UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Electrónica y Telecomunicaciones

Sistema de supervisión y control de una celda de manufactura a través de tecnologías habilitantes de internet industrial de las cosas

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Autores:

Alex Fernando Chuya Machuca
CI: 070658936-3
alexfernand_17@hotmail.com

Edwin Xavier Alvarez Murudumbay
CI: 030251045-8
xavier.alvarezm@gmail.com

Director:

Ing. Luis Ismael Minchala Ávila, PhD
CI: 030145348-6

Cuenca, Ecuador

14-marzo-2022

Resumen

Este documento describe el desarrollo de un sistema de supervisión de una celda de manufactura. La implementación de este sistema utiliza herramientas habilitantes del internet industrial de las cosas (Industrial Internet of the Things (IIoT)).

En el laboratorio de máquinas eléctricas de la Universidad de Cuenca existe una celda de manufactura flexible (Flexible Manufacturing Cell (FMC)) que está en desuso. Este trabajo tiene el propósito de poner en marcha esta FMC. La FMC consta principalmente de actuadores y sensores. Estos dispositivos interactúan con un (Programmable Logic Controller (PLC)) (*PiXtend*) perteneciente a la FMC.

Adicionalmente, un dispositivo *Raspberry Pi* actúa como interfaz entre la parte industrial y la computación en la nube. La comunicación entre el PLC y la *Raspberry Pi* se desarrolla a través del protocolo (Open Platform Communications-Unified Architecture (OPC-UA)). Un cliente instalado en la *Raspberry Pi* recibe datos del servidor OPC-UA. Por otra parte, la *Raspberry Pi* interactúa con un servidor en la nube a través del protocolo de comunicación (Message Queuing Telemetry Transport (MQTT)). Este servidor procesa, administra y almacena en una base de datos el flujo de datos industriales.

Este trabajo considera el *hardware* y *software* requerido para producir, recopilar, transferir, almacenar y procesar flujo de datos. Los procesos de recolección de datos se ejecutan en el *PiXtend*, mientras que el almacenamiento de datos de manera local y transferencia al servidor en la nube se ejecutan en la *Raspberry Pi*. El control y monitoreo por parte del usuario se realizan mediante un ordenador de bajo coste local y en dispositivos de manera remota.

El servidor en la nube envía la información a través de Websockets. Mediante una página web es posible realizar la supervisión. Se realizan pruebas de tráfico entre los diferentes protocolos de comunicación y una comparación cualitativa del funcionamiento de la FMC utilizando otras herramientas tecnológicas dirigidas al IIoT.

Palabras clave : Supervisión. OPC-UA. FMC. MQTT. PiXtend. WebSocket. IIoT.

Abstract

This document describes the development of a monitoring system for a manufacturing cell. The implementation of this system utilizes enabling technologies from the Industrial Internet of Things IIoT.

The University of Cuenca's Electrical Machines Laboratory owns a flexible manufacturing cell FMC that has been mostly left in disuse. This project is purposed to take advantage of this FMC. This cell consists primarily of actuators and sensors. These devices all interface with a PiXtend, a component of the FMC.

Additionally, a Raspberry Pi acts as an Interface between the industrial components and a cloud computing platform. The communication between the PLC and the Raspberry Pi is by use of the OPC-UA protocol. A client software program on the Raspberry Pi receives data from an OPC-UA server. It interfaces with the cloud server by way of the MQTT protocol. This online server manages, processes and stores a database of the generated industrial data.

This project considers both the hardware and software necessary for producing, compiling, transmitting, storing and processing the data stream. The process of compiling the data is carried out on the PiXtend, while the data storage is done locally as well as uploaded to the cloud server by way of the Raspberry Pi. The control and monitoring can be done with a low-cost local computer as well as remotely through other devices.

The cloud server communicates through WebSockets. With the use of a webpage interface, it's possible to monitor the various industrial processes. Various traffic tests are also carried out between the different protocols as well as a qualitative comparison of the FMC performance with the use of specific tools designed for the IIoT.

Keywords : Monitoring. OPC-UA. FMC. MQTT. PiXtend. WebSocket. IIoT.

Índice general

Resumen 1

Abstract 2

Índice general 3

Índice de figuras 6

Índice de tablas 8

Cláusula de Propiedad Intelectual 9

Cláusula de Propiedad Intelectual 10

Cláusula de licencia y autorización para publicación en el Repositorio Institucional 11

Cláusula de licencia y autorización para publicación en el Repositorio Institucional 12

Certifico 13

Dedicatoria 14

Dedicatoria 15

Agradecimientos 16

Agradecimientos 17

Abreviaciones y acrónimos 18

1. Introducción 19

1.1. Antecedentes . . . 19

1.2. Definición del problema . . . 20

1.2.1. Antecedentes . . . 20

1.2.2. Definición tácita del problema . . . 20

1.2.3. Revisión del estado del arte . . . 21

1.2.4. Propuesta . . . 24

1.3. Objetivos . . . 25

1.3.1. Objetivo general . . . 25

1.3.2. Objetivos específicos . . . 25
1.4. Contribuciones de la tesis . . . 25

2. Fundamentos teóricos **26**
2.1. Celda de manufactura flexible . . . 26
2.2. Neumática . . . 26
2.2.1. Nivel de presión . . . 26
2.2.2. Elementos principales de un sistema neumático . . . 26
2.3. Comunicaciones industriales . . . 28
2.3.1. Tecnologías de comunicación . . . 28
2.3.2. Arquitectura unificada OPC . . . 29
2.4. Internet industrial de las cosas . . . 29
2.4.1. Industria 4.0 . . . 29
2.4.2. Protocolos de aplicación IIoT en las TI . . . 30

3. Metodología **31**
3.1. Descripción del proceso . . . 31
3.2. Componentes principales del sistema de supervisión . . . 32
3.2.1. Celda de manufactura flexible . . . 32
3.2.2. PLC PiXtend V1.3 . . . 34
3.2.3. *Raspberry* Pi . . . 34
3.3. Implementación . . . 35
3.3.1. Levantamiento de la FMC . . . 35
3.3.2. Automatización de la FMC . . . 36
3.3.3. Niveles de jerarquía del sistema . . . 43
3.4. Comunicaciones . . . 44
3.5. Visualización . . . 45
3.6. Almacenamiento . . . 47

4. Resultados **48**
4.1. Descripción narrativa . . . 48
4.2. Pruebas de detección de perforaciones en los cilindros . . . 49
4.3. Pruebas de comunicación . . . 50
4.3.1. Tráfico en la *Raspberry* Pi . . . 50
4.3.2. Tráfico en el dispositivo de supervisión remota . . . 53
4.4. Comparación cualitativa . . . 57

5. Conclusiones y Recomendaciones **61**
5.1. Conclusiones . . . 61

A. Diseño del plano de la celda de manufactura **62**

B. Diseño del circuito neumático **64**

C. Manual de usuario del PiXtend **66**

D. Librería de programación de PiXtend para Python **82**

Bibliografía **91**

Índice de figuras

1.1. Arquitectura de la plataforma web desarrollada [5]. 22
1.2. Arquitectura IoT- SDN del prototipo implementado [8]. 23
1.3. Arquitectura propuesta para el desarrollo de un dispositivo de control para la industria 4.0 [10]. 24

2.1. Pirámide de automatización [13]. 28

3.1. Ilustración de un sistema de comunicación para la industria 4.0. 32
3.2. Sistema de comunicación implementado. 32
3.3. Modelo de FMC DE LORENZO. 33
3.4. PLC PiXtend V.1.3. 34
3.5. *Raspberry* Pi 3 Model B [17]. 34
3.6. Diagrama de flujo del funcionamiento de la FMC. 39
3.7. Diagrama de flujo de la estación de pesado y almacenamiento. 40
3.8. Diagrama de flujo de la estación de verificación, manipulación y transporte. 41
3.9. Arquitectura de la red de comunicación planteada. 43
3.10. Página de visualización en la web correspondiente al Dashboard del trabajo de titulación. . . . 47

4.1. Perforación de un cilindro detectado por la cámara con un valor de radio aproximado de 27 unidades. 50
4.2. Perforación de un cilindro detectado por la cámara con un valor de radio aproximado de 50 unidades. 50
4.3. Paquetes OPC-UA de tipo BrowseRequest y BrowseResponse. 51
4.4. Captura de un paquete OPC-UA en Wireshark. 51
4.5. Contenido del campo OpcUA Service. 52
4.6. Contenido de un objeto OPC-UA. 52
4.7. Captura de paquetes de mensajes MQTT. 53
4.8. Tráfico MQTT de Publicación. 53
4.9. Tráfico MQTT de Subscripción. 53
4.10. Contenido del protocolo HTTP al cargar la página web. 54
4.11. Petición de negociación WebSocket entre la aplicación y el servidor. 55
4.12. Intercambio de protocolo de HTTP a WebSockets. 55
4.13. Establecimiento de petición de negociación WebSocket y respuesta de negociación WebSocket. 55
4.14. Paquetes WebSockets de tipo Ping y Pong. 56
4.15. Gráfica round-trip time (RTT) entre el cliente en la *Raspberry* Pi y el servidor en el PiXtend. . 56
4.16. Gráfica RTT entre la Raspberry Pi y el broker. 57

4.17. Gráfica RTT entre el servidor AWS y el dispositivo del cliente. 57
4.18. Diagrama de araña de la solución IIoT Siemens, Schneider Electric, Mitsubishi Electric vs. la solución IIoT desarrollada. 59

A.1. Plano de la FMC 63

B.1. Representación del circuito neumático de la automatización de la FMC 65

C.1. Presentación manual técnico PiXtend [18]. 67
C.2. Instrucciones de conexión[18]. 68
C.3. Microcontrolador-PiXtend [18]. 69
C.4. Entradas y salidas digitales [18]. 70
C.5. Instrucciones de conexión de entradas digitales [18]. 71
C.6. Salidas digitales [18]. 72
C.7. Instrucciones de conexión de salidas digitales [18]. 73
C.8. Relés de salida [18]. 74
C.9. Instrucciones de conexión de relés de salida [18]. 75
C.10. Entradas y salidas GPIOs [18]. 76
C.11. Instrucciones de conexión de GPIOs [18]. 77
C.12. Entradas y salidas analógicas [18]. 78
C.13. Entradas analógicas [18]. 79
C.14. Salidas y entradas especiales [18]. 80
C.15. Instrucciones de conexión de salidas ([18]. 81

D.1. Presentación Librería PiXtend Python Library (PPL) Python [18]. 83
D.2. Ejemplo de programación [18]. 84
D.3. Funciones pertenecientes a la librería (PPL) Python [18]. 85
D.4. Funciones pertenecientes a la librería PPL Python [18]. 86
D.5. Funciones pertenecientes a la librería PPL Python [18]. 87
D.6. Funciones pertenecientes a la librería PPL Python [18]. 88
D.7. Funciones pertenecientes a la librería PPL Python [18]. 89
D.8. Funciones pertenecientes a la librería PPL Python [18]. 90

Índice de tablas

2.1. Simbología de componentes neumáticos [12]. 27

3.1. Especificaciones del modelo *Raspberry* Pi B [17]. 35
3.2. Conexión del sistema de borneras. 37
3.3. Conexión del PiXtend con los componentes de la FMC. 38
3.4. Variables de monitorización OPC-UA. 44
3.5. Variables de control OPC-UA. 44
3.6. Variables de control y monitorización MQTT. 45
3.7. Protocolos y servicios utilizados para la comunicación entre los dispositivos empleados en este proyecto. 45

4.1. Comparación entre la solución IIoT Siemens, Schneider Electric, Mitsubishi Electric y la solución desarrollada. 60

Cláusula de Propiedad Intelectual

Alex Fernando Chuya Machuca, autor del trabajo de titulación "Sistema de supervisión y control de una celda de manufactura a través de tecnologías habilitantes de internet industrial de las cosas", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 14 de marzo de 2022.

Alex Fernando Chuya Machuca

C.I: 070658936-3

Cláusula de Propiedad Intelectual

Edwin Xavier Alvarez Murudumbay, autor del trabajo de titulación "Sistema de supervisión y control de una celda de manufactura a través de tecnologías habilitantes de internet industrial de las cosas", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 14 de marzo de 2022.

Edwin Xavier Alvarez Murudumbay

C.I: 030251045-8

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Alex Fernando Chuya Machuca en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Sistema de supervisión y control de una celda de manufactura a través de tecnologías habilitantes de internet industrial de las cosas", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 14 de marzo de 2022.

Alex Fernando Chuya Machuca

C.I: 070658936-3

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Edwin Xavier Alvarez Murudumbay en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Sistema de supervisión y control de una celda de manufactura a través de tecnologías habilitantes de internet industrial de las cosas", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

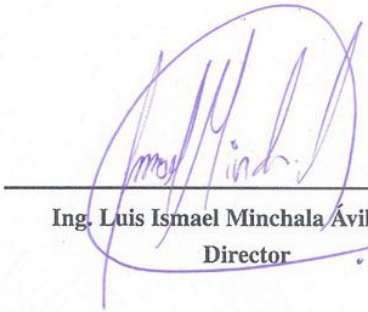
Cuenca, 14 de marzo de 2022.

Edwin Xavier Alvarez Murudumbay

C.I: 030251045-8

Certifico

Que el presente proyecto de tesis: Sistema de supervisión y control de una celda de manufactura a través de tecnologías habilitantes de internet industrial de las cosas, fue dirigido y revisado por mi persona.



Ing. Luis Ismael Minchala Ávila, PhD
Director

Dedicatoria

El presente trabajo esta dedicado a mi familia por haber sido mi apoyo a lo largo de toda mi carrera universitaria y a lo largo de mi vida. A todas las personas especiales que me acompanaron en esta etapa, aportando a mi formacion tanto profesional y como ser humano.

Alex Fernando Chuya Machuca

Dedicatoria

A mis padres, Luz y Valeriano, quienes con su amor y confianza me han permitido llegar a cumplir un sueño más, gracias por inculcar en mí el ejemplo de superación y perseverancia se los debo de manera infinita. A mis hermanos Cristian, Rosalia, Jennifer y Evelyn por su cariño y apoyo incondicional, durante todo este proceso, por estar conmigo en todo momento gracias. A mis sobrinos Jazmin y Junior, por esas pequeñas sonrisas que alegran mis días.

Edwin Xavier Alvarez Murudumbay

Agradecimientos

En primer lugar quiero agradecer a mi tutor Ing. Luis Minchala, quien con sus conocimientos y apoyo me guió a través de cada una de las etapas de este proyecto para alcanzar los resultados que buscaba.

También quiero agradecer a la Universidad de Cuenca por brindarme todos los recursos y herramientas que fueron necesarios para llevar a cabo la elaboración de este proyecto de tesis. No hubiese podido arribar a estos resultados de no haber sido por su incondicional ayuda.

Por último, quiero agradecer a todos mis compañeros y a mi familia, por apoyarme aún cuando mis ánimos decaían. En especial, quiero hacer mención de mis padres, que siempre estuvieron ahí para darme palabras de apoyo y un abrazo reconfortante para renovar energías.

Muchas gracias a todos.

Alex Fernando Chuya Machuca

Agradecimientos

A mis padres, por todo el apoyo brindado a lo largo de mis estudios y por haber estado en los buenos y malos momentos, a mis hermanos por brindarme siempre el apoyo incondicional, a mis sobrinos por alegrarme siempre los días, a mis compañeros, en especial a mi compañero de tesis Alex Chuya, por toda la dedicación brindada en este trabajo, a mis profesores, que a lo largo de mi carrera han brindado sus conocimientos y de manera especial a mi director de tesis Ing. Ismael Minchala, por el apoyo y tiempo dedicado a este proyecto.

Edwin Xavier Alvarez Murudumbay

Abreviaciones y Acrónimos

ACK Acknowledgement. 22, 56
AJAX Asynchronous JavaScript And XML. 21
AWS Amazon Web Services. 31, 51

BACNet Building Automation and Control Network. 58

CIM Computer-Integrated Manufacturing. 32, 43
CNC Computer Numerical Control. 21
CPS Cyber-Physical System. 29
CPU Central Processing Unit. 24
CSS Cascading Style Sheets. 21

FMC Flexible Manufacturing Cell. 1, 2, 6, 8, 20, 24–26, 31–33, 35, 36, 38, 39, 44, 48
FMS Flexible Manufacturing System. 26

HMI Human Machine Interface. 21, 23, 24
HTML HyperText Markup Language. 21, 54
HTTP Hypertext Transfer Protocol. 30, 31, 54, 55

IIoT Industrial Internet of the Things. 1, 2, 7, 19–21, 25, 26, 29, 35, 57–59, 61
IoT Internet of the Things. 22, 23, 29
IP Internet Protocol. 28, 30, 50, 52, 53

JSON JavaScript Object Notation. 22

KPI Key Performance Indicator. 58

MAPS Mitsubishi Adroit Process Suite. 58
MPS Modular Production System. 22, 24
MQTT Message Queuing Telemetry Transport. 1, 2, 8, 21, 30, 31, 44, 45, 50–52, 56, 58, 61

OP Open Platform Communications. 29
OPC-DA Open Platform Communications-Data Access. 58
OPC-UA Open Platform Communications-Unified Architecture. 1, 2, 8, 20–24, 29, 31, 43, 44, 50, 51, 58, 61
OT Operational Technology. 28, 29, 31, 58, 61

PLC Programmable Logic Controller. 1, 2, 20, 23–25, 31, 34–36, 43, 56
PPL PiXtend Python Library. 7, 43, 83, 85–90

PSH Push. 22

QoS quality of service. 52

RAMI Reference Architectural Model Industrie. 23, 24
REST Representational State Transfer. 31
RFID Radio Frequency Identification. 24
RTDE Real-Time Data Exchange. 23
RTT round-trip time. 6, 56, 61

SCADA Supervisory Control and Data Acquisition. 58
SDN Software Defined Networking. 22, 23
SPI Serial Peripheral Interface. 43

TCP Transmission Control Protocol. 30
TI Information Technology. 28, 29, 31, 57, 58, 61

CAPÍTULO 1

Introducción

Este capítulo presenta la definición del problema, los objetivos y el alcance del trabajo. Adicionalmente, incluye una revisión de literatura especializada en el área del internet industrial de las cosas (IIoT).

1.1. Antecedentes

En los últimos años una gran cantidad de organizaciones empresariales han obtenido un avanzado desarrollo industrial a nivel global. Sin embargo, existe una gran brecha de esta situación entre América latina y el resto del mundo [1].

Latinoamérica cuenta con la oportunidad de aprovechar las posibilidades que internet brinda para evolucionar y crecer. La investigación, colaboración y la aplicación de nuevas tecnologías promueven un cambio en el modelo productivo basado en internet [2].

Los conceptos del paradigma de la cuarta revolución industrial, denominada Industria 4.0 y manufactura inteligente, son relativamente nuevos. Este concepto contempla la integración conjunta de las tecnologías digitales en la industria de fabricación. Es decir, la incorporación al ambiente de manufactura de tecnologías como el internet de las cosas, cómputo móvil, la nube, el *big data*, redes de sensores, sistemas embebidos y dispositivos móviles, entre otros. La introducción de este concepto a las industrias provoca grandes capacidades de automatización, monitorización y optimización. Estos cambios mejoran la eficiencia operativa y el desempeño organizacional [3].

Hoy en día, muchas arquitecturas utilizadas para monitorización de procesos físicos están sujetas de algún modo al elevado coste y la ausencia de equipos necesarios para su implementación[4].

El éxito en la implementación de un sistema de monitorización, depende especialmente de la robustez del sistema, la comunicación con la capa física y el tratamiento de la información [4].

La aparición de la Industria 4.0 ha permitido que nuevas tecnologías utilicen el intercambio de información y se desarrollen como en el caso del protocolo de comunicación OPC-UA. Esto es así porque la mayor parte de la información actual tiende a transportase a través del internet. Este avance permite que las tecnologías para el intercambio de información y el desarrollo de interfaces web mejoren significativamente [5].

Una implementación preliminar de la Industria 4.0 para muchas empresas manufactureras e industriales considera la implementación de dispositivos de bajo coste, dispositivos compatibles con tecnologías de software libre.

Este proyecto se desarrolló en el laboratorio de máquinas de la Universidad de Cuenca. Entre los objetivos de este trabajo se incluye la supervisión de los procesos automatizados en una celda de manufactura de manera local y remota. La automatización de los procesos de manufactura se despliega a través de controladores lógicos programables (PLC) de bajo coste como PiXtend y tecnologías de comunicación compatibles con este dispositivo como el protocolo OPC-UA. Un computador de placa única como Raspberry Pi es utilizada como puerta de enlace, supervisión y almacenamiento local. La supervisión remota utiliza protocolos de comunicación en tiempo real. La construcción de la interfaz web y almacenamiento en la nube se desarrollan con herramientas de software libre.

Este proyecto de tesis aporta con conceptualizaciones técnicas sobre el tema. Adicionalmente, el enfoque experimental del trabajo, aplicado con el método científico de una investigación cuantitativa, permite replicar la propuesta tecnológica a través de la metodología reportada en este documento. La finalidad es mejorar la calidad y productividad en procesos a través de herramientas de bajo costo, habilitantes de la tecnología IIoT.

1.2. Definición del problema

1.2.1. Antecedentes

Los atributos de la Industria 4.0 son difíciles de reproducir por la escasez de recursos, grandes costos y falta de ingenio en el desarrollo e implementación de estas tecnologías. Esta incertidumbre tiene solución incorporando tecnologías habilitantes compatibles y accesibles para cualquier tipo de industria.[3]

1.2.2. Definición tácita del problema

La gran difusión de tecnologías del internet industrial a nivel global es el aspecto principal para incorporar nuevas técnicas y metodologías de producción en la industria manufacturera. La aplicación de esta forma de producción requiere condiciones de infraestructura técnica, formación especializada e integración de procesos [6].

Este proyecto técnico busca reincorporar el funcionamiento de una celda de manufactura flexible (FMC) que se encuentra en el laboratorio de máquinas eléctricas de la Universidad de Cuenca. Esta FMC funciona utilizando sistemas electro-neumáticos y eléctricos. La celda de manufactura, tiene sistemas automáticos des-actualizados y de herencia tecnológica (legacy).

1.2.3. Revisión del estado del arte

Este apartado presenta una síntesis estructurada de una recopilación de trabajos referentes al sistema de supervisión desarrollado para este proyecto con la finalidad de presentar el estado actual de proyectos relativos al IIoT.

