
**UNIVERSIDAD TÉCNICA DE AMBATO**

**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL**

**CARRERA DE INGENIERÍA INDUSTRIAL EN PROCESOS DE AUTOMATIZACIÓN**

**Tema:**

# DISEÑO DE COMUNICACIÓN BASADO EN CONTENEDORES PARA LA INDUSTRIA 4.0

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado para la obtención del título de Ingeniero Industrial en Procesos de Automatización.

**ÁREA:** Industrial y manufactura

**LÍNEAS DE INVESTIGACIÓN:** Tecnología de la Información y Sistemas de Control

**AUTOR:** Christian Emilio Puca Morales

**TUTOR:** Ing. Marcelo García, PhD

Ambato – Ecuador

marzo – 2023

## APROBACIÓN DEL TUTOR

En mi calidad de tutor del Trabajo de Titulación con el tema: DISEÑO DE COMUNICACIÓN BASADO EN CONTENEDORES PARA LA INDUSTRIA 4.0, desarrollado bajo la modalidad Proyecto de Investigación por el señor Christian Emilio Puca Morales, estudiante de la Carrera de Ingeniería Industrial en Procesos de Automatización, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, marzo 2023

Firmado electrónicamente por:
MARCELO VLADIMIR
GARCIA SANCHEZ

---

Ing. Marcelo García, PhD

TUTOR

## AUTORÍA

El presente Proyecto de Investigación titulado: DISEÑO DE COMUNICACIÓN BASADO EN CONTENEDORES PARA LA INDUSTRIA 4.0, es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2023

---

Christian Emilio Puca Morales

C.C. 1804790101

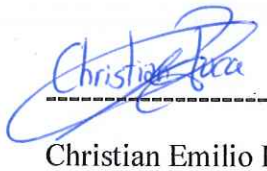
AUTOR

## DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, marzo 2023



Christian Emilio Puca Morales

C.C. 1804790101

AUTOR

## APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Christian Emilio Puca Morales, estudiante de la Carrera de Ingeniería Industrial en Procesos de Automatización, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado DISEÑO DE COMUNICACIÓN BASADO EN CONTENEDORES PARA LA INDUSTRIA 4.0, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

---

Ing. Elsa Pilar Urrutia Urrutia, Mg
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
LUIS ALBERTO MORALES PERRAZO

---

Ing. Luis Morales, Mg.
PROFESOR CALIFICADOR

ELIZABETH PAULINA AYALA BANO

Firmado digitalmente porELIZABETH PAULINA AYALA BANO
DN: cn=ELIZABETH PAULINA AYALA BANO c=EC l=QUITO o=BANCO CENTRAL DEL ECUADOR ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Motivo:He revisado este documento
Ubicaci- n:
Fecha:2023-03-14 13:48-05:00

---

Ing. Paulina Ayala, Mg.
PROFESOR CALIFICADOR

## DEDICATORIA

A mis padres, quienes con sus palabras de apoyo y amor han logrado que en sus hijos exista un espíritu de superación, mediante el valor de la humildad, respeto, compañerismo y responsabilidad, siendo ellos personas que a lo largo de su vida han conseguido pequeños y grandes logros que han motivado en mí el seguir adelante con el objetivo de ser un ser humano mejor para mi familia y la sociedad.

A mi familia que siempre han sido parte de mí, por ser siempre ese apoyo en cada momento de dificultad, demostrándome así ese cariño y amor sincero.

*Christian Emilio Puca Morales*

## AGRADECIMIENTO

Agradezco a Dios ante todas las cosas por haberme bendecido con el conocimiento y sabiduría necesaria para haber llegado hasta este punto de mi vida, además por haberme brindado la oportunidad de vivir experiencias que me formaron como ser humano.

A mis padres, por el apoyo incondicional que a lo largo de esta trayectoria me han brindado, por tener siempre esas palabras de motivación para de esta manera salir adelante sin importar los obstáculos que se han presentado, por los valores que me han enseñado, gracias por cada consejo y guía que me ayudo a cumplir varios de mis objetivos de vida. A mi familia y amigos por esas palabras de apoyo y su compañía a lo largo de este trayecto.

Gracias a mi tutor, Dr. Marcelo García por haberme ayudado con su conocimiento y experiencia, motivando en mí el desarrollo de la investigación e innovación para el mejoramiento de mi persona como futuro profesional.

*Christian Emilio Puca Morales*

## ÍNDICE GENERAL DE CONTENIDOS

## ÍNDICE DE TABLAS

## ÍNDICE DE FIGURAS

## RESUMEN EJECUTIVO

Es necesario conocer que la industria 4.0 se basa en el manejo de tecnologías avanzadas como el Internet de las Cosas (IoT) y la computación en la nube, lo que requiere la adopción de nuevas plataformas y estándares. El proyecto de investigación tuvo como objetivo el desarrollo de un sistema de control adaptable, dinámico e interactivo a cualquier proceso de una industria de manufactura. Esta interacción plantea desafíos en términos de seguridad, escalabilidad e interoperabilidad.

La metodología del diseño del programa se basó en los parámetros de la norma internacional IEC 61499 y utiliza contenedores para virtualizar procesos de producción, planteando la visualización de errores predecibles durante el manejo del programa. El diseño del sistema incluye la adquisición de datos a través de un CPU 1515SP PC2 F (IPC) y el envío de esos datos a través del Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP) hacia los contenedores Docker que se ejecutan en la Raspberry Pi. Los contenedores se almacenan en un repositorio y se cargan y descargan en la tarjeta a través de FBs creados para el control de los contenedores. El sistema de comunicación basado en contenedores permite monitorear el proceso a través de una interfaz gráfica mediante el protocolo de comunicación Message Queuing Telemetry Transport (MQTT).

Los resultados durante las pruebas de funcionamiento del comportamiento del sistema en relación con las salidas esperadas fueron que: para el protocolo TCP/IP el rango de respuesta fue de 0,006 a 0,109 ms y; para el protocolo MQTT el intervalo fue de 0.00095 µs a 0.000561 µs.



**Palabras clave:** Industria 4.0, contenedores, IEC 61499, Internet de las Cosas, bloques funcionales, arquitectura de memoria.

## ABSTRACT

It is necessary to know that industry 4.0 is based on the management of advanced technologies such as the Internet of Things (IoT) and cloud computing, which requires the adoption of new platforms and standards. The research project had as objective the development of an adaptable, dynamic and interactive control system to any process of a manufacturing industry. This interaction poses challenges in terms of security, scalability, and interoperability.

The program design methodology was based on the parameters of the international standard IEC 61499 and uses containers to virtualize production processes, proposing the visualization of predictable errors during program management. The system design includes data acquisition via a CPU 1515SP PC2 F (IPC) and sending this data via Transmission Control Protocol/Internet Protocol (TCP/IP) to the running Docker containers. on the Raspberry Pi. The containers are stored in a repository and uploaded and downloaded to the card through FBs created for container control. The container-based communication system allows monitoring the process through a graphical interface using the Message Queuing Telemetry Transport (MQTT) communication protocol.

The results during the performance tests of the system behavior in relation to the expected outputs were that: for the TCP/IP protocol the response range was from 0.006 to 0.109 ms and for the MQTT protocol the interval was from 0.00095 µs to 0.000561 µs.



**Keywords**: Industrial 4.0, containers, IEC 61499, Internet of Things, functional blocks, memory architecture.

## Introducción

El proyecto investigativo denominado DISEÑO DE COMUNICACIÓN BASADO EN CONTENEDORES PARA LA INDUSTRIA 4.0, tiene como objetivo principal desarrollar un sistema de control en base al uso de contenedores y el estándar IEC 61499, para el desarrollo de procesos mayormente flexibles dentro de la industria. La industria 4.0 es una evolución tecnológica que busca mejorar los procesos productivos mediante la conectividad, automatización y analítica de datos.

Para lograr esto, se analizaron los resultados obtenidos del sistema de control del proceso seleccionado, determinando los beneficios en cuanto a tiempos de respuesta y seguridad durante él envió de datos entre el proceso y los controladores. El uso de contenedores, una tecnología utilizada para empaquetar y desplegar aplicaciones permite una mayor flexibilidad en la configuración del sistema de control, permitiendo una mayor escalabilidad y facilidad de mantenimiento.

El sistema de control se desarrolló con la finalidad que pueda ser usado en múltiples dispositivos sin la distinción de la marca de la casa fabricante, permitiendo de esta manera que los procesos sean interoperables ya que no sería necesario buscar una marca en específico de los controladores, ya que gracias al estándar IEC 61499 y los contenedores el usuario dispondrá de los procesos desde un repositorio en la nube de tal manera que mediante un comando el podrá hacer uso del proceso deseado.

La metodología para el desarrollo de este proyecto se basó en la investigación y experimentación debido a que se hizo uso de equipos industriales de alta calidad y robustes como son los Computadores Industriales (IPC) y las tarjetas de desarrollo Raspberry Pi, permitiendo de esta manera tener un sistema de control con la capacidad de trabajar con dispositivos de diferentes procedencias.

# CAPÍTULO I

# MARCO TEÓRICO

## 1.1 Tema de investigación

"DISEÑO DE COMUNICACIÓN BASADO EN CONTENEDORES PARA LA INDUSTRIA 4.0"

## 1.2 Antecedentes investigativos

El estándar IEC 61499 es un estándar internacional para la programación y el control de sistemas de automatización distribuidos. Se basa en el concepto de Bloques Funcionales y proporciona un marco para la integración de sistemas de control y automatización. Uno de los beneficios del IEC 61499 es su capacidad para integrar sistemas de diferentes fabricantes y tecnologías, lo que permite una mayor flexibilidad y escalabilidad.

En los últimos años, el uso de contenedores ha ganado popularidad en el contexto de la automatización industrial. Los contenedores son una forma de empaquetar y distribuir aplicaciones y sus dependencias de manera que puedan ser ejecutadas de manera consistente en diferentes entornos. Esto permite la implementación rápida y sencilla de aplicaciones, y también puede mejorar la seguridad y la eficiencia en términos de recursos, como se observa en las siguientes investigaciones.

El estudio de Lyu T, Dwi U, y Vyatklin V, para la creación de aplicaciones de automatización distribuida con IEC 61499 y tecnología de contenedorización, en la que examinaron que la combinación de IEC 61499 y contenedores puede ser utilizada para el desarrollo de una Fábrica Inteligente (Smart Factory). Las fábricas inteligentes se destacan por la capacidad de poseer una gran flexibilidad, estabilidad, distribución y una ejecución basada en eventos [1]. Por lo tanto, la combinación del estándar IEC 61499 y los contenedores permiten que la industria utilice tecnologías avanzadas, como la automatización y la digitalización, para la mejorar la eficiencia y la productividad, de manera rápida y sencilla.

La investigación realizada por Dai W, Zhang Y, Kong L, Christensen J, et al., para el diseño de aplicaciones industriales basadas en IEC 61499 y contenedores, establece que la automatización industrial en la actualidad se enfoca en el IoT (Internet de las cosas), ya que este permite tener capacidades mejoradas a las industrias en cuanto a computación, comunicación y almacenamiento de información, al trabajar con la nube y los dispositivos de campo. De tal manera que los sistemas de automatización basados en la normativa ISA – 95 se encuentran en un cambio hacia la arquitectura de dos capas: la nube industrial y la computación periférica [2]. Por lo tanto, los softwares de programación industrial se encuentran en evolución debido a que es necesario que presenten la capacidad de crear aplicaciones en base a la nube, ya que de esta manera permiten el desarrollo de procesos más flexibles y eficientes, mejorando así la capacidad productiva de la industria.

El estudio realizado por Martinov G, Kozak N, Nezhmetdinov R, basado en el desarrollo de máquinas para la industria 4.0, mediante la integración de tecnologías avanzadas, como la automatización, la digitalización y la conectividad con el objetivo de mejorar los procesos de producción y fabricación, determinó que las tendencias actuales para el desarrollo y fabricación de máquinas y equipos industriales tienen como objetivo mejorar la eficiencia, la productividad y la flexibilidad de la producción y la fabricación a través de la utilización de tecnologías avanzadas como es el uso del estándar IEC 61499 y de los contenedores [3]. De esta manera, las nuevas máquinas tienen la capacidad de transferir datos a los niveles superiores de control para su integración en industrias de ingeniería digital. Esto significa que las máquinas y equipos industriales deben tener la capacidad de recopilar y enviar datos a sistemas de control y monitoreo, lo que permite una mejor visibilidad y control sobre los procesos de producción y fabricación. Esto también permite la integración de la información recopilada por las máquinas y equipos industriales en sistemas de ingeniería digital, lo que puede mejorar la toma de decisiones y la eficiencia de la producción y la fabricación.

La investigación desarrollada por Souri A y Ghobaei-Arani M para el desarrollo de aplicaciones de control basados en el uso de la nube, establece que las nuevas soluciones dentro de las industrias se encuentran orientadas al desarrollo de aplicaciones mediante el IoT (Internet de las cosas), ya que este tipo de sistemas

mejoran la capacidad de gestión de la información mediante una transición rápida, obteniendo de esta manera procesos más flexibles, más inteligentes, automatizados y reactivos [4]. Por ende, las industrias están orientadas al desarrollo de aplicaciones basadas en el IoT, como es el uso de los contenedores para la virtualización de los procesos, ya que estos mejoran la capacidad de gestión de la información y proporcionan procesos más flexibles, más inteligentes, automatizados y reactivos. Esto significa que las aplicaciones basadas en el uso de contenedores pueden proporcionar una mayor eficiencia y flexibilidad en la producción.

El estudio realizado por García C, García M, Irisarri E, et. al, basado en el desarrollo de una plataforma de contenedores para el control de robots industriales, establece que las tecnologías de virtualización en el desarrollo de software dentro de las industrias presentan un avance en cuanto a la flexibilidad y la innovación de los procesos industriales ya que permiten trabajar con soluciones emergentes relacionadas al Cloud Computing y Big Data [5]. Por lo tanto, el uso de contenedores permite el desarrollo de soluciones emergentes de tal manera que se aprovechan todas las ventajas del Cloud Computing y el Big Data, mejorando la eficiencia y la productividad de los procesos industriales, ya que facilita el despliegue de aplicaciones en diferentes entornos, como la nube o en diferentes sistemas operativos.

El estudio realizado por Zambrano T., basado en el uso del protocolo MQTT basado en la norma IEC 61499 para la integración de un robot KUKA YOUBOT hacia la nube, se utilizó la prueba Q de Cochran para determinar si había una diferencia significativa en la precisión de envío y recepción de datos del robot KUKA Youbot hacia la nube en diferentes momentos del día [6]. Los resultados de la prueba indicaron que no había una diferencia significativa en diferentes momentos del día, lo que sugiere que el diseño de comunicación utilizado para la integración del robot KUKA YOUBOT con la nube es capaz de realizar tareas de manera consistente y precisa a lo largo del día.

En resumen, el estándar IEC 61499 es una herramienta valiosa para la programación y el control de sistemas de automatización distribuidos, y su uso combinado con contenedores puede proporcionar beneficios adicionales en términos de flexibilidad,

escalabilidad y eficiencia en la implementación y ejecución de aplicaciones industriales.

### 1.2.1 Contextualización del problema

El uso de contenedores y el estándar IEC 61499 en la automatización se enmarca en el contexto de la transformación digital en la industria. La automatización es una pieza clave en la digitalización de los procesos industriales y la mejora de la eficiencia, flexibilidad y escalabilidad de los sistemas.

El uso de contenedores y el estándar IEC 61499 en la automatización también se enmarca en el contexto de la movilidad de la información y el aumento de la interconexión en los sistemas industriales. Los sistemas de automatización tradicionales a menudo son monolíticos y están diseñados para ejecutarse en un único sistema o dispositivo. Sin embargo, con la tendencia actual hacia la movilidad de la información y la interconexión, es necesario que los sistemas de automatización sean capaces de ejecutarse en diferentes dispositivos y entornos [7].

El uso de contenedores y el estándar IEC 61499 proporciona una solución para la distribución y ejecución de aplicaciones y la comunicación entre componentes de automatización. Los contenedores permiten empaquetar aplicaciones y sus dependencias en un formato portátil, lo que permite su ejecución consistente en diferentes entornos. El estándar IEC 61499 proporciona un marco común para la descripción de funciones de control y la comunicación entre componentes [8].

El uso de contenedores permite una fácil distribución y ejecución de las aplicaciones en diferentes entornos, incluyendo dispositivos edge, nubes públicas y privadas, y computadoras locales. El estándar IEC 61499 proporciona un marco común para la comunicación entre los componentes y la descripción de funciones de control, lo que permite una mayor interconexión entre los diferentes componentes del sistema [9].

En el nivel más específico, esto puede traducirse en la implementación de contenedores en un sistema de automatización para mejorar la flexibilidad y escalabilidad del sistema, mejorar la seguridad al aislar las aplicaciones y sus dependencias, y facilitar la actualización y mantenimiento de las aplicaciones.

Además, el uso del estándar IEC 61499 puede mejorar la interoperabilidad entre los diferentes componentes del sistema de automatización [10].

### 1.2.2 Fundamentación teórica

**Industria 4.0 y Manufactura Inteligente**

Es la nueva era de la revolución industrial, salto que hace referencia al manejo de una gestión de cadenas de valor y enfoque en la organización que desea adquirir un sistema automatizado por medio de la conexión entre el mundo digital y el mundo real, dando una vista real sobre el IoT y el Mundo de Datos (Big Data) con la finalidad de optimizar las líneas de procesos y la prestación de servicios de una fábrica en general [11] [12].

Este aspecto industrial de innovación se centra principalmente en puntos fundamentales como proporcionar resultados en el menor tiempo posible, incrementar el procesamiento de datos y supervisar la incidencia dentro de una organización que se plantea la mejora continua dentro de sus estadísticas, dar el siguiente paso es el compromiso de las industrias que desean prepararse para adaptarse a los cambios que se suscitan a lo largo del pasar del tiempo [13].

A partir de esta perspectiva, los cambios que se dan a nivel industrial son el resultado de la incursión de las nuevas tecnologías, la cuales se enfocan en: la digitalización de los procesos productivos, la automatización de las actividades de manufactura dentro de la industria, la integración de la de las capaz física y digitales de los sistemas de control industrial, etc. De esta manera la industria 4.0 se establece como un nivel más dentro de la composición organizacional de las cadenas de valor y gestión [11].

**CPS (Sistemas ciber físicos)**

Los sistemas ciber físicos son elementos compuestos de componentes físicos y digitales, que permiten gestionar información, comunicación y el almacenamiento de datos. Dichos sistemas son principalmente utilizados dentro de los CPS de la industria 4.0, en donde estos sistemas permiten supervisar productos y equipos proporcionando variables físicas como temperatura, humedad, ubicación, nivel, flujo, etc [11]. Los sistemas ciber físicos se identifican por las siguientes características:

**Tabla 1.** Características de los sistemas ciber físicos [11] [14]

| Características de los sistemas ciber físicos | |
|---|---|
| Capacidad de relacionarse | Tiene la capacidad de relacionarse con componentes físicos con finalidad de monitorear y/o controlar. |
| Capacidad de aprender y evolucionar | Tiene la capacidad de aprender y evolucionar a partir de la información disponible en el mundo virtual. |

De esta manera los sistemas ciber físicos presentan un amplio campo de aplicación como en el área de manufactura, energía, saluda e inclusive transporte.

**Estándar IEC 61499**

El estándar IEC 61499 es el resultado de la necesidad de establecer una arquitectura genérica para el control distribuido a nivel industrial. El avance tecnológico suponía una mejor arquitectura y requisitos software siendo esta la razón de la mejora del estándar 61131, llegando al estándar 61499 siendo una versión renovada y mejorada en comparación a su predecesor [15]. La primera versión del estándar 61499 se probó en agosto del 2005 por el Comité Técnico de medida, control y automatización de procesos industriales (Technical Commite) quienes son miembros de la IEC. En su primera versión se proporcionó de una guía de uso del FB en Sistemas de Control y Medición de Procesos Industriales Distribuidos (IPMCSs).

El estándar IEC 61499 tiene como objetivo establecer una arquitectura genérica, estableciendo sistemas heterogéneos mediante la integración de dispositivos de control de diferentes fabricantes, permitiendo el desarrollo de sistemas de control dinámicos, realizando modificaciones en la configuración mientras se encuentran en ejecución. Además, se establece como uno de los estándares orientados al desarrollo de sistemas de automatización con la facilidad de interoperabilidad de diferentes dispositivos [8].

**Elementos básicos del estándar IEC 61499**

El estándar IEC 61499 se caracteriza por los siguientes elementos:

- Funciones: son bloques de software que realizan tareas específicas, como leer un sensor o controlar un actuador.

- Eventos: son señales que indican el inicio o finalización de una tarea o un cambio en el estado de un sistema.

- Procesadores de eventos: son componentes que controlan el flujo de eventos y ejecutan las funciones adecuadas en respuesta a los eventos.

- Servicios: son funciones que proporcionan servicios comunes a varias funciones, como el acceso a una base de datos o la comunicación con otro sistema.

- Interfaces: son puntos de conexión entre diferentes componentes de un sistema, como entre un sensor y un procesador de eventos.

IEC 61499 también define conceptos como la configuración, la ejecución y el mantenimiento de los sistemas de control y automatización [15] [8]**.**

**Arquitectura**

El estándar IEC 61499 con el objetivo de mejorar la compresión del proceso o sistema industrial, establece una arquitectura en forma genérica y jerárquica de todos los modelos. Los modelos establecidos en dicha arquitectura se caracterizan por ser independientes del dominio y as su vez por ser extensibles a través del uso de FBs. De esta manera se establecen los siguientes modelos dentro del estándar:

**Modelo de un Bloque Funcional (FB)**

Un FB se establece como el elemento más pequeño e importante dentro de un sistema de control, componiéndose de una cabeza que se conecta al flujo de eventos con la finalidad de leer las entras, analizarlas y finalmente establecer las salidas adecuadas. En base a los eventos que se desarrollan, tanto en las entradas como en las salidas. De esta manera se establece que cada FB tiene un comportamiento dinámico permitiendo la entrada y salida de eventos.

En el estándar IEC 61499 un bloque funcional permanece en espera hasta recibir una señal de activación en forma digital para poder generar un valor de ingreso, lo cual permite la activación del FB generándose de forma simultánea los eventos de entrada y salida. Por lo tanto, un FB funciona de acuerdo a los algoritmos programados

internamente para el procesamiento de las entradas con los datos generados internamente, los cuales serán enviados por las salidas [15] [7].

**Modelo de Recurso**

El modelo de recurso permite la aceptación, procesamiento y retorno de eventos y/o información de los diferentes protocolos de comunicación y el proceso, además del control independiente de operación y los servicios a las funciones [10].

El modelo de recurso dentro del estándar IEC 61499 se compone de los siguientes elementos:

**Tabla 2.** Componentes del modelo de recurso [15] [16]

| **Componentes del modelo de recurso** | |
|---|---|
| Aplicación local | Posee las variables y eventos de entrada y salida de los FBs que desarrollaron diferentes procesos. |
| Interfaz de proceso | Realiza un mapeo, entras las diferentes aplicaciones y la información resultante entre proceso y los eventos a través de los bloques de función de interfaz de servicio. |
| Interfaz de comunicación | Realiza un mapeo entre las aplicaciones y la interfaz de comunicación utilizado de la misma manera los Bloques de Función de Interfaz de Servicio (SIFBs). |

**Modelo de dispositivo**

El modelo de dispositivo se establece como un receptáculo de recursos en el cual se ejecutan las aplicaciones, en el cual se encuentran dos clases de interfaces: la primera que se enfoca en el Proceso, en donde se encuentran las entradas y salidas de los dispositivos y la segunda que se enfoca en la Comunicación, en donde se hace uso de los diferentes protocolos de comunicación entre los diferentes dispositivos y/o aplicaciones. Por consiguiente, el modelo físico establece como un elemento independiente, el cual realiza una o varias funciones de acuerdo a un evento en particular teniendo en consideración las interfaces [15] [16].

