los mismos que en la del controlador PID, el único cambio fue el reemplazo de la imagen del controlador MPC por el del PID. La imagen del controlador incluye más librerías que la de los controladores PID, librerías relacionadas a la optimización de sistemas lineales. Esto hace a la imagen más pesada, tuvo un tamaño de 236.206 kilobytes (tomando en cuenta la base de las librerías necesarias para el MPC y la imagen del *script* con el controlador), por lo que tarda más en guardarse en los repositorios, en ser transferida e incorporada en el segundo IoT-PLC. Además, el controlador MPC es más exigente para la CPU de la Raspberry Pi 4B, por lo tanto es esperable que la migración de los contenedores tome mayor tiempo que en el caso de los controladores PID.

Como se puede ver en la Figura 6.8, el MPC pudo llevar y mantener el nivel de agua de cada tanque según las referencias ajustadas, de forma más lenta que el PID, pero resultó más estable durante la migración de los contenedores, solo hubo un pequeño cambio brusco durante el guardado de estados en memoria.

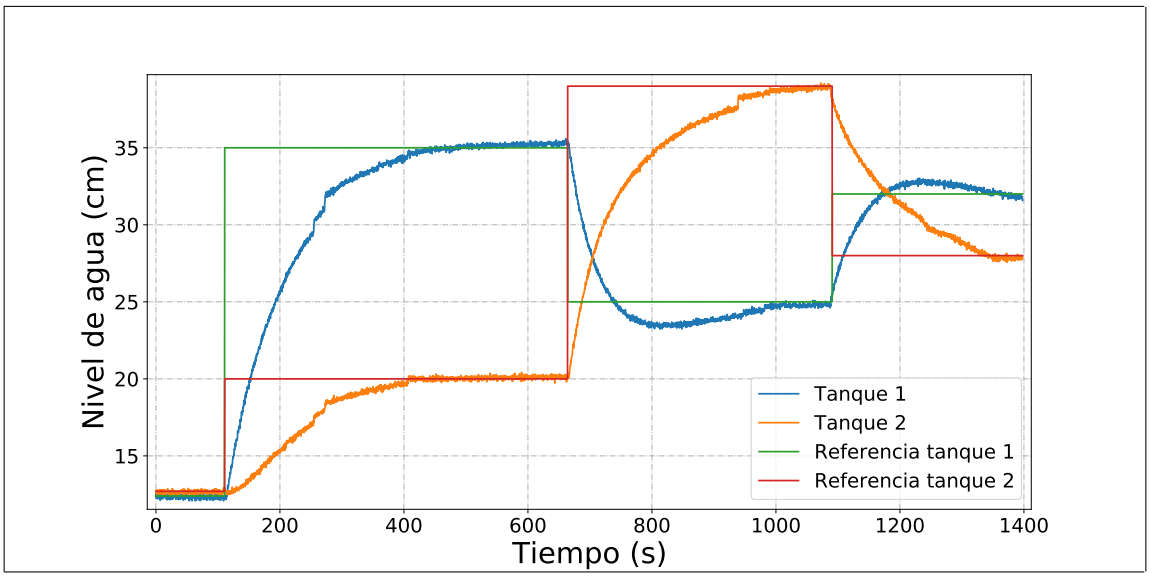


Figura 6.8. Resultado de evaluación con controlador MPC.

En el proceso de migración, los contenedores a los que se les guardó el estado en memoria y a los que no se les guardó son los mismos que en el caso de los controladores PID, por las mismas razones dadas anteriormente.

En cuanto a la duración total de la migración, en esta prueba fue de diecisiete minutos. De este tiempo ocho minutos y veintiún segundos fueron requeridos para generar el repositorio con las imágenes de los contenedores en el primer IoT-PLC, cuatro minutos y treinta segundos fueron requeridos para la transferencia de las imágenes entre IoT-PLCs y su posterior carga en el sistema. El tiempo total que se utilizó para la obtención de los estados en memoria y su transferencia fue de tres minutos y veintidós segundos, de ese tiempo la duración de la pausa por la que cada contenedor tuvo que pasar se muestra en la Tabla 6.5.

Tabla 6.5. Tiempo que se pausa cada contenedor al guardar su estado en memoria con el controlador MPC.

Bloque	Controlador	Administrador	Bluetooth	XBee	Filtro	HMI
Pausa (segundos)	5	6,7	3,2	14,4	17,8	41,4

6.2.2.1. Comparación de latencias entre funcionamiento con y sin contenedores

La prueba para comparar la carga extra que añaden los contenedores fue exactamente la misma que la usada para el caso del controlador PID. Los resultados del tiempo promedio y la desviación estándar de las mediciones se pueden ver en la Tabla 6.6, estos fueron obtenidos con cerca de tres mil doscientas mediciones. Además, en las Figuras 6.9, 6.10, 6.11 y 6.12 se presentan histogramas y boxplots donde se concentran más del $99,7\%$ de las mediciones.

Tabla 6.6. Comparación de latencias entre funcionamiento con y sin contenedores.

Tiempo (ms)	Promedio	Desviación estándar
Latencia total (sin contenedores)	288	73,4
Tiempo ocupado por IoT-PLC (sin contenedores)	32	41,2
Latencia total (con contenedores)	280	92,7
Tiempo ocupado por IoT-PLC (con contenedores)	32	38,7

Los resultados obtenidos con el controlador MPC son similares a los que se obtuvieron con el controlador PID, por lo que las conclusiones a las que se pueden llegar son las mismas.

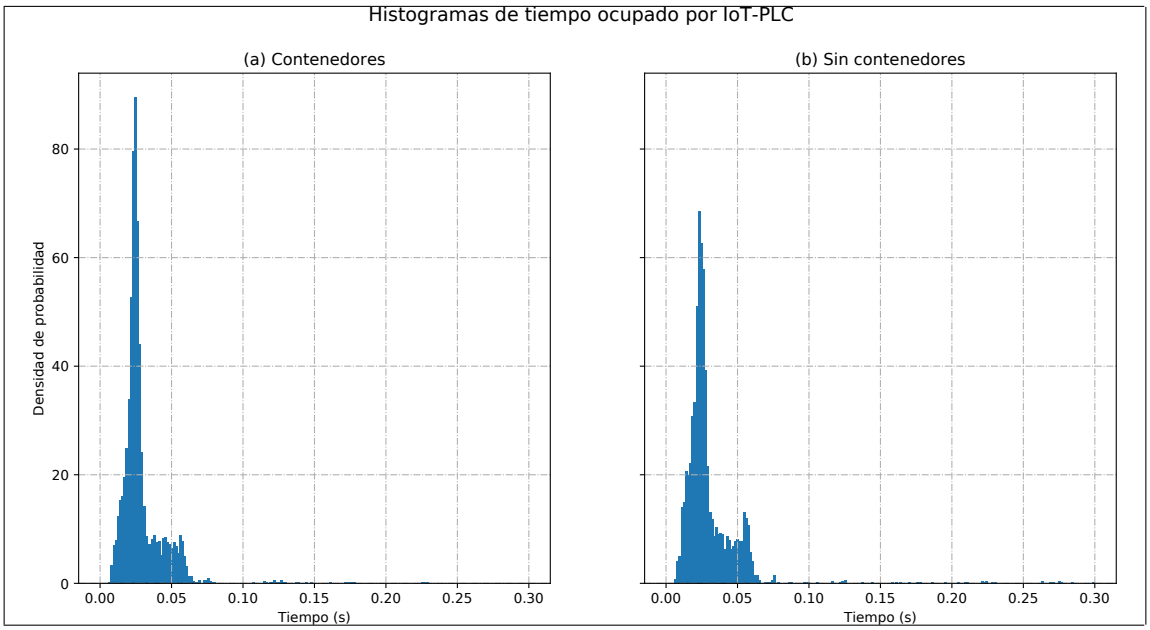


Figura 6.9. Histogramas de tiempo ocupado por IoT-PLC con controlador MPC.

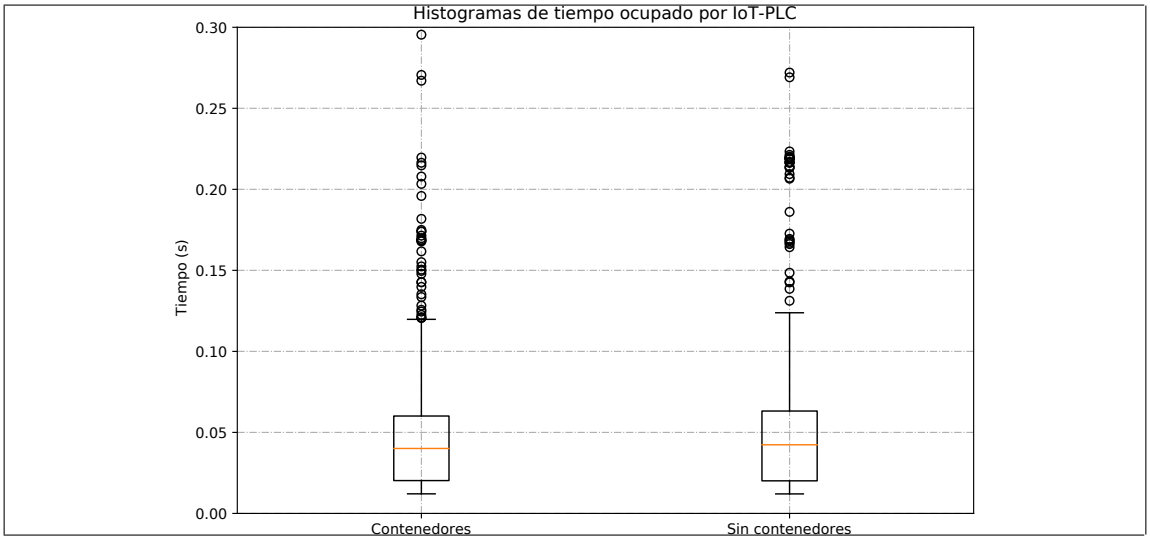


Figura 6.10. Boxplot de tiempo ocupado por IoT-PLC con controlador MPC.