La deficiente monitorización de procesos industriales es la problemática presentada en [2]. Alrededor de esta postura el autor propone una arquitectura con un *Broker* central que controla la comunicación entre clientes MQTT en diferentes etapas de comunicación. El proyecto consiste en una simulación de un proceso industrial en *Factory* I/O. El proceso industrial es monitorizado en tiempo real de manera remota mediante una interfaz web. Los resultados evidencia la teoría que la Industria 4.0 incide en la monitorización adecuada de procesos industriales. Para confirmar este hecho es analizado el tráfico de paquetes de información transferida por el protocolo MQTT. La prueba de chi cuadrado confirma la hipótesis de que la Industria 4.0 incide en la adecuada monitorización de procesos industriales.

Los altos costes y carencia de dispositivos necesarios para la implementación de arquitecturas para monitorización industrial es la principal dificultad para el despliegue de la Industria 4.0. En [4] propone la implementación de una arquitectura para un sistema de monitorización de procesos basados en sistemas ciberfísicos utilizando herramientas de bajo coste.

El autor plantea un sistema de monitorización basado en condición. El sistema de monitorización basado en condición es una una estrategia de mantenimiento enfocada en la prevención de fallos, tiempos de inactividad y practicas de mantenimiento innecesarias. Este sistema implementado controla la salud de los activos de manera a poder determinar que acciones de mantenimiento necesitan completarse y cuando utilizando el intercambio de información entre dispositivos.

El procesamiento y monitorización de procesos es realizado con un dispositivo *Raspberry* Pi. Este dispositivo controla máquinas de control numérico (Computer Numerical Control (CNC)) mediante agentes externos. Durante monitorización son obtenidos indicadores de estado y alarmas para prevención de posibles fallas. Además se obtienen tendencias del comportamiento de las variables en un rango de tiempo determinado. Los resultados obtenidos muestran que el sistema es capaz de conectarse con múltiples máquinas obteniendo información de diversas variables de estado. La información es mostrada y usada para generar reportes de conexión, sobrecalentamiento y almacenamiento.

En [5] desarrollan una plataforma web para la creación de interfaces humano máquina (Human Machine Interface (HMI)) haciendo uso del protocolo OPC-UA y la técnica de programación web Asynchronous JavaScript And XML (AJAX). La Figura 1.1 ilustra la arquitectura del sistema de comunicación.

La plataforma web recupera información en el controlador a través del protocolo de subscripción a las variables establecidas en el servidor OPC-UA. El HMI permite identificar el estado del proceso monitorizado. El diseño de la interfaz web se basó en una estructura sencilla a través de HyperText Markup Language (HTML), Cascading Style Sheets (CSS) y *JavasScript*.

El cliente web posee servicios para transferencia de datos. Los métodos para ofrecer estos servicios son

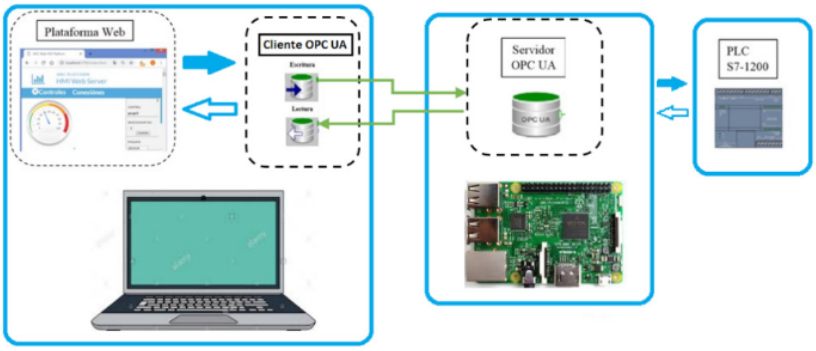


Figura 1.1: Arquitectura de la plataforma web desarrollada [5].

Get_Data_Source y *Find_Servers* y *Read_Data*. El servicio *Monitor_Service* administra las instancias de monitoreo. Estas instancias acceden a la información de las variables del servidor utilizando un identificador único para recuperar su valor en tiempo real.

El servidor OPC-UA crea nuevas variables del proceso. La información en el servidor OPC-UA ingresa al cliente como una cadena (JavaScript Object Notation (JSON)). Las pruebas de conexión se realizan analizando el tráfico de red. Se observan los paquetes (Push (PSH)) y (Acknowledgement (ACK)) generados al establecer una conexión cliente-servidor.

Los resultados del análisis de tráfico evidencian un pico máximo y un ancho de banda consumido de 0.14 Mb/s. Estos resultados concluyen un intercambio de información en la plataforma muy bajo. En consecuencia, es un proyecto ideal para trabajar con sistemas de bajos recursos.

En [7] realizan un estudio detallado para desarrollar un sistema de manufactura flexible utilizando tecnologías orientadas a la Industria 4.0. Esta investigación busca soluciones para una integración de los sistemas de producción modular (Modular Production System (MPS)) incorporando tecnologías para el aumento de flexibilidad en la verificación, manipulación, clasificación y agrupamiento de piezas metálicas. El investigador realiza simulaciones con el fin de obtener la mejor flexibilidad del sistema. Esta flexibilidad la analiza dividiendo las diferentes etapas de fabricación en estructuras maestro-esclavo.

La continua expansión de conectividad entre dispositivos dificulta la administración y control de la red. La cantidad de tráfico en IoT es más creciente que la cantidad de conexiones, la recopilación no regulada de datos personales y el crecimiento de IoT plantea problemas de privacidad y seguridad al consumidor. El costo de infraestructura IoT será extremadamente alto por limitaciones de ancho de banda.

En base a las dificultades anteriores el autor en [8] desarrolla un proyecto que integra el Internet of the Things (IoT) y las redes definidas por software (Software Defined Networking (SDN)). El proyecto establece un escenario SDN con aplicación IoT basada en la industria 4.0. La figura 1.2 ilustra la arquitectura implementada basada en la arquitectura IoT-SDN.

La implementación de comunicación de IoT-SDN utiliza un controlador SDN *OpenFlow* acoplado a una

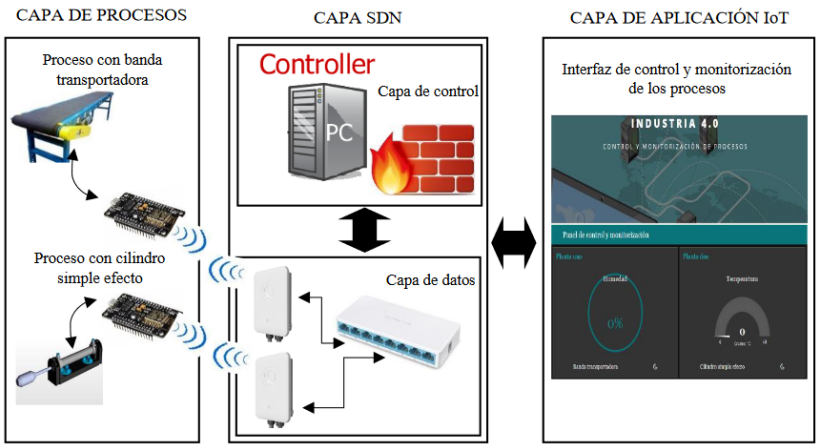


Figura 1.2: Arquitectura IoT- SDN del prototipo implementado [8].

placa *Raspberry* Pi, un *switch OpenFlow* dirigidos hacia dos puntos de acceso y un servidor IoT implementado en placas *Raspberry* Pi. Un punto de acceso permite comunicación entre procesos.

Con la herramienta Wireshark realizan pruebas de intercambio de mensajes de elementos *OpenFlow* de la SDN entre conmutadores y controlador. Las pruebas de tráfico de datos en la SDN utilizan la herramienta *iperf*. Estas pruebas determinan el *throughput* a través de los dos procesos y el servidor IoT.

En [9] realizan la simulación de una celda automatizada con robótica colaborativa. Este trabajo implementa intercomunicación a través de protocolos de comunicación industriales. Un PLC virtual maneja la lógica de control de la celda mientras dos robots colaborativos intercambian piezas. Estos robots están integrados con visión artificial. Una HMI es desarrollada como plataforma de monitorización y control. La solución se implementó para usuarios locales y remotos.

El proyecto investiga el desarrollo de un *Digital Twin*. La herramienta Codesys permite el control; la conexión virtual ocupa un dispositivo *Modbus_COM_Port*. La virtualización de los robots utiliza el *software* URSim. La simulación de secuencias gráficas emplea el *software* RoboDK. El proyecto utiliza el protocolo de comunicación OPC-UA para obtener los datos de simulación. Esta solución plantea la idea de obtener las mejores simulaciones para cargarlos a equipos físicos reales. Por otro lado, La interfaz HMI emplea la herramienta NB Designer de la empresa Omron.

La monitorización remota se realiza a través del protocolo Real-Time Data Exchange (RTDE). Las pruebas realizadas consistieron en la medición de los tiempos de secuencia de los procesos simulados por los robots virtuales. Como resultado de la medición consiguieron un tiempo de ciclo de secuencia de 50 a 58 segundos. Este resultado no es aceptable en ningún sector industrial. La solución consistió en un ajuste de las velocidades de secuencia de ciclo de virtualización.

En [10] implementaron un dispositivo de control para la Industria 4.0. El Modelo de Arquitectura Reference Architectural Model Industrie (RAMI) 4.0 fue utilizado para este trabajo. RAMI 4.0 es un mapa tridimensional que muestra cómo abordar el despliegue Industria 4.0 de una forma estructurada. La comunicación del dispositi-

vo ocupó el protocolo OPC-UA. Este dispositivo proporciona funcionalidades de programación de procesos, identificación de piezas por radiofrecuencia, comunicación en red y supervisión de equipos.

La Figura 1.3 presenta la arquitectura propuesta para el desarrollo del dispositivo de control. En la figura aparecen diferentes bloques de implementación desde el sistema de manufactura hasta los clientes OPC-UA. El control es realizado con el software OPEN PLC. El PLC interactúa con un Lector Radio Frequency Identification (RFID) para control de procesos. La arquitectura cliente servidor está presente en la comunicación mediante el protocolo OPC-UA. El sistema de manufactura es un identificador de piezas que interactúa con el lector RFID. Por otro lado, una HMI administra los procesos de manera local.

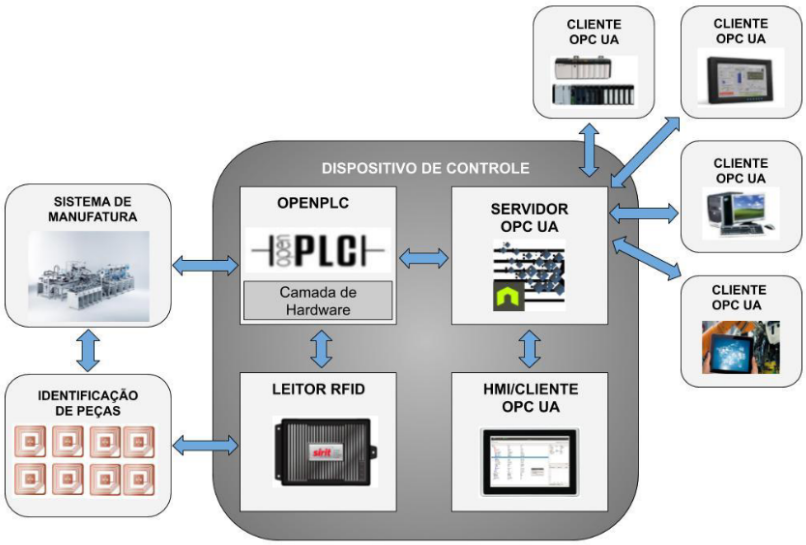


Figura 1.3: Arquitectura propuesta para el desarrollo de un dispositivo de control para la industria 4.0 [10].

Se realizaron pruebas experimentales identificando piezas de diferentes colores. Los resultados validaron la conexión entre las capas de activos técnicos, integración y comunicación. La estación de clasificación encontrada en la capa de activos técnicos de RAMI 4.0 forma parte del conjunto Festo MPS 200. Este conjunto vinculado con el lector RFID identifica las piezas de color en posiciones diferentes.

En las pruebas realizadas en base a la integración se evaluaron las particularidades del servidor OPC-UA, conectando uno o más clientes de diferentes tipos y plataformas. El funcionamiento del dispositivo de control fue robusto y sin interrupciones. Se produjeron picos en la monitorización del uso del (Central Processing Unit (CPU)) cuando cada cliente se conectó. Se comprobó que el uso de seguridad y políticas admitidas por el dispositivo de control no afecta el desempeño del dispositivo.

1.2.4. Propuesta

El propósito de este trabajo es activar el funcionamiento de la FMC utilizando dispositivos actuales de bajo costo con tecnologías de software compatibles con estos dispositivos. Esta implementación permite emular una solución inicial al incorporar esta celda de manufactura hacia el nuevo paradigma de la cuarta revolución

industrial o Industria 4.0.

Este proyecto aporta con una metodología de integración tecnológica que permite agilizar procesos, optimizar la productividad, eficiencia y calidad en manufactura. La manufactura flexible consigue una relación con la Industria 4.0 por las herramientas que otorga esta nueva revolución.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una solución de supervisión y control aplicado a una FMC usando tecnologías de IIoT de bajo costo.

1.3.2. Objetivos específicos

Este trabajo presenta los siguientes objetivos específicos.

- Reactivar la celda de manufactura flexible perteneciente al laboratorio de máquinas de la Universidad de Cuenca.
- Seleccionar la(s) tecnología(s) habilitantes de IIoT más convenientes para automatizar la FMC.
- Generar pruebas y resultados en la FMC.
- Analizar e interpretar los datos en la FMC con el propósito de ayudar en la toma de decisiones y automatización de procesos.

1.4. Contribuciones de la tesis

Este trabajo presenta las siguientes contribuciones.

- Una FMC integrada con una fuente de alimentación EDR-120-24, un dispositivo PLC PiXtend y una *Raspberry* Pi 3 B+.
- Un sistema de supervisión IIoT.
- Un programa de automatización de la FMC.
- Mantenimiento y reemplazo de equipos defectuosos en la FMC.

Capítulo 2

Fundamentos teóricos

Este capítulo presenta conceptos asociados con la supervisión y control de una celda de manufactura aplicando tecnologías habilitantes del IIoT.

2.1. Celda de manufactura flexible

Una celda de manufactura flexible FMC es un módulo de los sistemas de producción llamados (Flexible Manufacturing System (FMS)). Una FMC constituye un arreglo de máquinas de mecanizado o ensamble que atienden un propósito particular. Estos sistemas proveen flexibilidad a diferentes rutas del material a procesar o semi-procesar en las estaciones de trabajo. Las estaciones de trabajo ejecutan operaciones simples, producen o ensamblan una variedad de tamaños, formas y calidades de producción o ensamble [11].

2.2. Neumática

Neumática es la parte de la física que trata de las propiedades de los gases desde el punto de vista de su movimiento. En plantas industriales y sectores de todo el mundo utilizan la neumática y sistemas de control electro-neumáticos. Estos sistemas controlan el funcionamiento de equipos de fabricación, ensamblaje y envasado. Por lo general la neumática es utilizada en la técnica de manipulación [12].

2.2.1. Nivel de presión

Normalmente, los componentes neumáticos son concebidos para soportar una presión de funcionamiento desde 800 hasta 1000 kPa (8 hasta 10 bares). Para un funcionamiento económico, es suficiente una presión de 600 kPa (6 bares). Sin embargo, es recomendable que el compresor sea capaz de generar una presión desde 650 hasta 700 kPa (6.5 hasta 7 bares) debido a pérdidas en los componentes.[12]

2.2.2. Elementos principales de un sistema neumático

Un sistema neumático está constituido de varios componentes. Los principales elementos de un sistema neumático son los siguientes:

- **Compresor**:Es una máquina térmica utilizada para incrementar la presión de un fluido. La elección de un compresor depende de la presión de trabajo y de la cantidad de aire necesario en un sistema neumático.
- **Red de tuberías**:La selección de su diámetro depende de la presión de funcionamiento y la cantidad de aire proveniente del compresor.
- **Filtro de aire comprimido**: Protege los componentes del sistema neumático de impurezas y contaminantes
- **Válvula reguladora de presión**: Garantiza un nivel constante de aire comprimido.
- **Sistemas de accionamiento y actuadores**:Un sistema de accionamiento o actuador convierte energía en trabajo. Los actuadores neumáticos se clasifican en dos grupos: actuadores de movimiento lineal y actuadores de movimiento giratorio.
- **Válvulas distribuidoras**: Son utilizadas para desviar el caudal de aire comprimido. El accionamiento puede ser manual, mecánico, neumático o eléctrico. Las electroválvulas funcionan con energía eléctrica en la parte de control y neumática en la parte funcional.

La Tabla 2.1 muestra una simbología de los componentes que se usan con frecuencia en sistemas de control neumáticos.

Tabla 2.1: Simbología de componentes neumáticos [12].

Función	Simbología
Compresor	
Filtro	
Válvula reguladora de presión	
Válvula de 3/2 vías	
Válvula de 4/2 vías	
Válvula de 5/2 vías	
Retorno por muelle	
Válvula servopilotada	
Cilindro de simple efecto	
Cilindro de doble efecto	
Pinza neumática de doble efecto	
Actuador neumático giratorio	
Manómetro	

2.3. Comunicaciones industriales

Los sistemas de automatización actuales están basados en una arquitectura jerárquica conocida como pirámide de automatización. Esta arquitectura data de la década de 1970 y fue ideada para estructurar la complejidad de los sistemas de automatización de ese momento [13]. La Figura 2.1 presenta una ejemplificación general de la pirámide de automatización. En la ilustración se aprecian los niveles de jerarquía, los elementos de cada nivel y también las tecnologías de comunicación.

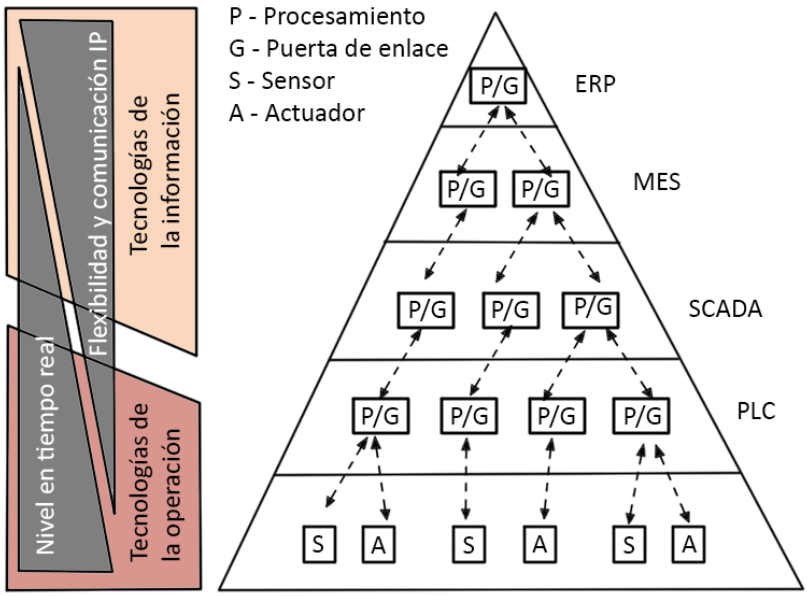


Figura 2.1: Pirámide de automatización [13].

Un elemento importante en la pirámide de automatización es la comunicación. Un sistema de comunicación es fundamental para la transferencia de datos dentro y entre las capas funcionales de la pirámide de automatización. Un sistema de comunicación industrial se origina de la necesidad de intercambiar, procesar y almacenar información entre componentes industriales.

2.3.1. Tecnologías de comunicación

La necesidad de un sistema de comunicación incentivó el desarrollo de varios sistemas de intercambio de datos críticos en tiempo real en las capas bajas de la jerarquía de automatización. Estás capas inferiores se resumen hoy en día bajo el término de tecnología operativa Operational Technology (OT) [13].

Los sistemas de comunicación industrial basados en ethernet y los sistemas de bus de campo se utilizan para cumplir requisitos de baja latencia, baja fluctuación y sincronización estrecha.

Las capas superiores de la pirámide aplican tecnologías de la información Information Technology (TI) estándar hoy en día basadas en la comunicación por protocolo de internet Internet Protocol (IP). Una estrategia

para conectar las OT con las TI es usar una puerta de enlace [13].

Existe una gran variedad de sistemas de comunicación que cumplen aspectos como modelado de datos, acceso a datos y con soluciones independientes de la red.

2.3.2. Arquitectura unificada OPC

OPC-UA es un protocolo para comunicación desarrollado como sucesor del Open Platform Communications (OP) *Classic*. Este protocolo permite una comunicación flexible en comunicaciones industriales.

OPC-UA es un protocolo de comunicación cliente-servidor. El servidor brinda acceso a datos y funciones estructurados en un modelo de información orientado a objetos. Los clientes OPC-UA pueden interactuar a través de un conjunto de servicios estandarizados. El modelo de información sobre el que operan los servicios combina ideas de orientación a objetos y tecnologías semánticas[14].

El modelo de información OPC-UA es una red de nodos, o un diagrama estructurado compuesto por nodo y referencias, llamado el espacio de direcciones OPC-UA. Esta estructura gráfica puede describir una amplia variedad de información estructurada (objetos). Cada nodo puede representar una variable, un método o un tipo de datos [14].

2.4. Internet industrial de las cosas

El internet industrial de las cosas IIoT es un sistema que comprende objetos inteligentes en red, sistemas ciber-físicos, tecnologías de información, y plataformas en la nube o de borde. Estas tecnologías permiten acceso, recopilación, análisis, comunicaciones e intercambio de información en tiempo real, inteligente y autónomo. Este sistema recopila información de procesos, productos y/o servicios, dentro del entorno industrial, para optimizar la producción [15].

2.4.1. Industria 4.0

Existe una superposición considerable entre el concepto de Industria 4.0 desarrollado en Alemania y el concepto de Internet industrial originado en Estados Unidos. Una definición de Industria 4.0 es la siguiente:

La industria 4.0 es un término colectivo para tecnologías y conceptos de cadena de valor. Dentro de las fábricas inteligentes modulares estructuradas de la Industria 4.0, los sistemas ciber-físicos Cyber-Physical System (CPS) monitorean los procesos físicos. Estos sistemas crean una copia virtual del mundo físico y toman decisiones descentralizadas. A través del IoT, los CPS se comunican y cooperan entre sí y con los operarios en tiempo real [15].

2.4.2. Protocolos de aplicación IIoT en las TI

Existe una variedad de protocolos de aplicación para comunicación en tiempo real en la Industria 4.0. Los siguientes protocolos son usados en la elaboración de este proyecto.

Websocket: El protocolo Websocket se desarrolló para cumplir con el intercambio constante de datos entre el cliente y servidor. Anteriormente este proceso no era soportado por Hypertext Transfer Protocol (HTTP).