**Modelo de sistema**

El modelo de sistema se encarga de la recolección de la cantidad de dispositivos que se encuentran conectados entre sí mediante un protocolo de comunicación por medio de segmentos y enlaces con la finalidad de una correlación entra aplicaciones [17].

**Modelo de aplicación**

El modelo de aplicación se define como una unidad con la capacidad de distribuirse en todos los dispositivos que se encuentren interconectados a través de recursos, dispositivos o aplicaciones, obteniendo de esta manera una respuesta eficaz a los eventos resultantes del proceso y de la comunicación. Además, presenta la capacidad de modificar variables y generar eventos de acuerdo a las necesidades del proceso, permitiendo la interacción entre la interfaz del proceso y la comunicación por medio de la programación establecida [18].

De esta manera la aplicación se define a través del flujo de datos de la red de FBs, teniendo en consideración los eventos que inicializan las variables, los algoritmos internos y a los eventos que emiten información a las entradas y salidas de los FBs.

**Modelo de distribución**

Es la sección en la que la aplicación se basa en las especificaciones del estándar IEC 61499, de acuerdo con eso los FBs son mapeados de acuerdo con los dispositivos de control en donde serán inicializados y ejecutados. De esta manera se aprecia la compatibilidad del estándar IEC 61499 en varios dispositivos, obteniendo un sistema de control distribuidos mediante la ejecución de FBs desde diferentes dispositivos [19]. Entre algunas consideraciones que tiene la norma IEC 61499 está:

- **Bloque de función básico:** Similar a los diagramas de estado, se rigen a eventos o algoritmos de 1 y 0 según estándares compatibles.
- **Bloque de función de interfaz de servicio:** Describe la secuencia del servicio.
- **Bloque de funciones compuesto:** Es una red de bloques de funciones.
- **Formato DTD:** Representa los requisitos de representación y portabilidad de los elementos conforme la IEC 61499.

- **Compatibilidad:** No es necesario el formato DTD, se requiere de un entorno de tiempo de ejecución y desarrollo compatible al sistema propuesto.

**Modelo de gestión**

En esta sección el estándar IEC 61499 establece un ejemplo de FB para la gestión y configuración de un IPMCS distribuido en base al uso de las funciones de gestión incluidas en los diferentes dispositivos [15] [20].

**4diac IDE**

El software 4diac IDE es un software de ingeniería de código abierto desarrollada por Eclipse, para el diseño e implementación de sistemas de control distribuidos en diferentes dispositivos. 4diac IDE fue dada a conocer en el año 2000 por parte de PORFACTOR GmbH y la Universidad Tecnológica de Viena [21].

De esta manera 4diac IDE fue desarrollada con el objetivo de proporcionar herramientas de acuerdo con el estándar, estableciendo un entorno de automatización y control, basado en la portabilidad, re-configurabilidad e interoperabilidad. 4diac IDE se caracteriza por los siguientes aspectos:

**Tabla 3.** Características de 4diac IDE [10]

| **Características de 4diac IDE** |
|---|
| Es un software de programación industrial basado en el estándar IEC 64199. |
| Presenta una variedad de datos de acuerdo con el estándar IEC 61131-3. |
| Permite la conexión de eventos y datos. |
| Permite hacer uso de las funciones de crear, escribir e iniciar de acuerdo con el estándar IEC 61499. |
| Posee FBs de los protocolos de comunicación como cliente/servidor, Publicación/Suscripción. |

Por lo tanto, 4diac IDE es una herramienta de gran utilidad dentro del campo del automatismo industrial, ya que cuenta con la capacidad de establecer el valor de las

variables remotamente, además de permitir la depuración y prueba de aplicaciones en tiempo real.

**Runtime - FORTE**

Forte es una aplicación portable del runtime de 4diac IDE en base al estándar IEC 61499, desarrollada para la ejecución en pequeños dispositivos de control con sistemas embebidos de 16/32 Bit; permite ejecutarse en una gran variedad de plataformas ya que permite integrar C++ dentro de su programación [9].

Forte permite ejecutar en tiempo real diversas configuraciones de control establecidas en base al IEC 61499, las cuales se inicializan por medio de eventos externos, de tal manera que los procesos se ejecutan de acuerdo con la prioridad establecida dentro de las configuraciones.

Los eventos dentro de forte se ejecutan de tal manera que el primero que ingresa será el primero en salir (FIFO), separando de esta manera la ejecución y envió de eventos del bloque receptor, creando el periodo de bloqueo del FB de forma independiente a la topología de la red [22].

La optimización de la ejecución de forte se mantiene en constante cambio en conjunto a la interfaz de comunicaciones, de esta forma esta aplicación funciona de forma independiente de la plataforma empleada, permitiendo el uso de diversos tipos de hardware y plataformas con sistemas embebidos [23].

**Raspberry Pi**

Raspberry Pi es una tarjeta de control creada en Reino Unido, por un grupo de investigadores de la Universidad de Cambridge; dicha tarjeta de control fue desarrollada con el objetivo de obtener un dispositivo de bajo coste para el desarrollo de habilidades de programación de acuerdo a las necesidades de la industria [24].

La primera Raspberry Pi fue presentada en el año 2012, después de arduas investigaciones, estudios y aprobaciones. Posteriormente en el año 2014 se dieron a conocer dos modelos de dicha tarjeta, el Modelo A y el Modelo B. La Raspberry Pi del modelo A poseía una menor potencia en comparación al modelo B, ya que no permitía realizar conectividad por Ethernet, además que el tamaño de la memoria

RAM y la velocidad de procesamiento eran relativamente pequeñas. El modelo B de Raspberry PI por otro lado presento una amplia aceptación en el mercado y la comunidad académica por sus características que permitían el desarrollo de investigaciones a nivel industrial a un costo accesible [24].

Raspberry Pi a lo largo de los años en el campo educativo e industrial se establece como un ordenador compacto y económico, empleado en su mayoría para el desarrollo de aplicaciones IoT.

Esta tarjeta cuenta con un procesador de cuatro núcleos a 1,4 GHz basado en el diseño ARM Cortex-A53, lo que le permite manejar tareas de computación de alta demanda. Además, cuenta con 1 GB de memoria RAM, lo que permite ejecutar varios programas simultáneamente. En cuanto al almacenamiento, la tarjeta Raspberry Pi B+ no viene con un disco duro o una unidad flash integrada, sino que se debe utilizar una tarjeta SD o un disco duro externo para almacenar el sistema operativo y los datos.

La tarjeta Raspberry Pi B+ cuenta con una amplia variedad de conectividad, incluyendo puertos Ethernet y HDMI para conectarse a Internet y a dispositivos de pantalla, así como puertos USB para conectarse a teclados, ratones y otros dispositivos. También cuenta con un puerto de audio de 3,5 mm y un puerto de salida de video compuesto. Además, cuenta con una conectividad inalámbrica mediante wifi y/o Bluetooth. La tarjeta Raspberry Pi B+ se alimenta mediante una fuente de alimentación micro, lo que le permite funcionar de manera estable y eficiente en términos energéticos. Es una excelente opción para proyectos de computación en el hogar y la educación, ya que es accesible y fácil de usar. [25]**.**

**Controlador Cpu 1515SP PC2 F (IPC)**

La CPU 1515SP PC2 F se establece como un controlador diferente a los controladores S7 1500 estándar, ya que este controlador permite trabajar como una PC en base a un sistema operativo de Windows 10 IOT, obteniendo una flexibilidad para trabajar como PLC y PC al mismo tiempo [26].

Este controlador al ser útil para las industrias basadas en trabajos en serie, son capaces de controlar máquinas automáticas distribuidas en función de la necesidad de trabajo. El IPC ET 200SP tiene una amplia variedad de conectividad, incluyendo puertos

Ethernet, USB y HDMI para conectarse a Internet y a dispositivos de pantalla, así como puertos para conectarse a sensores y actuadores. También cuenta con un puerto de audio de 3,5 mm y un puerto de salida de video compuesto. Además, cuenta con una conectividad inalámbrica, permite la conexión a redes y dispositivos mediante wifi o Bluetooth.

El IPC ET 200SP se alimenta mediante una fuente de alimentación externa con un rango de tensión de 18-30 VDC, lo que le permite funcionar de manera estable y segura en entornos industriales. También cuenta con un software de control y programación integrado, como STEP 7 o TIA Portal, que permite programar y controlar procesos industriales de manera fácil y eficiente. Además, cuenta con una carcasa robusta y resistente a la intemperie, diseñada para su uso en entornos industriales y ambientes hostiles [27] [28].

**Contenedores**

Los contenedores son aplicaciones que trabajan con su respectiva imagen, de tal manera que solo se debe facilitar la aplicación debido a que en el contenedor se incluyen el tiempo de ejecución y el servidor de esta, facilitando el despliegue y el mantenimiento de las aplicaciones.

Los contenedores permiten eliminar gran parte de los problemas que se generan al tener configuraciones de entorno incoherentes, ya que permiten empaquetar dentro de un solo conjunto las aplicaciones, dependencias y configuraciones de entorno [29].

**Docker**

Docker es un software de código abierto que se desarrolló con la finalidad de automatizar el despliegue de aplicaciones en contenedores portátiles, permitiendo ejecutarse en la nube o localmente según lo requiera.

De esta manera Docker es parte fundamentar de la estrategia para la implementación de contenedores que está siendo adoptada por gran parte de proveedores de plataformas de virtualización. Debido a que la construcción y ejecución de contenedores se desarrolla en cualquier sistema operativo sin distinción alguna.

Además, cabe mencionar que Docker permite trabajar con contenedores en plataformas Windows y Linux [30].

**Componentes de Docker**

Docker eta compuesto por los componentes descritos en la siguiente tabla:

**Tabla 4.** Componentes de Docker [30]

| Componentes de Docker | |
|---|---|
| Contenedor | Es una o varias instancias compuestas en las cuales se almacenan la aplicación o servicio. |
| Imagen | Es una compilación de los archivos del sistema y los parámetros necesarios para el uso del aplicativo en el tiempo de ejecución del contenedor. |
| Repositorio | Es un servicio que posee los repositorios de imágenes de Docker, los cuales pueden ser descargados desde el internet, existen repositorios públicos y privados. |

## 1.3 Objetivos

**Objetivo general**

Diseñar un módulo de comunicación basado en contenedores para la industria 4.0.

**Objetivos específicos**

- Identificar parámetros de comunicación en procesos automatizados mediante investigación bibliográfica para la determinación del enfoque del estándar de una industria 4.0.

- Modelar un entorno de contenedores Docker con comunicación MQTT y TCP/IP para el monitoreo de variables de control.

- Comparar el módulo de comunicación de ordenadores industriales IPC frente a dispositivos similares a través de comparación directa para el sustento de ventajas de un entorno de contenedores en la industria 4.0.

# CAPÍTULO II

# METODOLOGÍA

## 2.1 Materiales

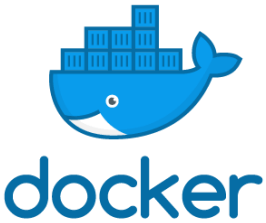
La Tabla 5, muestra los materiales físicos utilizados en el proyecto de investigación.

**Tabla 5.** Materiales del proyecto de investigación

| Material | Fotografía | Descripción |
|---|---|---|
| Máquina IPC |  | Controlador del sistema donde se elaboró la programación requerida para observar el comportamiento de los componentes. |
| Raspberry Pi |  | Microcontrolador programable para responder a la secuencia de líneas de código introducidas. |
| Ordenador |  | Dispositivo físico requerido para elaborar el informe de investigación. |
| Router |  | Dispositivo físico requerido para establecer la conexión entre los componentes del sistema. |

La Tabla 6, muestra los softwares utilizados en el proyecto de investigación.

**Tabla 6.** Software del proyecto de investigación

| Software | Fotografía | Descripción |
|---|---|---|
| Microsoft Office |  | Software requerido para plasmar la información del estudio elaborado. |
| Microsoft Excel |  | Software de tabulación de datos. |
| Tía Portal |  | Software de elaboración de estructuras funcionales de código. |
| Docker |  | Software requerido para elaborar contenedores de programación bajo código abierto. |
| 4DIAC IDE |  | Software utilizado para elaborar los códigos de programación bajo lineamientos de la norma IEC 61499. |
| Runtime - FORTE |  | Software requerido para elaborar programas portables bajo la norma IEC 61499. |
| ODK |  | Paquete de desarrollo que permite programar y generar archivos ejecutables. |

## 2.2 Métodos

### 2.2.1 Modalidad de la investigación

En esta investigación se utilizó un enfoque cuantitativo, ya que se buscó medir y analizar datos numéricos relacionados con el protocolo de comunicación MQTT en diferentes momentos del día. Se utilizó la prueba Q de Cochran para analizar los datos y determinar si había una diferencia significativa en la precisión de las mediciones. Además, se hizo uso de las técnicas y metodologías de investigación descritas a continuación.

**Investigación bibliográfica**

El desarrollo se fundamentó en investigaciones desarrolladas anteriormente con casos similares de implementación módulos de comunicación con la finalidad de mejorar el envío y recepción de datos entre departamentos de una industria, los desarrollos verificados cuentan con un porcentaje mayor a la velocidad normal fueron el sustento como medio para desarrollar la investigación planteada.

**Investigación de campo**

El trabajo de investigación se enfocó en la programación de un módulo de comunicación entre cliente – servidor, por tanto, fue necesario analizar el comportamiento que tienen los componentes del sistema en tiempo real, determinar los parámetros de respuesta y la eficiencia dentro de un ambiente de trabajo con proyección de automatización sobre el correcto envío y recepción de datos.

**Investigación experimental**

Se utilizó un diseño experimental, ya que se realizaron mediciones del sistema de comunicación en diferentes instancias para evaluar la precisión de las mediciones en diferentes condiciones. Se midieron la integridad de los datos durante el funcionamiento del sistema de comunicación mientras los procesos se encontraban en ejecutándose de forma automática, y se utilizó la prueba Q de Cochran para determinar si había una diferencia significativa en la precisión de las mediciones.

**Investigación aplicada**

La metodología aplicada se aplicó por los conocimientos obtenidos de los módulos de automatización, bajo los cuales se consideró el manejo adecuado del cableado de los equipos, los correctos manejos de programación y diseño adecuado de la interfaz gráfica de uso del sistema en concreto.

### 2.2.2 Recolección de información

La recolección de información para la validación del proyecto investigativo se enfocó en el desarrollo de pruebas de funcionamiento, con el objetivo de establecer el comportamiento del programa en un entorno real, la Tabla 7, muestra la recolección de información según los objetivos planteado.

**Tabla 7.** Tabla de recolección de información

| Objetivo | Técnica/Método | Instrumento |
|---|---|---|
| Identificar parámetros de comunicación en procesos automatizados mediante investigación bibliográfica para la determinación del enfoque del estándar de una industrial 4.0. | Recolección de información de datos sobre los componentes y softwares utilizados.<br>Revisión documental. | Libros.<br>Trabajos de investigación.<br>Manual de usuario de cada componente. |
| Modelar un entorno de contenedores Docker con comunicación MQTT y TCP/IP para el monitoreo de variables de control. | Diseño de la programación.<br>Diseño del archivo ejecutable.<br>Diseño de interfaz gráfica.<br>Pruebas de funcionamiento. | Softwares de modelado de programación abierta. |
| Comparar el módulo de comunicación de ordenadores industriales IPC frente a dispositivos similares a través de comparación directa para el sustento de ventajas de un entorno de contenedores en la industria 4.0. | Toma de tiempos.<br>Toma de envío/recepción de datos. | Fichas de recolección de datos. |

**Selección del software para el desarrollo del sistema de control**

El software utilizado para el desarrollo del sistema de control fue 4diac IDE, ya que permite crear programas mediante código abierto de forma rápida. Se caracteriza por tener una interfaz intuitiva y soporte oficial y permite programar a través de FBs propios del software o FBs creados por el usuario según se requiere.

Una de las ventajas de 4diac IDE es que las aplicaciones creadas mediante este software pueden ser cargadas y ejecutadas por diferentes dispositivos de campo bajo estándar IEC 61499, lo que significa que son compatibles con un gran número de dispositivos de la industria. Esto hace que 4diac IDE sea una opción atractiva para el desarrollo de sistemas de control en entornos industriales. Es importante tener en cuenta que el estándar IEC 61499 es un estándar internacional para la automatización distribuida de sistemas. Se utiliza para diseñar, implementar y mantener sistemas de control y automatización. Al utilizar el estándar IEC 61499, es posible integrar varios componentes de automatización como sensores, actuadores y controladores en un sistema de control y automatización distribuido.

**Selección del software de ejecución**

El software de ejecución seleccionado para el desarrollo del sistema de control es 4diac Forte, ya que permite trabajar de forma independiente con el dispositivo que contiene al software elaborado. Esto significa que es posible ejecutar el software Forte en equipos y sistemas operativos diferentes, lo que hace que sea ideal para utilizar en diferentes entornos industriales, de investigación y educativos. Una de las ventajas de 4diac Forte es que tiene una infraestructura de comunicación flexible que se puede adaptar a las necesidades del usuario. Esto permite utilizar el software Forte en una amplia variedad de aplicaciones y entornos, ya que es capaz de comunicarse con diferentes dispositivos y sistemas de control.

**Selección del software para la adquisición de datos**

El software seleccionado para la adquisición de datos fue ODK 1500S el cual permite recopilar y almacenar información de diferentes dispositivos y sistemas de control. Una de las ventajas de ODK 1500S es que permite hacer uso de librerías para la compilación de archivos necesarios para la creación de la extensión .dll. Este archivo

se utiliza para la interacción del equipo encargado de la adquisición y envío de datos con el software de desarrollo.

El uso de un archivo .dll permite que el equipo de adquisición de datos y el software de desarrollo puedan comunicarse entre sí de forma eficiente y rápida, lo que agilita el proceso. El uso de librerías permite agregar funcionalidades adicionales al software de adquisición de datos, como la capacidad de procesar y analizar información de forma automática.

**Selección de la plataforma para la creación de contenedores**

La plataforma seleccionada para la creación de los contenedores a utilizar en el desarrollo del sistema de control fue Docker, ya que permite automatizar y crear campos de información bajo software de forma rápida y sencilla. Una de las principales ventajas de Docker es que arranca los contenedores en cuestión de segundos, lo que hace que sea ideal para entornos de producción y desarrollo rápido.

Otra ventaja de Docker es que permite crear contenedores de software compacto, lo que reduce el consumo de recursos de los dispositivos, siendo útil en entornos de IoT y automatización se cuenta con recursos limitados.

**Selección del hardware para la adquisición/envío de datos**

El equipo de control utilizado para la adquisición de datos fue un computador industrial IPC ET 200SP de la marca siemens, debido a que cuenta con los parámetros de controlador de sistemas automatizados. Está equipado con un procesador Intel Core i7 de cuatro núcleos y 2,9 GHz, lo que le permite procesar y almacenar grandes cantidades de datos de forma rápida y eficiente. Además, es compatible con software de control como 4diac IDE y 4diac Forte.

El IPC ET 200SP tiene una carcasa robusta y resistente a la intemperie diseñada para su uso en entornos industriales y es capaz de trabajar sin descanso ni mantenimiento. Cuenta con puertos Ethernet, USB y HDMI para conectarse a Internet y dispositivos de pantalla, así como puertos para conectarse a sensores y actuadores. También tiene un puerto de audio de 3,5 mm y un puerto de salida de video compuesto. Se alimenta mediante una fuente de alimentación externa con un rango de tensión de 18-30 VDC.

**Selección del hardware para el control del proceso**

El hardware elegido para el proceso de control fue la Raspberry Pi modelo PI B+, debido a que es una placa de computadora de bajo costo y tamaño reducido que se ha vuelto muy popular entre los desarrolladores de proyectos de electrónica. En particular, el modelo PI B+ tiene un procesador de cuatro núcleos que trabaja a una frecuencia de 1.4 GHz realizando tareas de procesamiento intensivas sin problemas.

Además, esta placa cuenta con una tarjeta de red Gigabit Ethernet que ofrece una velocidad de transmisión de datos de hasta 300 Mbps cuando se conecta a través de USB 2.0. Esta tarjeta de red también se destaca por su bajo costo y por tener características técnicas superiores en comparación con otras tarjetas de control de bajo costo. En resumen, la Raspberry Pi modelo PI B+ es una buena opción para proyectos de control gracias a su procesador potente, tarjeta de red rápida y bajo costo.

### 2.2.3 Procesamiento y análisis de datos

El procedimiento utilizado en esta investigación consistió en realizar mediciones de la precisión del sistema de comunicación basado en el protocolo de comunicación MQTT. En cada momento del día se realizaron 10 mediciones durante la ejecución automática del proceso. Los datos recopilados se analizaron utilizando la prueba Q de Cochran para determinar si había una diferencia significativa en la precisión de las mediciones en durante la mañana, tarde y noche.

**Parámetros de comunicación en procesos automatizados**

La Tabla 4, muestra el análisis los parámetros de comunicación a nivel industrial.

**Tabla 8.** Parámetros de comunicación [15]

| Parámetro | Descripción |
|---|---|
| Tiempo de respuesta | El tiempo entre las áreas industriales debe ser mínimo debido a que, si existen problemas en la línea de producción, el sistema debe ser capaz de reaccionar al instante. |
| Envío/recepción de datos | El factor más importante de una comunicación es el movimiento de los datos en toda la línea de producción, por lo que debe existir una pérdida del 0%. |

| | |
|---|---|
| Enlace cliente/servidor | Se debe tomar en cuenta los puertos de enlace entre las partes que componen el sistema. |
| Fiabilidad | El sistema debe ser seguro, no se puede exponer los datos internos de una empresa bajo programas infecciosos. |
| Entradas y salidas del hardware | El número de entradas y salidas debe ser capaz de cubrir con la necesidad de los componentes que conforman el sistema. |

**Elaboración del Runtime incluido en los contenedores**

La construcción de Forte para la arquitectura de la Raspberry Pi se procedió a realizarla por compilación cruzada desde Windows 10 Home.

1. El proceso inicia con el código fuente pre-compilado en el paquete **paho.mqtt.c** donde se puede desarrollar el proyecto mediante la comunicación MQTT.
2. Abrir el software CMake en el cual se genera todos los archivos de esta librería automáticamente.
3. En primer lugar, seleccionar el directorio en el que se extrajo el código fuente **paho.mqtt.c**.
4. En el directorio de salida donde se crean los binarios se otorga la dirección de la carpeta a crear con el nombre “binPaho”.
5. Descargar e instalar las herramientas de Windows preconstruidas para Raspberry pi.
6. Al presionar configurar con los módulos por defeco se seleccionó como generador Unix Makefiles, con la opción para compilación cruzada. sus parámetros y librerías fueron el medio necesario para establecer la compilación del protocolo de comunicación MQTT.