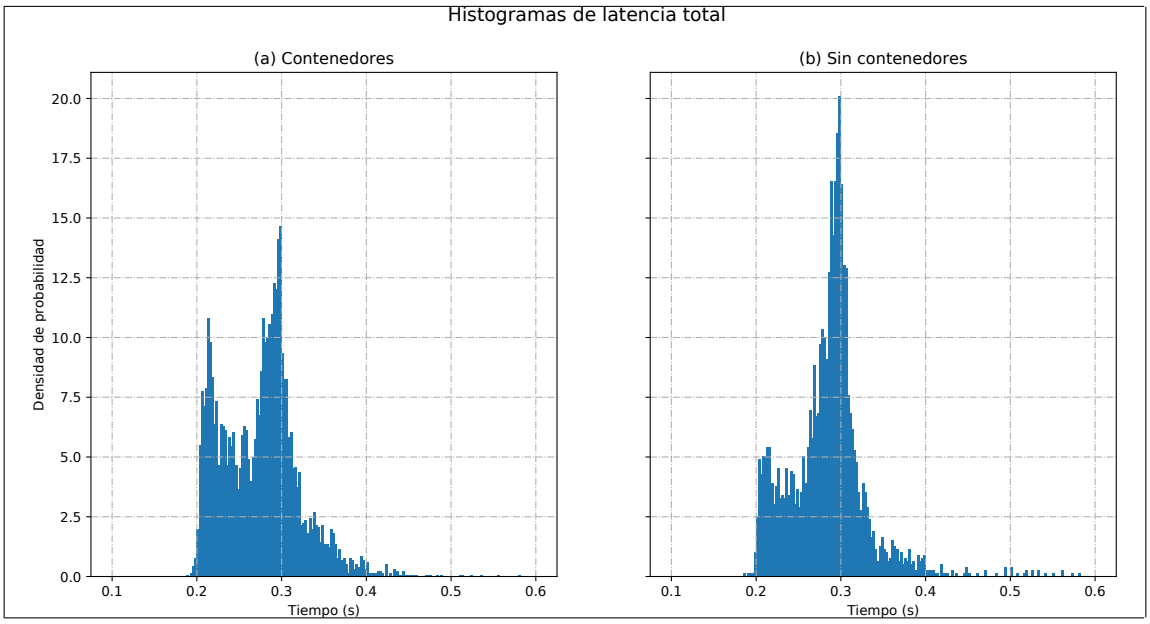


Figura 6.11. Histogramas de latencia total con controlador MPC.

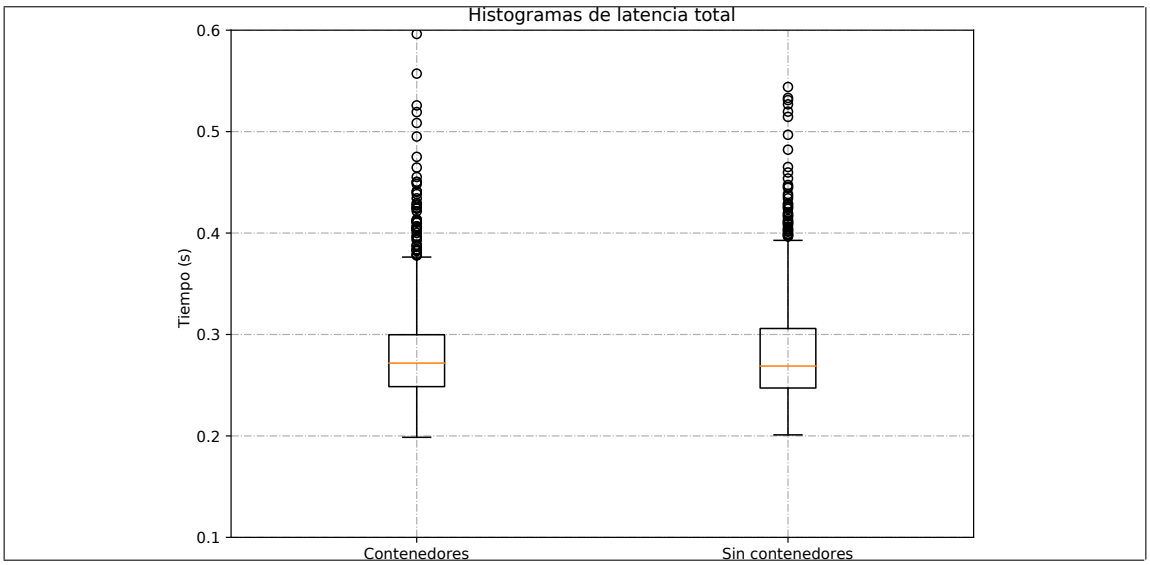


Figura 6.12. Boxplot de latencia total con controlador MPC.

6.2.3. Prueba de robustez frente a detención de contenedores

Al IoT-PLC se le hizo enfrentar una caída de varios contenedores en secuencia para determinar la respuesta de este y el sistema controlado. Esta prueba consiste en realizar un cambio de referencia y después de diez segundos detener el contenedor Administrador, diez segundos después detener el contenedor Filtro y, por último, el contenedor HMI. Para esta prueba se utilizó el controlador PID mencionado anteriormente, el IoT-PLC fue configurado para revisar cada medio segundo si un contenedor no estaba ejecutándose y se esperaba que los pudiese reiniciar.

En la Figura 6.13 se ve el comportamiento de las variables controladas. El tiempo que le tomó al IoT-PLC reiniciar cada contenedor fue de 0,93 segundos para el Administrador, 0,95 segundos para el Filtro y 1,26 segundos para el contenedor HMI. Los resultados muestran que el tiempo en que no se ejecuta el contenedor Filtro es lo suficientemente bajo como para no afectar el rendimiento del control. En cuanto a los contenedores Administrador y HMI, sus detenciones tampoco afectaron en el control, dado que estos no participan directamente en la generación de una señal de control.

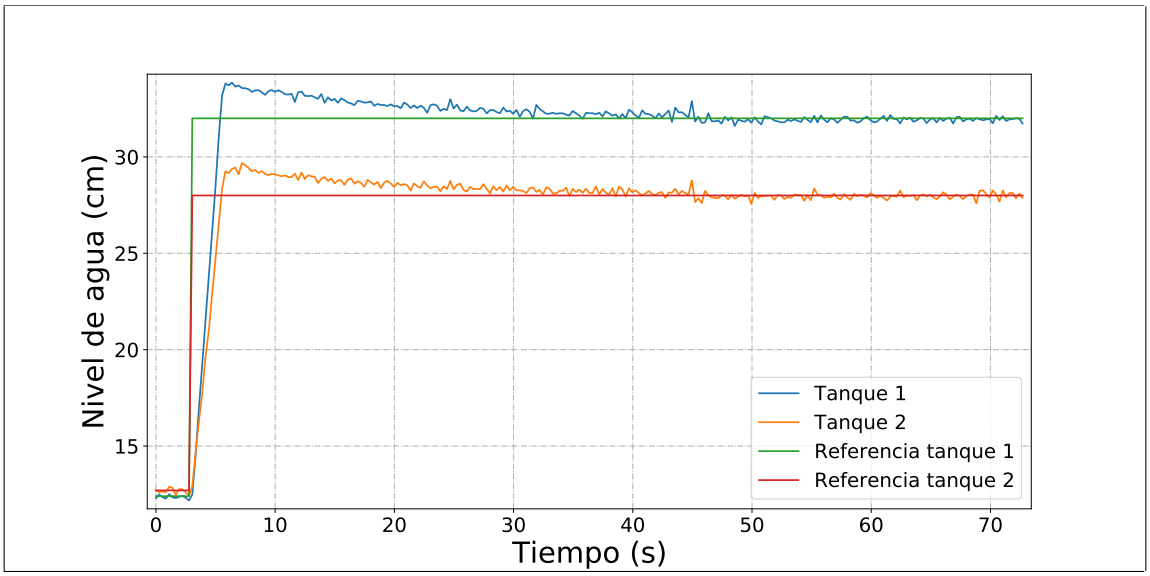


Figura 6.13. Prueba de robustez frente a caídas.

6.2.4. Análisis de los resultados

Los resultados muestran que el prototipo del IoT-PLC es capaz de mantener estable al sistema de cuatro tanques luego del cambio de referencia y durante el proceso de migración completo, sin embargo, al usar el controlador PID se ve un momento en que el nivel de agua se aleja levemente del la referencia. Esto ocurrió porque al pausar los contenedores necesarios para el control (interfaces XBee y Bluetooth, Filtro y controlador PID) el integrador del controlador no pudo actualizarse junto con las variaciones del nivel de la planta, por lo que produjo un gran salto brusco cuando se retomó la continuidad de operación, con el controlador MPC eso no sucedió, porque una de las restricciones limitaba la variación de la señal de control.