El protocolo consiste de un canal de comunicación bidireccional que funciona a través de un *socket*, además de tener una comunicación asíncrona. El funcionamiento de Websocket se divide en dos partes: protocolo de enlace y transferencia de datos [16].

Message Queue Telemetry Transport (MQTT): Es un protocolo de mensajería de intercambio que utiliza el estándar de publicación-subscripción para transportar mensajes entre un servidor y clientes. Se ejecuta sobre (Transmission Control Protocol (TCP))/IP.

En el patrón de publicación-subscripción los mensajes intercambiados entre diferentes clientes se realizan a través de un servidor llamado *Broker*. El *Broker* filtra los mensajes y los distribuye a los clientes según el *topic*. El *topic* es un identificador contenido en cada mensaje. Los que publican un mensaje al *Broker* con un *topic* se llaman publicadores, y los que suscriben uno o más temas para leer mensajes específicos se llaman suscriptores [16].

Capítulo 3

Metodología

Este Capítulo describe la metodología aplicada para el desarrollo e implementación de la comunicación, automatización, control y monitorización del sistema de supervisión de la FMC. Este apartado presenta con detalle la arquitectura, protocolos de comunicación, componentes principales y herramientas de *software* empleados para el desarrollo de este sistema.

3.1. Descripción del proceso

El sistema implementado en este trabajo tiene el propósito de supervisar los procesos de producción automatizados en una FMC. La implementación de este sistema está basada en una arquitectura de integración de tecnologías OT y TI [13]. Esta arquitectura está expuesta en la Figura 3.1. Esta Figura expresa la forma de transferencia de datos desde los sistemas OT (PLC) hacia un servidor OPC-UA, y hacia aplicaciones TI. Los sistemas TI usan protocolos como HTTP, Websockets, MQTT y estilos arquitectónicos como Representational State Transfer (REST), los sistemas OT usan protocolos de comunicación industriales y bus de campo. La integración entre estas dos tecnologías es posible mediante una conversión de protocolos realizada mediante un dispositivo intermediario.

La Figura 3.2 presenta el sistema de comunicación desarrollado con los componentes y herramientas de *software* utilizados para la implementación de este trabajo. Los datos de la FMC se almacenan en variables OPC-UA. El procesamiento para automatización de tareas lo maneja un PLC de bajo costo como el PiXtend. Mientras que una *Raspberry* Pi administra la conversión de protocolos para la posterior supervisión remota y almacenamiento en la nube. La librería PPL PiXtend de Python es ocupada para la automatización de tareas, mientras que las librerías *Channels*, Paho MTT, y OPC-UA de Python son ocupados para la comunicación hacia la nube. El servicio de Amazon Web Services (AWS) es utilizado para la supervisión remota y almacenamiento en la nube, mientras que la aplicación de control está diseñada y desarrollada utilizando el *framework* de Django y herramientas de desarrollo web. El almacenamiento ocupa la herramienta de *software* SQLite3.

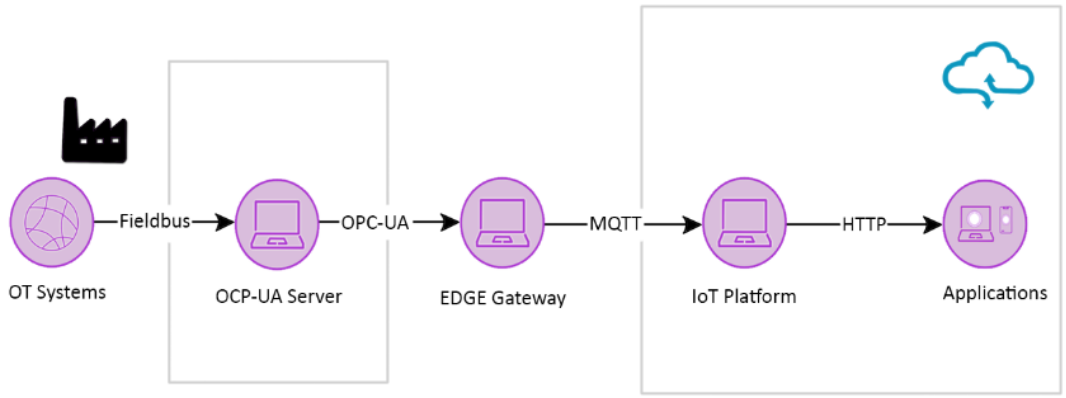


Figura 3.1: Ilustración de un sistema de comunicación para la industria 4.0.

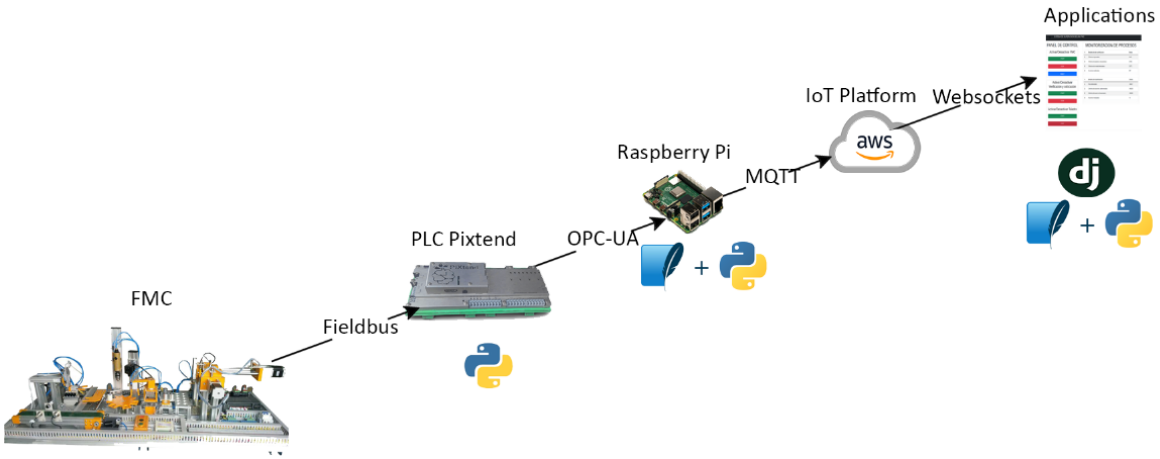


Figura 3.2: Sistema de comunicación implementado.

3.2. Componentes principales del sistema de supervisión

3.2.1. Celda de manufactura flexible

La FMC empleada para este proyecto está diseñada para el estudio y trabajo práctico en conocimientos profundos sobre Computer-Integrated Manufacturing (CIM). La FMC está compuesta por diferentes módulos, cada uno para reproducir una aplicación básica regularmente utilizada a la industria. Cada módulo está implementado con componentes electrónicos, mecánicos y neumáticos reales.

La FMC completa se presenta en la Figura 3.3. El funcionamiento de todos los dispositivos de la FMC requiere una alimentación de 24V; adicionalmente, los actuadores necesitan aire comprimido a una presión mínima de 5 bares para funcionar.

Los módulos de la FMC reproducen aplicaciones usualmente utilizadas en la industria. Cada módulo está implementado con componentes electrónicos, mecánicos y electro-neumáticos. La salida de la actividad de un

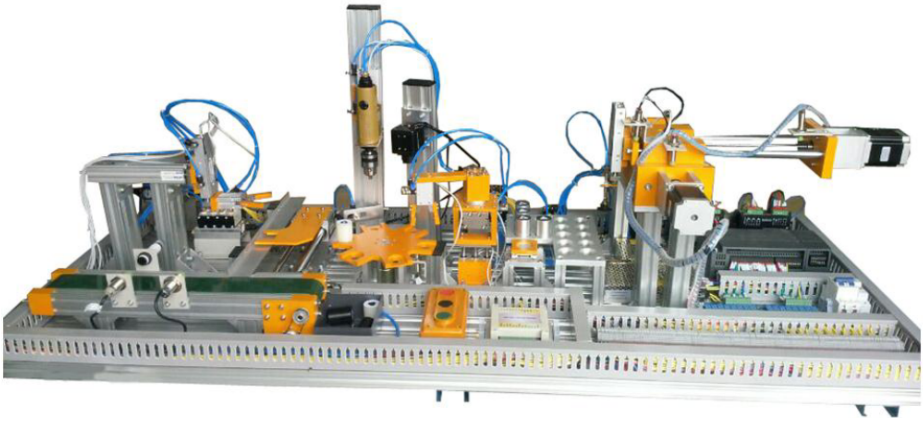


Figura 3.3: Modelo de FMC DE LORENZO.

módulo sirve como entrada de actividad al siguiente. Los módulos de la FMC son los siguientes:

- **Unidad transportadora**:A través de esta unidad os materiales se transfieren en movimiento lineal por medio de una banda transportadora impulsada por un motor DC. En esta unidad se encuentran sensores fotoeléctricos y capacitivos.

- **Unidad de transferencia lineal**: Es un actuador lineal controlado electro-neumáticamente.

- **Unidad pick & place**: Se compone de tres partes principales: un brazo vertical (cilindro vertical de doble acción), un brazo horizontal (cilindro horizontal de doble acción) y un gripper angular (de doble acción para sostener las piezas de trabajo).

- **Mesa rotativa de seis estaciones**:Consiste de una mesa de trabajo circular horizontal de seis posiciones. Está equipado con un sensor capacitivo de proximidad para conocer la posición de giro de la mesa rotativa.

- **Unidad de perforación**:Esta unidad está conformada por un cilindro de carrera corta que sujeta la pieza, un cilindro vertical de doble acción que desplaza el taladro sobre la pieza y el taladro neumático.

- **Brazo de descarga**: Es un sistema electro-neumático en el cual el movimiento vertical y giro es controlado por un gripper angular, actuador lineal y rotativo.

- **Unidad de medición de peso**:Consiste de dos elementos: una celda de carga y el circuito de interface. La celda de carga genera una resistencia variable relacionada con el peso del objeto que se coloca sobre ella. El circuito de interfaz recibe, procesa y envía este valor a la entrada analógica del PiXtend.

- **Unidad paletizadora**:Esta unidad está dirigida por dos motores de pasos. El primer motor controla la posición X y el otro motor la posición Y. Cada motor cuenta con sensores capacitivos de proximidad. Para manejar las piezas de trabajo, se provee una copa de succión que es controlada por medio de un cilindro plano de doble acción.

3.2.2. PLC PiXtend V1.3

El PiXtend V1.3 (Figura 3.4) es un PLC basado en la computadora de placa única *Raspberry* Pi de alto rendimiento. Está compuesto de entradas y salidas digitales/analógicas que permiten conectar una variedad de dispositivos como sensores y actuadores industriales. La utilidad de PiXtend reside en su compatibilidad con plataformas de desarrollo/programación de software libre como Python. El Apéndice C presenta la hoja de especificaciones técnicas del PiXtend V1.3.

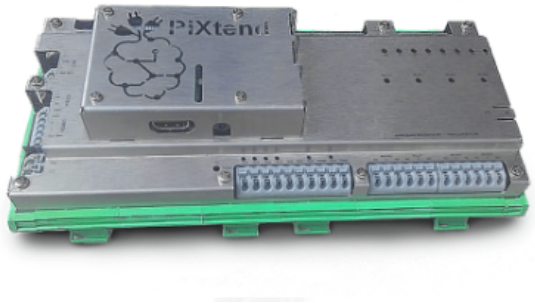


Figura 3.4: PLC PiXtend V.1.3.

3.2.3. *Raspberry* Pi

La *Raspberry* Pi es una serie de ordenadores de placa reducida de bajo costo, desarrollado con el objetivo de poner en manos de cualquier persona el manejo de la informática y creación digital. Específicamente la *Raspberry* Pi 3 *Model* B (Figura 3.5) es utilizada como punto de interconexión entre la etapa de control y la nube. Este dispositivo proporciona el almacenamiento y supervisión local. La Tabla 3.1 muestra las especificaciones de parámetros claves en términos de potencia, memoria y comunicación de la placa utilizada.

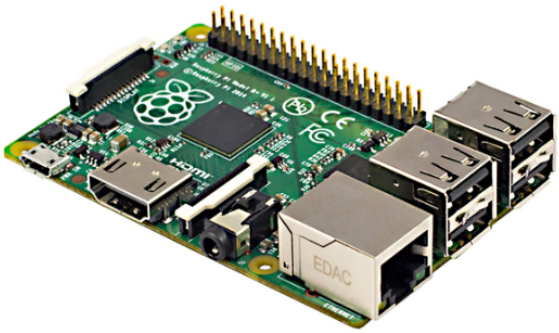


Figura 3.5: *Raspberry* Pi 3 Model B [17].

Tabla 3.1: Especificaciones del modelo *Raspberry* Pi B [17].

Parámetros	*Raspberry* Pi B+
Procesador	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC
Tensión de funcionamiento	5V/2.5A DC
Velocidad de Reloj	1.4 GHz
Ancho de bus (bits)	32
Memoria	1GB LPDDR2 DRAM
Memoria flash	-
EEPROM	-
Conectividad de red	Gigabit Ethernet over USB 2.0 (300 Mbps de máximo teórico)
Conectividad inalámbrica	2.4GHz / 5GHz IEEE 802.11.b/g/n/ac Bluetooth 4.2,BLE
Conectividad de E/S	SPI, DSI, UART, SDIO, CSI, GPIO

3.3. Implementación

Esta sección presenta con detalle la implementación del sistema completo, desde la automatización de la FMC hasta el desarrollo de la comunicación a través de internet.

3.3.1. Levantamiento de la FMC

La FMC cuenta con un PLC Schneider Electric. Este PLC está descontinuado del mercado, este trabajo pretende utilizar las tecnologías habilitantes de IIoT reemplazando el antiguo PLC por un PLC PiXtend, realizando nuevas conexiones en el sistema de borneras. Los módulos de la FMC se han diseñado para procesos industriales a escala, los sistemas y estaciones de la FMC funcionan de forma continua y secuencial de acuerdo al sistema programado.

Estructura: La FMC está estructura de la siguiente manera:

- **La máquina**:Elementos neumáticos, eléctricos, mecánicos, sensores y actuadores que componen la estación.
- **Conexiones**:Las interconexiones entre los diferentes módulos se realizan mediante cables XS2.
- **Botoneras**: Se conectan al PLC, situado en el cuadro eléctrico, mediante cables XS2.
- **Cuadro eléctrico**: El cuadro eléctrico está constituido por la fuente, el PLC y un sistema de bornas, necesario para la distribución de alimentación a los dispositivos electrónicos, electro-neumático y eléctrico.

El sistema eléctrico está centralizado en un sistema de borneras. Los componentes neumáticos están conectados a un compresor de aire de presión máxima de 7 bares. Los módulos pueden funcionar como elementos independientes, o en conjunto, como proceso productivo, de acuerdo a la programación del sistema. La comunicación entre estaciones se lleva a cabo mediante el PLC, el panel de control ejecuta un grupo de instrucciones previamente programadas.

Conexiones: La FMC cuenta con un PLC Schneider Electric. Este PLC está descontinuado del mercado, por lo tanto, es sustituido por un PLC PiXtend. También la fuente de alimentación anterior es sustituida por una

fuente de alimentación de modo conmutado DIN. Esta fuente es adecuada para los nuevos equipos instalados en la FMC. Las conexiones de cada componente de la FMC están organizadas en un sistema de borneras. La Tabla 3.2 describe las conexiones de los actuadores de la FMC con el PLC, ordenados de manera numérica. La Tabla 3.3 muestra las conexiones realizadas de los componentes de la FMC a las terminales del PiXtend y la fuente de alimentación.

3.3.2. Automatización de la FMC

Funcionamiento: Los materiales de entrada a la FMC son cilindros de metal, aluminio y plástico. Estos materiales son verificados, transportados y clasificados. Los cilindros de plástico son desechados en la etapa de verificación. Por otro lado, los cilindros de aluminio y acero son clasificados de acuerdo a su material de construcción. Los cilindros metálicos atraviesan cada módulo de la FMC. Los módulos forman parte de las dos estaciones. La Figura 3.6 ilustra el funcionamiento de la FMC.

Adicionalmente se incorpora una unidad para verificación y medición de la perforación de la emulación del taladrado realizado como paso anterior. Esta unidad se implementa utilizando la librería *Open* CV de Python. Lo más relevante del programa desarrollado es la función *HoughCircle*(). Esta función utiliza la técnica de extracción de características básicas utilizada en el procesamiento de imágenes digitales para detectar círculos en imágenes imperfectas.

La cámara que detecta la perforación en el cilindro está activa todo el tiempo. El programa permite capturar y subrayar el circulo detectado, también imprime el valor del radio del círculo.

Entrada de cilindros en la estación: La FMC realiza la gestión de entradas de la siguiente manera. La cinta transportadora funciona independientemente del sistema. El usuario es el encargado de ingresar los cilindros. No se permite el ingreso masivo de cilindros. Los cilindros de plástico se ingresan cada 5 segundos y los cilindros metálicos una vez terminado el proceso de almacenamiento.

Estaciones de la FMC: La FMC está dividida en dos estaciones, cada estación puede trabajar de forma individual o de manera conjunta. Cada estación abarca un conjunto de módulos. Las estaciones y sus módulos correspondientes se listan a continuación:

- **Estación de verificación, manipulación y transporte:**:Esta estación está conformada por la unidad transportadora, unidad de transferencia lineal, unidad *pick* & *place*, mesa rotativa de seis estaciones, unidad de perforación y brazo de descarga.

- **Estación de pesado y almacenamiento**: Esta estación está conformada por la unidad de pesado y unidad paletizadora.

La Figura 3.7 ilustra el diagrama de flujo de la estación de pesado y almacenamiento. El proceso de almacenamiento inicia pesando el cilindro de metal para ubicarlo en el almacén. El diagrama de flujo indica que el cilindro es desplazado mediante una cadena de decisiones, es decir, si se ejecuta la tarea anterior la siguiente

Tabla 3.2: Conexión del sistema de borneras.

Numeración	Módulos	Lateral izquierdo	Niveles izquierdo	Lateral derecho	Niveles derecho
0	Unidad de transferencia lineal	Cilindro sin vástago neumático	1(GND) 3(+24V)	DO0	1(DO0) 3(+24V)
1	Unidad pick & place	Cilindro con vástagos paralelos	1(GND) 3(+24V)	DO1	1(DO1) 3(+24V)
2	Unidad pick & place	Cilindro con vástagos paralelos	1(GND) 3(+24V)	DO2	1(DO2) 3(+24V)
3	Unidad pick & place	Pinza neumática	1(GND) 3(+24V)	DO3	1(DO3) 3(+24V)
4	Unidad de perforación	Cilindro carrera corta	1(GND) 3(+24V)	DO0	1(DO0) 3(+24V)
5	Unidad de perforación	Cilindro con vástagos paralelos	1(GND) 3(+24V)	DO0	1(DO0) 3(+24V)
6	Unidad de perforación	Taladro neumático	1(GND) 3(+24V)	DO0	1(DO0) 3(+24V)
7	Brazo de descarga	Cilindros con vástagos paralelos	1(GND) 3(+24V)	DO2	1(DO2) 3(+24V)
8	Brazo de descarga	Cilindro rotativo neumático	1(GND) 3(+24V)	DO4	1(DO4) 3(+24V)
9	Brazo de descarga	Pinza neumática	1(+24V) 2(GND)	Relay0	1(Relay0) 2(GND)
10	Unidad paletizadora	Succionador de pistón neumático	1(+24V) 2(GND)	Relay3	1(Relay3) 2(GND)
11	Unidad paletizadora	Mini cilindro neumático	1(+24V) 2(GND)	Relay2	1(Relay2) 2(GND)
13	Unidad paletizadora	Sensor capacitivo (DI1)	2(GND) 3(+24V)	Motor a pasos 1	2(GND) 3(+24V)
14	Unidad paletizadora	Sensor capacitivo (DI4)	2(GND) 3(+24V)	Motor a pasos 2	2(GND) 3(+24V)
17	Unidad transportadora	Sensor capacitivo	1(DI0) 2(GND) 3(+24V)	DI0	1(DI0)
18	Unidad transportadora	Sensor fotoeléctrico	1(DI2) 2(GND) 3(+24V)	DI2	1(DI2)
20	Mesa rotativa de seis estaciones	Sensor capacitivo	1(DI5) 2(GND) 3(+24V)	DI5	1(DI5)
21	Unidad transportadora	Motor DC	2(GND) 3(+24V)	1	1
22	Mesa rotativa de seis estaciones	Motor DC	1(+24V) 2(GND)	Relay1	1(Relay1) 2(GND)
4	Alimentación			+/-24V	2 (-24V) 3 (+24V)

continua.

Tabla 3.3: Conexión del PiXtend con los componentes de la FMC.

Módulo	Componente	Terminal de conexión	Punto de conexión
Unidad transportadora	Sensor capacitivo	24V Salida GND	Vcc DI0 GND
Unidad transportadora	Sensor Fotoeléctrico	24V Salida GND	Vcc DI2 GND
Unidad transportadora	Motor DC	24V GND	Vcc GND
Unidad de transferencia lineal	Electroválvula del actuador lineal	24V GND	Vcc DO0
Unidad “PICK AND PLACE”	Electroválvula del cilindro vertical de doble acción	24V GND	Vcc DO1
Unidad “PICK AND PLACE”	Electroválvula del cilindro horizontal de doble acción	24V GND	Vcc DO2
Unidad “PICK AND PLACE”	Electroválvula del gripper angular de doble acción	24V GND	Vcc DO3
Mesa rotativa de seis estaciones	Sensor capacitivo	24V Salida GND	Vcc Relay 1 GND
Mesa rotativa de seis estaciones	Motor DC	24V GND	C del relé 0 GND
Unidad de perforación	Electroválvula del cilindro vertical	24V GND	Vcc DO0
Unidad de perforación	Electroválvula del cilindro horizontal	24V GND	Vcc DO0
Unidad de perforación	Electroválvula del Taladro neumático	24V GND	Vcc DO0
Brazo de descarga	Electroválvula del Actuador lineal	24V GND	Vcc DO2
Brazo de descarga	Electroválvula del actuador Rotativo	24V GND	Vcc DO4
Brazo de descarga	Electroválvula del gripper angular de doble acción	24V GND	Relay0 GND
Unidad de pesado	Circuito de interface	24V Salida GND	Vcc AI1 GND
Unidad paletizadora	Drivers motor de pasos 1 y 2	Señal Habilitación Dirección	PWM0, PWM1 GPIO0, GPIO1 GPIO2, GPIO2
Unidad paletizadora	Succionador a presión	24V GND	Relay3 GND
Unidad paletizadora	Electroválvula del Actuador lineal bidireccional	24V GND	Relay2 GND
Unidad paletizadora	Sensor capacitivo 1	24V Salida GND	Vcc DI1 GND
Unidad paletizadora	Sensor capacitivo 2	24V Salida GND	Vcc DI4 GND

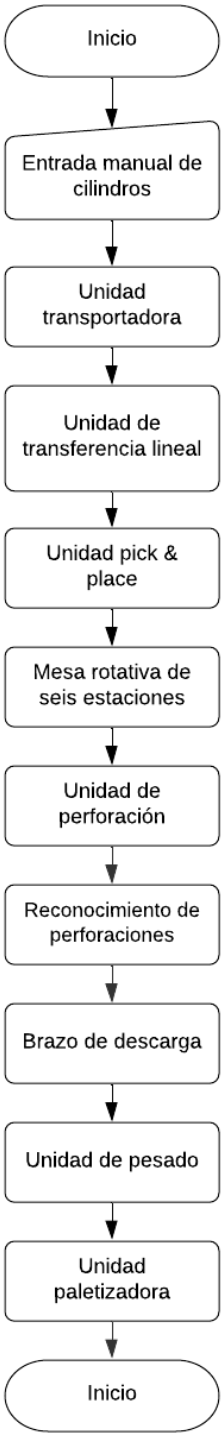


Figura 3.6: Diagrama de flujo del funcionamiento de la FMC.