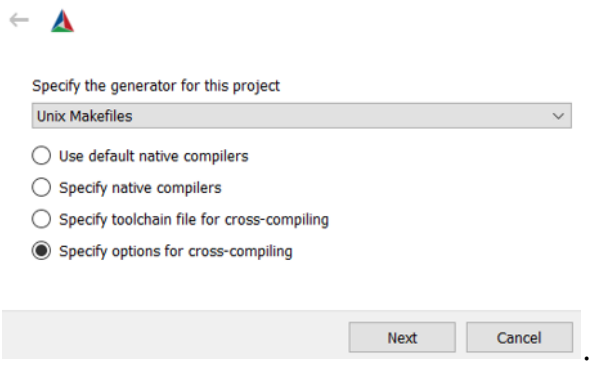



**Figura 1.** Selección de tipo de proyecto

7. Seleccionar las herramientas preconstruidas resultado de la instalación en el paso 5.

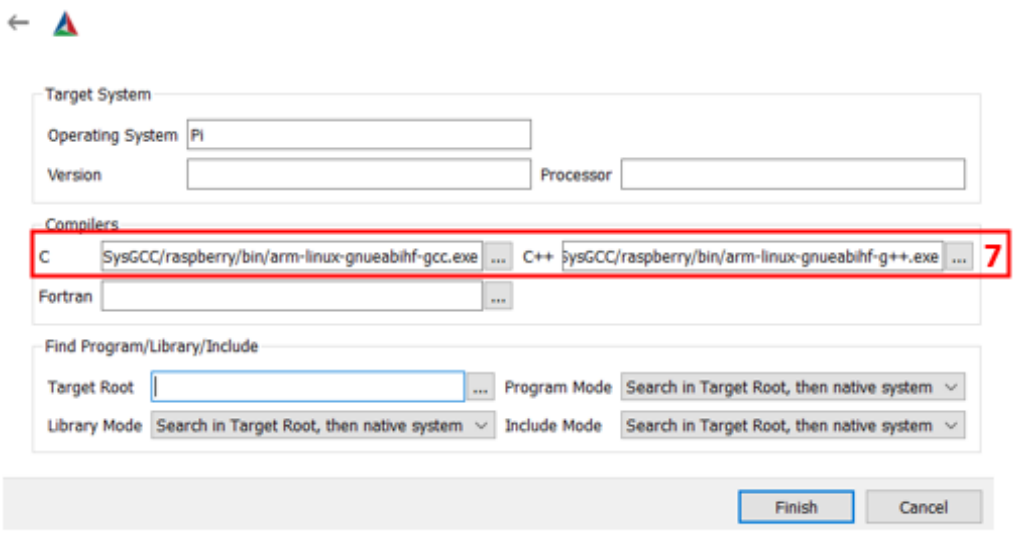



**Figura 2.** Dirección de cada archivo

8. **Configurar**, **generar** y **abrir el proyecto.**

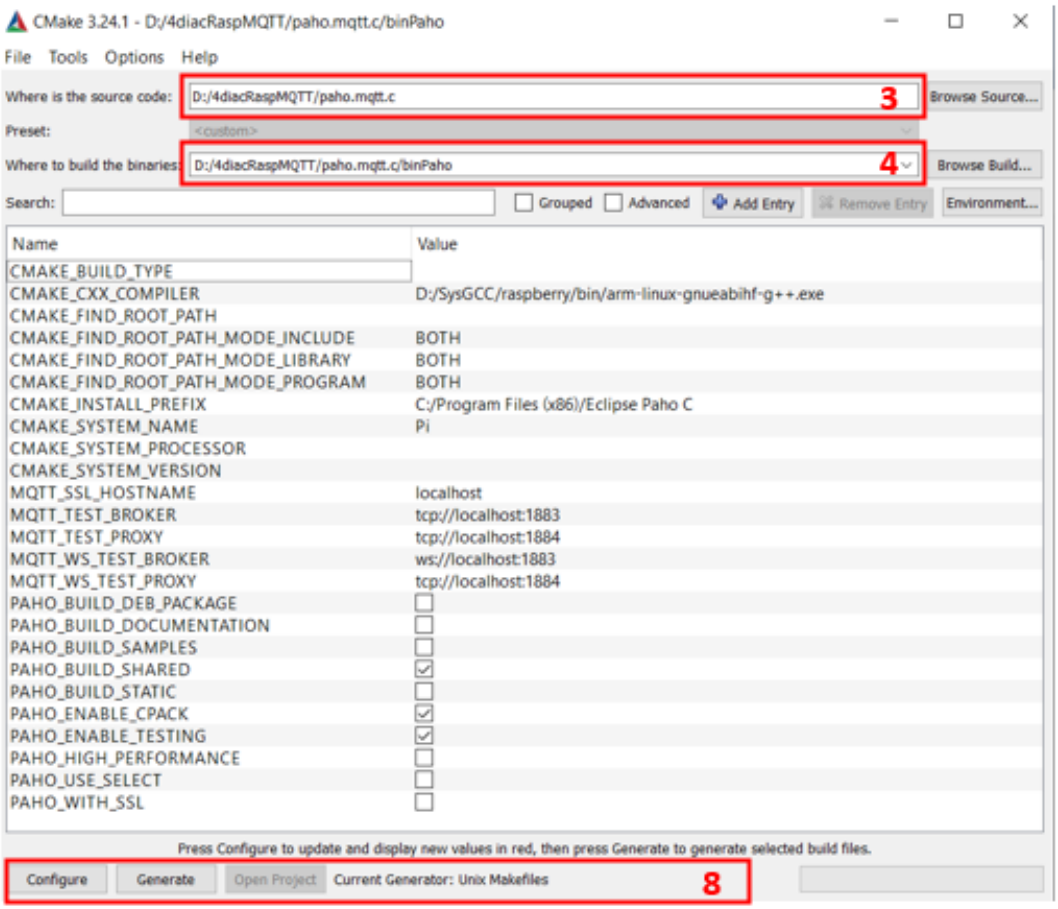



**Figura 3.** Proyecto generado en CMake

9. Compilar la solución usando las herramientas de compilación cruzada, generando la **biblioteca MQTT Paho.**
10. En una nueva ventana de CMake, configurar los archivos de compilación Forte.
11. En el requerimiento del código fuente, seleccionar el directorio del código fuente **Forte**.
12. En el directorio de salida, otorgar la dirección de la carpeta a crear con el nombre "build".
13. Agregar y configurar las siguientes librerías:

- FORTE_ARCHITECTURE Posix
- FORTE_COM_ETH
- FORTE_COM_FBDK
- FORTE_COM_PAHOMQTT
- FORTE_MODULE_CONVERT
- FORTE_MODULE_Container_1
- FORTE_MODULE_IEC61131
- FORTE_MODULE_UTILS

14. Agregar los parámetros sobre la librería **MQTT PAHO** para incluir los recursos finales, resultado de la generación de la biblioteca MQTT Paho.
15. **Configurar**, **generar** y **abrir el proyecto.**

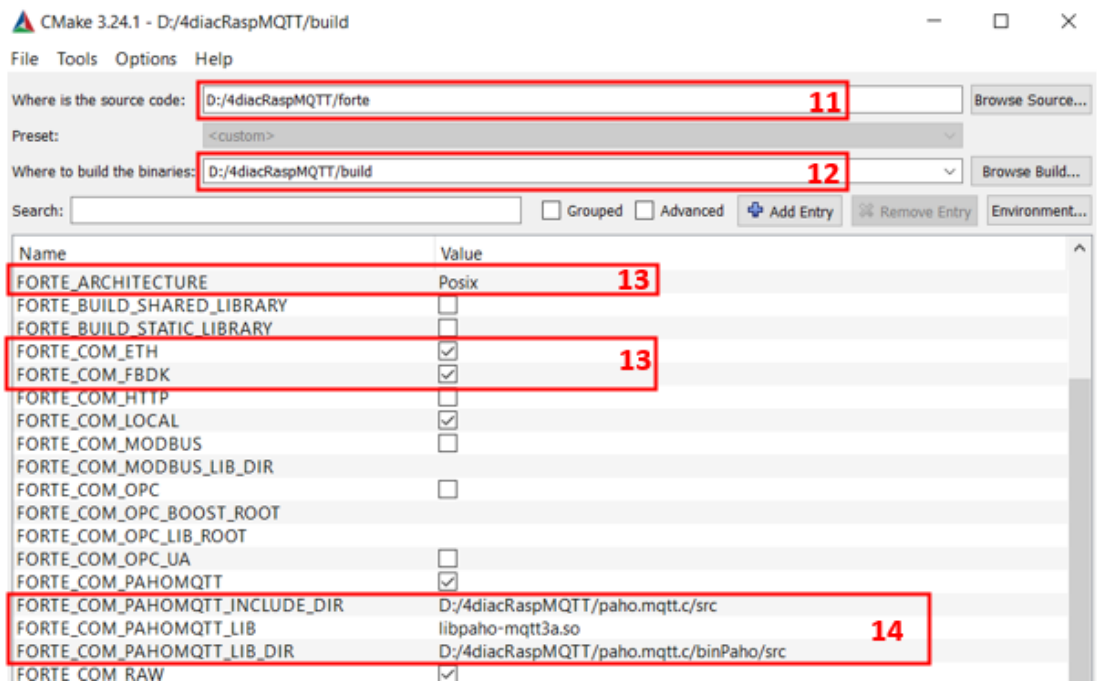



**Figura 4.** Proyecto cargado

16. Una vez compilada la solución con las herramientas de compilación cruzada, se genera el archivo Forte con todas las librerías configuradas anteriormente. Al cargarse el Forte en el contenedor respectivo es necesario cargar el archivo paho-mqtt3a.so en la carpeta lib del sistema, debido a que el ejecutable Forte requiere el archivo para el correcto funcionamiento de la librería.

**Implementación de Forte en contenedores**

Se crearon dos carpetas llamadas Container1 y Container2 haciendo referencia a cada contenedor los cuales abarcaran los archivos necesarios para construir la imagen base de los contenedores.

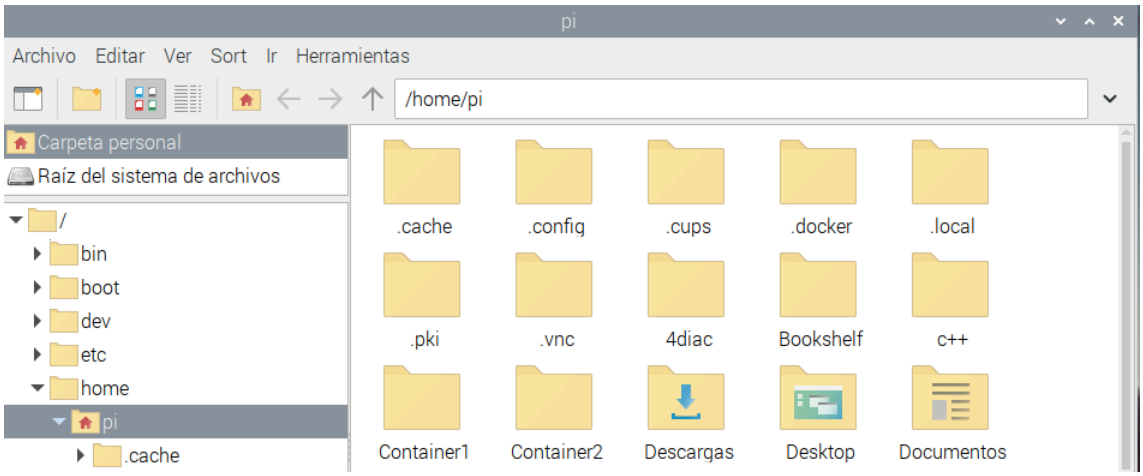



**Figura 5.** Dirección del archivo

Además del archivo Forte y librerías se creó un archivo Dockerfile en el cual se especificó las instrucciones que se ejecutarán para llevar a cabo la construcción de la nueva imagen.


**Figura 6.** Librerías utilizadas

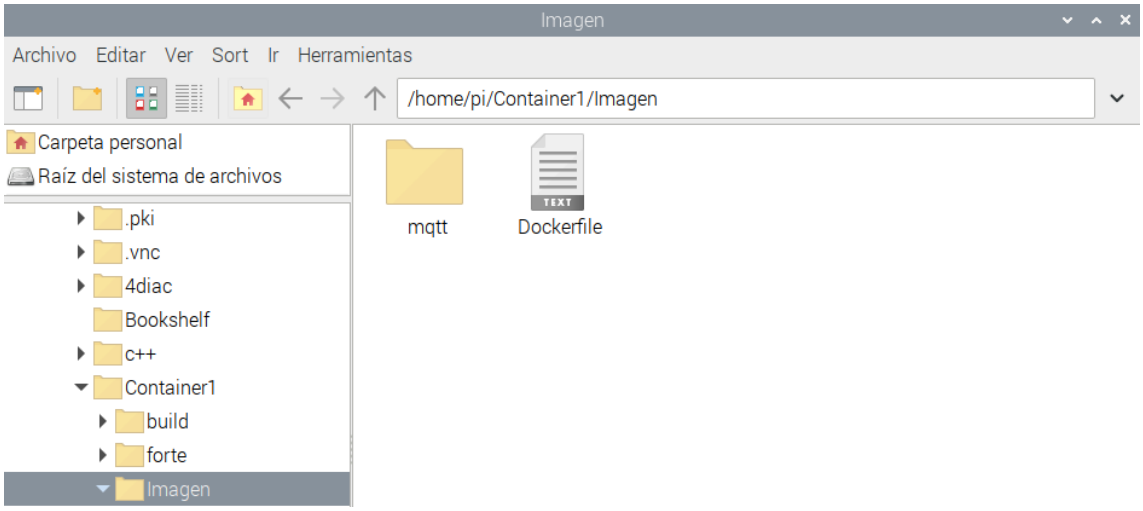



**Figura 7.** Programas creados

Cada archivo se ubicó en una carpeta diferente para mantener el orden y comprensión. La carpeta Forte contiene el runtime Forte y su respectivo forte.fboot el cual sirve para arrancar el dispositivo automáticamente con las configuraciones especificadas en 4diac IDE y la carpeta library abarca archivos de extensión so resultado de la compilación, mismos que son necesarios para el arranque de Forte.

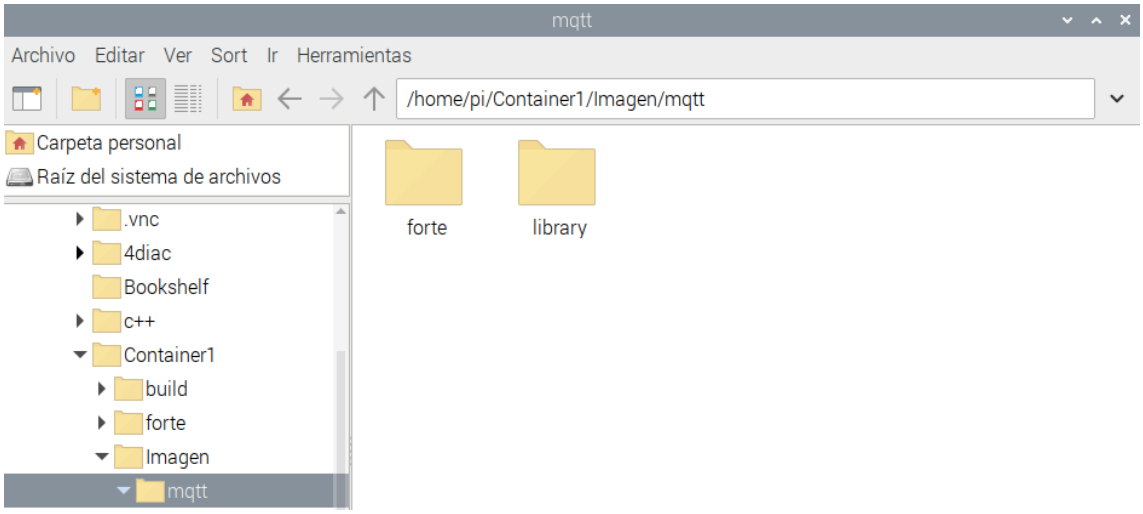



**Figura 8.** Carpetas para el arranque de Forte

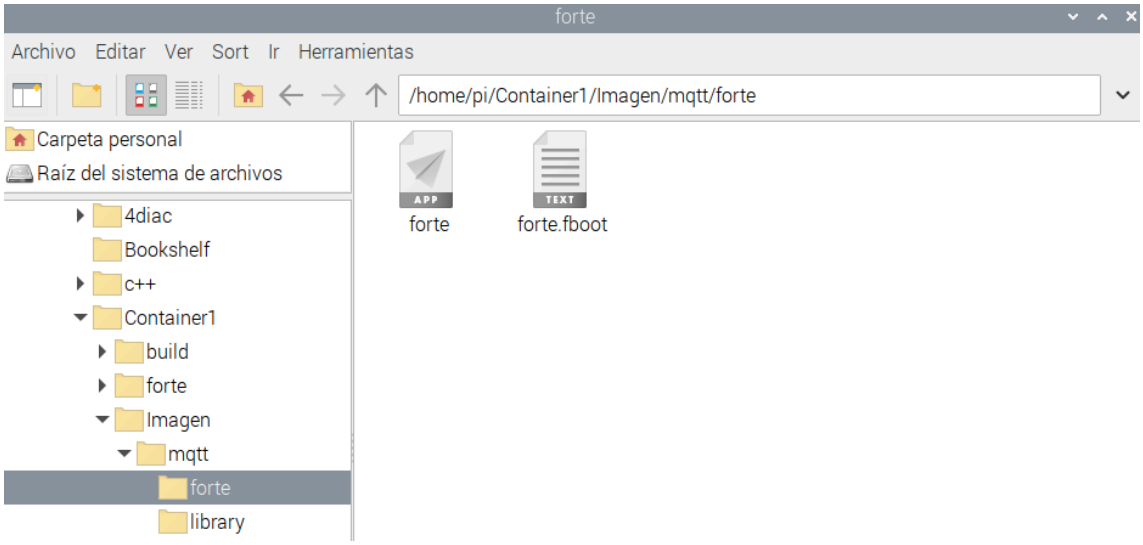



**Figura 9.** Archivos para el arranque de Forte

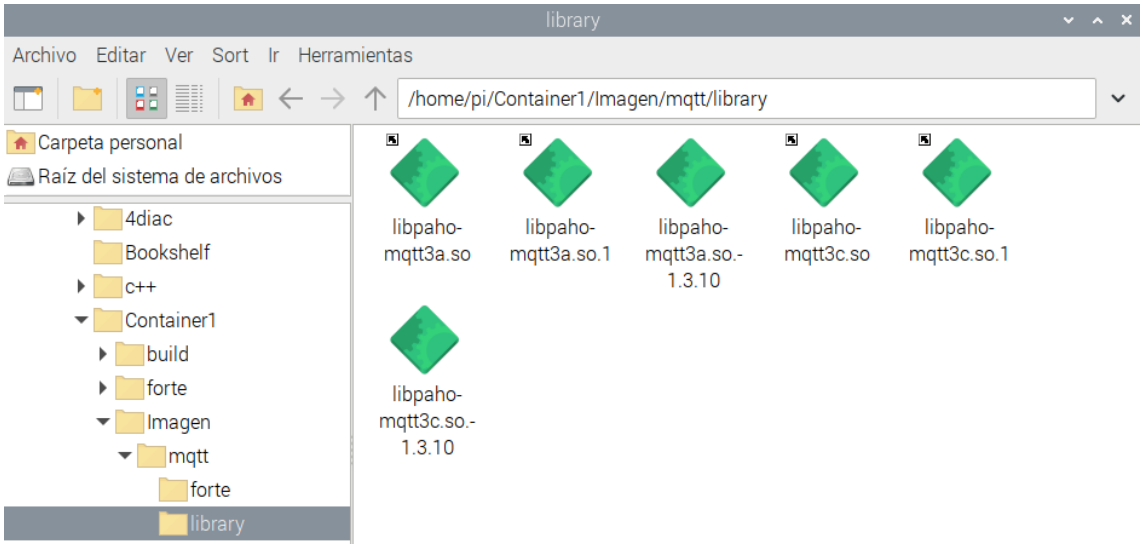



**Figura 10.** Librerías para el arranque de Forte

### Creación de imagen en Docker

Desde la terminal ubicar el archivo Dockerfile ejecutando el comando siguiente:

$docker build -t “Nombre de la imagen a crear”.

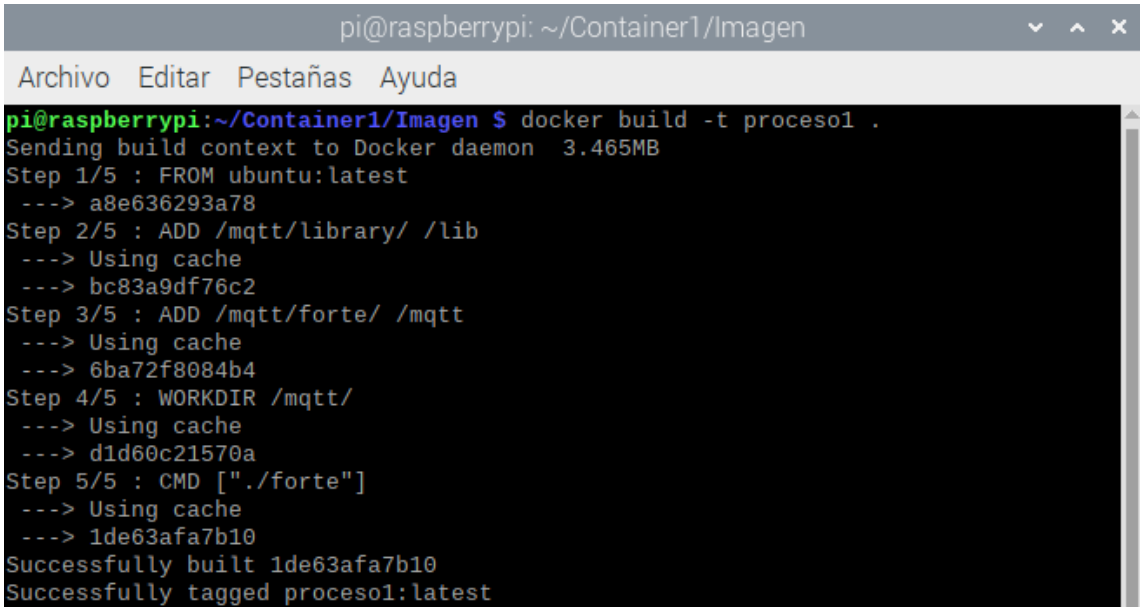



**Figura 11.** Programación en software de Raspberry Pi

Inspeccionando en el repositorio local Docker se observa la creación de la imagen del proceso.

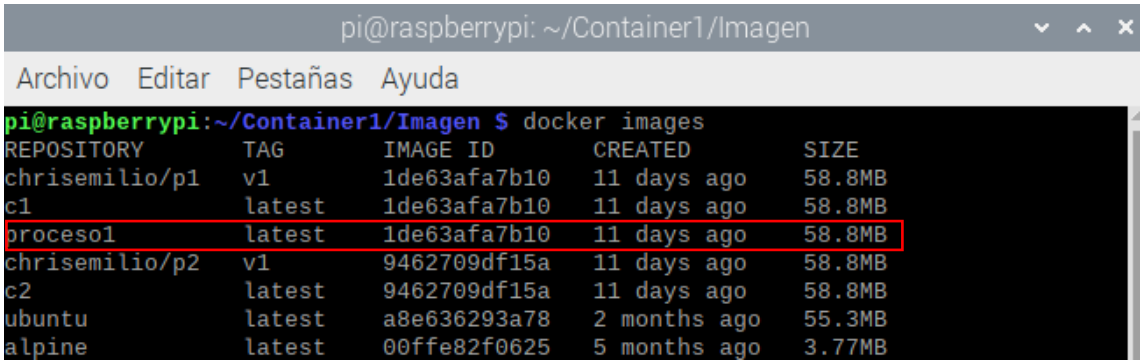



**Figura 12.** Muestra de la imagen creada

Se realiza una prueba del funcionamiento del contenedor para posteriormente arrancar el mismo desde 4diac IDE y su familiarización con la nube.

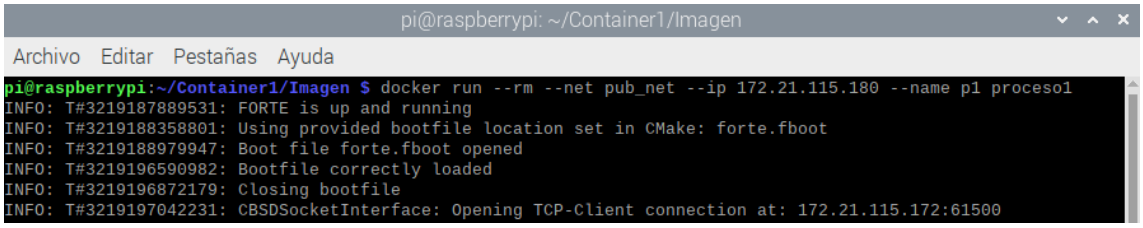



**Figura 13.** Arranque del programa

**Parámetro de comparación entre la IPC y la Raspberry Pi**

El trabajo de investigación al basarse en protocolos de comunicación que se adapten al sistema cuenta con los parámetros de estudio descritos en la Tabla 9.