La duración de las pausas que no permiten que el IoT-PLC pueda enviar una nueva señal de control están dadas por el tiempo que se toma en guardar los estados en memoria de los bloques de interfaces XBee y Bluetooth, bloque Filtro y el bloque del controlador. Para el caso del controlador PID el tiempo en que no se puede enviar una nueva señal de control es de 17,5 segundos, este tiempo tiene una fuerte influencia negativa en el control de los niveles de agua. Mientras que para el caso del controlador MPC, este tiempo es de 40,4 segundos y la evaluación muestra que no interfiere de forma importante en el rendimiento del control para esta planta. Este tiempo en que no se puede controlar no es continuo, es decir, entre el final de la pausa de un contenedor y el inicio de otra pausa hay un período en que el IoT-PLC funciona con todos sus componentes activos.

Además, el prototipo demuestra que es posible mantener funcionando de forma continua todos los bloques funcionales que lo componen mientras se controla y se comunica por medio de las interfaces inalámbricas, CoAP y OPC UA.

Los resultados de la evaluación permiten concluir que este prototipo de IoT-PLC puede controlar un sistema mientras realiza un proceso de migración cuando este tiene largas constantes de tiempo, dado que así el efecto es menor en las variables controladas. Sin embargo, para el control de una planta que tenga una dinámica más variable, el prototipo

podría tener problemas al tratar de mantener a las variables controladas cerca del nivel de referencia, principalmente debido a las pausas de varias decenas de segundos y, en menor medida, a la latencia propia de los canales inalámbricos Xbee y Bluetooth que supera los 200 ms y son susceptibles a pérdidas de paquetes por interferencias externas. El problema de latencia originado por la transmisión inalámbrica podría mejorarse, hasta cierto punto, con el uso de tecnologías enfocadas a la industria como WirelessHART e ISA100.11a dado que, estas son más robustas y tienen tiempos de transmisión cercanos a los 100 ms. Una tecnología inalámbrica similar a WirelessHART e ISA100.11a que podría ser útil en el futuro para enfrentar los problemas mencionados es el protocolo 6tisch, el que aun no tiene una presencia en el mercado de dispositivos industriales y tiene entre sus características ser un estándar abierto, permitir conexiones IPv6 de forma nativa y asegurar tiempos de transmisión determinísticos.

Por otro lado, el problema originado por las pausas de contenedores podría resolverse al iniciar un nuevo contenedor igual al que va a ser pausado y enviarle los datos básicos necesarios para que la variable controlada no se aleje de la referencia, para luego de la pausa parar el contenedor nuevo y seguir con el original. De esta manera se podría mantener el funcionamiento del contenedor en su estado mínimo y sin afectar notablemente el rendimiento del control. Entre las dificultades que tiene esta propuesta están las siguientes: se haría a la migración más larga y los recursos computacionales serían más exigidos al tener más contenedores abiertos; se requeriría una gran coordinación entre el contenedor nuevo y el original para no aumentar el tiempo de la pausa, en caso de que el contenedor necesite el uso de recursos limitados como una interfaz UART o Bluetooth; y en el caso del contenedor del controlador la coordinación necesaria sería más compleja, dado que su pausa es crítica en el control del proceso y tener dos controladores operando al mismo tiempo podría desestabilizar al sistema.

El funcionamiento del prototipo de IoT-PLC al operar de manera normal (sin migración de contenedores) muestra que podría ser utilizado en plantas que tengan tiempos de respuestas lo suficientemente largos para que puedan ser controladas con una latencia

total mayor a 150 ms , ya que el IoT-PLC necesita en promedio 30 ms y en más del 99% de los casos menos de 130 ms para procesar la información y enviar una señal de control. Esto supone que se usan medios de comunicación que permitan la obtención de tal latencia, como tecnologías alámbricas del tipo bus de campo, Ethernet o tecnologías como la de redes determinísticas. Esta última, tratada en el Anexo E, permite reservar recursos a las aplicaciones en una red administrada para asegurar cierto nivel de calidad de servicio, bajas tasas de pérdidas de paquetes y tiempos de transmisión determinísticos, características que ayudarían en el proceso de comunicación entre IoT-PLCs durante el intercambio de datos del proceso de control y la transferencia de imágenes de contenedores.

7. CONCLUSIONES

En este trabajo se presentó un diseño de PLC orientado a la Industria 4.0, llamado IoT-PLC, que tiene cabida en las arquitecturas enfocadas en los sistemas de control industriales del futuro. Como evaluación se probó un prototipo del IoT-PLC, al controlar un sistema de cuatro tanques.

Se mostró que es factible ubicar al PLC orientado a la Industria 4.0 como un nodo de capa *fog* que incluya tareas que no están directamente relacionadas con el lazo de control, como procesamiento de datos y exponer a los dispositivos de campo como CPSs, al ser representados como objetos virtuales.

Con el uso de contenedores se logró la incorporación de características útiles en sistemas de control industrial como la adaptación del uso de los recursos del sistema para cada tarea, aislamiento entre los procesos para evitar fallos en cadena y confiabilidad de recuperación en caso de caídas inesperadas de las tareas.

La evaluación muestra que los resultados en el seguimiento de la referencia hacen viable a la arquitectura del IoT-PLC en futuros diseños orientados a la Industria 4.0.

Además, los resultados de la evaluación muestran que la migración de contenedores entre PLCs es una alternativa viable para sistemas con grandes constantes de tiempo, ya que los tiempos en que se detienen los bloques de los que depende el funcionamiento del controlador pueden llegar a los dieciséis segundos.

La arquitectura del IoT-PLC demuestra viabilidad en el control de sistemas que requieran de latencias del orden de 30 ms, siempre y cuando las interfaces de comunicación utilizadas lo permitan.

Como trabajo futuro se propone la inclusión de un mecanismo que replique los contenedores de forma selectiva que van a ser pausados para obtener su estado en memoria, con el fin de mitigar los efectos adversos que tienen las pausas de algunos en el

rendimiento del controlador. En una primera etapa se sugiere la creación de un contenedor hijo que reciba el valor de las variables fundamentales desde el contenedor original antes de que este sea pausado.

También, se sugiere el diseño de un prototipo destinado a operar en entornos industriales, incluyendo el *hardware* y *software* para certificar que la arquitectura propuesta cumpla con los requisitos de latencia y confiabilidad que exige la industria.

REFERENCIAS

Amghar, S., Cherdal, S., & Mouline, S. (2018). Which nosql database for iot applications? In *2018 international conference on selected topics in mobile and wireless networking (mownet)* (p. 131-137).

Anybus. (2018). Industrial ethernet is now bigger than fieldbuses. Retrieved from `https://www.anybus.com/about-us/news/2018/02/16/industrial-ethernet-is-now-bigger-than-fieldbuses`

Azarmipour, M., Elfaham, H., Gries, C., & Epple, U. (2019, Oct). Plc 4.0: A control system for industry 4.0. In *Iecon 2019 - 45th annual conference of the ieee industrial electronics society* (Vol. 1, p. 5513-5518).

Badar, A., Lou, D. Z., Graf, U., Barth, C., & Stich, C. (2019, Oct). Intelligent edge control with deterministic-ip based industrial communication in process automation. In *2019 15th international conference on network and service management (cnsm)* (p. 1-7).

Bangemann, T., Riedl, M., Thron, M., & Diedrich, C. (2016, May). Integration of classical components into industrial cyber–physical systems. *Proceedings of the IEEE*, *104*(5), 947-959.

Bellavista, P., & Zanni, A. (2016, Sep.). Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP. In *2016 ieee 2nd international forum on research and technologies for society and industry leveraging a better tomorrow (rtsi)* (p. 1-6).

Candell, R., Kashef, M., Liu, Y., Lee, K. B., & Foufou, S. (2018). Industrial wireless systems guidelines: Practical considerations and deployment life cycle. *IEEE Industrial Electronics Magazine*, *12*(4), 6-17.

Chen, B., Wan, J., Celesti, A., Li, D., Abbas, H., & Zhang, Q. (2018, Sept). Edge computing in iot-based manufacturing. *IEEE Communications Magazine*, *56*(9), 103-109.

Chiang, M., & Zhang, T. (2016, Dec). Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, *3*(6), 854-864.

CISCO. (2014). The internet of things reference model.

Colombo, A. W., Karnouskos, S., Kaynak, O., Shi, Y., & Yin, S. (2017). Industrial cyberphysical systems: A backbone of the fourth industrial revolution. *IEEE Industrial Electronics Magazine*, *11*(1), 6-16.

Consortium, I. I. (2017). The industrial internet of things volume g1: Reference architecture..

Cook, J. (2017). Docker for data science. In *Docker for data science.* Berkeley, CA: Apress.

Delsing, J. (2017). Local cloud internet of things automation: Technology and business model features of distributed internet of things automation solutions. *IEEE Industrial Electronics Magazine*, *11*(4), 8-21.

Dolui, K., & Kiraly, C. (2018, Dec). Towards multi-container deployment on IoT gateways. In *2018 ieee global communications conference (globecom)* (p. 1-7).

Faruque, M. A. A., & Vatanparvar, K. (2016, April). Energy management-as-a-service over fog computing platform. *IEEE Internet of Things Journal*, *3*(2), 161-169.

Ferrer, B. R., Mohammed, W. M., Chen, E., & Lastra, J. L. M. (2017, Oct). Connecting web-based IoT devices to a cloud-based manufacturing platform. In *Iecon 2017 - 43rd annual conference of the ieee industrial electronics society* (p. 8628-8633).