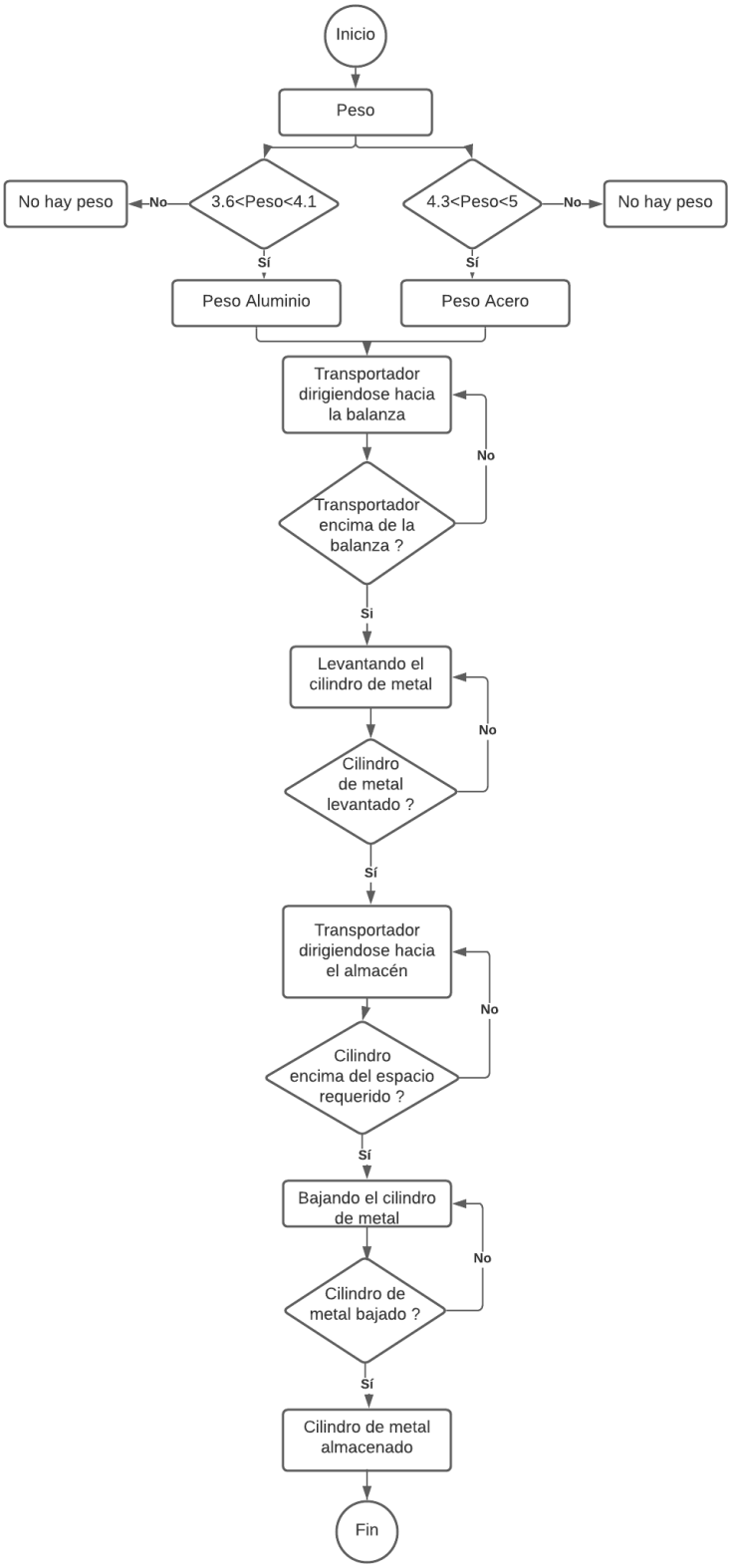


Figura 3.7: Diagrama de flujo de la estación de pesado y almacenamiento.

La Figura 3.8 ilustra el diagrama de flujo para la automatización de la estación de verificación, manipulación y transporte. El diagrama de flujo explica secuencialmente la cadena de decisiones de cada tarea realizada desde el ingreso del cilindro en la estación de trabajo hasta que finaliza en la unidad de pesado.

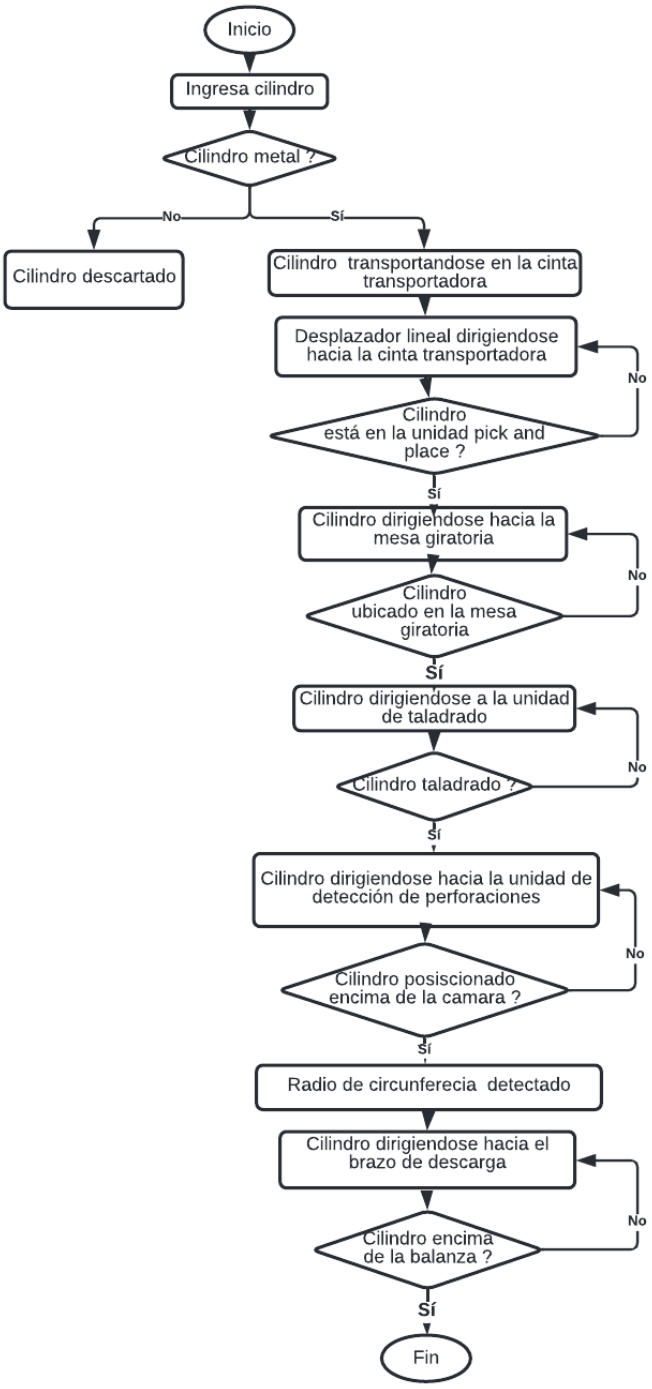


Figura 3.8: Diagrama de flujo de la estación de verificación, manipulación y transporte.

Secuencia de trabajo: Al pulsar inicio en el panel de mando remoto o local, se activa la secuencia de trabajo, que funciona de forma ininterrumpida, detectando si hay pieza en la entrada, y seleccionando de acuerdo al material de las piezas. Mientras el contador de piezas metálicas en proceso sea diferente de 0, se activan los programas de cada paso del sistema de proceso y la mesa realiza avances para transferir las piezas en procesos correspondientes. El programa se ramifica para ejecutar varias tareas en paralelo de acuerdo a los procesos.

A continuación, se describe el proceso para una sola pieza metálica:

1. Se verifica la presencia de un cilindro metálico en la cinta transportadora (entrada) mediante el sensor capacitivo. La cinta transportadora esta activada de manera interrumpida. El sensor fotoeléctrico detecta la presencia de cilindros de plástico. Este sensor permite realizar el conteo de estos cilindros. El sensor capacitivo da comienzo al siguiente proceso mediante la presencia de un cilindro metálico y realiza el conteo de cilindros metálicos.

2. El módulo de desplazamiento lineal avanza hacia la cinta transportadora para recibir el cilindro metálico, orden recibida previamente por el sensor capacitivo. El módulo de deslizamiento lineal regresa a su posición inicial después de un tiempo y envía una señal al siguiente proceso, llevando consigo al cilindro.

3. Al recibir la señal, el módulo de *pick & place* desciende y el *gripper* angular recoge el cilindro, luego asciende y se desplaza horizontalmente para descender y desligar el cilindro. El cilindro debe ubicarse en la ranura correspondiente de la mesa giratoria. El módulo de *pick & place* regresa a su posición inicial enviando una señal al siguiente proceso.

4. La mesa giratoria recibe la señal del proceso anterior y avanza un paso.

5. Se fija la pieza mediante un cilindro de carrera corta, el taladro desciende mediante cilindros con vástagos paralelos y se enciende. Este proceso es realizado por el módulo de taladrado por un corto tiempo. Al terminar la operación envía la señal al siguiente proceso.

6. La mesa giratoria recibe la señal del proceso anterior y avanza dos pasos enviando la señal al siguiente proceso.

7. El brazo de giro neumático sujeta el cilindro mediante el agarrador angular, asciende, gira, desciende y desliga el cilindro en la unidad de pesado. El brazo neumático regresa a su posición inicial enviando la señal al siguiente proceso.

8. La unidad de pesado obtiene el valor de peso del cilindro.

9. Los motores de pasos se encargan de ubicar al cilindro succionador encima del cilindro de metal luego de haber detectado su peso. El cilindro succionador desciende y succiona el cilindro de metal, luego asciende y mediante los motores se desplaza en movimientos XY para ubicar al cilindro, las posiciones XY están programadas. Finalmente, el cilindro desciende y el succionador desliga el cilindro en la posición final.

10. Vuelve al proceso inicial.

3.3.3. Niveles de jerarquía del sistema

La Figura 3.10 presenta la arquitectura de los sistemas de instrumentación y comunicación entre el nivel de campo, nivel de proceso y nivel de supervisión respecto a la pirámide CIM.

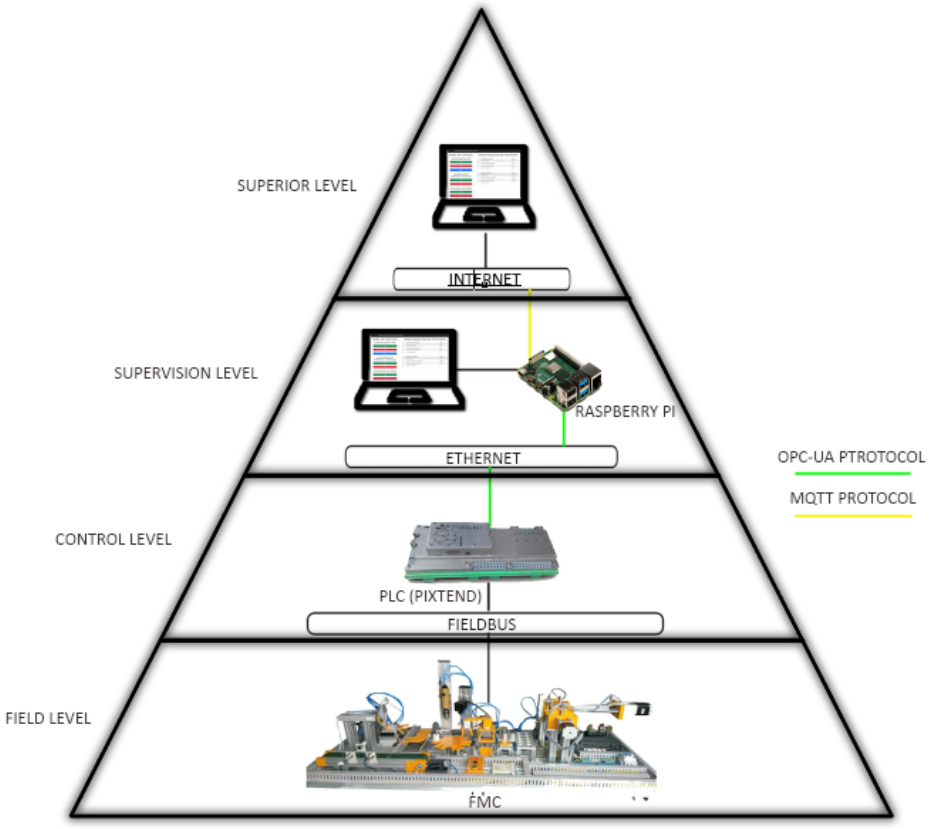


Figura 3.9: Arquitectura de la red de comunicación planteada.

Nivel de campo A nivel de campo la estación de trabajo realiza una secuencia de tareas programadas en el PLC PiXtend, mientras que ejecuta los resultados obtenidos de la interpretación de procesos de control y la medición de sensores.

Nivel de control El programa de automatización de procesos y servidor OPC-UA son ejecutados en el PLC PiXtend. Este programa es implementado como un archivo ejecutable .py utilizando principalmente las librerías PPL Python y Python OPC-UA.

El código de automatización es implementado a partir de los diagramas de flujo descritos en la Sección 3.3.2. Este código itera en ciclos de trabajo de 100ms. Esta restricción de tiempo se debe a la comunicación realizada entre el *Raspberry* Pi y el microcontrolador Atmega32A que controla los diferentes puertos de entrada/salida del PiXtend. La comunicación es establecida a través del bus Serial Peripheral Interface (SPI) (interfaz de periféricos

en serie).

El servidor OPC-UA está implementado fuera del bucle de iteración, En esta parte del programa se definen las variables de monitorización y control de la FMC. Las variables de control modifican el funcionamiento de la celda y las variables de monitorización fluctúan de acuerdo a la lectura de sensores y tareas realizadas en la FMC.

La Tabla 3.4 presenta las variables de monitorización, mientras que la Tabla 3.5 expone las variables de control de la FMC implementadas en el servidor OPC-UA.

Tabla 3.4: Variables de monitorización OPC-UA.

Descripción de variable de proceso	Nodo	Variable en el servidor OPC-UA
Lista con la información de pesos obtenidos en la etapa de clasificación	ns=2; i=3	PesosObtenidos
Mensajes del proceso de clasificación	ns=2; i=4	Notificaciones
Cilindros de metal	ns=2; i=6	CPua
Cilindros de plástico	ns=2; i=7	CMua
Mensajes del proceso de manipulación	ns=2; i=8	MVua

Tabla 3.5: Variables de control OPC-UA.

Descripción de variable de proceso	Nodo	Variable en el servidor OPC-UA
Iniciar/Detener estación de trabajo	ns=2; i=2	InicioPrograma
Iniciar/Detener manipulación de piezas	ns=2; i=9	Estado
Activar/Desactivar taladrado	ns=2; i=10	DesactivarTaladro
Reinicio de la estación de trabajo	ns=2; i=11	PosicionInicial

Nivel de supervisión El nivel de supervisión está encargado de monitorear las variables de procesos y controlar su ejecución en la FMC. Para realizar esta supervisión se implementan un servidor web local y un servidor web remoto. Estos servidores difieren únicamente en su *backend* de comunicación. El servidor web es creado como un proyecto Django.

La supervisión local es ejecutada en la *Raspberry* Pi. El *backend* de este servidor posee un cliente OPC-UA que intercepta y envía datos hacia servidor OPC-UA.

3.4. Comunicaciones

La *Raspberry* Pi también funciona como puerta de enlace entre la comunicación local y la comunicación a través de internet. Este proceso es realizado utilizando el protocolo de comunicación MQTT.

El programa de conversión de protocolos está implementado como un archivo .py utilizando las librerías Python OPC-UA y Paho MQTT. Este programa contiene un cliente de comunicación OPC-UA. Un publicador

de mensajes MQTT y cuatro subscriptores MQTT. La tabla 3.6 presenta los tópicos establecidos para realizar la comunicación con el servidor web remoto.

Tabla 3.6: Variables de control y monitorización MQTT.

Descripción de variable de proceso	Topic
Iniciar/Detener estación de trabajo	Activar_Celda
Iniciar/Detener manipulación de piezas.	Estado
ActivarDesactivar taladrado	Detener_taladrado
Reinicio de la estación de trabajo	Posicion_Inicial
Lista de mensajes de monitorización	Lista_mensajes

Un servidor web permite supervisar la FMC a través de internet. El backend del servidor web remoto contiene el broker MQTT. Este servidor es montado en una instancia de AWS. La instancia de AWS utiliza el SO Ubuntu 18.04. El servidor web del proyecto es implementado en una máquina virtual a través de un servidor Apache.

Existen múltiples dependencias que operan en el servidor web remoto como las librerías Django, channels, Paho MQTT, también se instala el servicio de Redis-server que funciona como un almacén de clave de datos.

La librería channels otorga herramientas para manejo de WebSockets en Django. Este protocolo de comunicación es utilizado para enviar y recibir información desde la página de visualización hacia el servidor de AWS en la nube. La Tabla 3.7 muestra los puertos y dirección de red utilizados para la comunicación local y remota en este trabajo.

Tabla 3.7: Protocolos y servicios utilizados para la comunicación entre los dispositivos empleados en este proyecto.

Protocolo o Servicio	Puerto	Dirección de host
OPC-UA	48400	169.254.121.159
MQTT	1883	EMQX
WebSockets	8000	18.221.158.203
Redis-server	6379	dirección de la máquina virtual AWS

3.5. Visualización

La página de visualización en la web correspondiente al Dashboard del proyecto es implementado utilizando herramientas para desarrollo como Django, Javascript, HTML y CSS.
La página de visualización está diseñada en 2 secciones descritas a continuación:

1. **Panel de control**: En este panel se encuentran los botones de mando para ejecutar los procesos de trabajo descritos en la Sección 3.3.2.

2. **Panel de monitorización**: En este panel se muestran los valores obtenidos de la ejecución de los procesos de control.

La Figura 3.10 ilustra el panel de control y la monitorización de procesos. El panel de control permite ejecutar los procesos de la FMC (botón start), detener los procesos de la FMC (botón stop) y reestablecer los procesos (botón reset). Además, permite ejecutar y detener los procesos verificación y colocación, que está conformado por la unidad transportadora, la unidad de transferencia lineal, unidad pick & place, mesa rotativa de seis estaciones, unidad de perforación, reconocimiento de perforaciones y brazo de descarga.

También permite activar y desactivar el módulo de taladrado que está conformado por la unidad de perforación, que simula el taladrado de cilindros. Si el taladro se encuentra desactivado, los procesos restantes se ejecutan con normalidad a excepción de la unidad de perforación.

La sección de monitorización de procesos muestra el avance del sistema en tiempo real, distribuido en detalles de la verificación y detalles de clasificación. Los datos correspondientes a detalles de verificación son:

- **Cilindros ingresados**: Muestra el número de cilindros ingresados en la unidad transportadora sin importar el material.
- **Cilindros de plástico descartados**: Muestra el número de cilindros de plástico ingresados en la unidad transportadora.
- **Cilindros de metal detectados**: Muestra el número de cilindros de metal ingresados en la unidad transportadora.
- **Acciones realizadas**: Muestra la acción que se realiza o la última que realizó.

Los datos correspondientes a detalles de clasificación son:

- **Peso detectado**: Muestra el peso de los cilindros en gramos.
- **Cilindros de aluminio almacenados**: Muestra el número de cilindros de aluminio almacenados.
- **Cilindros de acero almacenados**: Muestra el número de cilindros de acero almacenados.
- **Acciones realizadas**: Muestra la acción que se realiza o la última que realizó.

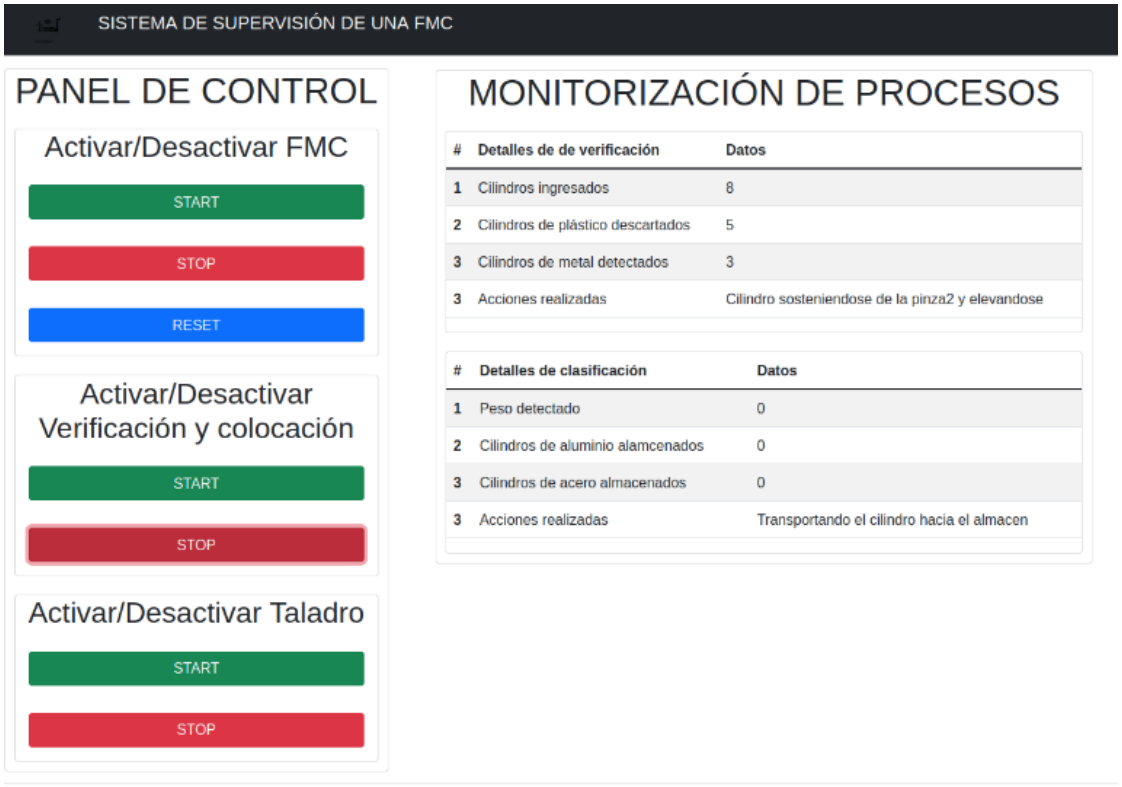


Figura 3.10: Página de visualización en la web correspondiente al Dashboard del trabajo de titulación.