**Tabla 9.** Parámetros de comparación entre dispositivos

| Parámetro | Descripción |
|---|---|
| Tiempo de respuesta | El tiempo de respuesta del hardware debe ser el mínimo debido a los manejos de grandes cantidades de datos. |
| Fiabilidad | El sistema debe tener un grado de protección frente a la seguridad y confidencialidad de los datos. |

# CAPÍTULO III

# RESULTADOS Y DISCUSIÓN

## 3.1 Desarrollo de la propuesta

El trabajo investigativo muestra el desarrollo de un sistema de control basado en la comunicación contenedores para la industria 4.0, para lo cual se crearon diferentes FBs y runtimes Forte, de acuerdo a las necesidades del proceso y los protocolos de comunicación a utilizar, los FBs creados se diseñaron mediante el software 4diac IDE y fueron compilados mediante el software Cmake para de esta manera crear el runtime Forte de acuerdo al diseño que se estableció para comunicación los diferentes dispositivos. Además, se crearon contenedores que se almacenaron en un repositorio, lo mismos contenían las imágenes de los procesos a controlar. Por otro lado, se diseñó una interfaz gráfica basada en Python y Lenguaje de Marcas de Hipertexto (HTML) con la finalidad de que el usuario pueda supervisar los procesos.

El sistema de control propuesto se basó en el uso de los siguientes protocolos de comunicación; para la comunicación entre el Computador Industrial y Raspberry PI se utilizó el protocolo de comunicación TCP/IP, en cuanto al envío de datos desde el Computador Industrial a la interfaz gráfica se usó el protocolo de comunicación MQTT. Los protocolos de comunicación utilizados fueron compilados con las demás librerías requeridas en los runtimes FORTE.

Para la adquisición de datos desde el proceso mediante la IPC se desarrollaron FBs que permiten adquirir datos de las entradas digitales y analógicas por medio del uso de la DLL de ODK 1500S de memoria compartida, también se crearon FBs para el envío de datos a las salidas digitales y analógicas de la IPC. Además, se crean FBs para subir, bajar, correr y parar los contenedores que poseen las imágenes de los procesos, para de esta manera elegir que contenedor correr desde el repositorio de imágenes, como se muestra en la Figura 14.

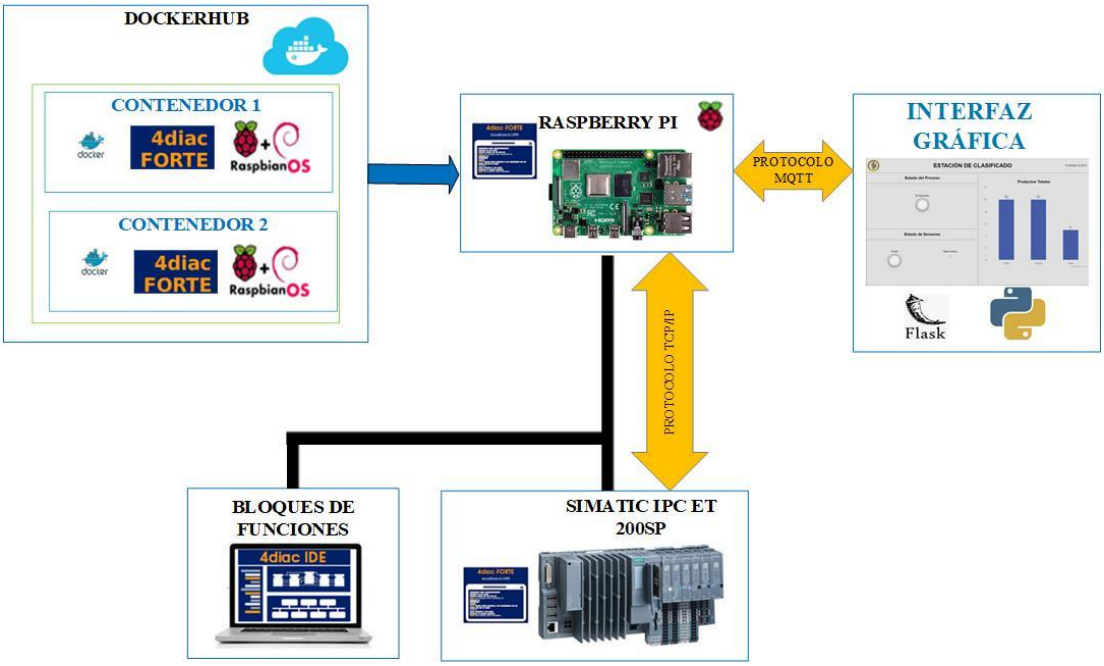



**Figura 14.** Esquema general del sistema de control

El sistema de control basado en la comunicación contenedores para la industria 4.0, de esta manera cumple cuatro de los cinco niveles de la pirámide de la automatización industrial como se observa en la Figura 15. Llegando hasta el nivel de integración el cual corresponde según la literatura de la pirámide al inicio de la industria 4.0, debido a que en dicho nivel los procesos presentan una mayor flexibilidad y eficiencia.

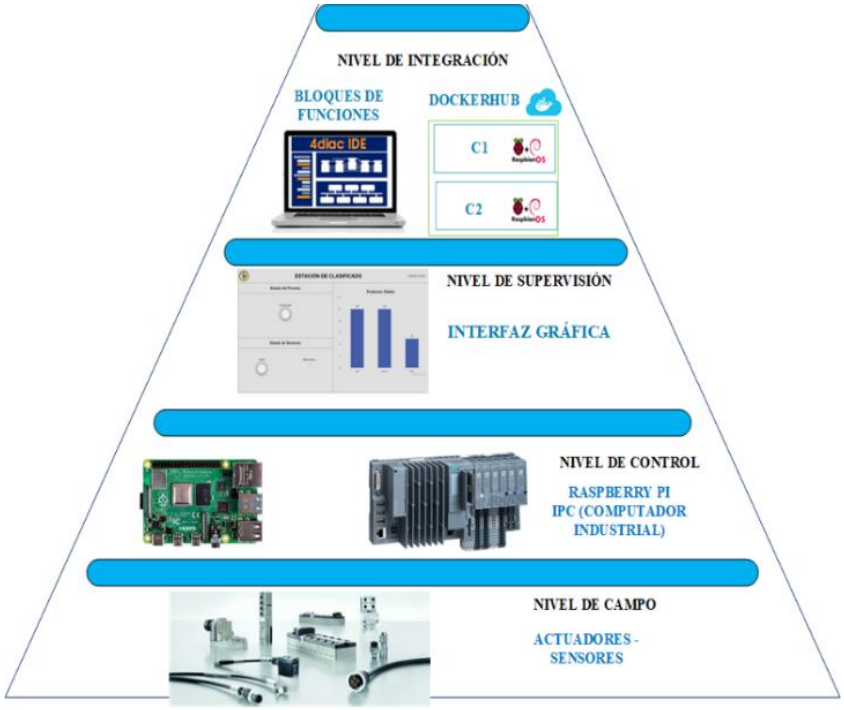



**Figura 15.** Pirámide de la automatización industrial del sistema de control

### 3.1.1 Diseño del entorno virtual mediante el estándar IEC 61499 basado en protocolos de comunicación MQTT y TCP/IP

**Bloques Funcionales para la adquisición/envió de datos de la IPC**

Los bloques relacionados a las entradas y salidas de la IPC se desarrollaron bajo: el tipo de bloques funcionales básicos que trabajan con "INIT" para la inicialización del FB; "REQ" para adquirir la señal de solicitud para la ejecución; "QI" que habilita el bloque funcional y; "PARAMS" que registra la dirección de la entrada o salida digital o analógica con la que se trabaja de la manera: 'I0.0' o 'AI0' en el caso de ser entrada digital o analógica respectivamente y; 'Q0.0.' o 'AO0' en el caso de ser salida digital o analógica respectivamente. La Figura 16, muestra el FB para las entrada y salidas digitales de la IPC.

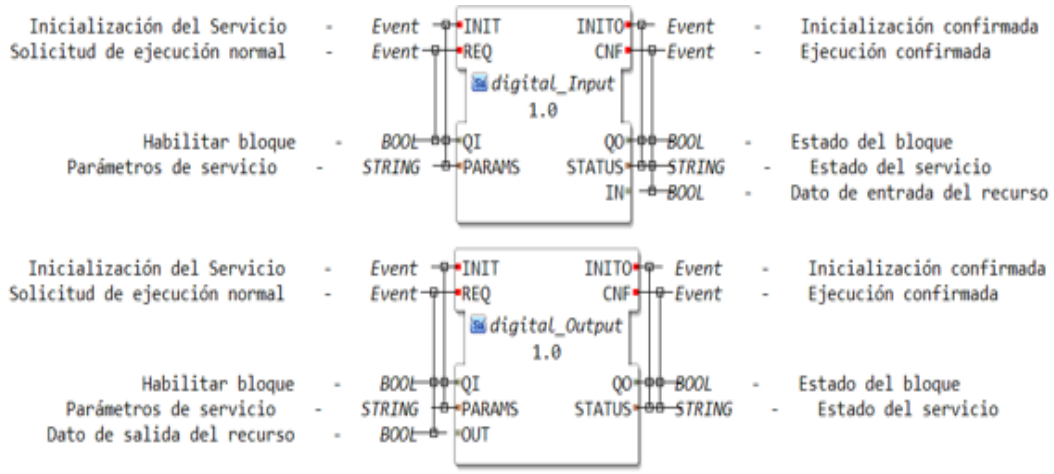



**Figura 16.** Bloques Funcionales para las entrada y salidas digitales de la IPC

Análisis

En el caso de que el bloque funcional sea una salida digital o analógica de la IPC se tiene: el parámetro "OUT", que permite la escritura de las salidas del controlador; el parámetro "INITO", que permite tener una señal que confirma la inicialización al siguiente FB; el parámetro "CNF" que envía una señal de ejecución confirmada al siguiente FB; el parámetro "QO" que muestra el estado del Bloque y; el parámetro "IN" que es la lectura del dato de entrada digital o analógica del controlador.

Para el estudio, la memoria compartida sirve para adquirir y enviar datos desde el proceso que se está ejecutando en la tarjeta Raspberry Pi a un contenedor por medio

del IPC. Esto implica que se ha establece una conexión entre el proceso y el IPC a través de una porción de memoria compartida, lo que permite que ambos puedan acceder a la misma información y compartir datos de forma rápida y eficiente. Además, se mapea la memoria para encontrar la información adquirida por la DLL de ODK del IPC, lo que significa que se realiza un análisis de la estructura para determinar las direcciones en las que se encuentran los datos relevantes. Una vez que se adquiere la información, se procesa de acuerdo con el tipo de entrada o salida y se envía a la aplicación Forte para su manipulación en la IPC, como se presenta en el diagrama de la Figura 17.

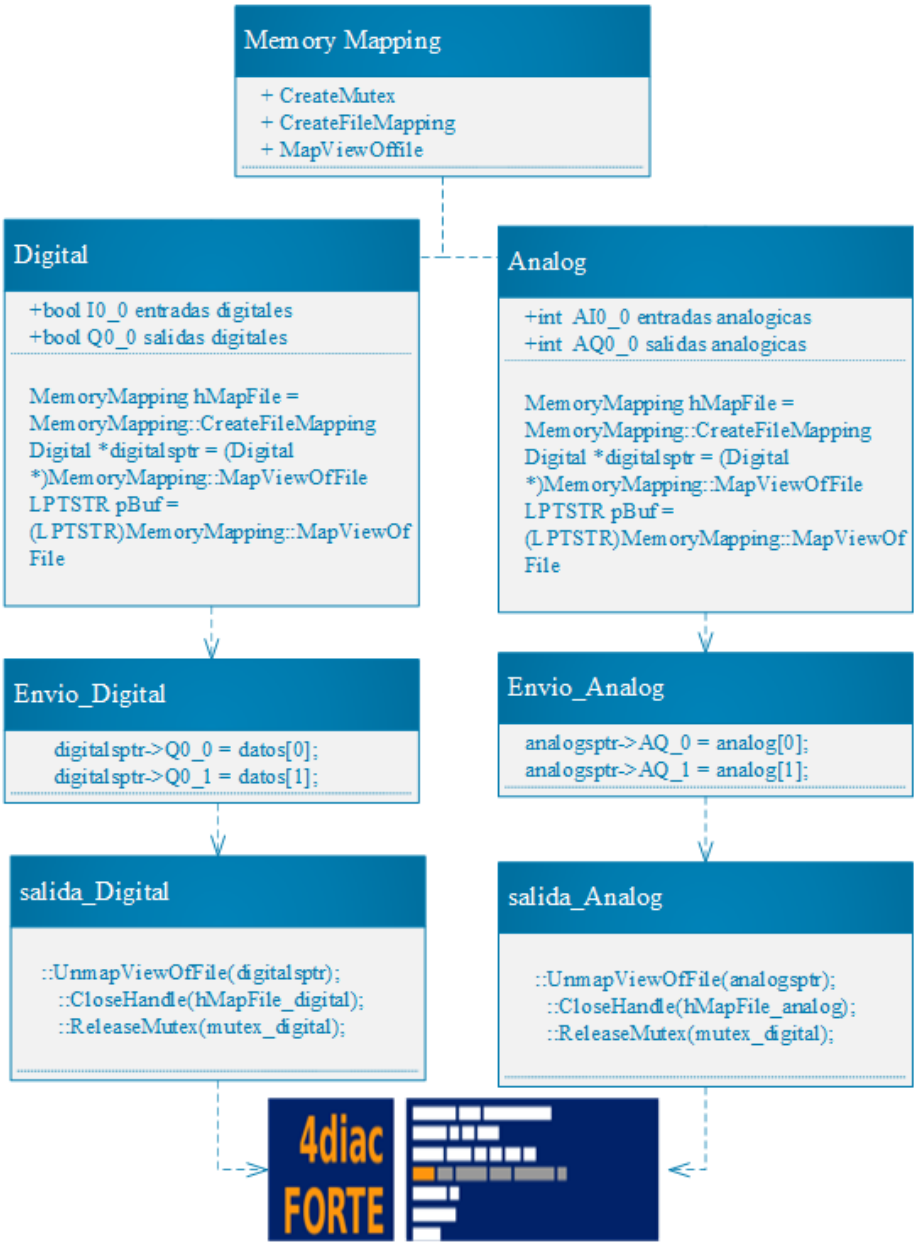



**Figura 17.** Diagrama de clase de la aplicación Forte para la adquisición de datos de las entradas y salidas de la IPC SIEMENS

### 3.1.2 Bloques Funcionales de los procesos automatizados

Los FBs se crearon mediante el software 4diac, dichos bloques se diseñaron de acuerdo con las especificaciones del estándar IEC 61499, partiendo de una plantilla básica que proporcionan el software, la Figura 18, muestra el resultado de la modificación.

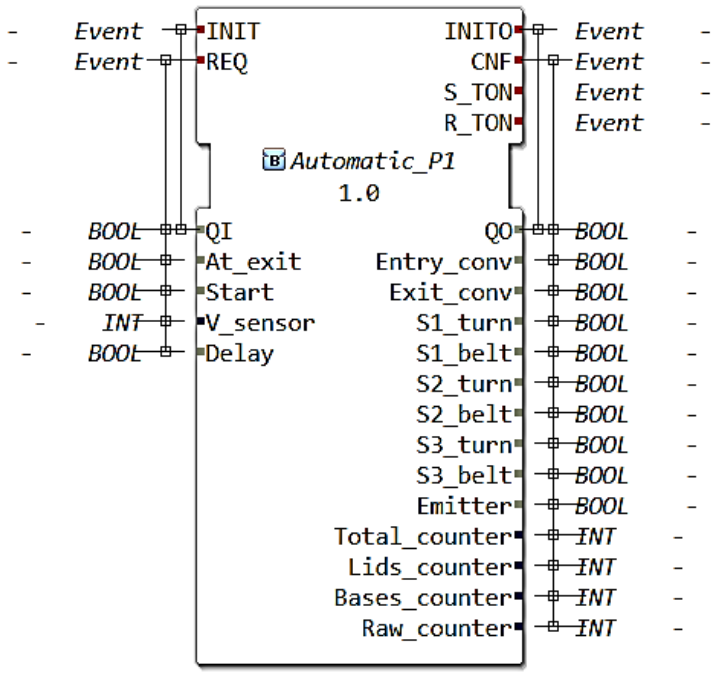



**Figura 18.** Bloque funcional del Proceso uno creado

El FB consta de eventos que son inicializados de acuerdo con la señal en la entrada "INIT" y "REQ", en cuanto a los datos son enviados de acuerdo a la señal de ejecución presente en el "CNF", obteniendo dos señales de salida que controlan el paro del proceso. Además, se tiene "INITO" que permite conectar con otro FB para activarlo. El bloque funcional se configuró en el Control Gráfico de Ejecución (ECC), el cual une los eventos con los datos activando "INIT" y posteriormente el "REQ", hasta recibir la señal de "Start" que, en conjunto a "REQ" y otros requisitos, darán inicio al proceso automatizado, como se observa en la Figura 19.

El FB del proceso 2 se creó con una estructura similar a la del proceso 1, con la diferencia que se presenta una entrada adicional de tipo entero que se detiene una vez que se cumple el tiempo de ciclo programado. El FB del proceso 2, se configuró en el ECC de una forma similar a la del proceso 1, con la diferencia que se presenta el evento del contador que una vez llegado al setpoint el proceso termina.

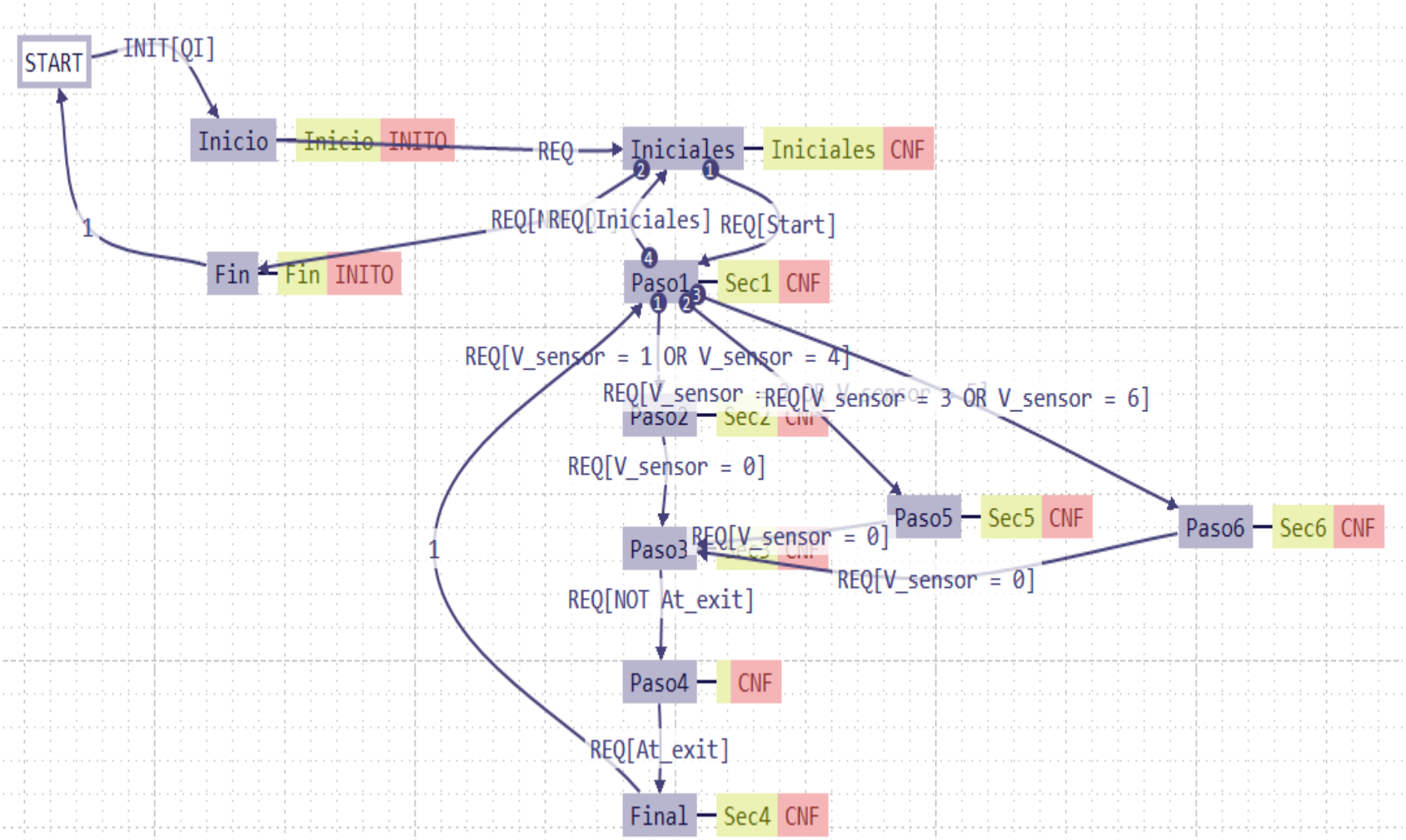



**Figura 19.** Gráfico de Control de Ejecución (ECC) del proceso 1

### 3.1.3 Bloques Funcionales de los protocolos de comunicación según los parámetros identificados

**Bloques Funcionales de acuerdo con los parámetros del protocolo de comunicación TCP/IP**

Los FBs del protocolo de comunicación TCP/IP creados fueron diseñaron a partir de los FBs básicos de cliente – servidor que el software proporciona, de forma que: el FB del cliente se encuentra en los contenedores de la Raspberry pi y; el FB del servidor se encuentra en el runtime (FORTE) de la IPC.

El FB del cliente presenta una estructura similar al FB básico del software, con la diferencia que en este caso se trabaja con el "ID" en donde es establece el parámetro de comunicación mediante la dirección y el puerto de enlace necesarios para la conexión cliente/servidor. Además, presenta los parámetros de ingreso y salida de datos desde el proceso hacia el servidor, como se muestra en la Figura 20.

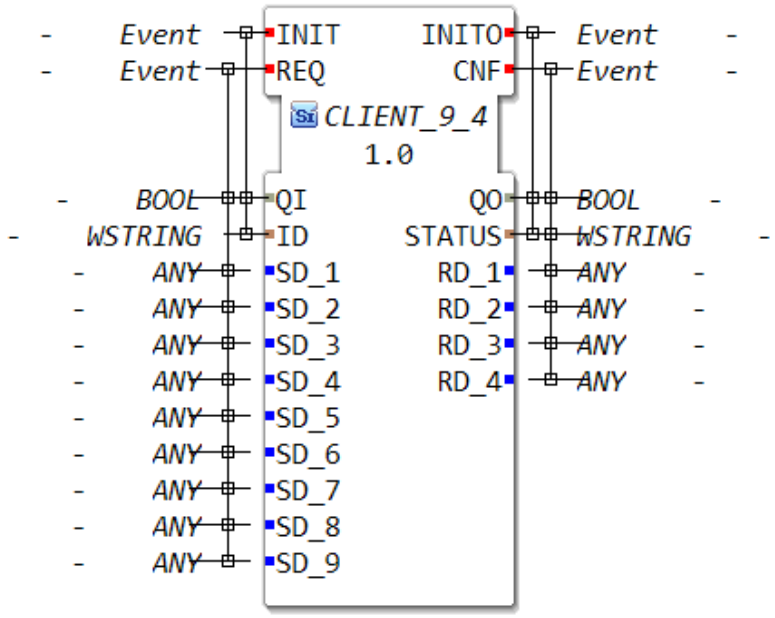



**Figura 20.** Bloque Funcional- Cliente en base a los parámetros identificados

El FB del servidor presenta una estructura similar a la del bloque funcional del cliente de estableciendo del parámetro de comunicación mediante la dirección y el puerto de enlace, además, de los formatos de las entradas y salidas para el envío y recepción de datos desde la IPC, como se observa en la Figura 21.