Freeman, A. (2017). Essential docker for asp.net core mvc. In *Essential docker for asp.net core mvc.* Berkeley, CA: Apress.

Galloway, B., & Hancke, G. P. (2013, Second). Introduction to industrial control networks. *IEEE Communications Surveys Tutorials*, *15*(2), 860-880.

Gangakhedkar, S., Cao, H., Ali, A. R., Ganesan, K., Gharba, M., & Eichinger, J. (2018). Use cases, requirements and challenges of 5g communication for industrial automation. In *2018 ieee international conference on communications workshops (icc workshops)* (p. 1-6).

Garcia, C. A., Salinas, G., Perez, V. M., Salazar L., F., & Garcia, M. V. (2019). Robotic arm manipulation under iec 61499 and ros-based compatible control scheme. In M. Botto-Tobar, L. Barba-Maggi, J. González-Huerta, P. Villacrés-Cevallos, O. S. Gómez, & M. I. Uvidia-Fassler (Eds.), *Information and communication technologies of ecuador (tic.ec)* (pp. 358–371). Springer International Publishing.

Goldschmidt, T., Murugaiah, M. K., Sonntag, C., Schlich, B., Biallas, S., & Weber, P. (2015, June). Cloud-based control: A multi-tenant, horizontally scalable soft-PLC. In *2015 ieee 8th international conference on cloud computing* (p. 909-916).

Habib, G., Haddad, N., & Khoury, R. E. (2015). Case study: Wirelesshart vs zigbee network. In *2015 third international conference on technological advances in electrical, electronics and computer engineering (taeece)* (p. 135-138).

Harrison, R., Vera, D., & Ahmad, B. (2016, May). Engineering methods and tools for cyber–physical automation systems. *Proceedings of the IEEE*, *104*(5), 973-985.

Holt, A., & Huang, C.-Y. (2018). Containers. In *Embedded operating systems: A practical approach* (pp. 41–67). Cham: Springer International Publishing.

Jangla, K. (2018). Containers. In *Accelerating development velocity using docker: Docker across microservices* (pp. 1–8). Berkeley, CA: Apress.

Jasperneite, J., Sauter, T., & Wollschlaeger, M. (2020). Why we need automation models: Handling complexity in industry 4.0 and the internet of things. *IEEE Industrial Electronics Magazine*, *14*(1), 29-40.

Johansson, K. H. (2000, May). The quadruple-tank process: a multivariable laboratory process with an adjustable zero. *IEEE Transactions on Control Systems Technology*, *8*(3), 456-465.

John, K. H., & Tiegelkamp, M. (2010). Concepts and programming languages, requirements for programming systems, decision-making aids. In *Iec 61131-3: Programming industrial automation systems.*

Kim, D.-S., & Tran-Dang, H. (2019). Wireless sensor networks for industrial applications. In *Industrial sensors and controls in communication networks: From wired technologies to cloud computing and the internet of things* (pp. 127–140). Cham: Springer International Publishing.

Kristiani, E., Yang, C.-T., Wang, Y. T., & Huang, C.-Y. (2019). Implementation of an edge computing architecture using openstack and kubernetes. In K. J. Kim & N. Baek (Eds.), *Information science and applications 2018* (pp. 675–685). Singapore: Springer Singapore.

Kämäräinen, T., Shan, Y., Siekkinen, M., & Ylä-Jääski, A. (2015). Virtual machines vs. containers in cloud gaming systems. In *2015 international workshop on network and systems support for games (netgames)* (p. 1-6).

Lan, L., Shi, R., Wang, B., & Zhang, L. (2019). An IoT unified access platform for heterogeneity sensing devices based on edge computing. *IEEE Access*, *7*, 44199-44211.

Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., & Colombo, A. W. (2016, May). Smart agents in industrial cyber–physical systems. *Proceedings of the IEEE*, *104*(5), 1086-1101.

Lennvall, T., Svensson, S., & Hekland, F. (2008). A comparison of wirelesshart and zigbee for industrial applications. In *2008 ieee international workshop on factory communication systems* (p. 85-88).

Luo, Z., Hong, S., Lu, R., Li, Y., Zhang, X., Kim, J., ... Liang, W. (2017, Sep.). Opc ua-based smart manufacturing: System architecture, implementation, and execution. In *2017 5th international conference on enterprise systems (es)* (p. 281-286).

Machen, A., Wang, S., Leung, K. K., Ko, B. J., & Salonidis, T. (2018, February). Live service migration in mobile edge clouds. *IEEE Wireless Communications*, *25*(1), 140-147.

Meier, U., Witte, S., Helmig, K., Hoing, M., Schnuckel, M., & Krause, H. (2007). Performance evaluation and prediction of a bluetooth based real-time sensor actuator system in harsh industrial environments. In *2007 ieee conference on emerging technologies and factory automation (efta 2007)* (p. 33-37).

Moghaddam, M. H. Y., & Leon-Garcia, A. (2018, April). A fog-based internet of energy architecture for transactive energy management systems. *IEEE Internet of Things Journal*, *5*(2), 1055-1069.

Morabito, R., Kjällman, J., & Komu, M. (2015). Hypervisors vs. lightweight virtualization: A performance comparison. In *2015 ieee international conference on cloud engineering* (p. 386-393).

Ni, J., Zhang, A., Lin, X., & Shen, X. S. (2017, June). Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine*, *55*(6), 146-152.

Núñez, F., Langarica, S., Díaz, P., Torres, M., & Salas, J. C. (2020). Neural network-based model predictive control of a paste thickener over an industrial internet platform. *IEEE Transactions on Industrial Informatics*, *16*(4), 1-9.

Pang, Z., Luvisotto, M., & Dzung, D. (2017). Wireless high-performance communications: The challenges and opportunities of a new target. *IEEE Industrial Electronics*

Magazine, *11*(3), 20-25.

Popović, I. T., & Rakić, A. Z. (2018, Nov). The fog-based framework for design of real-time control systems in internet of things environment. In *2018 international symposium on industrial electronics (indel)* (p. 1-6).

Ramos, B., Savazzi, S., Winter, J. M., Ojeda, V., Chalen, M., Rosario, E. D., & Á lvarez, M. A. (2018). A perfomance comparison of wirelesshart and zigbee in oil refinery. In *2018 ieee-aps topical conference on antennas and propagation in wireless communications (apwc)* (p. 846-849).

Sahin, D., & Ammari, H. M. (2014). Network architectures and standards. In H. M. Ammari (Ed.), *The art of wireless sensor networks: Volume 1: Fundamentals* (pp. 789–827). Berlin, Heidelberg: Springer Berlin Heidelberg.

Salazar, L. A. C., & Alvarado, O. A. R. (2014, Oct). The future of industrial automation and iec 614993 standard. In *2014 iii international congress of engineering mechatronics and automation (ciima)* (p. 1-5).

Schulte, D., & Colombo, A. W. (2017, Oct). Rami 4.0 based digitalization of an industrial plate extruder system: Technical and infrastructural challenges. In *Iecon 2017 - 43rd annual conference of the ieee industrial electronics society* (p. 3506-3511).

Tanenbaum, A. S., & Renesse, R. V. (1992). *Modern operating systems.*

Tang, B., Chen, Z., Hefferman, G., Pei, S., Wei, T., He, H., & Yang, Q. (2017, Oct). Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Transactions on Industrial Informatics*, *13*(5), 2140-2150.

Varun, & Thakur, R. (2015, 12). The role of modular programming in industrial control system. *International Research Journal of Engineering and Technology 2395-0056*, *2*, 1152-1157.

Vilajosana, X., Watteyne, T., Chang, T., Vučinić, M., Duquennoy, S., & Thubert, P. (2020). Ietf 6tisch: A tutorial. *IEEE Communications Surveys Tutorials*, *22*(1), 595-615.

Vohra, D. (2016). Kubernetes microservices with docker. In *Kubernetes microservices with docker.* Berkeley, CA: Apress.

Vohra, D. (2017). Getting started with docker. In *Docker management design patterns: Swarm mode on amazon web services* (pp. 1–7). Berkeley, CA: Apress.

Wollschlaeger, M., Sauter, T., & Jasperneite, J. (2017). The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, *11*(1), 17-27.

Yao, J., Xu, X., & Liu, X. (2016, May). Mixcps: Mixed time/event-triggered architecture of cyber–physical systems. *Proceedings of the IEEE*, *104*(5), 923-937.

Zeydan, E., Bastug, E., Bennis, M., Kader, M. A., Karatepe, I. A., Er, A. S., & Debbah, M. (2016, September). Big data caching for networking: moving from cloud to edge. *IEEE Communications Magazine*, *54*(9), 36-42.

Zhang, H., Yan, Q., & Wen, Z. (2020). Information modeling for cyber-physical production system based on digital twin and automationml. In *The international journal of advanced manufacturing technology.*

Zhang, Y., Zhu, Z., & Lv, J. (2018, April). Cps-based smart control model for shopfloor material handling. *IEEE Transactions on Industrial Informatics*, *14*(4), 1764-1775.

ANEXO

A. PROTOCOLOS HART Y CANOPEN

A.1. Highway Addressable Remote Transducer Protocol (HART)

El Protocolo de Comunicación HART (Highway Addressable Remote Transducer) es un protocolo abierto de automatización industrial híbrido analógico-digital. Su ventaja más notable es que puede comunicarse a través de los lazos de corriente de instrumentación analógica 4-20 mA, al compartir el par de cables utilizados por los sistemas anfitriones únicamente analógicos. HART se utiliza ampliamente en sistemas de procesos e instrumentación que van desde pequeñas aplicaciones de automatización hasta aplicaciones industriales muy sofisticadas. El protocolo HART hace uso del estándar Bell 202 Frequency Shift Keying (FSK) para superponer señales de comunicación digital a bajo nivel sobre el 4-20mA. El estándar FSK permite la comunicación bidireccional a través de un circuito de 2 hilos de 4-20mA sin interrumpir la señal analógica.