3.6. Almacenamiento

EL almacenamiento de datos de forma local es desarrollado con el sistema de gestión de base de datos de dominio público SQLite3. Este *software* está enlazado con la biblioteca Django siendo parte integral del mismo programa.

El modelo de base de datos está constituido de tres tablas que contienen los registros de las variables descritas en la Tabla 3.6 El envío de datos es implementado en el backend del servidor local. La memoria de la *Raspberry* Pi funciona como *hosting* de almacenamiento para los ficheros de información de la base datos local.

El almacenamiento remoto funciona de la misma manera que el almacenamiento local. Sin embargo, el almacenamiento es realizado en la máquina virtual provista del servicio de AWS. La administración y gestión de la base de datos almacenada puede realizarse a través del panel de administración de Django. Este panel está configurado por defecto y es originado cuando se crea el proyecto Django.

Capítulo 4

Resultados

4.1. Descripción narrativa

En el laboratorio de máquinas eléctricas de la Universidad de Cuenca el día 26 de enero de 2022 es grabado un vídeo Automatización de la FMC[1] de la ejecución completa de los procesos automatizados en la FMC.

El vídeo inicia presentando la posición inicial de los componentes que conforman la FMC. Esta parte del vídeo muestra a la mesa rotativa ubicada en una posición de acople a la unidad *pick & place*. En la unidad paletizadora los motores a pasos están alejados de la unidad de pesado. En esta posición los componentes de la *FMC* están preparados para realizar el ciclo de trabajo.

Un cilindro de metal es ubicado en la banda transportadora cuyo componente es la entrada de piezas en la FMC. Este cilindro es monitorizado por los sensores mientras es transportado.

A continuación, el primer par de sensores detectan el cilindro de metal, por lo tanto, es activada la unidad de transferencia lineal desplazando el cilindro desde la cinta transportadora hacia la unidad *pick & place*. Luego el *gripper* angular despliega sus pinzas desplazándose hacia abajo para sujetar el cilindro. Posteriormente el *griper* cierra sus pinzas para sujetar el cilindro elevándose para desplazarse de forma horizontal. A continuación, el *gripper* desciende ubicando el cilindro en la mesa rotativa.

La mesa rotativa gira una posición ubicando el cilindro debajo del módulo de simulación de taladrado. Inmediatamente, el cilindro es taladrado. A continuación, la mesa giratoria gira una posición hacia el módulo de inspección y medición de radio de perforación. La cámara realiza el reconocimiento del círculo de simulación de taladrado. La pantalla de la *Raspberry* Pi expone a través de la grabación la circunferencia y coordenadas de perforación emulada.

Posteriormente, la mesa gira ubicando el cilindro debajo del brazo de descarga. El brazo de descarga des-

[1] https://drive.google.com/drive/folders/1ZG_zRxQBIj7VP6RSjUYZ-Q_cpashMhYA

ciende, a su vez el *gripper* sujeta el cilindro. Inmediatamente asciende y rota colocando el cilindro en la unidad de pesado. Esta unidad obtiene el peso enviando una señal de ejecución de la unidad paletizadora. La unidad paletizadora es desplazada en XY hacia la unidad de pesado, luego desciende y un succionador sujeta el cilindro elevándolo. Finalmente, la unidad paletizadora desplaza el cilindro a la posición de almacenamiento. En esta posición, desciende y suelta el cilindro.

El valor de presión estable de la celda de manufactura se obtuvo mediante varias pruebas realizadas. Estas pruebas demostraron una presión de agarre máxima con el *gripper* angular. Para las pruebas son utilizados varios cilindros con diferentes pesos, el *gripper* angular tiene un agarre máximo con una presión de aire de 6 bares. El ordenador *Raspberry* Pi conjuntamente con su pantalla se colocó cerca de la cámara agregada para el reconocimiento de perforaciones en los cilindros.

4.2. Pruebas de detección de perforaciones en los cilindros

La detección de la circunferencia de perforación es desarrollada utilizando la librería *Open* CV. El programa inicia tomando capturas del vídeo. Luego mediante iteraciones repetidas las imágenes capturadas son procesadas. El procesamiento de la imagen inicia con un ajuste del tamaño de imagen capturado. Posteriormente la imagen es transformada a escala de grises. A continuación son aplicadas las funciones *GaussianBlur*() y *medianBlur*() para reducir el ruido y suavizar la imagen.

La función *HoughCircles* es utilizada para la detección de círculos en la imagen procesada. Esta detección es lograda utilizando el método del gradiente de *Hough*. Una correcta detección del círculo se consigue ajustado un valor máximo y mínimo de circunferencia. También se ajusta el parámetro de umbral con el objetivo de eliminar círculos no deseados.

Las Figuras 4.1 y 4.2 presentan las capturas de los cilindros detectados por la cámara. Se puede apreciar el dibujo del círculo y sus coordenadas. La parte izquierda de la imagen muestra un arreglo de valores. Estos valores son las coordenadas y el radio detectado de la imagen capturada en un instante determinado.

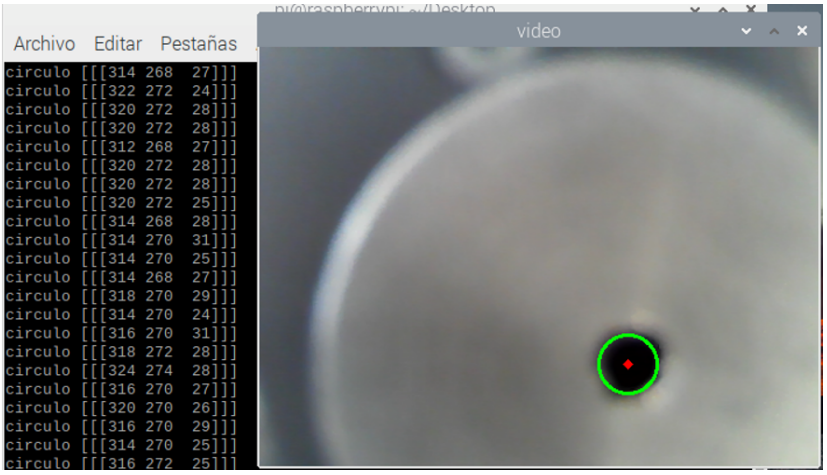


Figura 4.1: Perforación de un cilindro detectado por la cámara con un valor de radio aproximado de 27 unidades.

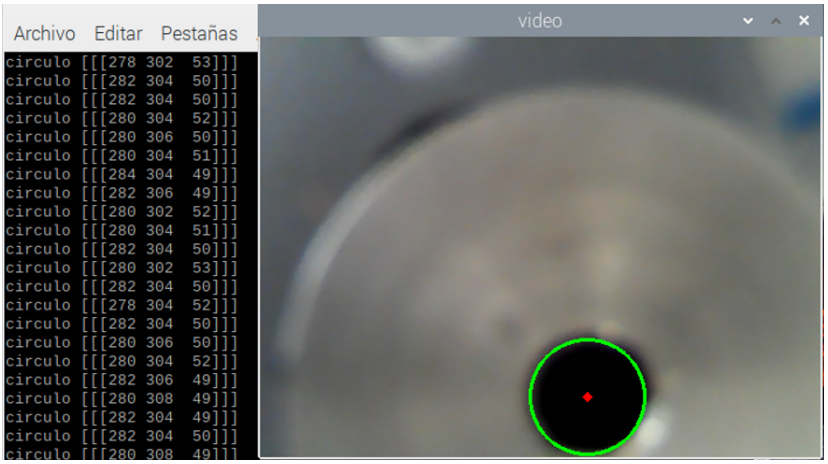


Figura 4.2: Perforación de un cilindro detectado por la cámara con un valor de radio aproximado de 50 unidades.

4.3. Pruebas de comunicación

4.3.1. Tráfico en la *Raspberry* Pi

El programa de conversión de protocolos envía y recibe mensajes OPC-UA y MQTT. Este programa genera paquetes de tráfico transmitidos a través de la interfaz de red ethernet e inalámbrica. Este tráfico es capturado con el *software* Wireshark.

La comunicación del servidor OPC-UA es establecida con la dirección IP 192.254.121.159 y puerto 48400, mientras que la comunicación con el protocolo MQTT opera con un *Broker* EMQX en el puerto 1883.

El protocolo OPC-UA arroja paquetes del tipo *BrowseRequest* y *BrowseResponse* (Figura 4.3). Estos paquetes examinan las referencias de uno o más nodos en el espacio de direcciones del servidor. *BrowseResponse* es originado desde el servidor hacia el cliente, mientras que *BrowseRequest* es originado desde el cliente hacia el servidor.

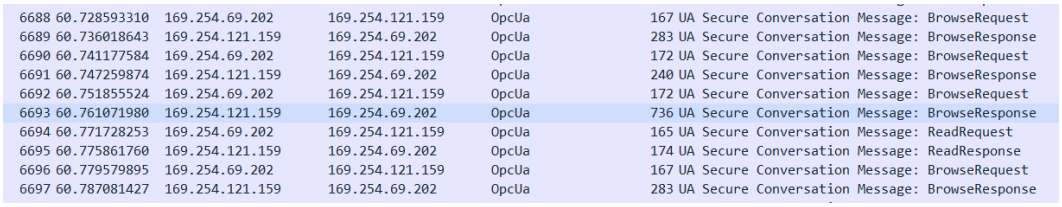


Figura 4.3: Paquetes OPC-UA de tipo BrowseRequest y BrowseResponse.

La Figura 4.4 muestra el contenido de un paquete OPC-UA de tipo *BrowseResponse*. En la imagen se visualizan las capas de operaciones de transferencia de datos. El tamaño total del paquete enviado es de 736 bytes. En la última capa aparecen los campos del protocolo OPC-UA. Entre estos campos aparecen los identificadores de seguridad, la longitud del mensaje y el objeto codificado.

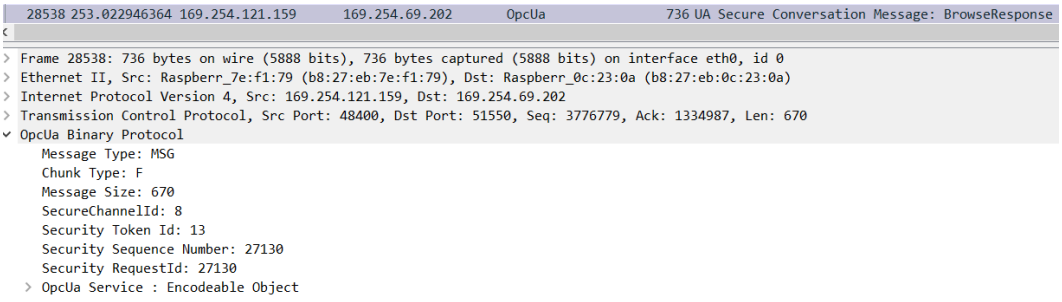


Figura 4.4: Captura de un paquete OPC-UA en Wireshark.

La Figura 4.5 muestra el contenido del campo *OpcUa Service*. Este campo contiene la marca de tiempo que se realizó durante la captura de tráfico. El campo *Results* contiene un arreglo que contiene información de los nodos definidos para el control y monitorización de variables en el programa de automatización. El campo *NodeId* contiene un identificador numérico y de índice de nodo con un valor igual a 2 para ambos identificadores. Por otro lado, el campo *BrowseName* contiene el nombre "InicioPrograma" y es asignado según el nombre de la variable del programa de automatización.

La Figura 4.6 expone los campos de identificación de un nodo. Este campo especifica los identificadores, el nombre del nodo y el tipo de nodo.

La Figura 4.7 presenta los paquetes de mensajes que usan el protocolo MQTT. Entre estos paquetes están los mensajes de subscripción y publicación. Los paquetes de publicación y subscripción MQTT son enviados a través de la dirección 192.168.43.221 perteneciente a la *Raspberry* Pi. El *Broker* EMQX con dirección 192.168.43.221 administra los mensajes de subscripción y publicación en el servidor instalado en AWS. La imagen posee mensajes de publicación y subscripción desde el *Broker* hacia la *Raspberry* Pi.

En la imagen se verifica que la *Rapsberry* Pi recibe mensajes del servidor web debido a que en el paquete 1980 se publica un mensaje de control con *topic* "pythonmqttActiva_Celda", mientras que en el paquete 2008 se obtiene el mensaje de control a este tópico por parte del subscriptor en la *Raspberry* Pi.

```
BrowseResponse
  ResponseHeader: ResponseHeader
      Timestamp: Jan 26, 2022 14:31:47.401488000 Hora est. Pacífico, Sudamérica
      RequestHandle: 27130
      ServiceResult: 0x00000000 [Good]
    > ServiceDiagnostics: DiagnosticInfo
    > StringTable: Array of String
    > AdditionalHeader: ExtensionObject
  Results: Array of BrowseResult
      ArraySize: 1
    [0]: BrowseResult
        StatusCode: 0x00000000 [Good]
        ContinuationPoint: <MISSING>[OpcUa Null ByteString]
      References: Array of ReferenceDescription
          ArraySize: 10
        [0]: ReferenceDescription
          > ReferenceTypeId: NodeId
            IsForward: True
          NodeId: ExpandedNodeId
            > EncodingMask: 0x02, EncodingMask: Numeric of arbitrary length
              Namespace Index: 2
              Identifier Numeric: 2
          BrowseName: QualifiedName
              Id: 2
              Name: InicioPrograma
          > DisplayName: LocalizedText
            NodeClass: Variable (0x00000002)
          > TypeDefinition: ExpandedNodeId
```

Figura 4.5: Contenido del campo OpcUA Service.

```
[1]: ReferenceDescription
  > ReferenceTypeId: NodeId
    IsForward: True
  NodeId: ExpandedNodeId
    > EncodingMask: 0x02, EncodingMask: Numeric of arbitrary length
      Namespace Index: 2
      Identifier Numeric: 3
  BrowseName: QualifiedName
      Id: 2
      Name: PesosObtenidos
  > DisplayName: LocalizedText
    NodeClass: Variable (0x00000002)
  > TypeDefinition: ExpandedNodeId
```

Figura 4.6: Contenido de un objeto OPC-UA.

La Figura 4.8 muestra el contenido del paquete MQTT de publicación. Este paquete posee todas las capas de transmisión de información. La primera capa especifica que la información es transmitida por medio de la interfaz wlan0. La capa de enlace presenta las direcciones físicas de la fuente y el destino. Por otro lado, la capa de red expone las direcciones IP del origen y destino. La capa de transporte muestra los puertos de origen y destino.

El contenido de la publicación de mensaje MQTT muestra diferentes campos como el tipo de mensaje, el quality of service (QoS), la longitud de mensaje en bytes, la longitud del *topic* en bytes, el tópico y el mensaje

```
1978 248.740905315 192.168.43.221   52.32.182.17      MQTT   213 Publish Message [python/mqtt/mensajes]
1980 249.011310515 52.32.233.61     192.168.43.221    MQTT    96 Publish Message [python/mqtt/Activar_Celda]
1982 249.012521456 192.168.43.221   52.32.233.61      MQTT    68 Disconnect Req
1984 249.143045286 192.168.43.221   52.32.182.17      MQTT   213 Publish Message [python/mqtt/mensajes]
1988 249.369404127 192.168.43.221   52.32.182.17      MQTT    68 Ping Request
1990 249.628198543 192.168.43.221   52.32.182.17      MQTT   212 Publish Message [python/mqtt/mensajes]
1991 249.628318278 52.32.182.17     192.168.43.221    MQTT    68 Ping Response
1994 249.969934844 192.168.43.221   52.32.182.17      MQTT   212 Publish Message [python/mqtt/mensajes]
2003 250.222788344 192.168.43.221   52.32.182.17      MQTT    80 Connect Command
2004 250.395378484 192.168.43.221   52.32.182.17      MQTT   212 Publish Message [python/mqtt/mensajes]
2006 250.433451524 52.32.182.17     192.168.43.221    MQTT    70 Connect Ack
2008 250.434869018 192.168.43.221   52.32.182.17      MQTT    98 Subscribe Request (id=1) [python/mqtt/Activar_Celda]
2011 250.624605650 52.32.182.17     192.168.43.221    MQTT    71 Subscribe Ack (id=1)
2013 250.839511634 192.168.43.221   52.32.182.17      MQTT   212 Publish Message [python/mqtt/mensajes]
```

Figura 4.7: Captura de paquetes de mensajes MQTT.

en formato hexadecimal.

```
1980 249.011310515 52.32.233.61     192.168.43.221    MQTT    96 Publish Message [python/mqtt/Activar_Celda]

Frame 1980: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface wlan0, id 0
Ethernet II, Src: d6:ae:05:6a:75:11 (d6:ae:05:6a:75:11), Dst: Raspberr_59:76:5f (b8:27:eb:59:76:5f)
Internet Protocol Version 4, Src: 52.32.233.61, Dst: 192.168.43.221
Transmission Control Protocol, Src Port: 1883, Dst Port: 54411, Seq: 9, Ack: 9, Len: 30
MQ Telemetry Transport Protocol, Publish Message
> [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
v Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
      0011 .... = Message Type: Publish Message (3)
      .... 0... = DUP Flag: Not set
      .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
      .... ...0 = Retain: Not set
   Msg Len: 28
   Topic Length: 25
   Topic: python/mqtt/Activar_Celda
   Message: 30
```

Figura 4.8: Tráfico MQTT de Publicación.

La Figura 4.9 presenta el contenido del paquete de subscripción desde la dirección 192.168.43.221 hacia la dirección 52.32.182.17. El campo *Header Flags* especifica que el mensaje es de tipo subscripción. La longitud del *topic* es de 25 bytes.

```
2008 250.434869018 192.168.43.221   52.32.182.17      MQTT    98 Subscribe Request (id=1) [python/mqtt/Activar_Celda]

Frame 2008: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface wlan0, id 0
Ethernet II, Src: Raspberr_59:76:5f (b8:27:eb:59:76:5f), Dst: d6:ae:05:6a:75:11 (d6:ae:05:6a:75:11)
Internet Protocol Version 4, Src: 192.168.43.221, Dst: 52.32.182.17
Transmission Control Protocol, Src Port: 48441, Dst Port: 1883, Seq: 15, Ack: 5, Len: 32
MQ Telemetry Transport Protocol, Subscribe Request
v Header Flags: 0x82, Message Type: Subscribe Request
      1000 .... = Message Type: Subscribe Request (8)
      .... 0010 = Reserved: 2
   Msg Len: 30
   Message Identifier: 1
   Topic Length: 25
   Topic: python/mqtt/Activar_Celda
   Requested QoS: At most once delivery (Fire and Forget) (0)
```

Figura 4.9: Tráfico MQTT de Subscripción.

4.3.2. Tráfico en el dispositivo de supervisión remota

Este análisis de tráfico es realizado conectándose a la página a través de internet. La dirección IP del dispositivo conectado de manera inalámbrica es: 192.168.43.168. Por otro lado, se accede a la página con la dirección http//18.221.158.203:8000.

La Figura 4.10 muestra el contenido de un paquete HTTP obtenido al cargar la página web a través de internet. Esta Figura indica que desde la fuente 192.168.43.168 al destinatario 18.221.158.203 se obtiene un código de respuesta satisfactorio de 200 OK. Este resultado es producido al realizar una petición GET, de modo que los recursos se transfieren a la página. La imagen muestra que en la capa de transporte se encuentran los puertos origen 8000 y destino 42290. La última capa posee el contenido del código HTML, la lógica y los estilos implementados.

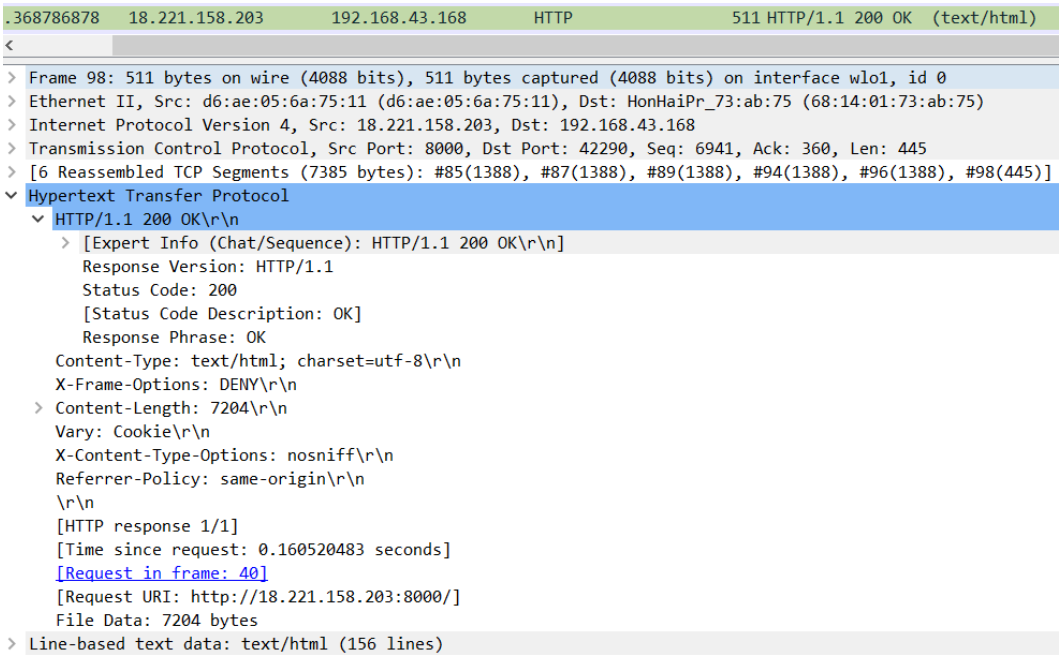


Figura 4.10: Contenido del protocolo HTTP al cargar la página web.