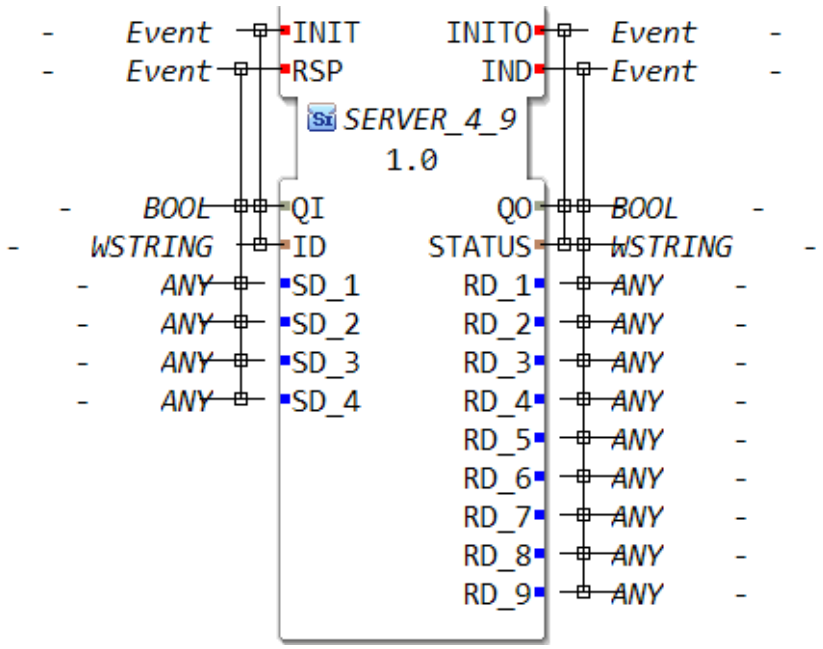



**Figura 21.** Bloque Funcional- Servidor en base a los parámetros identificados

**Bloques Funcionales de acuerdo con los parámetros del protocolo de comunicación MQTT**

El FB del protocolo de comunicación MQTT creado se diseñó a partir del bloque funcional "Publish" que el software proporciona como se observa en la Figura 22 y en la Figura 23. El Bloque Funcional del publicador presenta una estructura similar a la del FB del cliente y del servidor utilizados en el protocolo de comunicación TCP/IP, con la diferencia que los parámetros para la publicación de datos trabajan de acuerdo con la estructura del siguiente bróker.

- "raw []. mqtt[tcp://test.mosquitto.org:1883, forte, Proceso #/Variable #",

Este se modifica de acuerdo a los tópicos del proceso. Además, presenta parámetros de ingreso acorde a los tipos de variables que pueden ser de INT (dato entero) y BOOL (dato booleano).

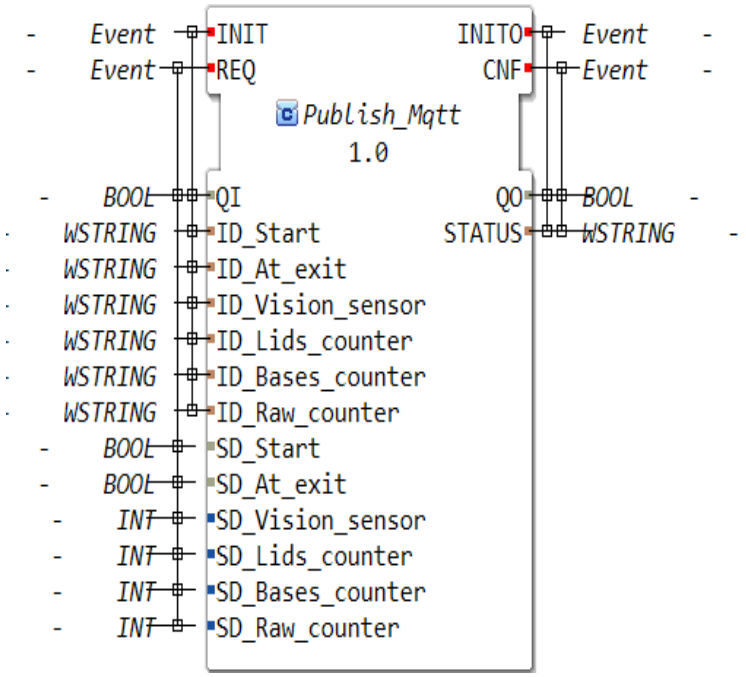



**Figura 22.** Bloque Funcional compuesto del protocolo de comunicación MQTT

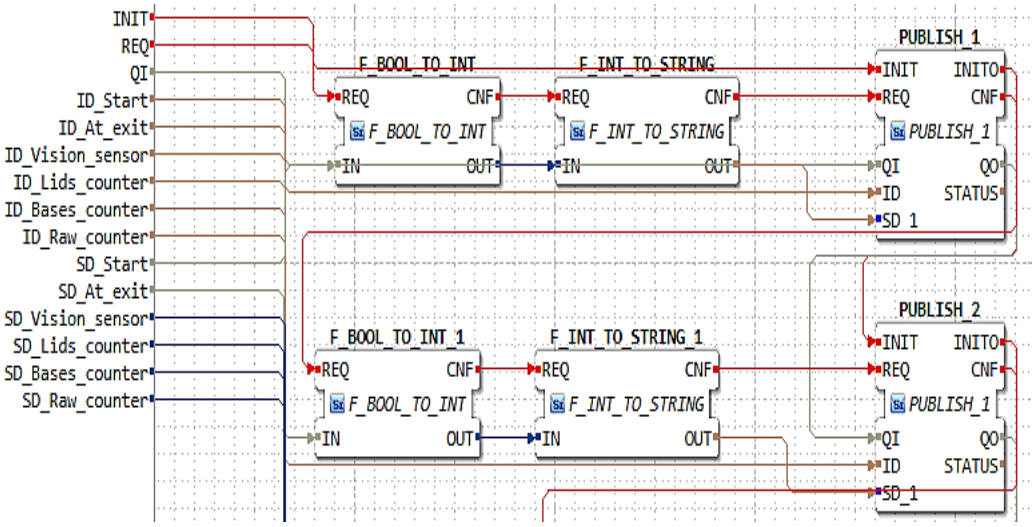



**Figura 23.** Estructura interna del FB de la comunicación MQTT

### 3.1.4 Bloques Funcionales para el Control de Contenedores Docker

Los componentes básicos que controlan los contenedores de Docker se fundamentaron en los parámetros de inicio y detención del contenedor y su capacidad de carga y descarga desde el repositorio en la nube. El FB para el arranque y paro de los contenedores presenta una estructura idéntica al FB básico como se muestra en la Figura 24, con la diferencia que en el ingreso "PARAMS" se registra los comandos como String.

**Comandos para arrancar contenedores**

- 'run; “Nombre que se llamará al contendor”; “IP estática que se le asignará al contenedor”; “Número de proceso”'

**Comandos para parar y matar contenedores**

- 'stop; “Nombre del contenedor”'
- 'kill; “Nombre del contenedor”'

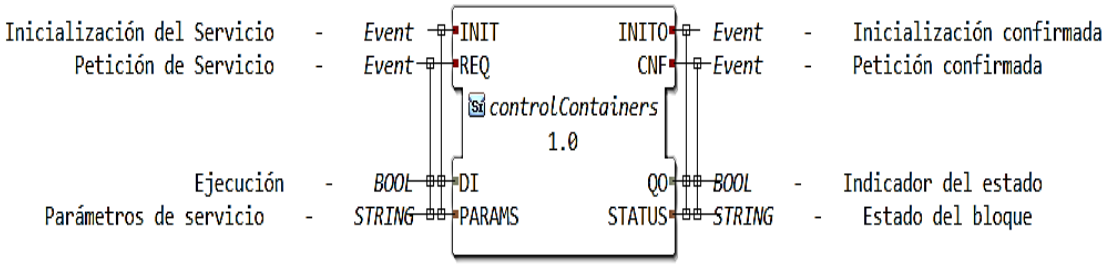



**Figura 24.** Bloque Funcional para arrancar y parar contenedores

El FB para subir y bajar contenedores desde la nube de igual manera presenta una estructura similar a la anterior con la diferencia que en el ingreso “PARAMS” se ingresa los siguientes comandos:

**Comandos para subir y bajar contenedores**

- Para subir un contenedor: 'push; “Número del proceso”
- Para bajar un contenedor: 'pull; “Número del proceso”'

### 3.1.5 Creación de contenedores de los procesos

Los contenedores se crearon desde la tarjeta Raspberry Pi, por lo cual se instaló y configuro Docker en dicha tarjeta, los archivos Forte generados por compilación cruzada de los procesos a controlar en conjunto con las librerías necesarias para el funcionamiento de estos se incluyeron dentro de la tarjeta Raspberry Pi, con los cuales se crearon los contenedores identificados como Proceso1 y Proceso2 respectivamente.

De esta manera los contenedores creados desde la Shell de Raspbian presentan la estructura que se visualiza en la Figura 25.

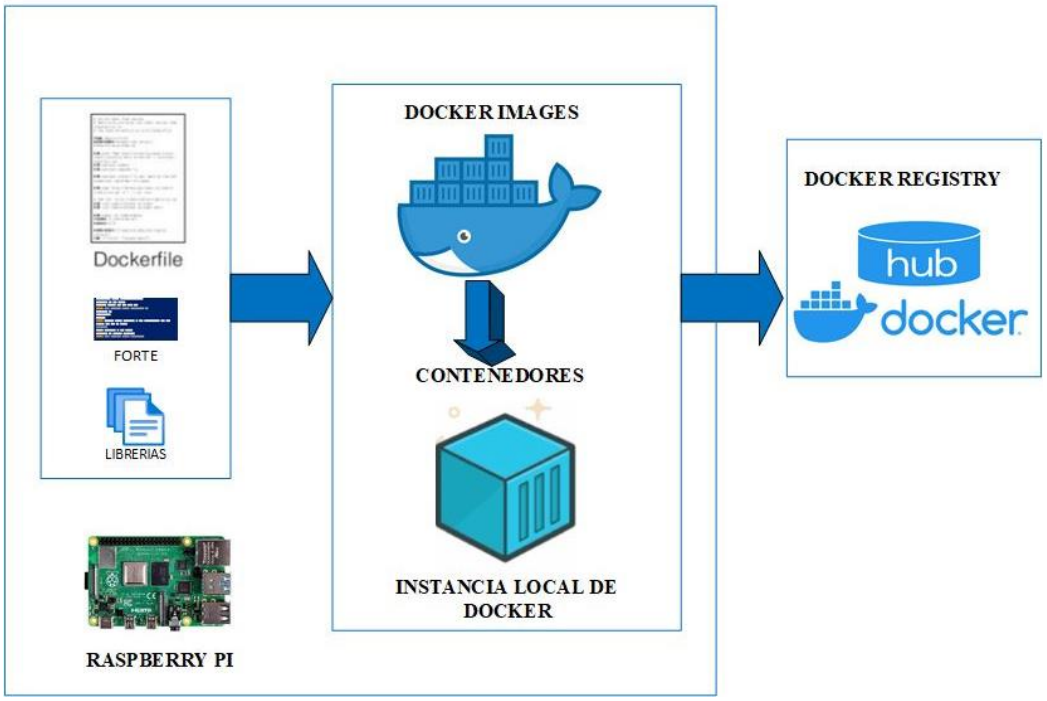



**Figura 25.** Estructura de los Contenedores Docker

Los contenedores y/o dispositivos se identificaron por medio de la asignación de una dirección IP estática configurada desde 172.21.115.172 hasta la 172.21.115.176, la Figura 26, muestra el proceso.

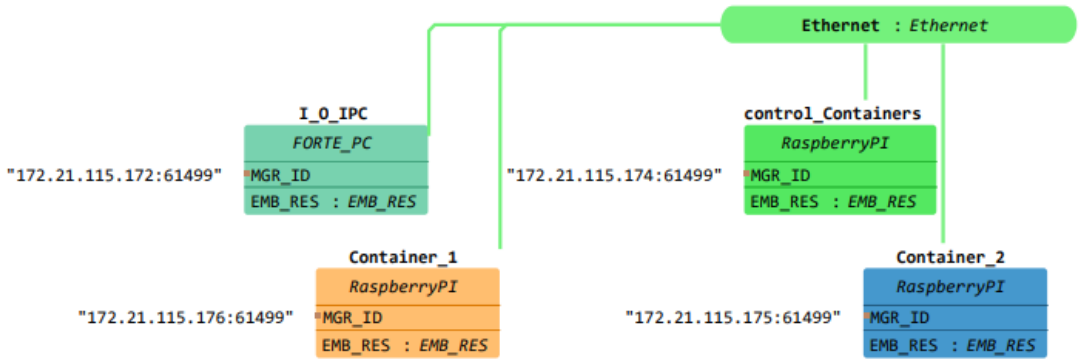



**Figura 26.** Dispositivos conectados virtualmente a la red

Dentro de cada contenedor se encuentran los archivos necesarios para la creación de las imágenes de los procesos, los cuales se ejecutan, como se observa en la Figura 27.

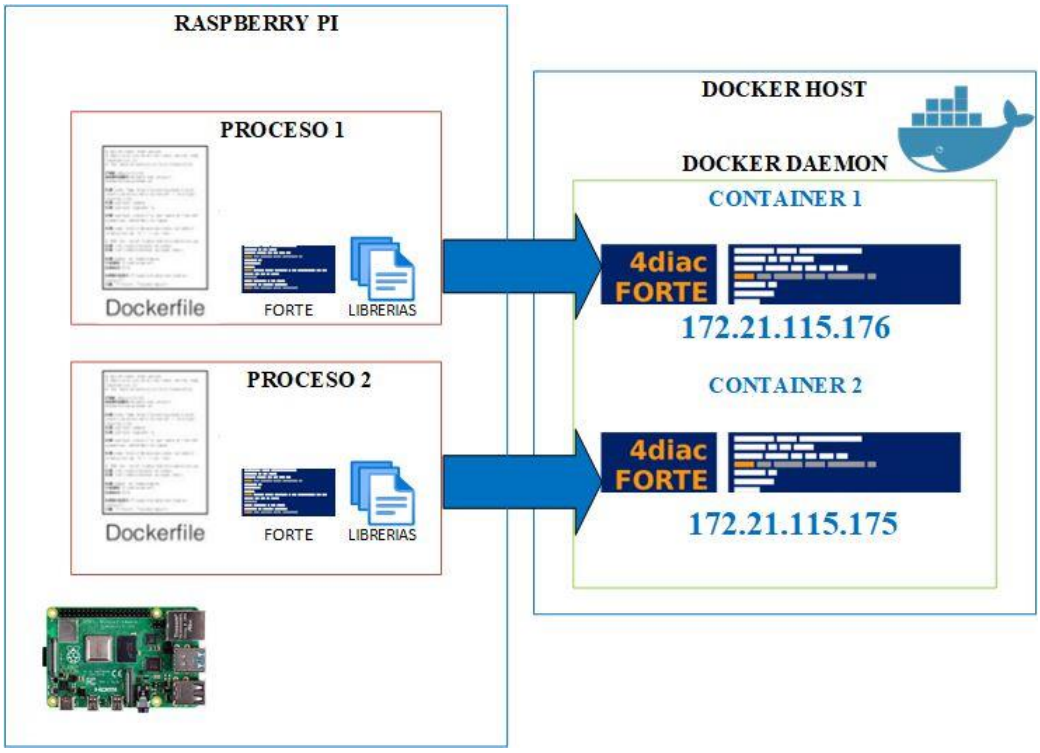



**Figura 27.** Ejecución de los contenedores en Raspberry Pi

### 3.1.6 Diseño del entorno de control para los diferentes dispositivos

El diseño de entorno de control de los diferentes dispositivos se desarrolló a través de la implementación de los FB creados con los nombres de "proceso1" y "proceso2", para la adquisición/envío de datos de las entradas y salidas de la IPC. Los protocolos de comunicación usados para la conexión entre dispositivos, el envío de datos a la interfaz gráfica y para el control de los contenedores fueron TCP/IP y MQTT.

Para el proceso1, la aplicación del entorno de control se distribuye en los siguientes servicios de acuerdo a los dispositivos conectados a la red de 4diac, iniciando de esta manera con la adquisición de datos de las entradas y salidas digitales o analógicas de la IPC mediante memoria compartida por nombre, siendo los datos enviados hacia el contenedor que se encuentra en ejecución en la Raspberry Pi de donde se envían datos a la interfaz gráfica para la supervisión del proceso, de igual manera para proceso2 y por último se establece el control de los contenedores tanto para su descarga/carga como para la ejecución/detención de los mismos.

El sistema de control se basa en la adquisición y envío de datos de la IPC mediante los FBs de las entradas y salidas analógicas o digitales de la IPC y de los FBs del protocolo

de comunicación TCP/IP para la comunicación entre los dispositivos. En esta etapa se establece como servidor el IPC y como cliente los contenedores de los procesos, los cuales se ejecutan en la Raspberry Pi, como se observa en la Figura 28.

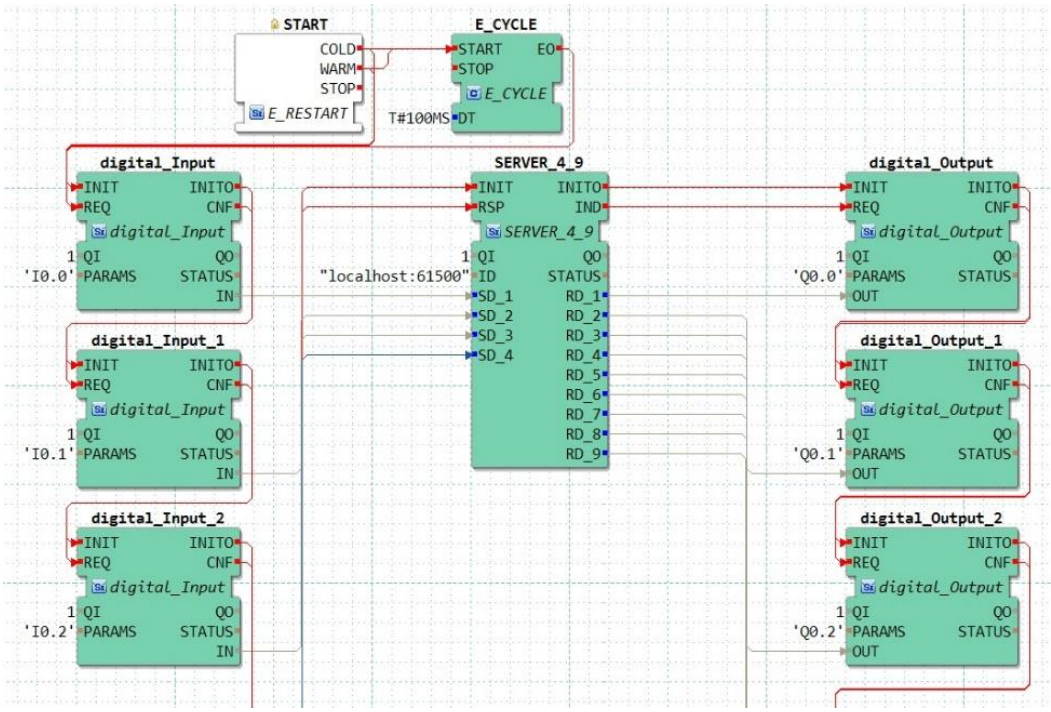



**Figura 28.** Adquisición y envió de datos desde la IPC

La Figura 29, muestra parte de la comunicación de los dispositivos mediante el protocolo de comunicación TCP/IP, en donde el FB del cliente recibe y envía los datos procesados a la IPC, además posee de un FB de publicación del protocolo MQTT con el que se envía los datos a la interfaz gráfica.

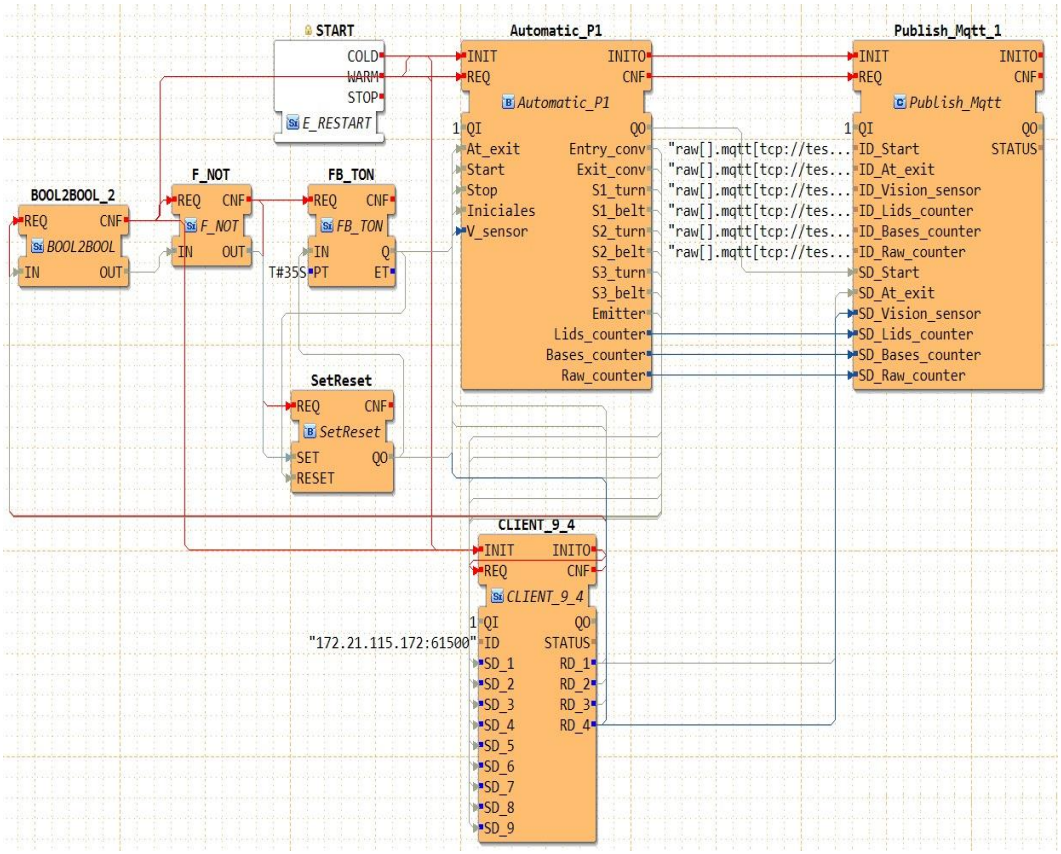



**Figura 29.** Control del proceso1

De la misma manera se tiene el control del proceso2 con la diferencia que se maneja el proceso mediante un contador, con el cual se controla el fin del proceso según el setpoint establecido, como se observa en la Figura 30.