La tecnología HART ofrece una solución para los operadores de plantas que buscan las ventajas de los dispositivos inteligentes con comunicación digital, a la vez que preservan las inversiones existentes en instrumentación analógica y cableado de plantas.

A.2. CANopen

CANopen es un protocolo de comunicación de alto nivel y una especificación de perfil de dispositivos que se basa en el protocolo CAN (*Controller Area Network*). El protocolo fue desarrollado para los sistemas incorporados utilizados en la automatización, CANopen abarca un marco de programación de redes, descripciones de dispositivos, definiciones de interfaz y perfiles de aplicación. Además, CANopen proporciona un protocolo que estandariza la comunicación entre dispositivos y aplicaciones de diferentes fabricantes, por esto se ha utilizado en una amplia gama de industrias, destacándose en las aplicaciones de automatización y movimiento.

En cuanto a CAN, este protocolo cubre los niveles de la capa física y la capa de enlace de datos. La capa física define las líneas utilizadas, los voltajes, la naturaleza de alta velocidad, etc. La capa de enlace de datos incluye el hecho de que CAN es un protocolo basado en marcos (mensajes). CANopen abarca las cinco capas superiores a las que especifica CAN: red (direccionamiento, enrutamiento), transporte (fiabilidad de extremo a extremo), sesión (sincronización), presentación (datos codificados de forma estándar, representación de datos) y aplicación. La capa de aplicación describe cómo configurar, transferir y sincronizar los dispositivos CANopen.

B. APLICACIONES PRÁCTICAS DEL PARADIGMA FOG EN LA LITERATURA

En la literatura el paradigma *fog* se ha utilizado para monitorear variados tipos de sistemas, ya que al estar cerca del borde de la red otorga varias ventajas, entre ellas está el permitir que los nodos que actúan en la capa *fog* de un modelo estén descentralizados y, por lo tanto, se puedan generar zonas autónomas en caso de alguna falla en capas superiores y el obtener tiempos de respuesta cortos frente a anomalías. A continuación se presentan algunas de sus aplicaciones descritas en la literatura.

B.1. Manejo de materiales en zona de producción

En (Y. Zhang, Zhu, & Lv, 2018) se plantea una arquitectura para mejorar la eficiencia de los vehículos automatizados que se utilizan en las fábricas para transportar materiales, en ella existen tres capas: la primera es la capa física que incluye al vehículo, sus sensores, elementos que le dan conectividad y maquinaria en el entorno, la segunda es la capa cibernética que incluye procesamiento en tiempo real de los datos que los vehículos entregan módulos que están cercanos a ellos y la tercera capa consiste en la de sistema, que es en donde están las aplicaciones que supervisan el servicio para evitar choques, asignar tareas y hacer seguimiento de vehículos, la Figura B.1 muestra el esquema de la arquitectura.

En esta arquitectura la capa con características de capa *fog* es la segunda, porque toma los datos y los procesa antes de enviarlo hacia las capas superiores.

Los resultados mostraron una mejora en la eficiencia del transporte, ya que se disminuyeron los espacios de trabajo entre vehículos, se alcanzaron mejores estrategias anticolisiones y en las intersecciones se superaban más rápido.

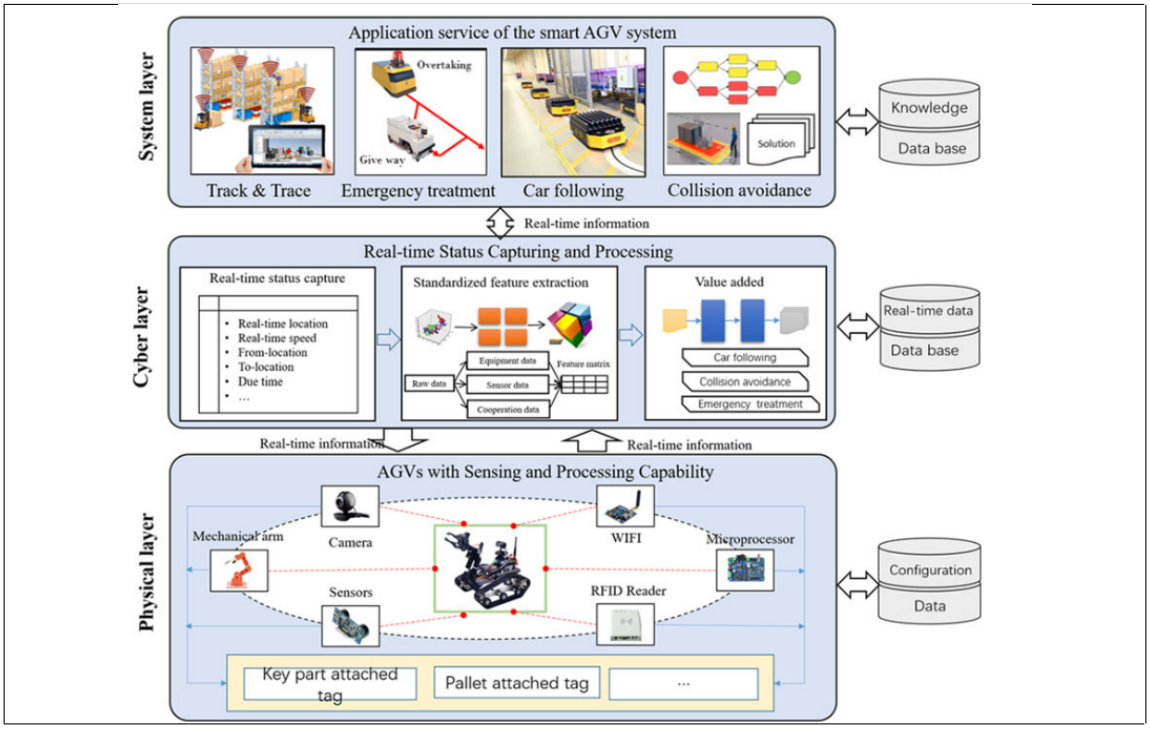


Figura B.1. Arquitectura propuesta para el manejo de materiales en zona de producción (Y. Zhang et al., 2018).

B.2. Caché en redes

En (Zeydan et al., 2016) se propone una arquitectura para la infraestructura de un operador de redes móviles que permita guardar el contenido más solicitado en el caché de las estaciones base, para disminuir la carga en el núcleo de la infraestructura del operador. La arquitectura utiliza las plataformas de análisis de *big data* que posee el operador móvil para determinar que contenidos deben ser guardados en las unidades de almacenaje en las estaciones base de forma proactiva, cuando un contenido debe ser guardado se lo informa a la estación base y esta realiza las operaciones de guardado y distribución local cuando se demande un contenido. De esta forma se llevaron parte de las tareas hacia el borde de la red, en este caso las estaciones base pueden ser reconocidas como nodos en la capa *fog*. Las conexiones entre elementos de la arquitectura pueden verse en la Figura B.2.

En las simulaciones se utilizó *Hadoop Distributed File System* (HDFS) para la selección, limpieza y análisis de los datos de un operador móvil real, el sistema de selección proactivo estaba a cargo de técnicas de aprendizaje de máquina. Los resultados obtenidos indican que se logra un alto porcentaje de satisfacción del usuario al utilizar la nueva arquitectura.

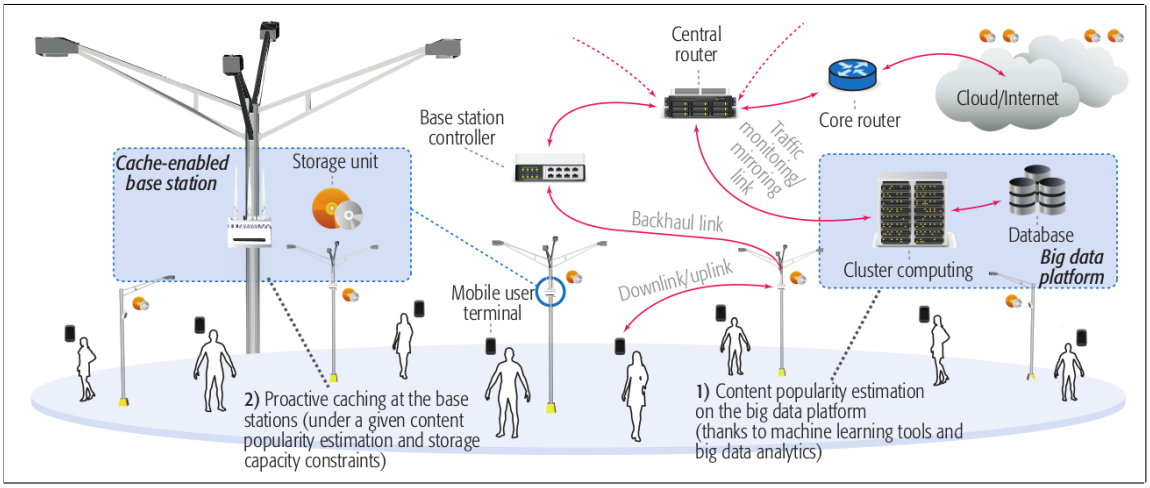


Figura B.2. Arquitectura propuesta para uso de caché en redes (Zeydan et al., 2016).