La Figura 4.11 presenta una petición de negociación WebSocket desde el *host* 192.168.43.168 hacia la dirección 18.221.158.203. Esta petición está vinculada hacia la ruta wsdatos_recibidosHTTP1.1. Esta ruta pertenece a la configuración de conexión con el WebSocket que proporciona los datos de monitorización de la FMC.

Los campos SecWebSocketKey1 y SecWebSocketKey2 son tokens aleatorios que el servidor utiliza para construir un token de 16 bytes al final de la negociación para confirmar que ha leído correctamente la petición de negociación del cliente.

La Figura 4.12 presenta la respuesta *Hankshake* del servidor hacia el cliente. La cabecera Sec-WebSocket-Accept es derivada a partir de los tokens enviados en la petición de negociación WebSocket. Cuando el servidor envía las cabeceras se completa el *Handshake* y existe un intercambio de datos.

El campo WebSocket presenta información del intercambio de *Dataframes*. El bit FIN: *True* indica que el mensaje es entregado. El campo .ºpcode"define los datos de carga útil, en este caso se envía texto por lo tanto el campo es 0x1. Por otro lado, el campo *Payload* contiene la longitud del mensaje en bytes.

La Figura 4.13 presenta el establecimiento de una conexión WebSocket para cada una de las variables de

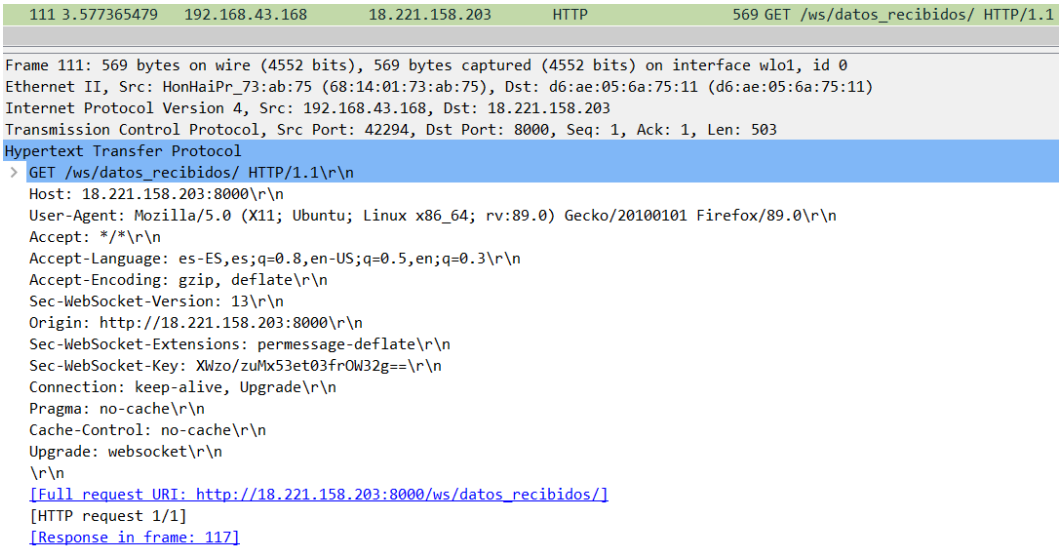


Figura 4.11: Petición de negociación WebSocket entre la aplicación y el servidor.

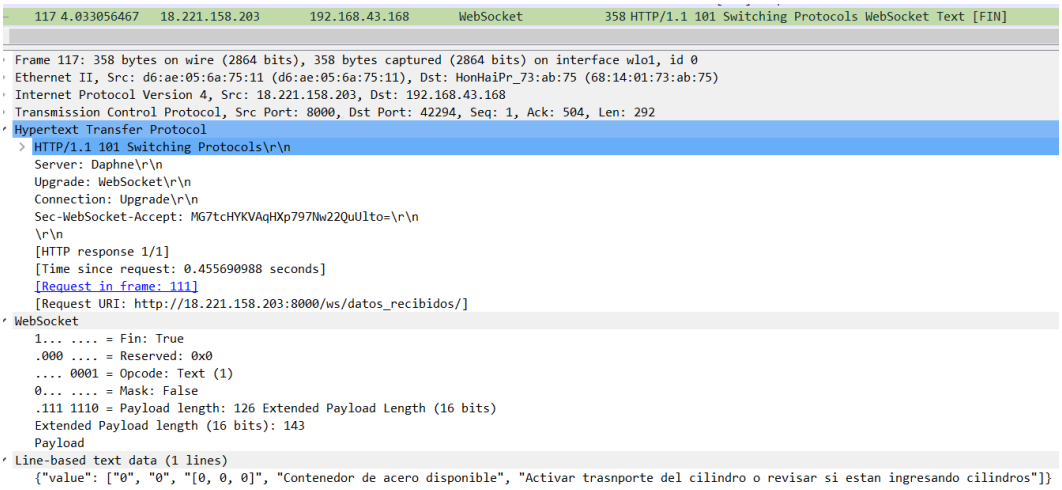


Figura 4.12: Intercambio de protocolo de HTTP a WebSockets.

control. Las peticiones de negociación están en los paquetes HTTP de tipo GET, mientras que la respuesta de negociación está en el cambio de protocolos.

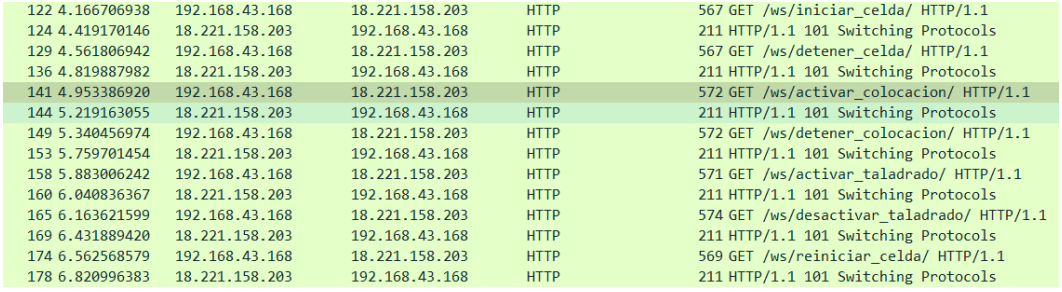


Figura 4.13: Establecimiento de petición de negociación WebSocket y respuesta de negociación WebSocket.

Después de producirse el *Handshake*, el cliente o el servidor envían un Ping a la otra parte. Figura 4.14. Cuando se recibe el Ping, el destinatario envía un Pong instantáneamente. Esta configuración está programada por defecto. Por lo tanto, el tiempo de envío de estos paquetes está controlado con un tiempo específico para asegurar la conexión.

2933	270.878133388	18.221.158.203	192.168.43.168	WebSocket	72 WebSocket Ping [FIN]
2934	270.878210030	192.168.43.168	18.221.158.203	TCP	66 40246 → 8000 [ACK] Seq=624 Ack=47206 Win=
2935	270.878373452	18.221.158.203	192.168.43.168	WebSocket	72 WebSocket Ping [FIN]
2936	270.878392767	18.221.158.203	192.168.43.168	WebSocket	72 WebSocket Ping [FIN]
2937	270.878410726	192.168.43.168	18.221.158.203	TCP	66 40252 → 8000 [ACK] Seq=655 Ack=254 Win=64
2938	270.878563199	192.168.43.168	18.221.158.203	WebSocket	76 WebSocket Pong [FIN] [MASKED]

Figura 4.14: Paquetes WebSockets de tipo Ping y Pong.

Las Figuras 4.15, 4.16 y 4.17 muestran las gráficas RTT durante el envío de mensajes desde un dispositivo hacia otro. La primera gráfica muestra el RTT entre la *Raspberry* Pi con dirección 169.254.69.202 y el servidor PLC PiXtend con dirección 169.254.121.59. La segunda gráfica muestra el envío de mensajes desde la *Raspberry* Pi hacia el *Broker* MQTT. La tercera gráfica muestra el envío de mensajes desde el servidor web hacia la página del cliente.

Estas gráficas permiten conocer el tiempo trascurrido entre el envío de un paquete y la recepción del ACK por parte del servidor. Las gráficas de las Figuras 4.15 y 4.17 evidencian un RTT mínimo menor a 30 milisegundos. Sin embargo, la gráfica de la Figura 4.16 de la comunicación entre la *Raspberry* Pi y el *Broker* muestra un RTT promedio de 250 ms.

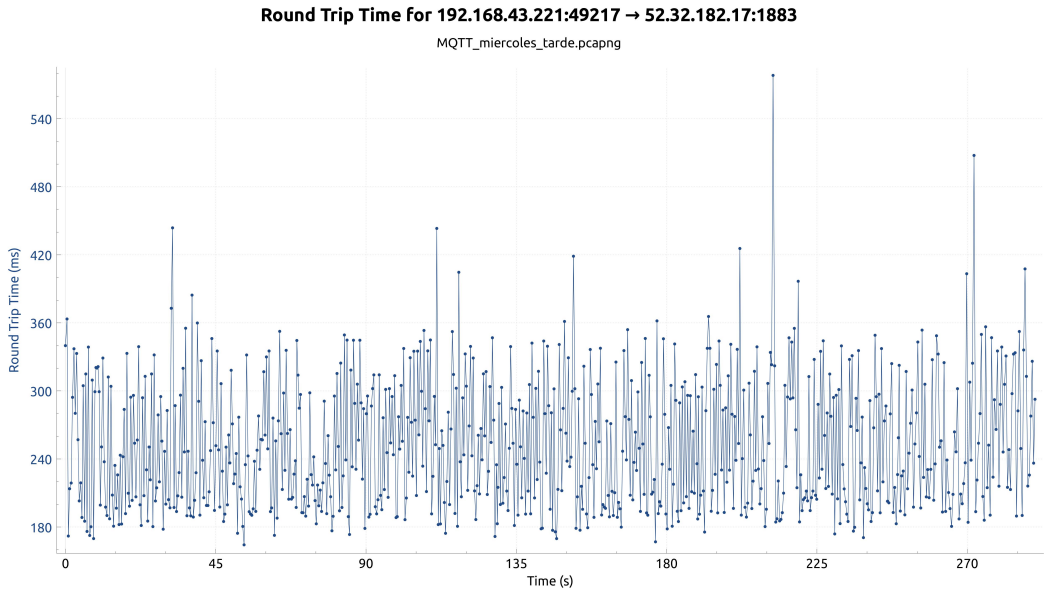


Figura 4.15: Gráfica RTT entre el cliente en la *Raspberry* Pi y el servidor en el PiXtend.

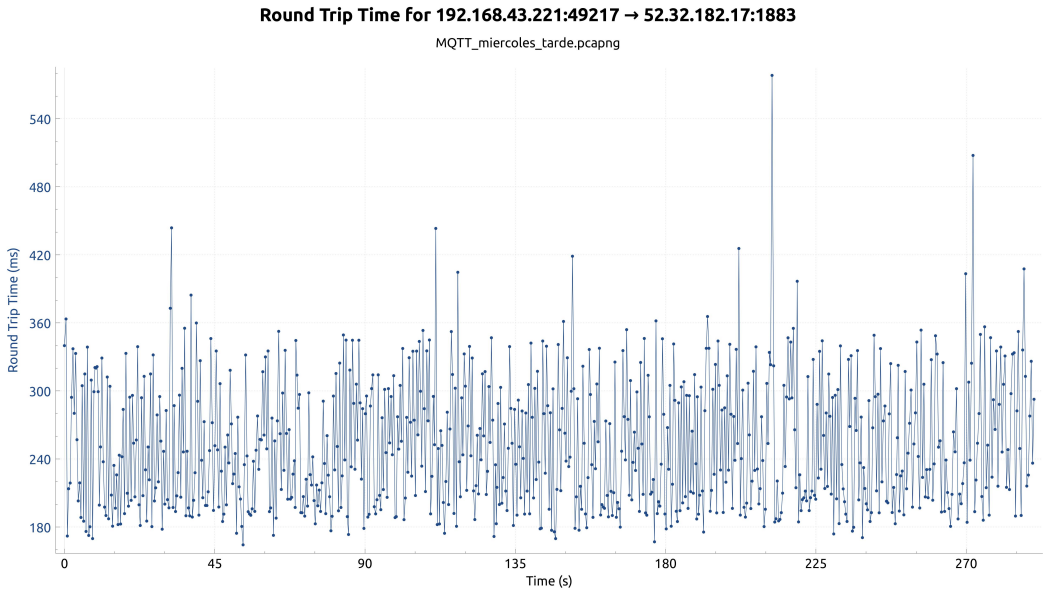


Figura 4.16: Gráfica RTT entre la Raspberry Pi y el broker.

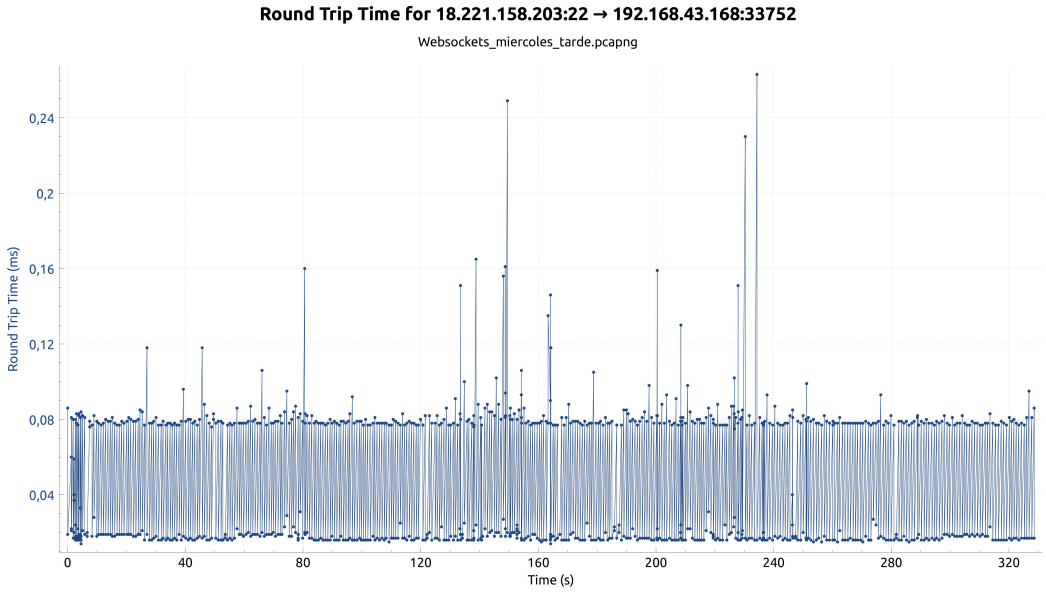


Figura 4.17: Gráfica RTT entre el servidor AWS y el dispositivo del cliente.

4.4. Comparación cualitativa

Entre los exponentes con más auge en el desarrollo de soluciones IIoT están: Siemens, Schneider Electric y Mitsubishi Electric. Cada uno ofrece soluciones para IIoT. A continuación, se realiza una comparación cualitativa entre los tres exponentes y el sistema desarrollado en el trabajo de titulación.
Siemens ofrece las siguientes soluciones para IIoT:

- **Indutrial Edge**: *Edge* permite la computación en el borde o en la nube, simplificando la TI de la planta. Además, permite procesar los datos en tiempo real de manera local o remota.

- **MindSphere**: Es un sistema operativo basado en la nube para dispositivos del sector industrial y de fabricación, potenciando las soluciones IIoT desde el borde hasta la nube. Este sistema recopila datos de la planta para análisis en la nube y presenta resultados analíticos en tiempo real en una ubicación centralizada.

- **Mendix**: Es una plataforma de aplicaciones de alta productividad que le permite crear y mejorar continuamente aplicaciones móviles y web a escala. Esta plataforma está diseñada para acelerar la entrega de aplicaciones empresariales en todo el ciclo de vida de desarrollo de aplicaciones, desde la ideación hasta la implementación y las operaciones. Mendix permite un entorno colaborativo para desarrolladores, permitiendo participar de forma activa en el desarrollo de la solución.

Schneider Electric ofrece las siguientes soluciones para IIoT:

- ***Edge Computing***: Está enfocado en IIoT con adaptación a una nube híbrida, permitiendo infraestructura local como de nube pública (Microsoft Azure, Amazon Web Services, Google Cloud, etc.).

- ***EcoStruxure Plant*** : Permite la integración de tecnologías de la información (TI con las tecnologías operacionales OT en múltiples sistemas y sitios. Esta plataforma brinda a las empresas industriales información en tiempo real sobre los indicadores clave de rendimiento Key Performance Indicator (KPI) necesarios para la optimización comercial.

Mitsubishi Electric ofrece las siguientes soluciones para IIoT:

- ***Mitsubishi Adroit Process Suite 4*** :Mitsubishi Adroit Process Suite (MAPS) 4, una solución Supervisory Control and Data Acquisition (SCADA) para la gestión, automatización, control, visualización, conectividad y auditoria de procesos. Además, proporciona una plataforma completa de planificación, gestión e integración del ciclo de vida de las aplicaciones, desde la fabricación hasta la infraestructura.
  Esta solución puede ser alojada localmente, en la nube o de forma privada. Desarrollado en torno a estándares industriales como OPC-UA, Open Platform Communications-Data Access (OPC-DA), MQTT y Building Automation and Control Network (BACNet).

- ***Industrial Computer*** **MELIPC**: Está diseñado para realizar el control en tiempo real de los dispositivos de control y la computación de borde que permite la recopilación y análisis de datos en coordinación con sistemas TI. Estos dispositivos incluyen el microprocesador en tiempo real Q06CCPU diseñado para aplicaciones de control industrial. Este módulo microprocesador implementa el Sistema operativo VxWorks de Wind River.

La Tabla 4.1 ilustra la comparación entre las características de la solución de Siemens, Schneider Electric, Mitsubishi Electric y la solución presentada en el trabajo de titulación. La figura 4.18 ilustra el diagrama araña con la comparación de las 4 soluciones. Cada solución tiene un valor entre 1 y 5 en cada característica. El valor de 1 significa que la solución posee una integración pobre con esa característica, mientras que el valor de 5 significa que la solución posee una excelente integración con dicha característica.

La solución desarrollada presenta la ventaja de heterogeneidad, debido al ordenador *Raspberry* Pi que utiliza el sistema. Este ordenador permite la instalación de sistemas operativos libres como Raspbian que a su vez permiten la conexión de dispositivos externos sin ningún inconveniente. Además, permite la adaptabilidad con diferentes lenguajes de programación y con diferentes protocolos IIoT, permitiendo de esta manera el procesamiento de información en tiempo real. Sin embargo, estos dispositivos ofrecen capacidad de cómputo limitado, obligando a adquirir nuevos dispositivos para aumentar la capacidad de computo.

Las soluciones de Siemens, Schneider Electric y Mitsubishi Electric poseen plataformas IIoT escalables, permitiendo un rápido despliegue en los sistemas. Además, poseen compatibilidad con nubes públicas y capacidad de cómputo en tiempo real, lo cual brinda robustez al sistema. Sin embargo, trabajan con dispositivos de la misma marca y utilizan plataformas únicas ya establecidas por cada marca.

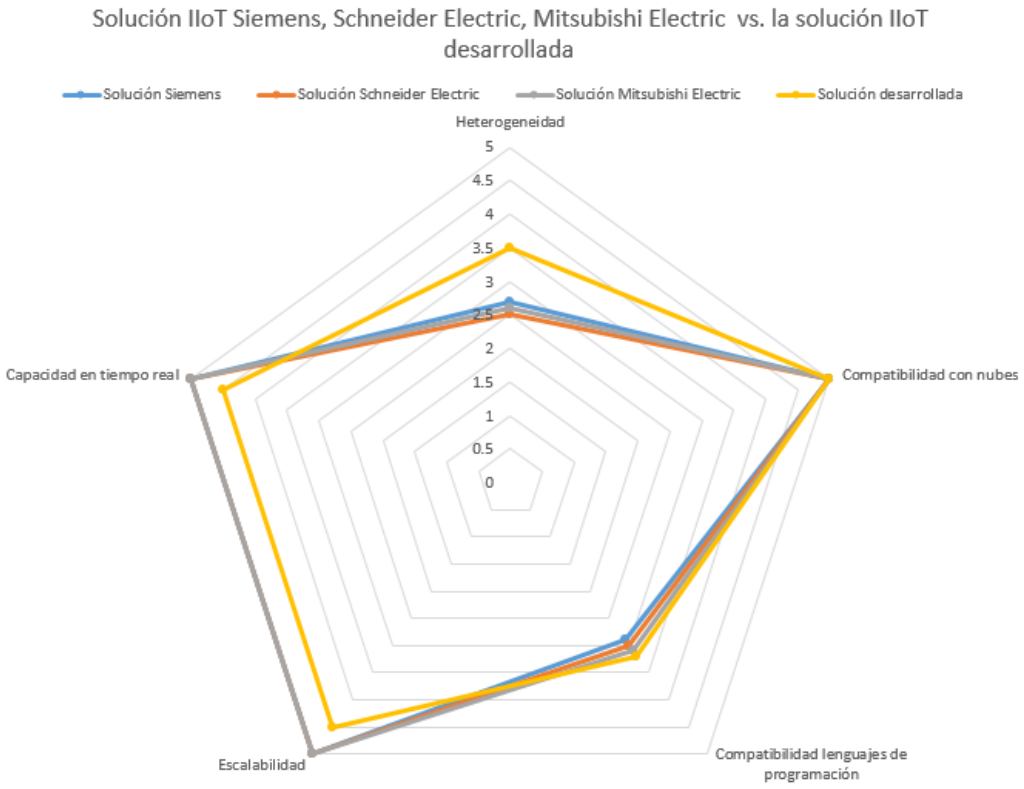


Figura 4.18: Diagrama de araña de la solución IIoT Siemens, Schneider Electric, Mitsubishi Electric vs. la solución IIoT desarrollada.

La solución desarrollada maneja diferentes lenguajes de programación gracias al sistema operativo libre. Sin embargo, las soluciones de Siemens, Schneider Electric y Mitsubishi Electric poseen su propio sistema operativo y permiten lenguajes de programación admitidos por cada marca.

Tabla 4.1: Comparación entre la solución IIoT Siemens, Schneider Electric, Mitsubishi Electric y la solución desarrollada.