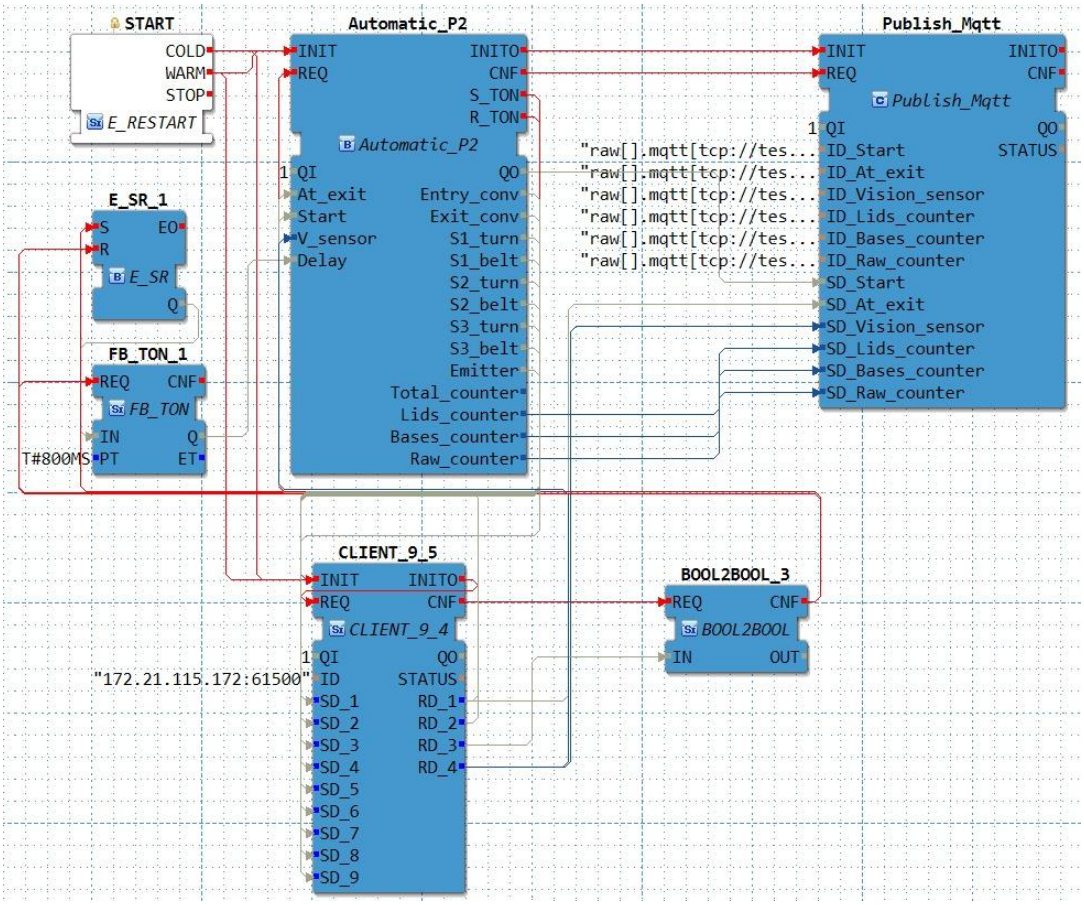



**Figura 30.** Control del proceso2

Docker Hub fue un repositorio del que se puede extraer contenedores según la necesidad de los procesos automatizados tanto en el dispositivo de control como en el repositorio de contenedores, como se muestra en la Figura 31.

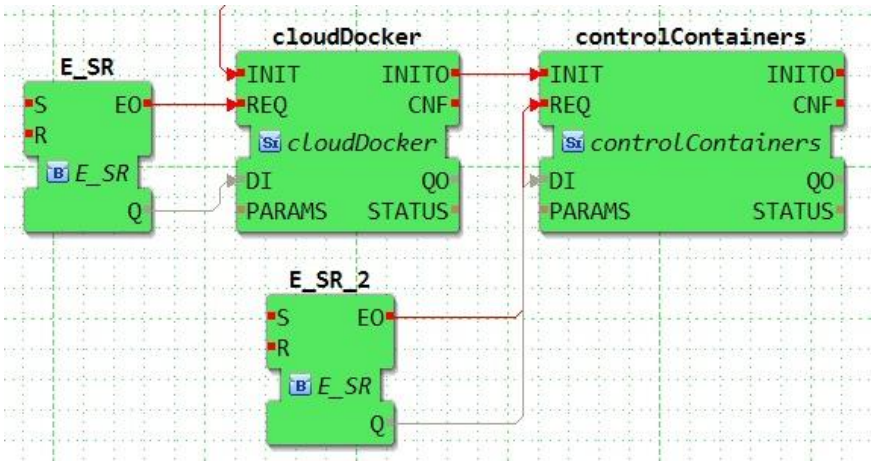



**Figura 31.** Control de contenedores

### 3.1.7 Comunicación entre la IPC y contenedores en ejecución en la tarjeta Raspberry Pi

La adquisición de los datos del proceso se realizó a través del uso de la IPC de la marca Siemens, creando una aplicación FORTE con las librerías necesarias para adquirir y enviar datos mediante el protocolo de comunicación TCP/IP. La comunicación entre los contenedores ejecutados en la Raspberry Pi y la IPC se muestra en la Figura 32.

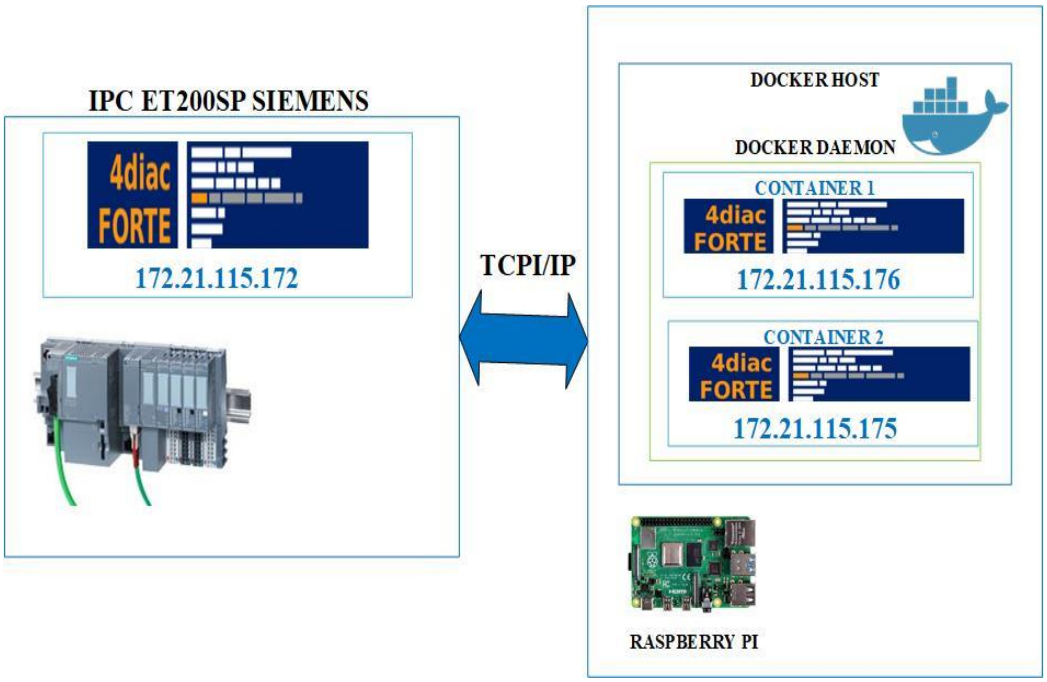



**Figura 32.** Comunicación entre Forte de IPC y Contenedores ejecutados en la tarjeta Raspberry

La aplicación FORTE se encuentra en la computadora industrial IPC funciona como servidor y los FORTE de los contenedores como clientes, enviando así los datos adquiridos por la IPC desde el proceso al cliente que podría ser cualquier contenedor que se esté ejecutando.

### 3.1.8 Diseño de la interfaz gráfica

Para la supervisión del proceso se diseñó una interfaz gráfica en base a HTML como se muestra en la Figura 33, con la cual se monitorea el estado de sensores desde una página Web en base a un bróker del protocolo de comunicación MQTT, permitiendo tener una adecuada interacción entre el usuario y el sistema de control industrial.

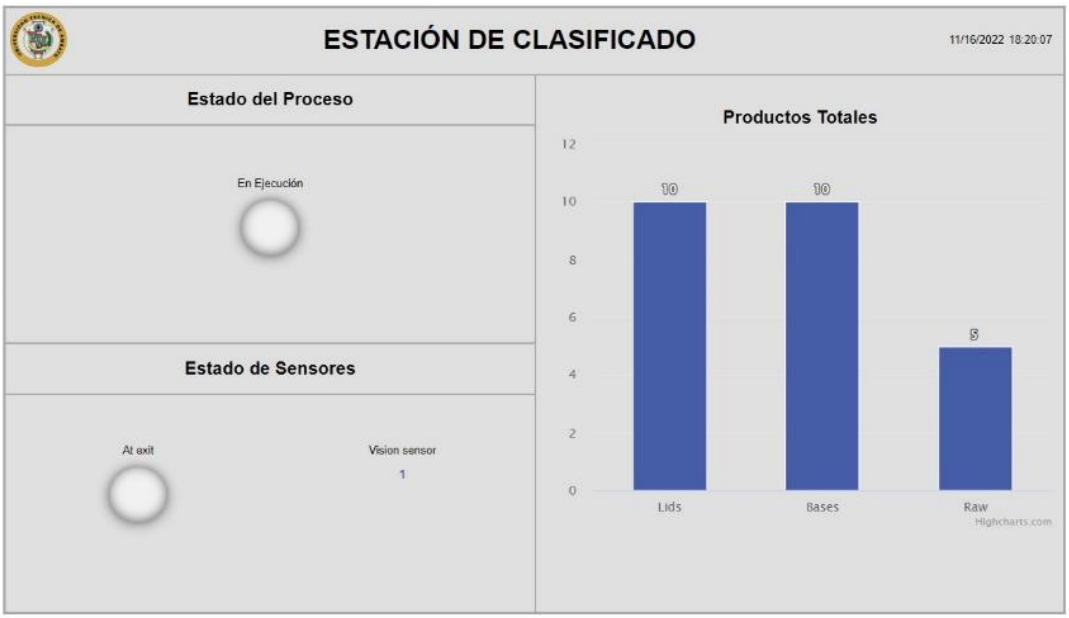



**Figura 33.** Interfaz gráfica para la supervisión del sistema de control

La interfaz gráfica, permite estar al tanto del estado de ejecución del proceso, los sensores y el total de productos clasificados. Se presenta la factibilidad de acceder a la misma desde cualquier parte del mundo, ya que internamente se encuentra configurada con todas las credenciales de acceso al bróker designado para la recepción de datos.

Los datos visualizados en la interfaz de supervisión son adquiridos de acuerdo con el protocolo de comunicación MQTT, como se observa en la Figura 34.

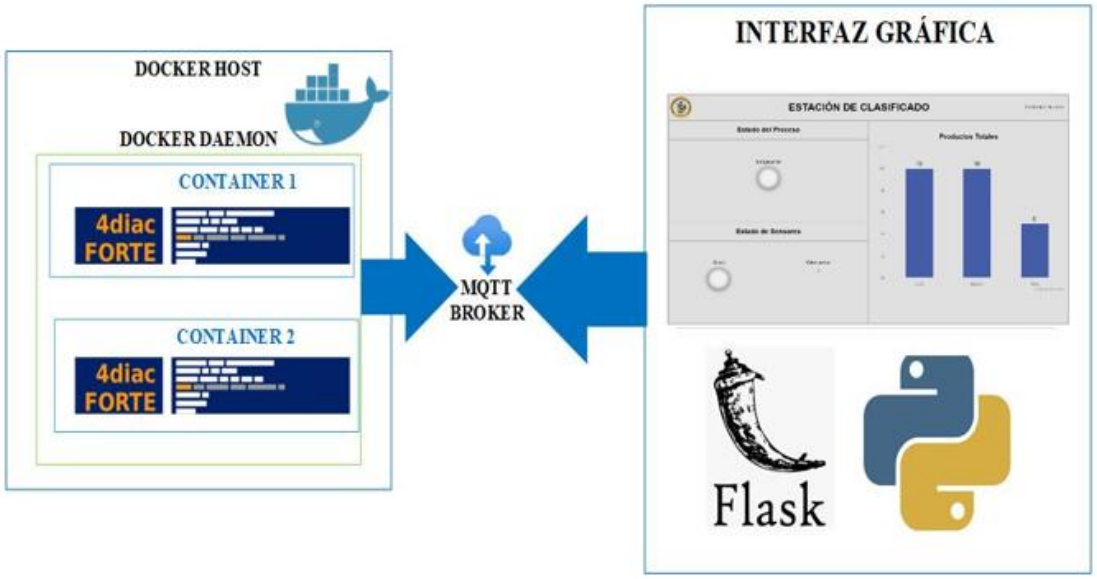



**Figura 34.** Comunicación de la Interfaz gráfica con los contenedores

La Figura 35, muestra la interfaz gráfica de supervisión bajo el diagrama de clase, en donde: la composición del archivo en donde se establecen los parámetros gráficos está en formato .css; la programación de la página web en el el archivo index.html y; el archivo main.by en el cual establece el suscriptor del protocolo de comunicación MQTT para la adquisición de datos desde el proceso ejecutado hacia las variables de la interfaz creada.

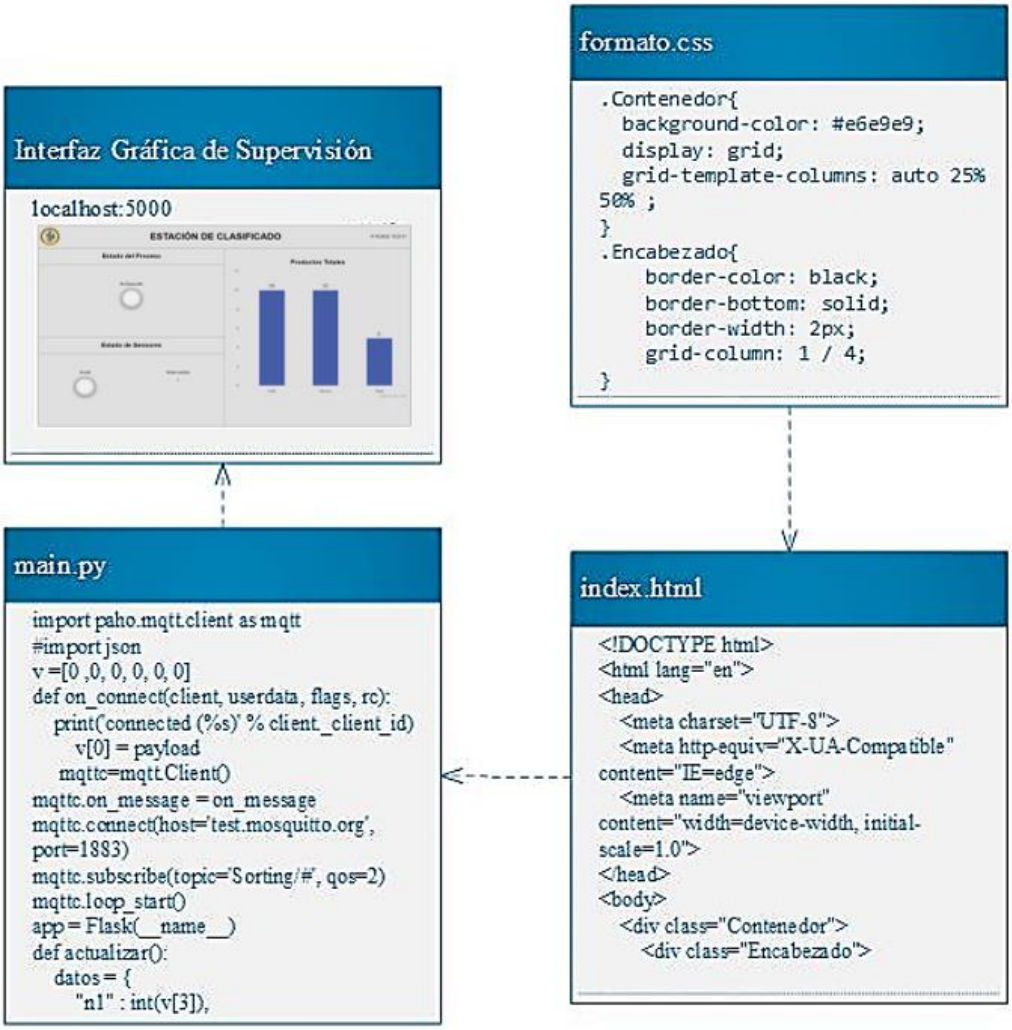



**Figura 35.** Diagrama de clase para la configuración de la interfaz

### 3.1.9 Resultado final del proyecto investigativo

El desempeño del sistema de control realizado bajo el estándar IEC 61499 y el uso de los contenedores para la Industrial 4.0, resultó satisfactorio ya que la interfaz gráfica y la comunicación con el contenedor ejecutado en la Raspberry Pi, permite tener una

supervisión en tiempo real del proceso, sensores y cantidad de ítems producidos como se observa en la Figura 36 a).

**Figura 36.** a) Entorno virtual


**Figura 36. b)** Ejecución del sistema de control (programación)

En el Figura 36 b) se aprecia por las líneas rojas, que el cliente y el servidor se están comunicando y se encuentran en el envió/recepción de datos, por lo tanto, se observa que el protocolo TCP/IP y el protocolo MQTT, están funcionando correctamente.

El desarrollo del proyecto cumplió con los objetivos establecidos, teniendo como resultado un módulo de comunicación basado en contenedores para la industria 4.0 mediante el uso del estándar IEC 61499, de tal manera que se evidenció el eficaz rendimiento de los contenedores dentro de los sistemas de control industriales, y el uso de los diferentes protocolos de comunicación para el envío y recepción de datos entre los diferentes dispositivos, por consiguiente se obtuvo la capacidad de controlar dispositivos de control de diferentes marcas y características bajo un solo estándar de programación, como se observa en la Figura 37, en donde tenemos el control de dos dispositivos diferentes para el desarrollo de sistemas de control automatizados.

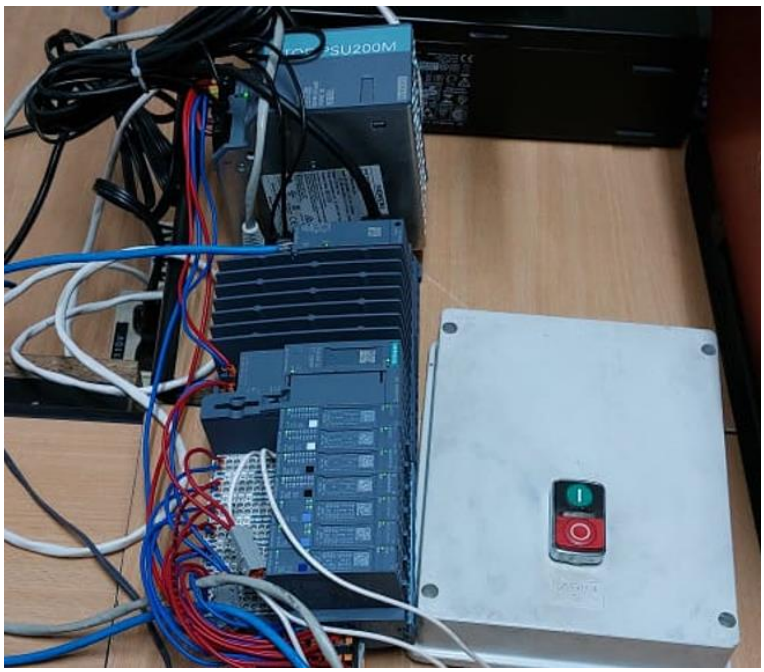


**Figura 37.** Computador industrial IPC utilizado para la adquisición y envió de datos desde el proceso

### 3.2 Análisis y Discusión de los Resultados

Los diferentes resultados obtenidos a lo largo del desarrollo del proyecto investigativo se analizaron con el objetivo de establecer las ventajas y características de la implementación del estándar IEC 61499 y la virtualización de los procesos mediante

el uso de contenedores. De esta manera se realizó un análisis de los tiempos de ejecución, carga y descarga de los contenedores y de los tiempos de procesamiento de los controladores utilizados. Además, se elaboró la verificación de la integridad de los datos enviados desde los diferentes protocolos de comunicación utilizados para el desarrollo del sistema de comunicación.

Se utilizaron técnicas estadísticas para analizar los datos recopilados en esta investigación. En primer lugar, se realizó una verificación de la calidad de los datos para asegurar que estuvieran completos y precisos. Los datos se tabularon y se organizaron de manera adecuada para su posterior análisis.

Posteriormente, se realizó una descripción de los datos utilizando medidas estadísticas como la media, la desviación estándar y el rango. La media y la desviación estándar se calcularon para cada momento del día (mañana, tarde y noche) y para cada una de las 10 mediciones realizadas en cada momento del día. El rango se calculó para cada momento del día y para el conjunto completo de datos.

Una vez realizada la descripción de los datos, se llevó a cabo una prueba de normalidad de los datos utilizando la prueba de Kolmogorov – Smirnov. Los resultados indicaron que los datos no seguían una distribución norma, ya que $p < 0.05$ debido a que presenta un valor de 0.0, como se puede observar en la figura 38.

**Prueba de Kolmogorov-Smirnov para una muestra**

| | | Datos |
|---|---|---|
| N | | 360 |
| Parámetros normales[a,b] | Media | 132,84 |
| | Desv. Desviación | 53,082 |
| Máximas diferencias extremas | Absoluto | ,254 |
| | Positivo | ,254 |
| | Negativo | -,235 |
| Estadístico de prueba | | ,254 |
| Sig. asintótica(bilateral) | | ,000[c] |

**Figura 38.** Prueba de Kolmogorov – Smirnov para comprobar distribución de los datos.

Los datos recopilados se analizaron utilizando la prueba Q de Cochran para determinar si había una diferencia significativa en la precisión de las mediciones durante el envío y recepción de datos entre el suscriptor y el publicador. Se utilizó un nivel de significancia del 0,05 para determinar si los resultados eran estadísticamente significativos.

**Análisis del tiempo de ejecución de los contenedores del repositorio**

Contenedor 1

El tiempo de ejecución de los contenedores fue analizado a través de 10 mediciones por contenedor, debido a que fue una investigación de carácter experimental, la cantidad de medicines se realizó de manera aleatoria, de acuerdo a los comandos "run; nombre del proceso; ip contenedor; #proceso" y "kill; nombre del proceso; ip contenedor; #proceso", para de esta esta manera determinar el tiempo máximo, mínimo y promedio en el que se tarda el contenedor en ejecutarse desde que se envían dichos comandos, como se observa en la Figura 39.

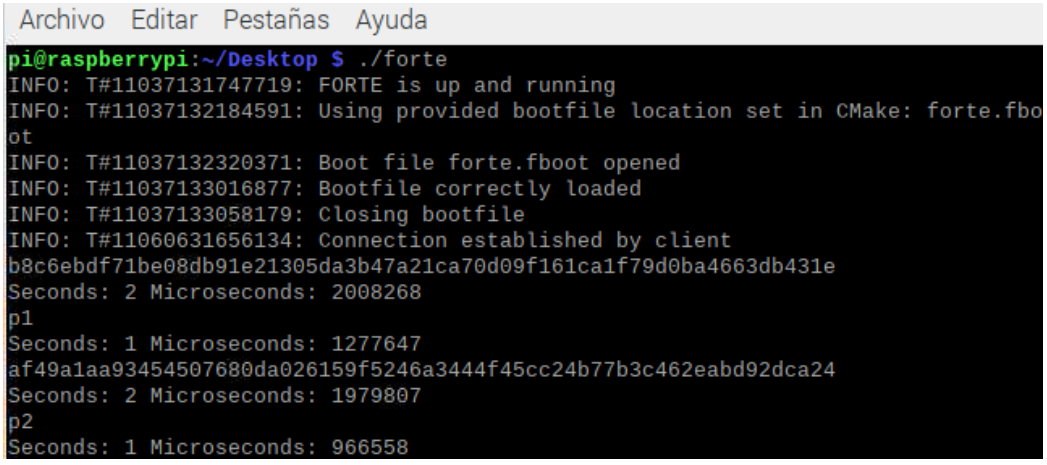



**Figura 39.** Toma del tiempo de retardo para la ejecución del contenedor 1

De esta manera de los datos obtenidos en las mediciones se tiene gráfica mostrada en la Figura 40.