B.3. Recolección masiva de datos vehiculares

En (Ni, Zhang, Lin, & Shen, 2017) se formula la recolección de los datos de sensores de vehículos por medio de una arquitectura de tres capas, tal como se presenta en la Figura B.3: la primera es de los vehículos con sus sensores que recolectan datos continuamente, la segunda es la capa *fog* que tiene nodos distribuidos geográficamente y recibe los datos de los vehículos, los filtra y agrega para generar reportes y luego los envía a la tercera capa, la cual corresponde a la nube, en esta capa se realizan análisis para asignar beneficios a los vehículos si es que cumplen determinadas tareas, que en el fondo permiten mejorar el tráfico vehicular y disminuir el tiempo de estacionamiento. Para asegurar la confidencialidad y privacidad de los datos que se comparten se incluyen métodos de cifrado de datos utilizando agentes de confianza.

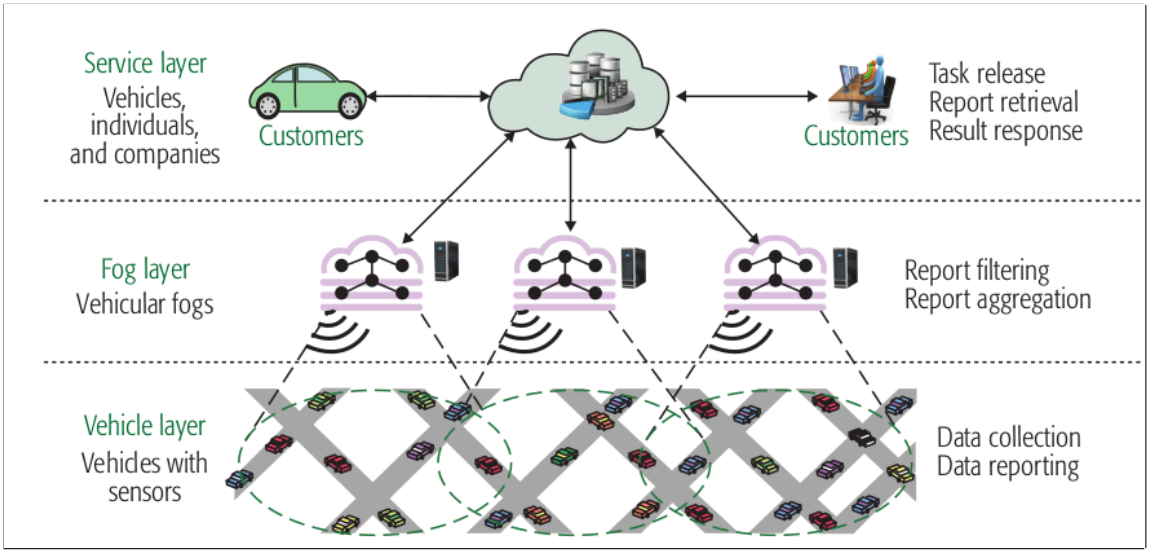


Figura B.3. Arquitectura propuesta para recolección masiva de datos vehiculares (Ni et al., 2017).

B.4. Manejo de energía

En (Faruque & Vatanparvar, 2016) se propone un servicio de manejo de energía que utiliza a la capa *fog*, para ello implementan un arquitectura en que los sensores se conectan a través de puntos de accesos a un *gateway*, a este se conecta el calefactor y el sistema de aire acondicionado directamente. Toda la información pasa del *gateway* al administrador de energía del hogar (nodo *fog*) que monitorea la energía consumida y las condiciones físicas del entorno (temperatura, humedad, luz, etc.) y entrega comandos de control al calefactor y al sistema de aire acondicionado, según lo que indique el algoritmo que incorpora, luego desde el nodo *fog* se conecta a internet para que toda la información esté disponible de forma remota. En la Figura B.4 se ve la estructura general de la arquitectura.

En (Moghaddam & Leon-Garcia, 2018) se propone algo similar al sistema anterior, pero a nivel más amplio, ya que en esta arquitectura se toma la información de varios consumidores de energía. La arquitectura propuesta se muestra en la Figura B.5, esta tiene tres capas: la primera involucra a los sensores en los hogares o instalaciones de los clientes que miden el consumo eléctrico y envían la información a los nodos *fog*, estos sensores se conectan a través de un *gateway* utilizando el protocolo OpenADR, la segunda

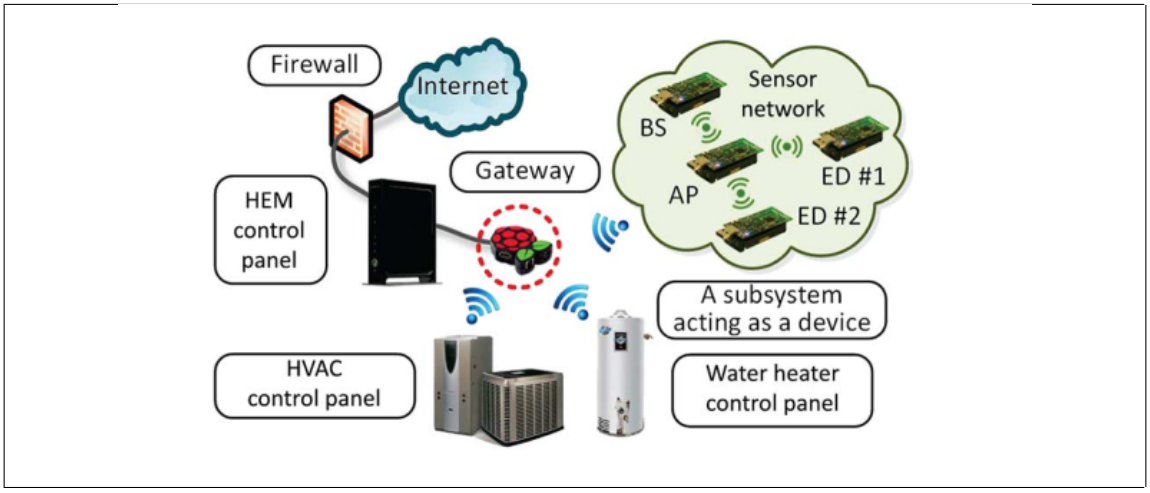


Figura B.4. Primera arquitectura propuesta para el manejo de energía (Faruque & Vatanparvar, 2016).

capa es la *fog* y se encarga de filtrar la información que se envía a la nube, de almacenar datos de estadísticas de los usuarios para entregar un servicio con baja latencia a través de LANs y actúan como interfaz entre los proveedores de energía y los clientes. La tercera capa es la relacionada con la nube, aquí se integra la información de los precios de la energía con la de los datos de los clientes y los resultados de los análisis se envían hacia los nodos *fog* para que los consumidores puedan alcanzar un consumo de energía óptimo en un periodo de tiempo cercano.

B.5. Procesos de fabricación

En (Chen et al., 2018) *fog* se ha utilizado para permitir que los procesos de fabricación puedan compartir información con bajos tiempos de retardo, mientras se utilizan las funciones de la nube para la toma de decisiones a nivel general. La estructura propuesta tiene tres capas: la primera es la de los sensores, actuadores y controladores que transmiten información en los protocolos OPC UA y EtherCAT hacia los dominios IT, a la vez que pueden comunicarse con los dispositivos dentro de la misma capa, la segunda capa es la de red que tiene la función de llevar los datos hacia los operadores de las instalaciones y

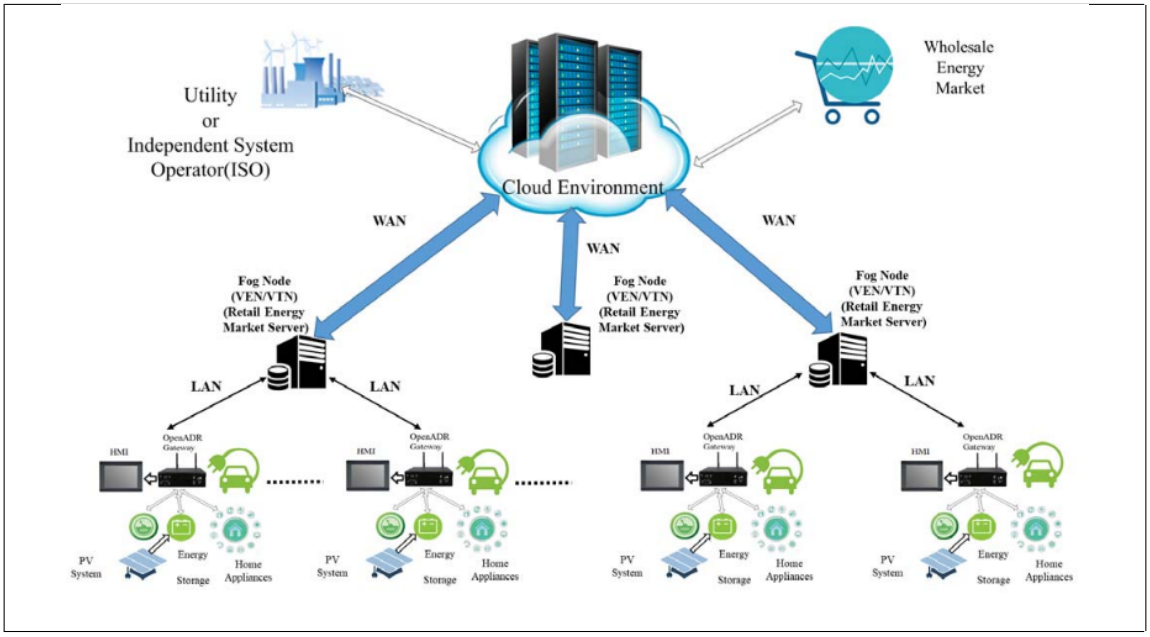


Figura B.5. Segunda arquitectura propuesta para el manejo de energía (Moghaddam & Leon-Garcia, 2018).

están cerca del borde de la red, además, transmiten la información a los centros de cómputo que realizan análisis, por último, la tercera capa es la del dominio de datos, la cual provee de servicios de procesamiento de datos, tales como limpieza de datos, extracción de características y respuestas en tiempo real basadas en datos provenientes de la primera capa. La estructura general de la arquitectura se muestra en la Figura B.6. Con la inclusión de esta arquitectura se logró disminuir los tiempos de operación total de cada robot en la instalación, junto con bajar la tasa de transmisión que había en la red.