Característica		
Heterogeneidad	**Solución Siemens**	**Solución Schneider Electric**
	Todos los dispositivos deben ser de la marca Siemens.	Todos los dispositivos deben ser de la marca Schneider Electric.
	Solución Mitsubishi Electric	**Solución desarrollada**
	Todos los dispositivos deben ser de la marca Mitsubishi Electric.	Permite integrar varios dispositivos y tecnologías industriales.
Compatibilidad con nubes	**Solución Siemens**	**Solución Schneider Electric**
	Posee compatibilidad con las nubes AWS, Azure, Alibaba Cloud.	Posee compatibilidad con nubes públicas.
	Solución Mitsubishi Electric	**Solución desarrollada**
	Posee compatibilidad con nubes públicas.	Posee compatibilidad con nubes públicas.
Compatibilidad lenguajes de programación	**Solución Siemens**	**Solución Schneider Electric**
	Los servicios IIoT se desarrollan en una única plataforma llamada Mendix.	Los servicios IIoT se desarrollan en una única plataforma llamada EcoStruxure Plant.
	Solución Mitsubishi Electric	**Solución desarrollada**
	Los servicios IIoT se desarrollan en una única plataforma llamada VxWorks.	Posee compatibilidad con lenguajes de programación como Python.
Escalabilidad	**Solución Siemens**	**Solución Schneider Electric**
	Los servicios IIoT son escalables.	Los servicios IIoT son escalables.
	Solución Mitsubishi Electric	**Solución desarrollada**
	Los servicios IIoT son escalables.	Posee capacidad limitante. Sin embargo, se puede adquirir nuevos dispositivos con mas capacidad.
Capacidades de tiempo real	**Solución Siemens**	**Solución Schneider Electric**
	Los servicios IIoT trabajan en tiempo real.	Los servicios IIoT trabajan en tiempo real.
	Solución Mitsubishi Electric	**Solución desarrollada**
	Los servicios IIoT trabajan en tiempo real.	Los servicios IIoT trabajan en tiempo real.

Capítulo 5

Conclusiones y Recomendaciones

5.1. Conclusiones

- El sistema realizado en este trabajo de titulación es heterogéneo, y los dispositivos utilizan software libre, razón por la cual se utiliza el protocolo de comunicación abierta OPC-UA. Este protocolo es útil debido a que brinda comunicación abierta interoperable multinivel y multiplataforma.

- En el análisis de la comparación con otras soluciones con IIoT se comprobó que la solución implementada está acoplada a parámetros de compatibilidad con diferente *software*, plataformas IIoT y heterogeneidad. La solución implementada se simplificó al utilizar herramientas de software libre referente a la automatización y comunicación OT y TI. Los dispositivos implementados brindan capacidad de cómputo limitado, presentado la desventaja en escalabilidad.

- Las pruebas de comunicación evidencian una adecuada transferencia de información. El análisis de los paquetes de tráfico para cada protocolo de comunicación muestra que los campos de cada capa contienen las direcciones, puertos y parámetros establecidos en la implementación de cada programa para la comunicación. La monitorización de los paquetes de tráfico en Wireshark no resultó interrumpida.

- En este proyecto se demostró que la implementación conjunta de estos protocolos de comunicación es adecuada para una solución IIoT. El análisis del tráfico confirmó un correcto envío y recepción de datos desde el servidor OPC-UA hacia MQTT, Websockets y viceversa. Las gráficas de tiempo RTT muestran un tiempo muy corto de intercambio de paquetes entre dispositivos.

- Se implementó un sistema IIoT entre dispositivos de bajo costo y *software* libre. Esta implementación condujo a la percepción de una adecuada integración entre diversas herramientas de comunicación y automatización.

ANEXO A

Diseño del plano de la celda de manufactura

Este anexo presenta el plano de la FMC.

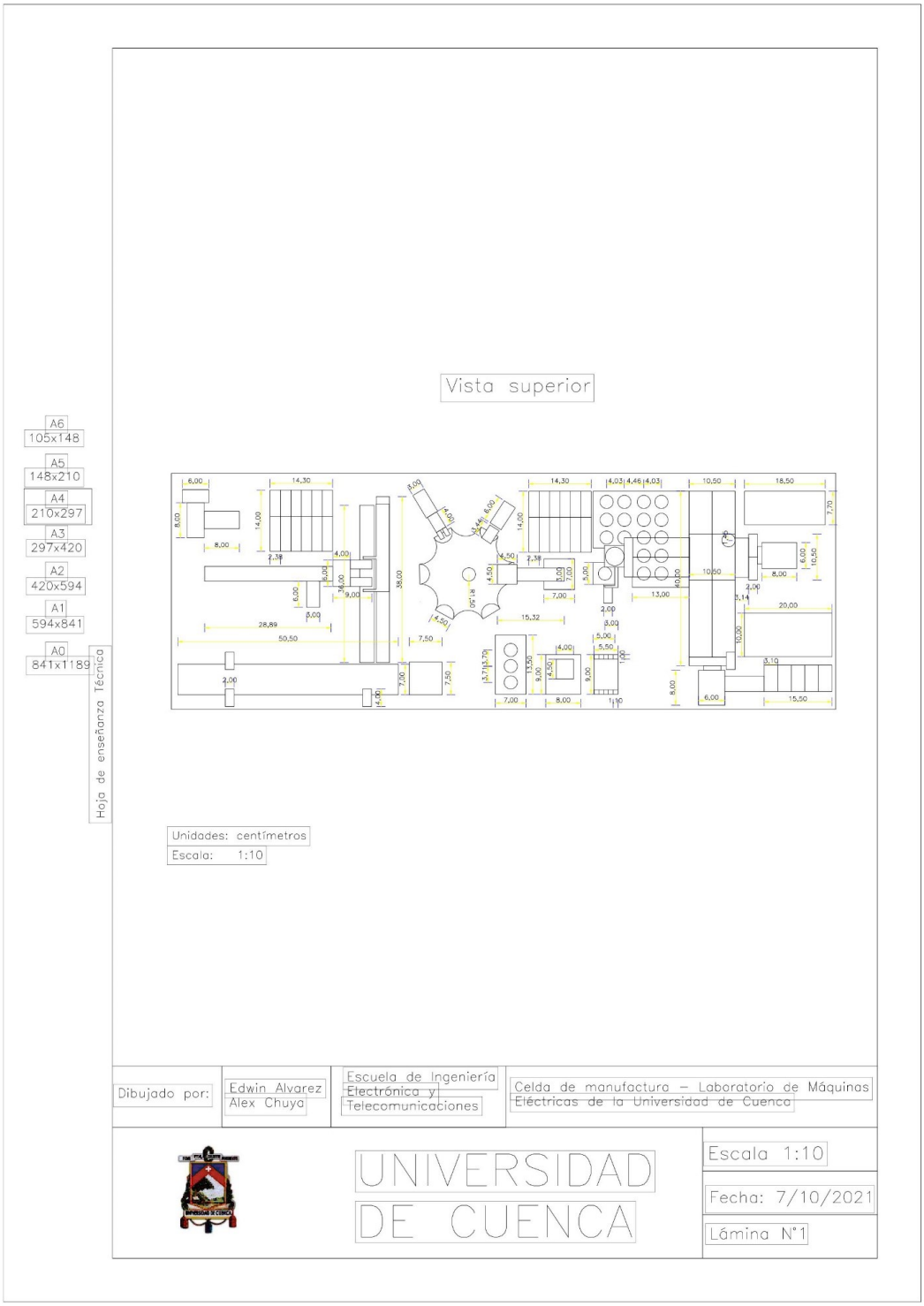


Figura A.1: Plano de la FMC

ANEXO B

Diseño del circuito neumático

Este anexo presenta el diseño del circuito neumático implementado en el proyecto.

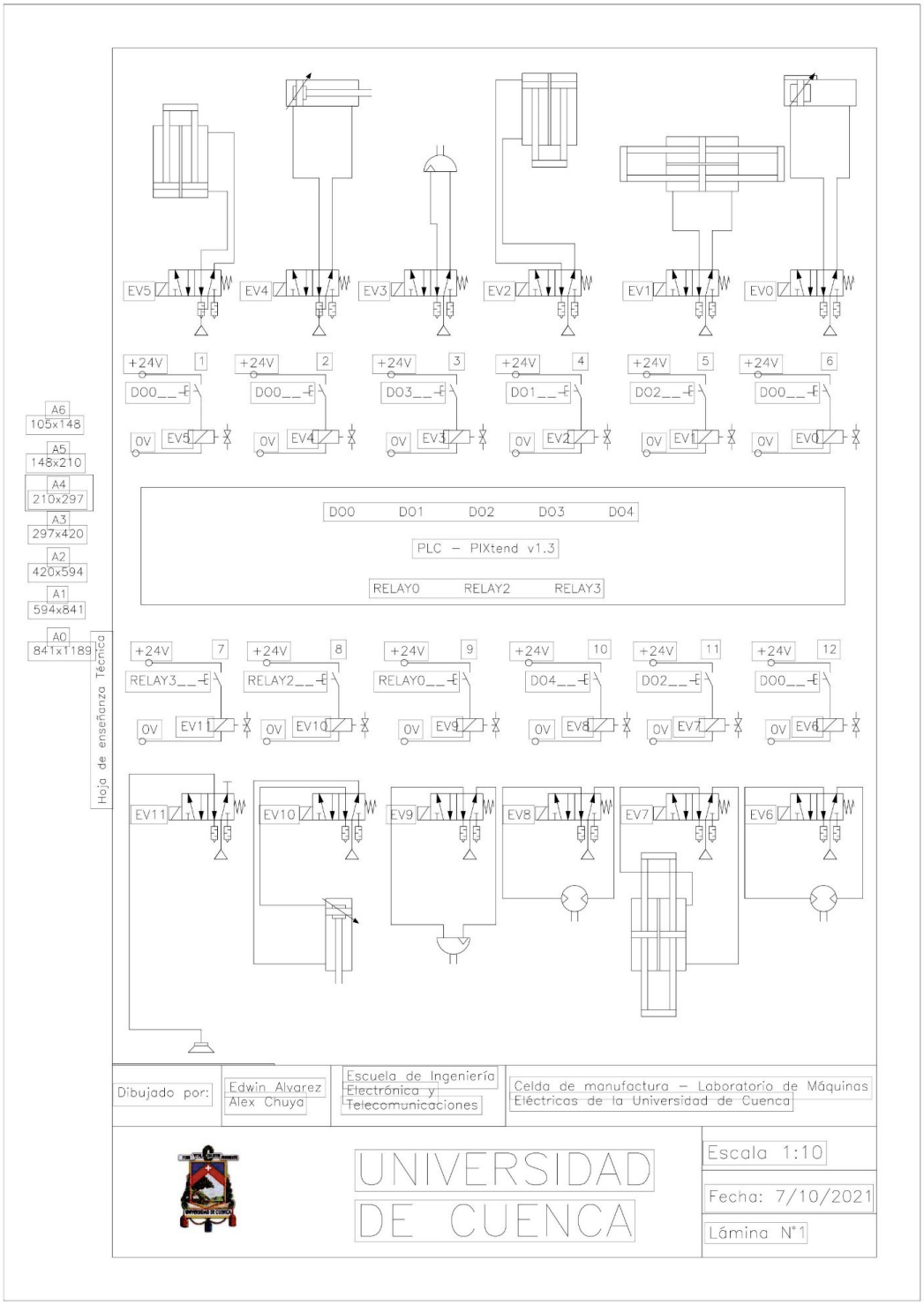


Figura B.1: Representación del circuito neumático de la automatización de la FMC

ANEXO C

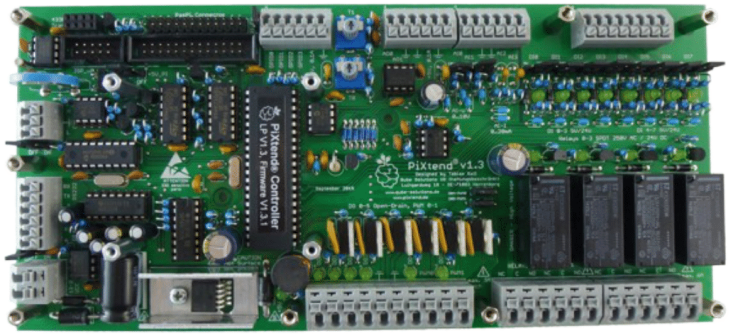
Manual de usuario del PiXtend

Este anexo presenta el manual del dispositivo PiXtend.

PiXtend V1.3
Data sheet: Technical Data and connection hints

PiXtend V1.3 Data sheet

Technical Data and connection hints



Stand 15.06.2016, V1.05

Qube Solutions UG (limited liability)
Arbachtalstr. 6, 72800 Eningen, Germany
http://www.qube-solutions.de
http://www.pixtend.de

Figura C.1: Presentación manual técnico PiXtend [18].

PiXtend V1.3

Data sheet: Technical Data and connection hints

Connection Instructions

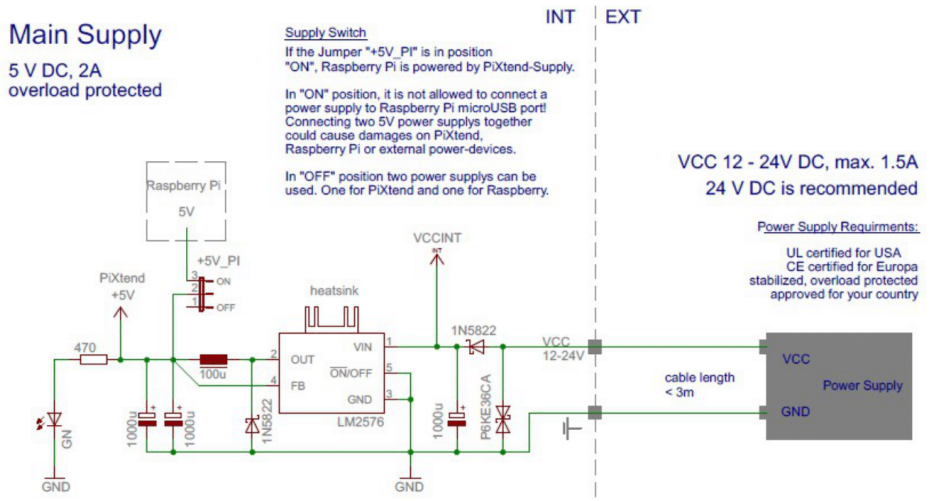


Figure 2: Principle-circuit diagram: Connection of the main supply

The power supply of the PiXtend is realized with a regulated and short-circuit proof power supply unit with an output voltage between 12 and 24 V DC. The power supply unit must correspond to the legal regulations of the country in which the PiXtend is used. If you are buying another power supply unit take care of the appropriate certification marks.

Usable power supply units you also find in our Online-Shop.

The internal power supply of the PiXtend has an energy reserve on the 12-24 V side. This takes care of an interruption-free power supply in case of voltage drops at the voltage input for minimum 10 ms. The energy reserve is so designed that the named hold-up time will last over the complete lifetime of the device.

Figura C.2: Instrucciones de conexión[18].

3. PiXtend- Microcontroller

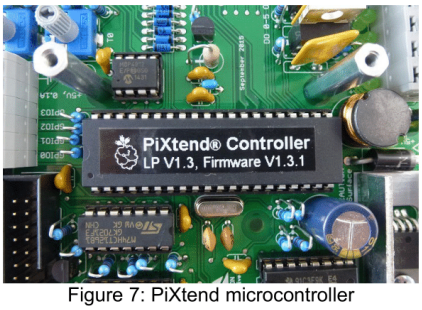


Figure 7: PiXtend microcontroller

The PiXtend- Controller is an 8 bit- RISC- Processor, the ATmega32A from the Atmel Corporation. The Atmega- series is very popular and widespread. Similar controller you can find for example on the Arduino-boards or on our LED-Qube 5.

The microcontroller takes over multiple tasks

- Control the digital outputs and relays
- Generates servo- and PWM- signals
- Reading of analogue and digital inputs
- Operation of the 4 GPIOs (as input, output or temperature- and humidity sensors (DHT11/22 / AM2302))
- Signal- and data processing
- Watchdog-functionality and voltage monitoring (drown out- detection), data security layer with 16 bit CRC- checksum

Raspberry Pi and PiXtend are connected over the SPI- bus (serial peripheral interface). The Raspberry Pi is the bus-master, the PiXtend- controller is the slave.

Our open source firmware is written in the programming language „C“ and compiled with the free of charge program AVC-GCC- Compiler. If needed the controller can be reprogrammed as wanted. But please consider that only with the original firmware the function and the CE- conformity can be guaranteed.

Figura C.3: Microcontrolador-PiXtend [18].

4. Digital in- and outputs

PiXtend has a huge number of digital in- and outputs. In this chapter you will find all relevant technical data, as well as connection- and safety instructions. Read always the relevant sections before you start with the wiring and the work at the in- or output.

4.1 Digital inputs

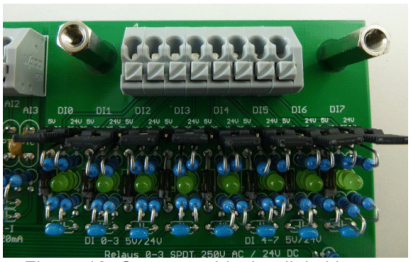


Figure 13: Connector block - digital inputs

The eight digital inputs (DI0-DI7) are realized in 1-wire-connection technology and have reference to the common ground (GND).

Example of use

- Switch, push button, end switch
- Sensors with switch output (proximity switch, light barrier)
- Outputs of other control devices
- Outputs of integrated circuits (TTL-level, CMOS-level)

It is possible to choose between two different voltage ranges (one jumper per input). In the 24 V area the inputs are according to legal requirements of the PLC-norm IEC 61131-2. Also signals with 12 V-level (for example in automotive) are detected reliably. Overload and polarity reversal up to +/- 30 V can't destroy the inputs. The 5V area is designed for 5 V TTL- and 3.3V CMOS-level.

Analog filtering levels increase the resistance to interference and provide a save signal-processing by the PiXtend- microcontroller. LEDs signalize the status of the inputs.

Figura C.4: Entradas y salidas digitales [18].

PiXtend V1.3

Data sheet: Technical Data and connection hints

Connection instructions

The following circuit diagram explains the connection of different signal sources to a digital input. On the left side (INT) is the internal circuit of an input, on the right side (EXT) the possible connection of an external circuit.

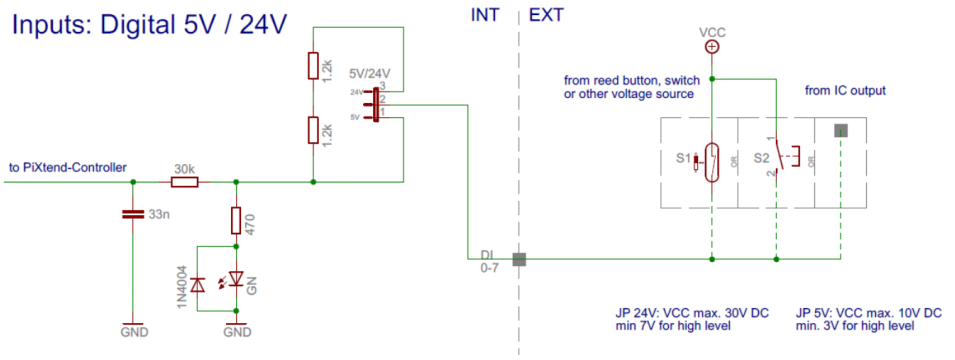


Figure 14: Principle-circuit diagram: Connection of the analog voltage inputs

On the left side of the connector block of the digital inputs is a 2-pole supply connector. Here the switches, sensors etc. can be connected and supplied. The power consumption of the connected signal sources should not exceed 100 mA.
A self-resetting fuse prevents damages at overload or short circuit.

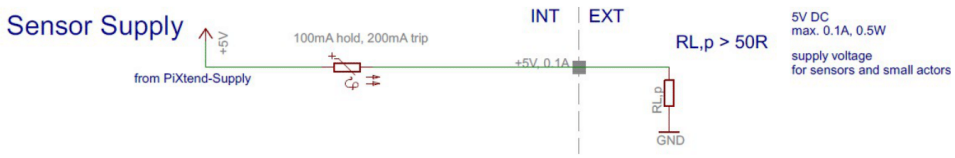


Figure 15: Principle-circuit diagram: Connection of the sensor-supply

Figura C.5: Instrucciones de conexión de entradas digitales [18].

4.2 Digital outputs

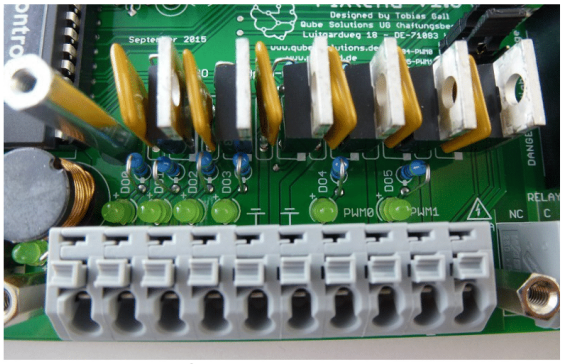


Figure 17: Connection block - digital outputs

Six digital outputs (DO0 – DO5) allow the switching of DC-consumers with voltages up to 30 V and currents up to 3 A.

Example of use

- Switching of external power relays or contactors
- Operation of DC-motors, model making servo motors
- Connection with the inputs of other control units
- Heating- and Peltier elements
- Fans and blowers
- Lamps for direct current and power-LEDs

All six digital outputs are short circuit- and overload protected. Self-resetting fuses (Polyfuse / PTC) save the MOSFET power switch. The outputs are in accordance with „protected and short circuit- resistant outputs“ like it is specified in the PLC-norm (IEC 61131-2) . Due to the open-drain- technique it is possible to switch up to 30 V DC.

LEDs signalize the status of the outputs.

Figura C.6: Salidas digitales [18].

PiXtend V1.3

Data sheet: Technical Data and connection hints

Connection instructions

The following circuit diagram explains the connection of different loads and the digital outputs. On the left side (INT) is the internal circuit of the outputs and on the right side (EXT) is shown the possible external circuit.

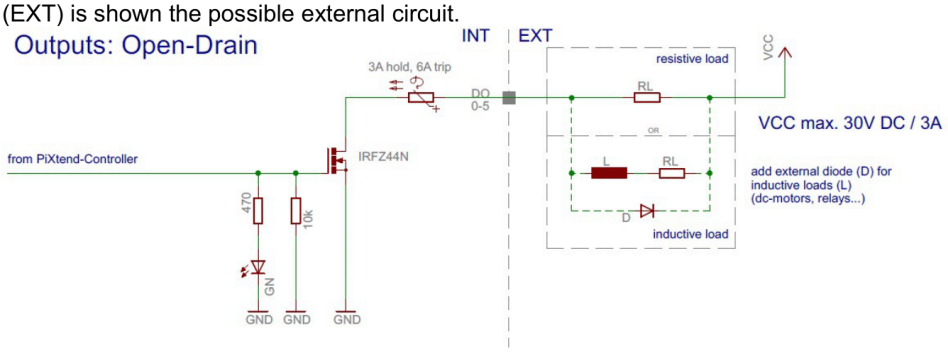


Figure 18: Principle-circuit diagram: Connection of the digital outputs

At inductive loads it needs a protection in the external circuit, to ensure that the voltage at the output never rises higher then 40 V. This can be realised for example with a free-wheel diode (1N4004), as it is shown in Figure 18.