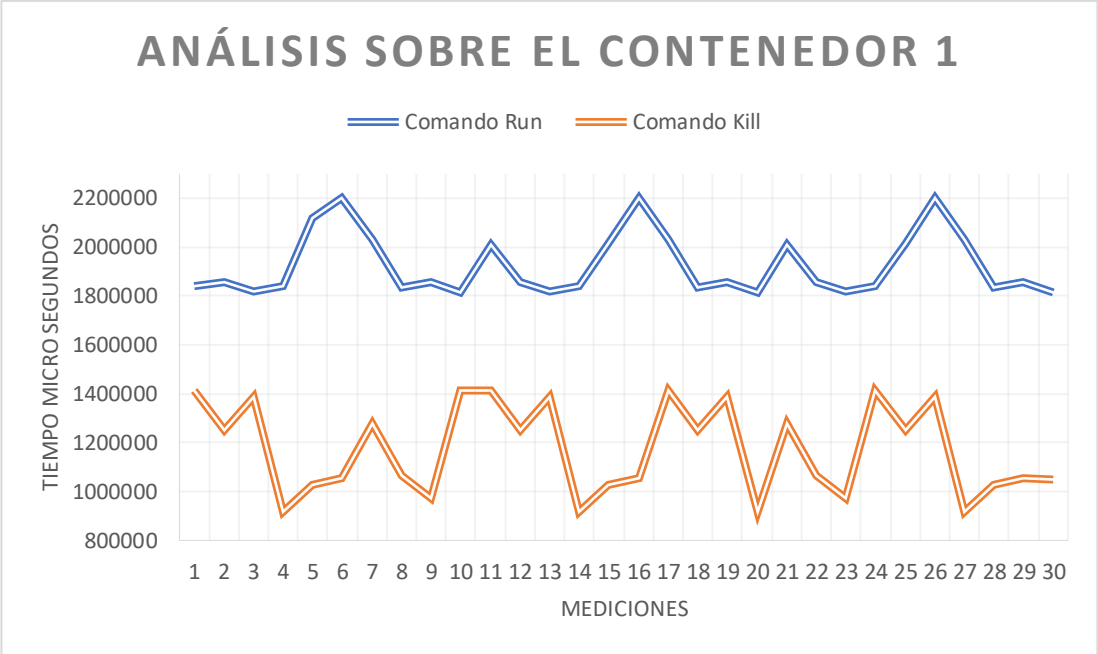



**Figura 40.** Análisis del tiempo de ejecución y paro del contenedor 1

La Tabla 10, muestra el intervalo de tiempos del contenedor 1.

**Tabla 10.** Tiempo máximo, mínimo y promedio de ejecución del contenedor 1

| Tiempo | Run | Kill |
|---|---|---|
| **Máx** | 2201442 µs | 1413437 µs |
| **Mín** | 1814542 µs | 916745 µs |
| **Prom** | 1926027,2 µs | 1174264,367 µs |

De las mediciones realizadas durante el funcionamiento del contenedor 1, se identificó que; el tiempo de paro es de 1174264,367 µs y; el tiempo de arranque es de 1174264,36 µs. Se considera que el tiempo de paro del contenedor es más rápido que el tiempo de arranque.

Contenedor 2

La Figura 41, muestra los tiempos de reacción del contenedor 2.

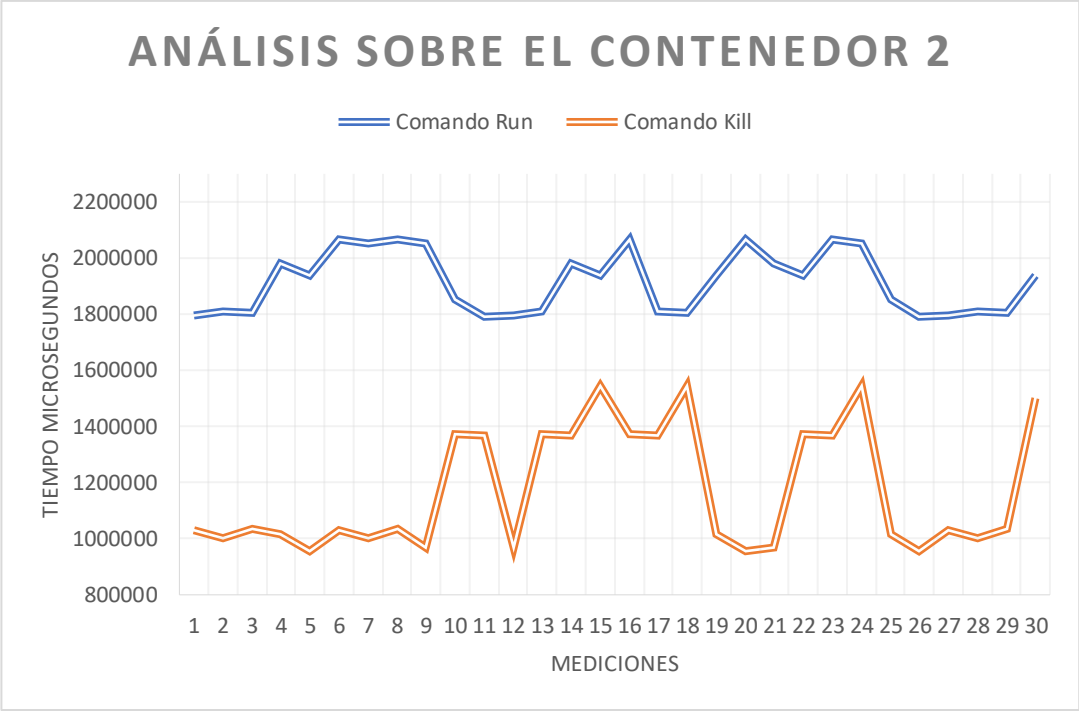



**Figura 41.** Análisis del tiempo de ejecución y paro del contenedor 1

La Tabla 11, muestra el intervalo de tiempo del contenedor 2.

**Tabla 11.** Análisis de tiempo para la ejecución del contenedor 2

| Tiempo | Run | Kill |
|---|---|---|
| **Máx** | 2066190 µs | 1544017 µs |
| **Mín** | 1790829 µs | 955501 µs |
| **Prom** | 1914137,967 µs | 1169769,333 µs |

De las mediciones realizadas durante el funcionamiento del contenedor 2, se identificó que; el tiempo de paro es de 1169769,33 µs y; el tiempo de arranque es de 1914137,96 µs. Se considera que el tiempo de arranque del contenedor es más rápido que el tiempo de paro.

**Análisis del tiempo de carga y descarga de los contenedores del repositorio**

Contenedor 1

El tiempo de carga y descarga fue analizado a través de 10 mediciones por contenedor, los datos se obtuvieron según los comandos “Push” y “Pull” determinando tiempo

máximo, mínimo y promedio en el que se tarda el contenedor en cargarse y descargarse de la nube desde que se envían dichos comandos, como se observa en la Figura 42.

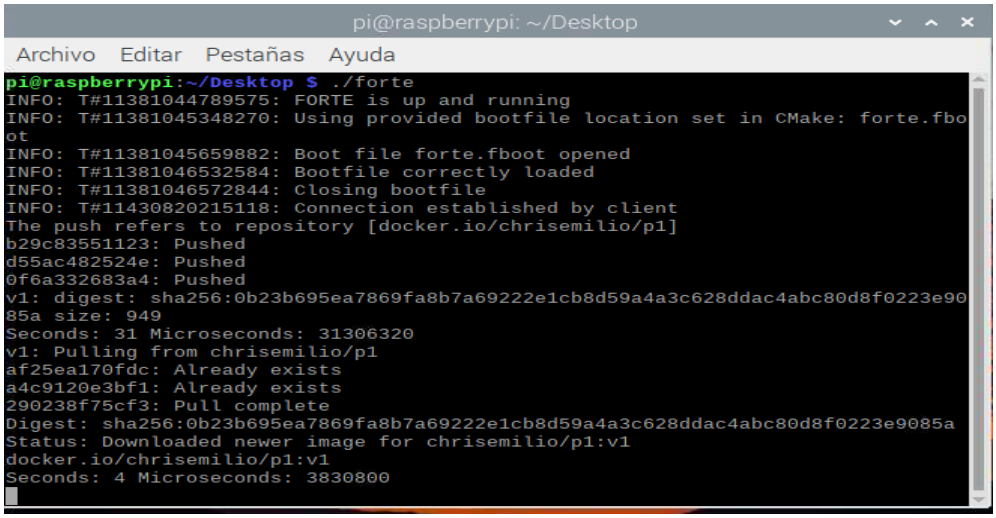



**Figura 42.** Toma del tiempo de retardo para la descarga y carga del contenedor 1

La Figura 43, muestra el análisis del tiempo de carga y descarga del contenedor 1.

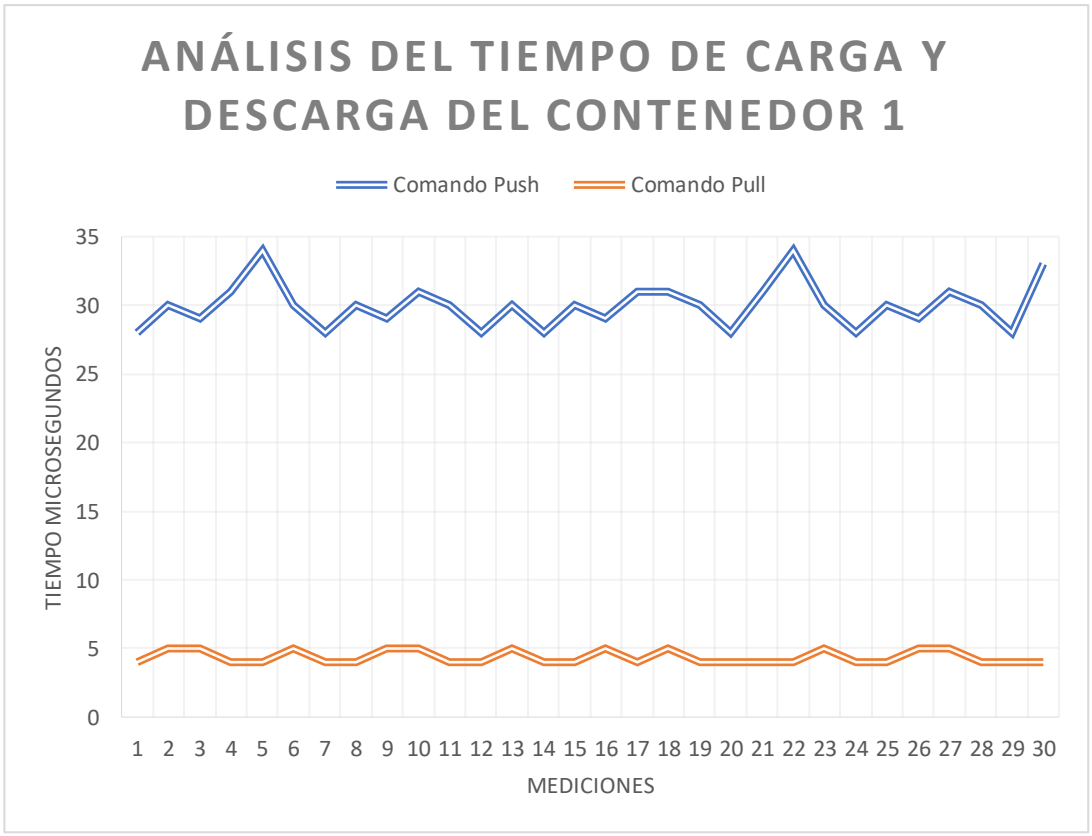



**Figura 43.** Análisis del tiempo de carga y descarga del contenedor 1

La Tabla 12, muestra el intervalo de tiempo de los comandos de carga y descarga del contenedor 1.

**Tabla 12.** Análisis de tiempo de los comandos de carga y descarga del contenedor 1

| Comandos | Push | Pull |
|---|---|---|
| Tiempo. Máx | 34 | 5 |
| Tiempo. Mín | 28 | 4 |
| Tiempo. Prom | 29,97 | 4,37 |

De las mediciones realizadas durante el funcionamiento del contenedor 1, se identificó que; el tiempo de descarga es de 4,37 s y; el tiempo de carga es de 29,97 s.

Contenedor 2

Mediante la misma metodología, se realizaron tomas de tiempo carga y descarga del contenedor que contiene el proceso 2, como se observa en la Figura 44.

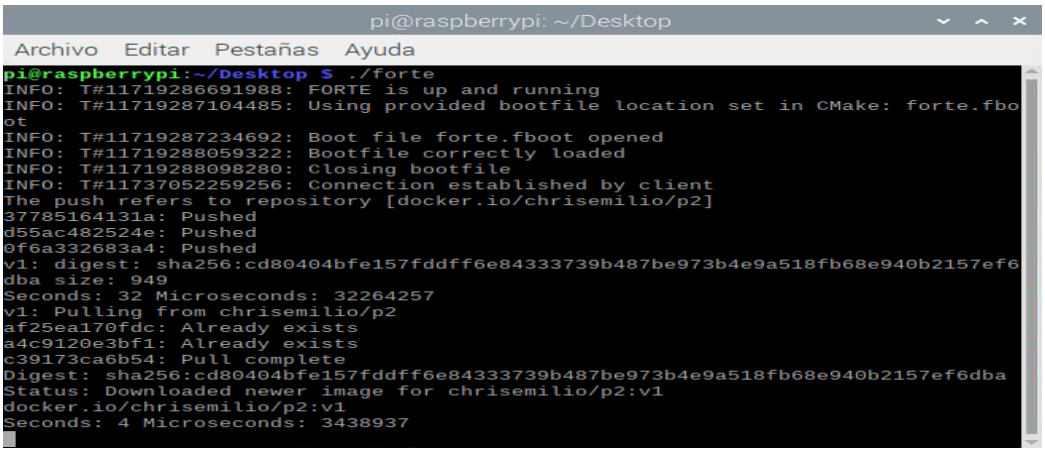



**Figura 44.** Toma del tiempo de retardo para la descarga y carga del contenedor 2

La Figura 45, muestra el análisis del tiempo de carga y descarga del contenedor 2.

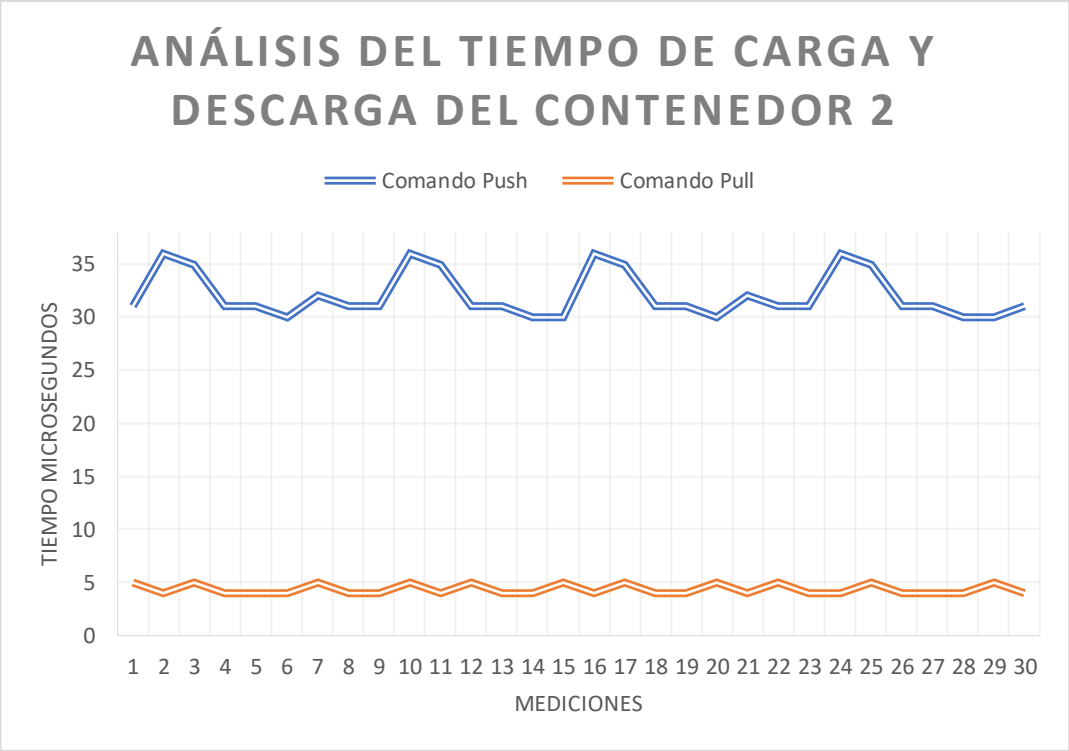



**Figura 45.** Análisis del tiempo de carga y descarga del contenedor 2

La Tabla 13, muestra el intervalo de tiempo de los comandos de carga y descarga del contenedor 2.

**Tabla 13.** Análisis de los tiempos de carga y descarga del contenedor 2

| Comandos | Push | Pull |
|---|---|---|
| Tiempo. Máx | 36 | 5 |
| Tiempo. Mín | 30 | 4 |
| Tiempo. Prom | 32,07 | 4,37 |

De las mediciones realizadas durante el funcionamiento del contenedor 1, se identificó que; el tiempo de descarga es de 4,37 s y; el tiempo de carga es de 32,07 s.

**Análisis del proceso de comunicación entre el servidor y cliente TCP/IP**

Los diferentes resultados obtenidos a lo largo del desarrollo del proyecto investigativo se analizaron con el software Wireshark el cual permitió identificar el tráfico de datos desde el computador industrial al contenedor ejecutado en la Raspberry Pi, obteniendo los datos mostrados en la Figura 46.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 527 | 0.000 | 172.21.115.172 | 172.21.115.176 | TCP | 74 | 61500 → 49472 |
| 534 | 0.006 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 536 | 0.101 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 538 | 0.093 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 543 | 0.016 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 545 | 0.109 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 548 | 0.066 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 553 | 0.019 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 555 | 0.093 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 557 | 0.093 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 562 | 0.022 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 566 | 0.016 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 569 | 0.062 | 172.21.115.172 | 172.21.115.176 | TCP | 66 | 61500 → 49472 |
| 570 | 0.047 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 574 | 0.027 | 172.21.115.172 | 172.21.115.176 | TCP | 66 | 61500 → 49472 |
| 576 | 0.009 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 579 | 0.057 | 172.21.115.172 | 172.21.115.176 | TCP | 66 | 61500 → 49472 |
| 580 | 0.031 | 172.21.115.172 | 172.21.115.176 | TCP | 72 | 61500 → 49472 |
| 582 | 0.062 | 172.21.115.172 | 172.21.115.176 | TCP | 66 | 61500 → 49472 |

**Figura 46.** Tráfico de datos de la comunicación TCP/IP

De esta manera se determinó que el envío de datos desde el servidor al cliente se encuentra dentro del rango de 0.006 ms hasta 0.109 ms. La toma de datos se realizó la comprobación del envío de paquetes entre el servidor y el cliente, obteniendo que, de 2,185 KiB enviados desde el servidor al cliente llegaron en su totalidad, de tal manera que el envío de datos desde la IPC al contenedor ejecutado en la Raspberry Pi se cumplió al 100%, como se observa en la Figura 47.

Ethernet · 2 | IPv4 · 2 | IPv6 | TCP · 2 | UDP

| Dirección | Port | Paquetes | Bytes | Total Packets | Percent Filtered | Tx Packets | Tx Bytes | Rx Packets | Rx Bytes |
|---|---|---|---|---|---|---|---|---|---|
| 172.21.115.172 | 61500 | 2,185 KiB | 155,055 KiB | 2237 | 100.00% | 1,449 KiB | 100,027 KiB | 753 bytes | 55,027 KiB |
| 172.21.115.176 | 49472 | 2.185 KiB | 155.055 KiB | 2237 | 100.00% | 753 bytes | 55.027 KiB | 1.449 KiB | 100.027 KiB |

**Figura 47.** Comprobación de la integridad de los paquetes entre los dispositivos

**Análisis del proceso de comunicación MQTT contenedores – interfaz gráfica**

El envío de información entre la aplicación Forte del contenedor con la interfaz gráfica de supervisión se realizó mediante el protocolo de comunicación MQTT, ya que dicho protocolo presenta una gran cantidad de ventajas sobre los demás protocolos de comunicación, siendo una de sus ventajas el grado de seguridad que presenta en cuanto a la integridad de los datos.

Los datos obtenidos se analizaron de manera cuantitativa de acuerdo a la cantidad de veces en las que se enviaron datos desde el Forte que se encuentra en el contenedor y la cantidad de veces en las que la interfaz gráfica recibió datos en la terminal MQTT enviadas desde el runtime que se encuentra ejecutando en la Raspberry Pi. De esta manera se planteó la siguiente Hipótesis General:

- El protocolo MQTT es efectivo para la comunicación entre los contenedores que se encuentran en ejecución en la tarjeta Raspberry Pi y la interfaz gráfica.

En la investigación se llevaron a cabo pruebas para evaluar el desempeño de los suscriptores y publicadores en el protocolo MQTT. Se definieron los criterios de éxito y fracaso de acuerdo a la entrega completa o pérdida de datos en cada prueba. En específico, se consideró un éxito la entrega completa de los 30 datos esperados, los cuales se representaron con el valor 1. En caso de que se perdiera uno o varios datos, se consideró un fracaso y se representó con el valor 0. De esta manera, se contó el número de éxitos y fracasos para cada prueba realizada en cada una de las dos variables evaluadas.

A continuación, la Tabla 14, muestra la cantidad de datos enviados mediante el protocolo MQTT y publicados por el Forte que se ejecuta en el contenedor.

**Tabla 14.** Número de envíos desde el Forte del contenedor a la interfaz gráfica

| **Número de envíos desde el Forte del contenedor a la interfaz gráfica** | | |
|---|---|---|
| **TÓPICO** | **ÉXITO** | **FRACASO** |
| Tópico 1 (Sorting/Start) | 30 | 0 |
| Tópico 2 (Sorting/Exit) | 30 | 0 |
| Tópico 3 (Sorting/Vision) | 30 | 0 |
| Tópico 4 (Sorting/Lids) | 30 | 0 |
| Tópico 5 (Sorting/Bases) | 30 | 0 |
| Tópico 6 (Sorting/Raw) | 30 | 0 |
| **Total** | **180** | **0** |

Se establece que gran parte de los datos enviados han llegado con éxito y solo un porcentaje mínimo ha fracaso, lo cual se muestra en la Figura 48.

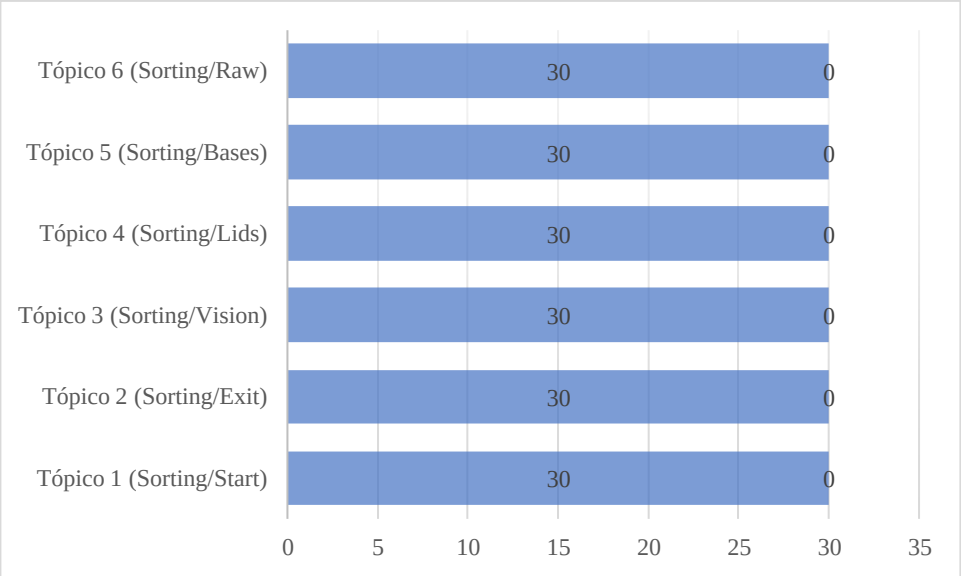



**Figura 48.** Número de envíos desde el Forte del contenedor a la interfaz gráfica

De igual manera, se realizó el análisis de la etapa de suscripción del modelo MQTT, pero desde la interfaz gráfica hacia el Forte que se está ejecutando en el contenedor.