B.6. Ciudades Inteligentes

En (Tang et al., 2017) se propone una arquitectura, presentada en la Figura B.7, que tiene como objetivo mejorar los tiempos de respuesta ante eventos de emergencia al usar la capa *fog*, tal arquitectura tiene cuatro capas: la primera corresponde a la que tiene sensores geodistribuidos por la ciudad y que recopilan información en el tiempo, la segunda capa tiene nodos *fog* que analizan los datos de pequeñas comunidades, aquí se procesan los datos de los sensores y se encarga de detectar anomalías con un bajo tiempo de respuesta

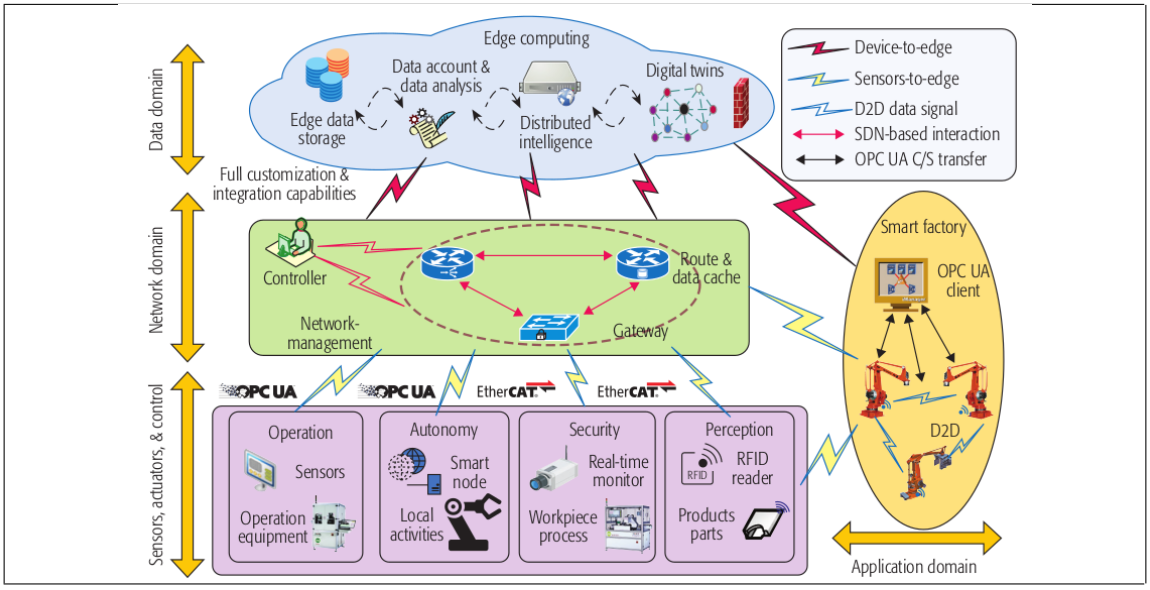


Figura B.6. Arquitectura propuesta para procesos de fabricación (Chen et al., 2018).

(segundos), la tercera capa realiza la misma tarea, pero cada nodo funciona en un área mayor para detectar anomalías, la última capa es la nube, la que recibe los datos de la capa tres y los analiza para generar estadísticas que puedan beneficiar a la ciudad, como optimización del funcionamiento de semáforos o niveles de iluminación.

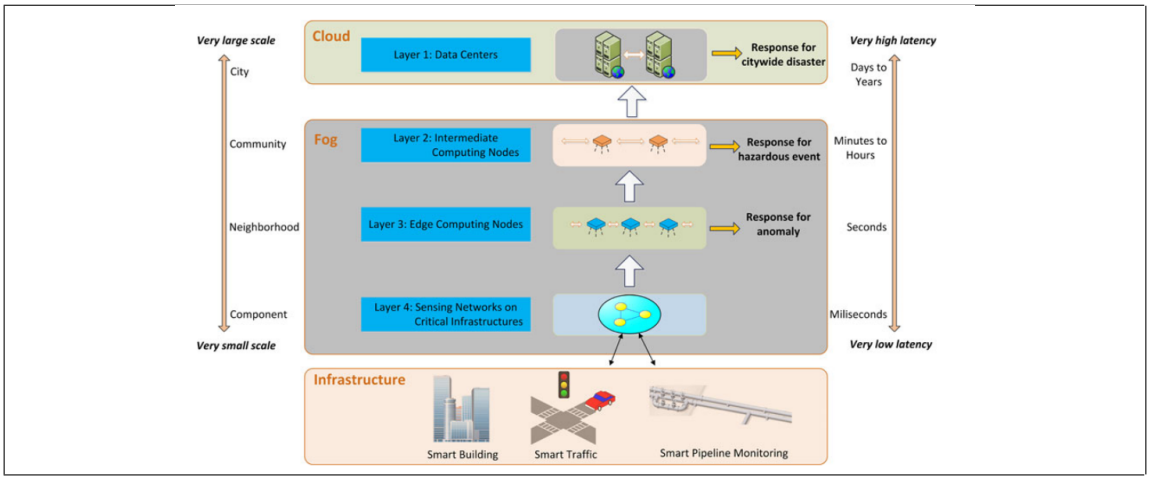


Figura B.7. Arquitectura propuesta para ciudades inteligentes (Tang et al., 2017).

C. PLATAFORMA DE ADMINISTRACIÓN DE CONTENEDORES: KUBERNETES

Tal como Docker es la plataforma más extendida para iniciar procesos en contenedores, el *software* Kubernetes lo es en lo que se refiere a la gestión de grupos de contenedores, y para su utilización es recomendado que Docker esté instalado junto con él. La orquestación de Kubernetes incluye la programación, la distribución de la carga de trabajo y el escalado.

Kubernetes lleva más allá el encapsulamiento de software proporcionado por Docker al introducir los Pods. Un Pod es una colección de uno o más contenedores Docker con características de interfaz única, como proporcionar una red y un sistema de archivos a nivel de Pod en lugar de a nivel de contenedor.

Kubernetes también introduce "etiquetas" (o *labels*) con las que los servicios y los controladores de replicación (el controlador de replicación se utiliza para escalar un grupo) identifican o seleccionan los contenedores o pods que gestionan. Kubernetes es ligero, portátil (adecuado para la arquitectura de nube), modular y puede ejecutarse en casi cualquier plataforma. La estructura general puede apreciarse en la Figura C.1.

La necesidad de tener un orquestador como Kubernetes aparece con el uso de contenedores en los sistemas de producción, ya que ahí se ponen de manifiesto problemas prácticos como qué contenedor debe funcionar en qué nodo, cómo aumentar/disminuir el número de contenedores en funcionamiento para una aplicación (escalado) y cómo comunicarse dentro de los contenedores. Kubernetes fue diseñado para superar todos estos y otros problemas prácticos de la gestión de agrupaciones de contenedores. Kubernetes proporciona una orquestación dinámica de agrupaciones de contenedores en tiempo real, que como orquestador proporciona las siguientes ventajas (Vohra, 2016).

- Facilitar la utilización de microservicios, al dividir una aplicación en componentes más pequeños, manejables y escalables que podrían ser utilizados por grupos con diferentes requisitos.

- Desplegar conjuntos de contenedores tolerantes a los fallos, en el que si una sola réplica de un Pod falla (debido a un fallo en un nodo, por ejemplo), otra se inicia automáticamente.
- Permitir el escalamiento horizontal en el que se podrían ejecutar réplicas adicionales o en menor cantidad de un Pod con sólo modificar el parámetro "réplicas" en el dispositivo maestro.
- Mayor utilización y eficiencia de los recursos.