La Tabla 15, muestra el resumen de 30 datos enviados desde la interfaz gráfica hacia el Forte que se ejecuta en el contenedor.

**Tabla 15.** Número de envíos desde Forte del Contenedor a la interfaz gráfica

| **NÚMERO DE ENVIOS DESDE LA INTERFAZ GRÁFICA AL FORTE DEL CONTENEDOR** | | |
|---|---|---|
| **TÓPICO** | **ÉXITO** | **FRACASO** |
| Tópico 1 (Sorting/Start) | 30 | 0 |
| Tópico 2 (Sorting/Exit) | 30 | 0 |
| Tópico 3 (Sorting/Vision) | 30 | 0 |
| Tópico 4 (Sorting/Lids) | 30 | 0 |
| Tópico 5 (Sorting/Bases) | 30 | 0 |
| Tópico 6 (Sorting/Raw) | 30 | 0 |
| **Total** | **180** | **0** |

La Figura 49, muestra el envío de 360 datos.

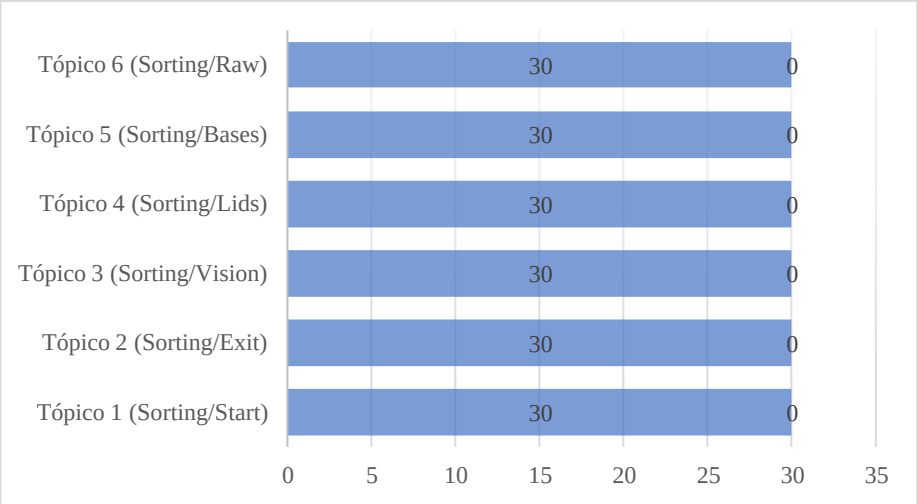



**Figura 49.** Número de envíos desde la interfaz gráfica al Forte del contenedor

A continuación, la Tabla 16, muestra el número total de éxitos y fracaso en el envío y recepción de datos (Publicación y Suscripción).

**Tabla 16.** Número de éxitos y fracasos durante el envío de datos

| | **Publicación** | **Suscripción** |
|---|---|---|
| **Éxitos** | 180 | 180 |
| **Fracasos** | 0 | 0 |
| **Total** | 180 | 180 |

De esta manera se obtuvo una efectividad del protocolo de comunicación con un porcentaje de efectividad del 100 % como se muestra en la Figura 50.

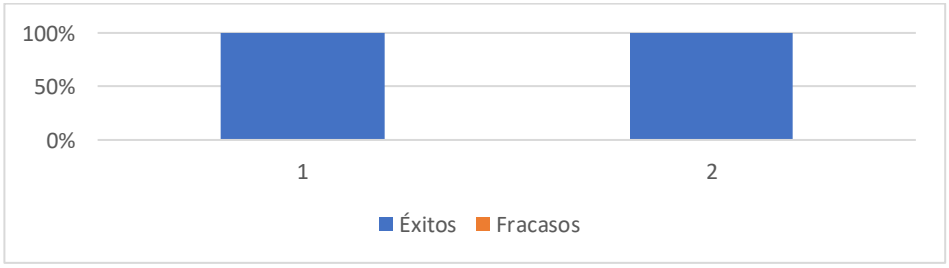



**Figura 50.** Número de éxitos y fracasos durante el envío de datos

La Figura 51, muestra la elaboración de la prueba de Cochran en el software MedCalc.

| | |
|---|---|
| Cases in spreadsheet | 6 |
| Cases with missing values | 0 |
| Cases included in the analysis | 6 |

**Frequencies**

| Variable | Value | | Proportion (%) |
|---|---|---|---|
| | 0 | 1 | |
| Publicador | 0 | 6 | 100,00 |
| Suscriptor | 6 | 0 | 0,00 |

**Cochran's Q test**

| | |
|---|---|
| n | 6 |
| Cochran's Q | 6,0000 |
| DF | 1 |
| Significance | 0,014 |

**Multiple comparisons**

| Variable | Different (P<0,05) from variable nr |
|---|---|
| (1) Publicador | (2) |
| (2) Suscriptor | (1) |

**Figura 51.** Número de éxitos y fracasos durante el envío de datos

Se estableció un nivel de significancia de 0.014 para la prueba Q de Cochran, el cual resultó ser menor que el nivel de significancia previamente establecido de 0.05. Por lo tanto, se concluyó que existe evidencia suficiente para aceptar la hipótesis de que el protocolo de comunicación MQTT es efectivo para la comunicación entre los contenedores que se encuentran en ejecución en la tarjeta Raspberry Pi y la interfaz gráfica.

Los datos entre el contenedor que trabaja como Publicador y la interfaz gráfica que funciona como Subscriptor se obtuvieron mediante la aplicación Wireshark de la misma manera que con el protocolo de comunicación TCPI/IP, la Figura 52, muestra los datos enviados y recibidos entre los dispositivos.

| Protocol | Length | Info |
|---|---|---|
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 68 | Ping Request |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 372 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 264 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 264 | Publish Message [Sorting/Start], Publish Message [Sorting/At_exit], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |
| MQTT | 174 | Publish Message [Sorting/Start], Publish Message [Sorting/At_exit], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 264 | Publish Message [Sorting/Start], Publish Message [Sorting/At_exit], |
| MQTT | 84 | Publish Message [Sorting/Start] |
| MQTT | 156 | Publish Message [Sorting/At_exit], Publish Message [Sorting/Vision], |

**Figura 52.** Tráfico de datos de la comunicación MQTT

Estableciendo que el tiempo de respuesta entre la interfaz y el contenedor que se encuentre ejecutado en la Raspberry Pi se encuentra en el rango de 0.00095 µs a 0.000561 µs. La Figura 53, muestra la entrega de 180 paquetes entre el publicador y el subscriptor con el 100% de satisfacción.

| Protocolo | Porcentaje de paquetes | Paquetes |
|---|---|---|
| Frame | 100.0 | 180 |
| Ethernet | 100.0 | 180 |
| Internet Protocol Version 4 | 100.0 | 180 |
| Transmission Control Protocol | 100.0 | 180 |
| MQ Telemetry Transport Protocol | 100.0 | 180 |

**Figura 53.** Comprobación de la integridad de los datos en los puntos finales

### Análisis del tiempo de respuesta entre la IPC y la tarjeta Raspberry Pi

El desempeño de los dispositivos de control se evidencia a través del análisis de los protocolos de comunicación utilizados, mostrando ventajas del uso de la IPC sobre la

tarjeta Raspberry PI. La Figura 54 muestra los porcentajes de satisfacción entre cada dispositivo.

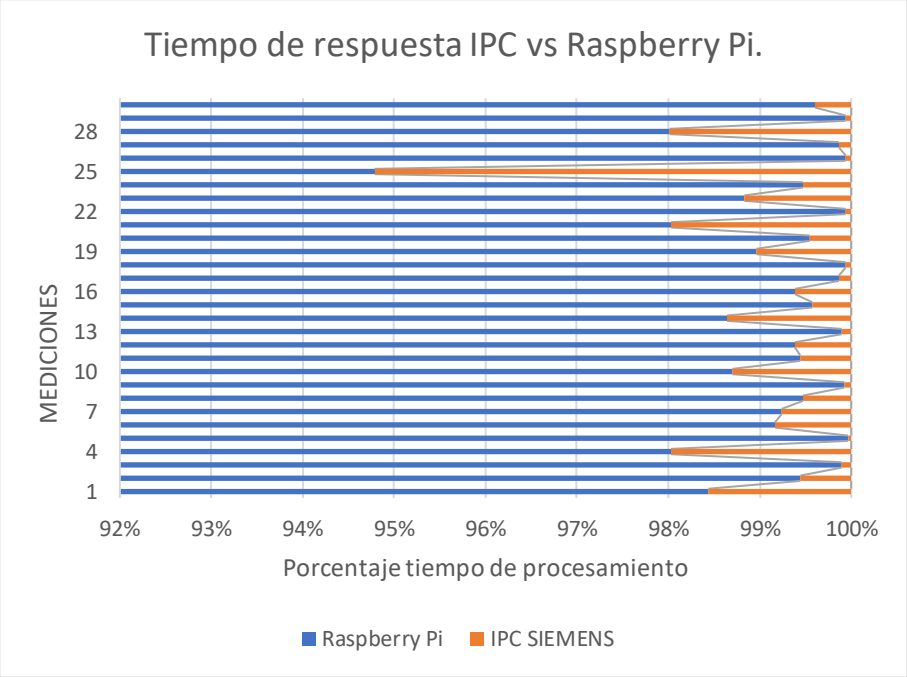



**Figura 54.** Tiempo de respuesta IPC vs Raspberry Pi

Del análisis se obtuvo que la IPC es superior por los tiempos de respuesta y manejo adecuado de datos. Entre algunas características destacables se tiene que: la Raspberry Pi B+ posee un procesador de cuatro núcleos a 1,4 GHz y con 1 GB de RAM y; la IPC ET 200SP tiene un procesador de cuatro núcleos a 1,8 GHZ con 4 GB de RAM.

# CAPÍTULO IV

# CONCLUSIONES Y RECOMENDACIONES

## 4.1 Conclusiones

- Utilizar el IPC junto con el estándar IEC 61499 y los contenedores permitió desarrollar aplicaciones de software escalables y fácilmente desplegables en diferentes entornos. Además, el IPC brindó acceso en tiempo real a los datos del proceso, lo que controla sistemas automatizados de manera eficiente mediante aplicaciones desarrolladas con el estándar IEC 61499 y la Raspberry Pi.

- Los protocolos de comunicación implementados en el caso de estudio mostraron un buen rendimiento. El protocolo TCP/IP envió 2237 paquetes del servidor al cliente al 100% y el protocolo MQTT publicó 180 datos que fueron recibidos por el suscriptor. Esto muestra que el sistema funcionó de manera segura, eficiente y sin producir ninguna pérdida de datos.

- De acuerdo al análisis de los tiempos de ejecución y paro de los contenedores se estableció un rango de tiempo promedio de 1914137,967 µs a 1926027,2 µs para que el contenedor se inicialice y se llega a parar en un rango de tiempo promedio de 1169769,333 µs a 1174264,367 µs, siendo estos tiempos muy reducidos de tal manera que los contenedores permiten tener procesos flexibles e interoperables, ya que el tiempo de ejecución y paro de los mismos son similares, permitiendo de esta manera cambiar de un proceso a otro de forma rápida sin tener grandes pérdidas de tiempo.

- Para los dispositivos físicos del sistema de control, se determinó que el IPC tiene un rendimiento superior a la Raspberry Pi en términos de tiempos de respuesta y ejecución de aplicaciones. Esto muestra que, debido a sus características de hardware y software, el IPC es uno de los controladores más eficientes para uso industrial en el ámbito de los equipos de control.

## 4.2 Recomendaciones

- Implementar el sistema desarrollado a un proceso físico ya que un proceso simulado presenta un ambiente controlado, mostrando muchas de las veces un proceso altamente eficiente ya que no se ve afectado por ninguna variable que se presente durante el desarrollo del proceso, por otro lado, un proceso físico se ve afectado por una gran cantidad de variables no controladas de tal manera que la implementación de la investigación en ese tipo de entornos presentaría mejores resultados.

- Desarrollar sistemas de control enfocados en el uso de controladores programables con características similares a las del computador industrial IPC siemens ET200SP, con la finalidad de promover el desarrollo de sistemas bajo un estándar de programación de código abierto, relacionado de esta manera al desarrollo tecnológico de lo que implica trabajar dentro de la industria 4.0.

- Para un mejor manejo de los contenedores, el envío y recepción de datos entre los dispositivos y todo lo que se encuentra relacionado al cloud computing, sería recomendable trabajar con una velocidad de internet eficiente igual o mayor que los 100 Mbps, ya que el tiempo de respuesta y ejecución de estos dependen de dicha velocidad.

**Referencias Bibliográficas**


[1] T. Lyu, U. Dwi Atmojo, y V. Vyatkin, «Towards cloud-based virtual commissioning of distributed automation applications with IEC 61499 and containerization technology», presentado en IECON Proceedings (Industrial Electronics Conference), 2021, vol. 2021-October. doi: 10.1109/IECON48115.2021.9589945.

[2] W. Dai, Y. Zhang, L. Kong, J. H. Christensen, y D. Huang, «Design of Industrial Edge Applications based on IEC 61499 Microservices and Containers», *IEEE Trans. Ind. Inform.*, pp. 1-11, 2022, doi: 10.1109/TII.2022.3214199.

[3] G. M. Martinov, N. V. Kozak, y R. A. Nezhmetdinov, «Approach in implementing of logical task for numerical control on basis of concept "industry 4.0"», presentado en Proceedings - 2018 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2018, 2018. doi: 10.1109/ICIEAM.2018.8728584.

[4] A. Souri y M. Ghobaei-Arani, «Cloud manufacturing service composition in IoT applications: a formal verification-based approach», *Multimed. Tools Appl.*, vol. 81, n.º 19, pp. 26759-26778, 2022, doi: 10.1007/s11042-021-10645-1.

[5] C. A. Garcia, M. V. Garcia, E. Irisarri, F. Perez, M. Marcos, y E. Estevez, «Flexible Container Platform Architecture for Industrial Robot Control», presentado en IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2018, vol. 2018-September, pp. 1056-1059. doi: 10.1109/ETFA.2018.8502496.

[6] T. P. Zambrano Valverde, «Uso del protocolo MQTT basado en la norma IEC 61499 para la integración de un robot Kuka Youbot hacia la nube.», dic. 2019, Accedido: 8 de marzo de 2023. [En línea]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/13353

[7] J. D. Llamuca Supe, «Integración de ISA-95 e IEC-61499 para el monitoreo de sistemas de control distribuido», *Integrating ISA-95 and IEC-61499 for distributed control system monitoring*, ene. 2020, Accedido: 9 de noviembre de 2022. [En línea]. Disponible en: https://repositorio.uta.edu.ec:8443/jspui/handle/123456789/30754

[8] M. V. Garcia, «Metodologías para el diseño de sistemas de control distribuido bajo el estándar IEC 61499 aplicados al control de procesos - CORE». https://core.ac.uk/works/51396279 (accedido 10 de enero de 2023).

[9] C. A. García, E. X. Castellanos, M. V. García, C. A. García, E. X. Castellanos, y M. V. García, «Desarrollo de sistemas ciber-físicos de producción para Procesamiento por lotes usando normas IEC-61499 e ISA-88», *Ingeniare Rev. Chil. Ing.*, vol. 27, n.º 3, pp. 443-453, sep. 2019, doi: 10.4067/S0718-33052019000300443.

[10] E. X. Castellanos Narváez y C. A. García Sánchez, «Control distribuido en procesos industriales utilizando sistemas empotrados de bajo costo, para verificar la aplicabilidad de la norma IEC-61499.», bachelorThesis, Universidad de las Fuerzas Armadas ESPE. Extensión Latacunga. Carrera de Ingeniería en Electrónica e Instrumentación., 2017. Accedido: 10 de enero de 2023. [En línea]. Disponible en: http://repositorio.espe.edu.ec/jspui/handle/21000/13944

[11] D. Mora-Sánchez y L. Guerrero-Marín, «Industria 4.0: el reto en la ruta hacia las organizaciones digitales», *Estud. Gest. Rev. Int. Adm.*, n.º 8, Art. n.º 8, nov. 2020, doi: 10.32719/25506641.2020.8.7.

[12] C. E. para A. L. y el Caribe, *Industria 4.0: oportunidades y desafíos para el desarrollo productivo de la provincia de Santa Fe*. CEPAL, 2019. Accedido: 10 de enero de 2023. [En línea]. Disponible en: https://www.cepal.org/es/publicaciones/44954-industria-40-oportunidades-desafios-desarrollo-productivo-la-provincia-santa-fe

[13] C. B. Y. Cortés, J. M. I. Landeta, J. G. B. Chacón, F. A. Pereyra, y M. L. Osorio, «El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras», *Concienc. Tecnológica*, n.º 54, 2017, Accedido: 9 de noviembre de 2022. [En línea]. Disponible en: https://www.redalyc.org/journal/944/94454631006/html/

[14] W. Montalvo, P. Encalada, A. Miranda, C. A. Garcia, y M. V. Garcia, «Implementación de OPC UA en una Plataforma Web para la integración de comunicación en el área de producción - ProQuest», *risti*, n.º 26, pp. 667-680.

[15] L. Cruz, «Estándar IEC 61499 como Complemento en el Control Distribuido», Universidad del Cauca, Cartagena de Indias. Accedido: 9 de noviembre de 2022. [En

línea]. Disponible en: https://1library.co/article/est%C3%A1ndar-iec-complemento-control-distribuido.z31313ey

[16] F. Chicaiza, C. A. García, E. X. Castellanos, C. Sánchez, C. Rosero, y M. García, «Arquitectura Flexible Basada en ISA-88 para el Diseño del Diagrama de Control de Ejecución en Aplicaciones Distribuidas mediante IEC-61499», *Enfoque UTE*, vol. 9, n.º 1, pp. 149-165, 2018.

[17] N. Jazdi, «Cyber physical systems in the context of Industry 4.0», en *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, Cluj-Napoca, Romania, may 2014, pp. 1-4. doi: 10.1109/AQTR.2014.6857843.

[18] L. Ferrarini, C. Veber, y G. Fogliazza, «IEC 61499 implementation of a Modular Control Model for Manufacturing Systems», en *2005 IEEE Conference on Emerging Technologies and Factory Automation*, Catania, Italy, 2005, vol. 1, pp. 315-321. doi: 10.1109/ETFA.2005.1612540.

[19] K. Thramboulidis, «Different perspectives [Face to Face; "IEC 61499 function block model: Facts and fallacies"», *IEEE Ind. Electron. Mag.*, vol. 3, n.º 4, pp. 7-26, dic. 2009, doi: 10.1109/MIE.2009.934788.

[20] J. A. B. Muñoz y C. D. S. Cobo, «Propuesta de Aplicación del estándar IEC 61499-1 en un caso de estudio», p. 121.

[21] Eclipse 4diac, «Eclipse 4diac - The Open Source Environment for Distributed Industrial Automation and Control Systems», *Eclipse 4diac*. https://www.eclipse.org/4diac/ (accedido 9 de noviembre de 2022).

[22] S. Logistic, «El método FIFO en la valoración de stock de un almacén», *Stock Logistic*, 2 de julio de 2018. https://www.stocklogistic.com/metodo-fifo-valoracion-stock-almacen/ (accedido 9 de noviembre de 2022).

[23] Tribalyte, «Sistema embebido y sus características | Conceptos fundamentales», *TRIBALYTE TECHNOLOGIES*. https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas (accedido 9 de noviembre de 2022).

[24] J. A. Gonzalez Ramos, «Implementación de un computador raspberry pi enfocado a la enseñanza de herramientas ofimáticas aplicadas a las tareas académicas

concretas de los niños y niñas del séptimo año de educación general básica, de la escuela de educación básica José Miguel Burneo de la ciudad de Loja, período 2015», bachelorThesis, 2016. Accedido: 10 de enero de 2023. [En línea]. Disponible en: https://dspace.unl.edu.ec//handle/123456789/11145

[25] Xataka, «Qué modelo de Raspberry Pi comprar: un repaso a las principales placas y los proyectos más habituales para dar con la mejor», *XATAKA*. https://www.xataka.com/seleccion/que-modelo-raspberry-pi-comprar-repaso-a-principales-placas-proyectos-habituales-para-dar-mejor (accedido 9 de noviembre de 2022).

[26] Siemens, «SIMATIC ET 200SP Open controlador, CPU 1515SP PC2 T, 8 GB de RAM, CFAST de 30 GB con Windows 10 IoT Enterprise 64 bits y controlador por software S7-1», *AUTYCOM*. https://www.autycom.com/producto/simatic-et-200sp-open-controlador-cpu-1515sp-pc2-t-8-gb-de-ram-cfast-de-30-gb-con-windows-10-iot-enterprise-64-bits-y-controlador-por-software-s7-1/ (accedido 9 de noviembre de 2022).

[27] Autycom, «SIMATIC ET 200SP: sistema IO para gabinetes de control», *AUTYCOM*, 14 de septiembre de 2021. https://www.autycom.com/simatic-e-200sp/ (accedido 9 de noviembre de 2022).

[28] Siemens, «Simatic St80 STPC Complete Spanish 2020 | PDF | Scada | Software de la aplicacion». https://es.scribd.com/document/539172168/Simatic-St80-Stpc-Complete-Spanish-2020 (accedido 9 de noviembre de 2022).

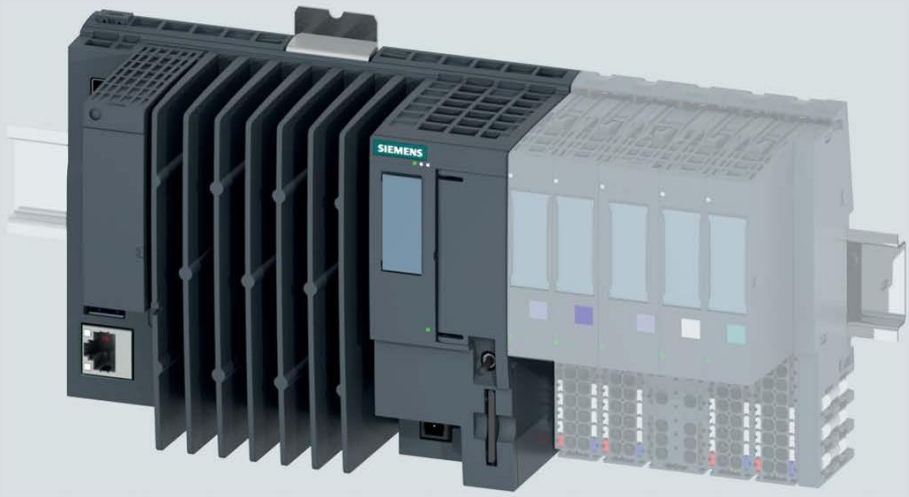
[29] NetApp, «¿Qué son los contenedores? - Docker | NetApp», *Netapp*. https://www.netapp.com/es/devops-solutions/what-are-containers/ (accedido 9 de noviembre de 2022).

[30] M. Rodríguez y N. Mariela, «Despliegue de una nube híbrida para entornos de desarrollo implementando Kubernetes y Dockers en la plataforma Openstack e infraestructura en Amazon», p. 121, 2021.

**Anexos**

**Anexo 1.** Manual de usuario de la IPC

SIEMENS




Manual de producto

SIMATIC

ET 200SP Open Controller

CPU 1515SP PC2 (F/T/TF)

Edición 09/2021

support.industry.siemens.com

**Anexo 2.** Manual de usuario de la Raspberry Pi

**Anexo 3.** Manual de ODK 1500s

SIEMENS

SIMATIC

# Opciones de STEP 7 (TIA Portal) Open Development Kit 1500S V2.5 SP1

Manual de programación y manejo

12/2019
A5E42356212-AE