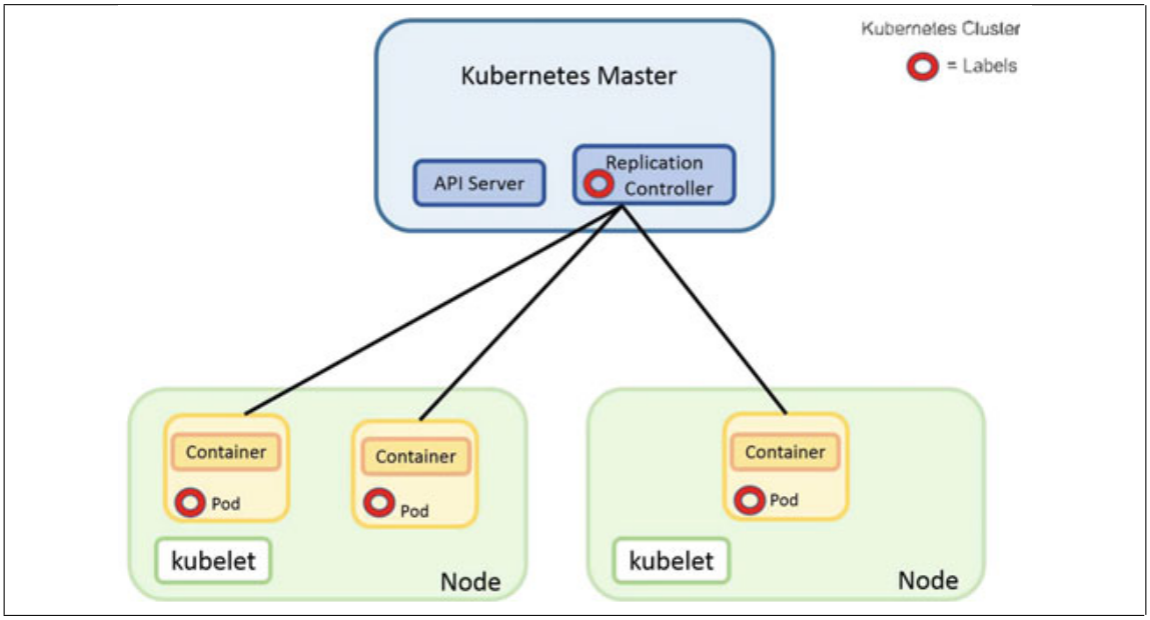


Figura C.1. Estructura general de sistemas orquestados por Kubernetes (Kristiani et al., 2019).

D. DMBS NOSQL: NEO4J Y CASSANDRA

A continuación se describirán dos de las DBMSs más populares, una perteneciente a la categoría de bases de estructuradas en grafos y la otra a bases de datos orientada a las columnas.

D.1. Neo4j

Neo4j es una base de datos estructurada en grafos de código abierto creada por Neo Technology, Inc. en 2007. Debido a su modelo de grafos, Neo4j no puede ser distribuida a través de nodos. Además, el grafo se replica con la arquitectura maestro-esclavo. Neo4j tiene un tipo de escalabilidad de alta disponibilidad, la que permite tres cosas: redundancia total de datos, tolerancia a fallos de servicio y escalabilidad lineal de lectura.

Entre las características de esta base de datos están la posibilidad de realizar operaciones en tiempo real, proporciona características de seguridad como la autenticación para asegurar que un usuario es quien dice ser, soporta funcionalidades geoespaciales, por lo que puede ser usado para análisis de rutas o búsquedas de proximidad proporcionando la librería Neo4j-Spatial.

Con Neo4j se pueden utilizar dos lenguajes de consulta diferentes para acceder a los datos de Neo4j, Cypher, que es declarativo y un poco similar a SQL, y el lenguaje transversal de gráficos de bajo nivel Gremlin.

D.2. Cassandra

Cassandra es una base de datos distribuida de código abierto orientada a las columnas, al ser una base de datos orientada a columnas, los datos se almacenan en forma de filas y columnas. Cada fila puede contener un número diferente de columnas, pero las familias de columnas se fijan cuando se crea una base de datos. Cassandra tiene un componente interno llamado particionador, que determina cómo se distribuyen los datos en los nodos,

esta base de datos tiene una arquitectura de anillo sin maestro, es decir, todos los nodos juegan el mismo papel y que los datos se replican usando el modelo maestro-maestro.

La arquitectura sin maestro de Cassandra hace que la escalabilidad sea más fácil de obtener, dado que solo se necesita añadir un nuevo nodo. En Cassandra se propone una técnica de geohazing para permitir consultas espaciales como extensión. Sin embargo, el manejo de datos espaciales sigue siendo una característica que falta en Cassandra. En cuanto a seguridad, se proporcionan opciones de encriptación para los datos que se envían desde un cliente a un grupo de base de datos, así como para las comunicaciones de nodo a nodo.

E. REDES DETERMINÍSTICAS

La arquitectura general de una red determinístca (DetNet) ofrece la posibilidad de transportar determinados flujos de datos del tipo unidifusión o multidifusión para aplicaciones en tiempo real con tasas de pérdida de datos extremadamente bajas y latencia limitada dentro de un dominio de red. DetNet es para redes que están bajo un único control administrativo o dentro de un grupo cerrado de control administrativo; entre ellas se incluyen las redes de todo el campus y las redes de área extendida privadas, por lo que DetNet no es para grandes grupos de dominios como la Internet.

Los objetivos de DetNet son 1) permitir la migración de aplicaciones con problemas críticos de tiempo y fiabilidad que actualmente utilizan tecnologías de bus de campo de propósito especial (por ejemplo, Interfaz Multimedia de Alta Definición (HDMI), Red de Área de Controladores (bus CAN), PROFIBUS o RS-232) a tecnologías de paquetes en general y a IP en particular y 2) soportar tanto estas nuevas aplicaciones como las aplicaciones de red de paquetes existentes sobre la misma red física. En otras palabras, una DetNet es compatible con el tráfico (capaz de transportar) estadísticamente multiplexado, preservando las propiedades de los flujos determinista aceptados.

Las técnicas utilizadas incluyen 1) reservar recursos del planos de datos para flujos DetNet individuales (o agregados) en algunos o todos los nodos intermedios a lo largo del trayecto del flujo, 2) proporcionar rutas explícitas para los flujos DetNet que no cambien inmediatamente con la topología de la red, y 3) distribuir los datos de los paquetes de flujo DetNet en el tiempo y/o el espacio para asegurar la entrega de los datos de cada paquete a pesar de la pérdida de un trayecto.

DetNet opera en la capa IP y presta servicio a través de tecnologías de capas más bajas como MPLS y IEEE 802.1 Time-Sensitive Networking (TSN). DetNet proporciona un servicio fiable y disponible dedicando recursos de red como el ancho de banda del enlace y el espacio del búfer a los flujos de DetNet y/o a las clases de flujos de DetNet, y

replicando los paquetes a lo largo de múltiples rutas. Los recursos reservados no utilizados están disponibles para los paquetes no DetNet siempre que se cumplan todas las garantías.

La asignación de recursos funciona asignando recursos, por ejemplo, espacio de memoria intermedia o ancho de banda del enlace, a un flujo DetNet (o agregado de flujo) a lo largo de su trayectoria. La asignación de recursos reduce en gran medida, o incluso elimina por completo, la pérdida de paquetes debido a la contención de paquetes de salida dentro de la red, pero sólo puede suministrarse a un flujo DetNet que esté limitado en la fuente a un tamaño de paquete y una velocidad de transmisión máximos.

La presencia de flujos DetNet no impide flujos no DetNet, y las ventajas que ofrecen los flujos DetNet no deben, salvo en casos extremos, impedir que los mecanismos de QoS existentes funcionen de manera normal, con sujeción al ancho de banda necesario para los flujos DetNet. Toda aplicación que genere un flujo de datos que pueda caracterizarse útilmente por tener un ancho de banda máximo debería poder aprovechar las ventajas de DetNet, siempre que se puedan reservar los recursos necesarios. Las reservas pueden hacerse por la propia aplicación, mediante la gestión de la red, de forma centralizada por un controlador de la aplicación, o por otros medios, por ejemplo, haciendo una reserva a la carta mediante un plano de control distribuido, por ejemplo, aprovechando el protocolo de reserva de recursos (RSVP). Los requisitos de calidad de servicio de los flujos DetNet pueden cumplirse si todos los nodos de la red en un dominio DetNet implementan las capacidades DetNet.

Los objetivos principales de la QoS DetNet pueden expresarse en términos de:

- Mínimizar y máximizar la latencia de extremo a extremo desde la fuente hasta el destino con un entrega puntual y fluctuación acotada (variación de la demora de los paquetes) derivada de estas limitaciones.
- Tasa de pérdida de paquetes bajo varios supuestos en cuanto a los estados operacionales de los nodos y enlaces.

- Límite superior en la entrega de paquetes fuera de orden (paquetes enviados primero que otros deben llegar primero). Cabe señalar que algunas aplicaciones de DetNet no pueden tolerar ninguna entrega fuera de orden.

El principal medio por el que DetNet logra sus garantías de calidad de servicio es reduciendo, o incluso eliminando completamente, la pérdida de paquetes debido a la contención de los paquetes de salida dentro de un nodo DetNet como causa de la pérdida de paquetes. Esto sólo puede lograrse mediante la provisión de suficiente almacenamiento intermedio en cada nodo de la red para garantizar que no se pierdan paquetes por falta de almacenamiento intermedio.

Las aplicaciones que están diseñadas para funcionar en enlaces en serie normalmente no proporcionan servicios para recuperar la variabilidad en tiempos de transmisión, porque la variabilidad simplemente no existe allí. Por lo general, se espera que los flujos de DetNet se entreguen en orden, y la hora precisa de recepción influye en los procesos. Para hacer converger esas aplicaciones existentes, se desea emular todas las propiedades del cable serie, como el transporte por reloj, el perfecto aislamiento del flujo y la latencia fija. Si bien DetNet soporta una fluctuación mínima (en forma de especificar una latencia mínima, así como máxima, de extremo a extremo), existen limitaciones prácticas en las redes basadas en paquetes a este respecto. En general, se alienta a los usuarios a que utilicen una combinación de sincronización temporal de submicrosegundos entre todos los sistemas finales de origen y destino, y campos de tiempo de ejecución en los paquetes de cada aplicación.