Inductive loads are DC-motors, relays, contactors, solenoids etc.

Should the outputs get connected with inputs (at positive switching) of another controlling device, it will be needed an external pull-up resistor for the input voltage of the controlling device.

The ground connections (GND) of external power supply units are to connect directly with the GND- connections of the connector block of the digital outputs.

Figura C.7: Instrucciones de conexión de salidas digitales [18].

4.3 Relay outputs

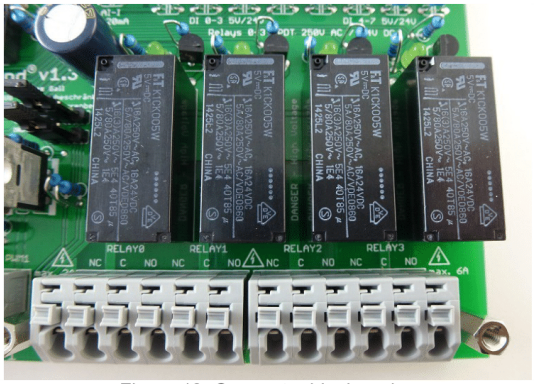


Figure 19: Connector block - relay

The four power relays (RELAY0 – RELAY3) allow the potential free switching from DC voltage and AC voltage consumers. Every relay has 3 connectors (changeover contact).

Example of use

- Turn on / off 115 V / 230 V AC devices
- Switch AC voltages of different voltages and frequencies
- rotational speed of fans with two velocities
- direct operation of DC- or AC- drives
- fans, blowers and lighting
- high loads with low switching cycles

With the universal-relay it is possible to switch everything what needs maximum 230 V and 6 A. All switching contacts are potential free and have no leading contact to the rest of the PiXtend circuit.

LEDs signalize the status of the relays.

Implicitly pay attention to the following safety instructions if you work with dangerous voltages (higher 50 V)!

Figura C.8: Relés de salida [18].

PiXtend V1.3
Data sheet: Technical Data and connection hints

Connection instructions
The following circuit diagram shows the connection of ohmic and inductive DC-loads with the relay outputs. On the left side (INT) is the internal circuit of a relay, on the right side (EXT) is shown a possible external circuit.

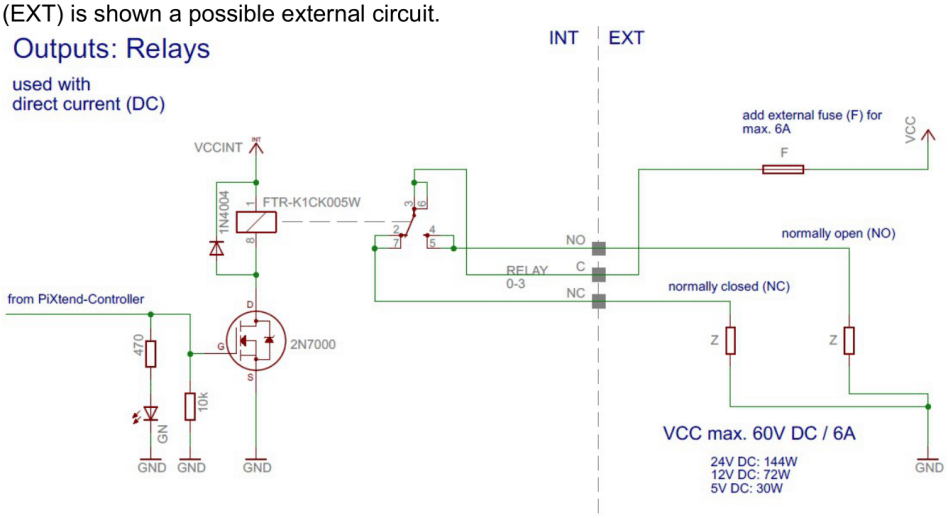


Figure 20: Principle-circuit diagram: Connection of the relays (DC)

The four relay-outputs have no integrated fuse or other overload protection. Because of that it is necessary to include an external protection which trips at 6A.
Inductive loads must have an external free-wheel diode or it must be installed a fitting snubber- network. Otherwise it can cause sparking while switching of the load. This can interfere other devices and damage the contacts.

Inductive loads are motors, relays, contactors, solenoids, power supply units with transformers etc.

The ground connectors (GND) of external power supply units, which are used together with relays, must not be connected to the common PiXtend-GND. The relay contacts have a potential separation and because of that no conducting connection to other components or potentials of the PiXtend or the Raspberry Pi.

The following circuit diagram shows the connection of ohmic and inductive AC-loads with the relay-outputs. On the left side (INT) is the internal circuit of a relay, on the right side (EXT) is shown a possible external circuit.

Figura C.9: Instrucciones de conexión de relés de salida [18].

4.4 GPIOs as digital in- and outputs

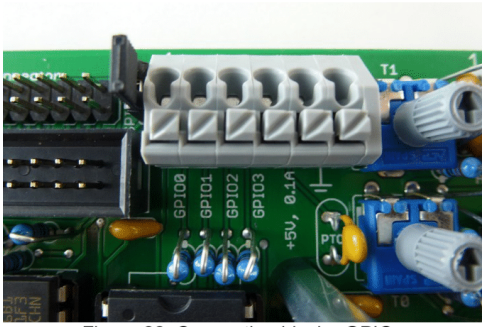


Figure 22: Connection block - GPIOs

The four GPIOs (GPIO0 -GPIO3) on the PiXtend- board can, depending on the requirement, be configured for different functions. Generally the PiXtend- GPIOs are made for a range between 0 V to 5 V.

Example of use

- Connection of sensors, push buttons and end switches (as input)
- Direct connection of small electrical loads like LEDs (as output)
- Connection to digital in- and outputs of other controlling devices or microprocessors
- Connection of temperature- and humidity sensors (DHT11 / DHT22 / AM2302)

In opposite to other in- and outputs on the PiXtend, the GPIOs have only low protection and no input filter. Because of that it are only short wires (shorter 3 m) allowed and if necessary it should be used a external signal processing. But as outputs the GPIOs have a short circuit protection (to GND).

The here described GPIOs are connected to the PiXtend- microcontroller and have nothing to do with the GPIOs of the Raspberry Pi.

The configuration of the GPIOs must be done per software (PiXtend-Linux- tools or CODESYS).

Figura C.10: Entradas y salidas GPIOs [18].

PiXtend V1.3

Data sheet: Technical Data and connection hints

Connection instructions

The following circuit diagram shows the connection of the signal generator to the GPIOs in operation as inputs. On the left side (INT) is the internal circuit of a GPIO, on the right side (EXT) is shown the possible external circuit.

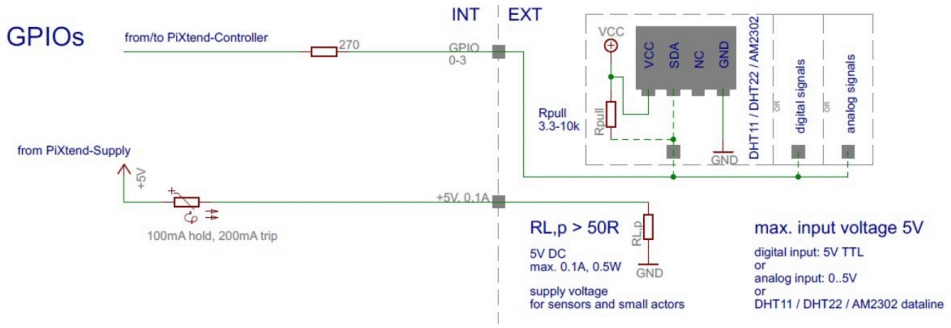


Figure 23: Principle-circuit diagram: Connection of the PiXtend- GPIOs (as inputs)

For the supply of sensors, switches and other signal generators can be used the +5 V connector, which is on the same connector block like the GPIOs. It must be looked for that the consumer doesn't use more then 100 mA from the power supply.
In case of doubt a self-resetting fuse (Polyfuse) saves the supply connector before overload or short circuit.

Should a temperature- or humidity sensor (DHT11 / DHT22 / AM2302) getting connected, so it should be attached a pull-up resistor to the data cable (SDA), like it's shown in figure 23. At cable lengths longer then 2 m we recommend pull-ups in range of 1 kΩ – 3,3 kΩ. Because of certain fluctuations of the sensors (different manufacturers / batches / variants), it is not possible to quote an exact value.
Four of the named sensors can be connected to the power supply without problems. Every sensor takes maximum 1,5 mA. At cable lengths longer then 2 m we recommend to attach a capacitor with minimum 100 nF directly at the sensor between VCC and GND.

Figura C.11: Instrucciones de conexión de GPIOs [18].

5. Analogue in- and outputs

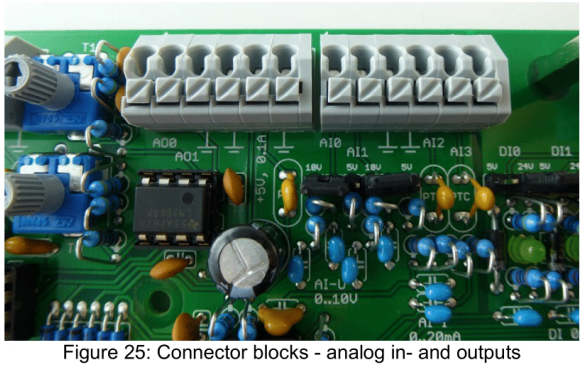


Figure 25: Connector blocks - analog in- and outputs

PiXtend has following in- and outputs:

- two voltage inputs (AI0 – AI1)
- two current inputs (AI2 - AI3)
- two voltage outputs (AO0 - AO1)

Further information and details to the named analogue I/Os you will find on the following sites.

Figura C.12: Entradas y salidas analógicas [18].

5.1 Analogue inputs

Figure 26: Connector block- analog inputs

PiXtend has four analogue inputs. Two for voltage measuring (AI0 – AI1) in the ranges 0..5 V / 0..10 V and two more for current measuring (AI2 – AI3) in the range 0..20 mA.

Example of use

- Analysis of sensors with analogue outputs
- capturing of potentiometer positions (rotary control)
- Current- and voltage measuring in laboratory and at test setups
- Voltage monitoring for accumulators (for example in robots)
- Current measuring with an external shunt (voltage at resistor)
- Analysis of PT100/1000 sensors (with pre-amplifier)
- Connection with the analogue outputs of other controlling devices

The inputs are robust designed and withstand surges up to 30 V DC. Per jumper the voltage areas can be easily changed and adapted to the circumstances.

All four channels are according to the standards of programmable logic controllers (IEC 61131-2) and are therefore usable for a variety of professional sensors and measurement equipment.

Analogue filtering levels ensure low-noise measurements with the 10 bit- analog/digital converter (integrated in the microcontroller).

Figura C.13: Entradas analógicas [18].

6. Special in- and outputs

Some in- and outputs of PiXtend are having special functions.
The functions and the possibilities that they open up are described in the following.

6.1 PWM/Servo-outputs

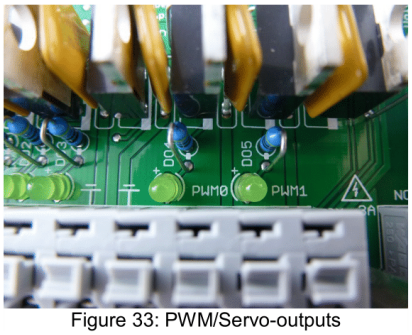


Figure 33: PWM/Servo-outputs

On the 10-pole connection block of the digital outputs are too two connectors named PWM0 and PWM1. At this special outputs it is possible to output pulse width modulated (PWM) signals with adjustable frequency and duty cycle or to directly connect a model making servo.

With help of the jumper „Dox- PWMx“ the PWM- signal directly layed at the digital outputs DO4 and DO5, so that it is possible to run loads up to 3 A.

Example of use

- control position of up to two model making servos
- rotation speed control of fans and other DC-motors
- dimming of DC-lamps and LEDs (high-power-LEDs too)
- controlling of heating element temperature
- fine adjustable clock source for a variety of electronical applications

Figura C.14: Salidas y entradas especiales [18].

PiXtend V1.3
Data sheet: Technical Data and connection hints

Connection instructions

The following circuit diagram shows how to connect consumers and devices to the PWM or digital-outputs of PiXtend. On the left side (INT) is the internal circuit of the outputs, on the right side (EXT) is shown a possible external circuit.

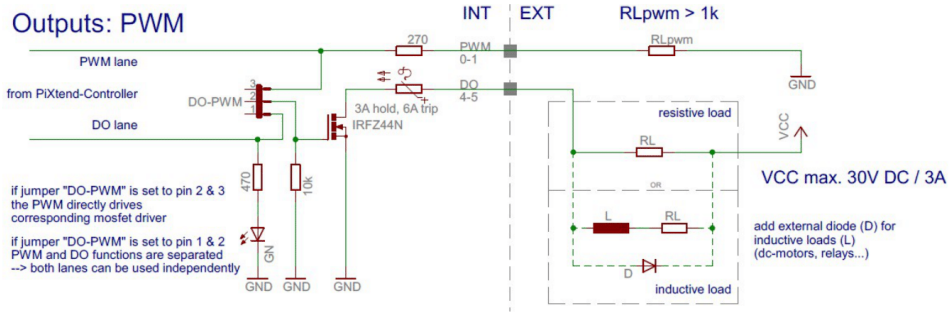


Figure 34: Principle-circuit diagram: Connection of the PWM- / Servo outputs

The PWM outputs PWM0 and PWM1 can be operated in 2 ways. The Jumper „Dox-PWMx“ is for the selection of the operation mode:

1. ***Jumper left (Pin 1 and 2 connected) → DO-mode***

   DO and PWM work independent from each other. At the PWM outputs can the PWM- signal be taken as 5 V-TTL-level. It is only allowed to charge the load with a resistor higher then 1 kΩ, connected with the high omic inputs of a servo motor or other another digital input.

2. ***Jumper right (Pin 2 and 3 connected) → PWM- mode***

   The PWM- signal is getting redirected to the power switch of the belonging output and is able to switch /tact the connected load of this DO because of that.
   The „normal“ DO-function is deactivated in this mode. The changing of the values from DO4 and DO5 in the software is not affecting the state of the respective power driver.

In the software (PiXtend-Linux- Tools and CODESYS- Demoprojekt) can be chosen between Servo- and PWM- mode:

Figura C.15: Instrucciones de conexión de salidas ([18].

ANEXO D

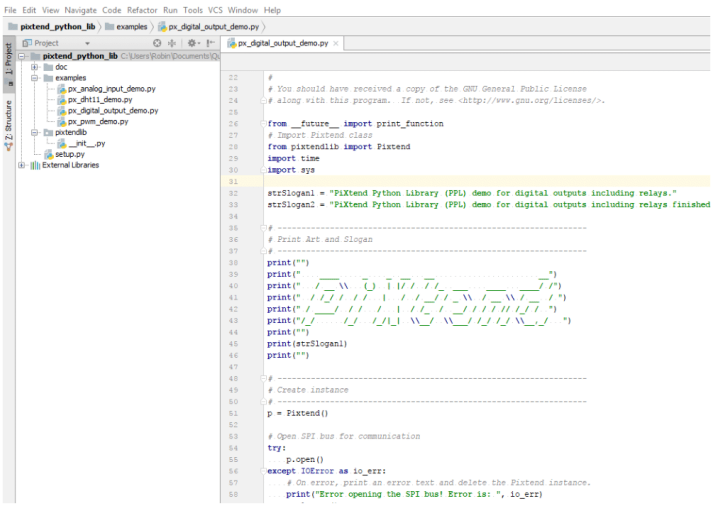
Librería de programación de PiXtend para Python

Este anexo muestra el manual de programación del PiXtend en Python.

PiXtend
Application-Note: PiXtend Python Library

Application-Note
PiXtend Python Library

Installation, Orientation and Programming



APP-PX-401

Status: 05.03.2018, V1.01

Qube Solutions UG (haftungsbeschränkt)
Arbachtalstr. 6, 72800 Eningen, Germany
http://www.qube-solutions.de/
https://www.pixtend.de

Figura D.1: Presentación Librería PPL Python [18].

3. Writing the program

Everything is ready, now we can start with the *Python* programming. As our first program we let relay 0 cyclically turn *on* and *off* every second.

The following program will do this task for us. Simply enter the program in *Notepad++* or copy it over and save it. For manual input please pay attention to the indentation and do not mix *blanks* with *tabulators*, otherwise the program does not run later on and there will be error messages, for *Python* this is not a joke.

The first program:

```
#!/usr/bin/env python

from pixtendlib import Pixtend
import time

p = Pixtend()
p.open()
while True:
    if p.auto_mode() == 0:
        if p.relay0 == p.OFF:
            p.relay0 = p.ON
        else:
            p.relay0 = p.OFF
        time.sleep(1)
```

Program explanation:

- In the first line we declare that this is a *Python* program, this will run later on the Raspberry Pi.
- To wait at the end of the program we import the *time* class, too.
- Then the “Pixtend” class has to be imported, so we get access to the mentioned properties and functions, and a communication with the micro-controller and DAC on the PiXtend board will be possible.
- In a *while* loop, we call the *auto_mode* function, which in *Python* we can combine with an *if* query, that gives us a short code line. We compare the response of the *auto_mode* function to the number 0 (zero), i.e. we check if everything is working. If the function returns the value -1, there is a problem with the communication with the micro-controller.
- **CAUTION**: Do not call the *auto_mode* function faster than ***100 ms*** (0,1 second)
- If the query was successful, we can turn the relay 0 cyclically on and off for example.

Figura D.2: Ejemplo de programación [18].

7. Appendix A

Following is a table showing all the public properties and functions of the *PiXtend Python Library* which an end user can use.

name	type	data type	access[6]	values	description
auto_mode()	function	int	-	0 = OK, -1 = Error	Activation of the auto mode and communication with the micro-controller on the *PiXtend* board, should be called cyclically, minimum is 100ms.
close()	function	-	-		Close the SPI driver and reset all internal variables and objects. Call up before the end of your own program.
open()	function	-	-		Open the SPI master 0 with Chip Select 0 to communicate with the micro-controller on the *PiXtend* board. May only be called once.
open_dac()	function	-	-		Open the SPI master 0 with Chip Select 1 to communicate with the DAC.
pwm_ctrl_configure()	function	-	-		Configuration of the PWM channels, must be called up after every change of the PWM settings.
set_dac_output()	function	-	-		Transfer of the configuration and analog values for DAC A and DAC B. Before calling, define *dac_selection* target DAC.
update_rtc()	function	-	-		Write Linux system time into the RTC on the *PiXtend* board.
analog_input0	property	float	R	0 V .. 5 V/10 V	Analog input on the *PiXtend*, provides values always in the unit *Volt*. Value range depending on *analog_input0_10volts_jumper*.
analog_input0_10volts_jumper	property	int	R / W	0 = OFF, 1 = ON	Information whether the 5V / 10V jumper on the *PiXtend* board is physically plugged in.
analog_input0_nos	property	int	R / W	1, 5, 10, 50	Number of samples to be taken by the MC. The default value is 10.
analog_input0_raw	property	int	R	16 bit value	Raw value from the micro-controller without conversion.
analog_input1	property	float	R	0 V .. 5 V/10 V	Analog input on the *PiXtend* board, always supplies values in the unit *Volt*. Value range depending on *analog_input1_10volts_jumper*.
analog_input1_10volts_jumper	property	int	R / W	0 = OFF, 1 = ON	Information whether the 5V / 10V jumper on the *PiXtend* board is physically plugged in.
analog_input1_nos	property	int	R / W	1, 5, 10, 50	Number of samples to be taken by the MC. The default value is 10.
analog_input1_raw	property	int	R	16 bit value	Raw value from micro-controller without conversion.
analog_input2	property	float	R	0 mA .. 20 mA	Converted analog value from the micro-controller in 0 to 20 milliAmps.

[6] R = Reading , W = Writing, R / W = Reading and Writing

Figura D.3: Funciones pertenecientes a la librería (PPL) Python [18].

PiXtend

Application-Note: PiXtend Python Library

name	type	data type	access	values	description
analog_input2_nos	property	int	R / W	1, 5, 10, 50	Number of samples to be taken by the MC. The default value is 10.
analog_input2_raw	property	int	R	16 bit value	Raw value from micro-controller without conversion.
analog_input3	property	float	R	0 mA .. 20 mA	Converted analog value from micro-controller in 0 to 20 milliAmps.
analog_input3_nos	property	int	R / W	1, 5, 10, 50	Number of samples to be taken by the MC. The default value is 10.
analog_input3_raw	property	int	R	16 bit value	Raw value of micro-controller without conversion.
analog_input_nos_freq	property	float	R / W	0.125, 0.250, 0.500, 1.0, 2.0, 4.0, 8.0 MHz	Sampling frequency of the analog inputs.
dac_selection	property	int	R / W	0 = DAC A, 1 = DAC B	Selection of the DAC that is to be changed. DAC A = AO 0, DAC B = AO 1.
dht0	property	int	R / W	0 = OFF, 1 = ON	Configuration of GPIO 0 as DHT11/22 input. If 1 (ON), all other GPIO settings are ignored.
dht1	property	int	R / W	0 = OFF, 1 = ON	Configuration of GPIO 1 as DHT11/22 input. If 1 (ON), all other GPIO settings are ignored.
dht2	property	int	R / W	0 = OFF, 1 = ON	Configuration of GPIO 2 as DHT11/22 input. If 1 (ON), all other GPIO settings are ignored.
dht3	property	int	R / W	0 = OFF, 1 = ON	Configuration of GPIO 3 as DHT11/22 input. If 1 (ON), all other GPIO settings are ignored.
di0	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di1	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di2	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di3	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di4	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di5	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di6	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
di7	property	int	R	0 = OFF, 1 = ON	State of the digital input in abbreviated form.
digital_input0	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_input1	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_input2	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_input3	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_input4	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_input5	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_input6	property	int	R	0 = OFF, 1 = ON	State of the digital input.

Figura D.4: Funciones pertenecientes a la librería PPL Python [18].

PiXtend

Application-Note: PiXtend Python Library

name	type	data type	access	values	description
digital_input7	property	int	R	0 = OFF, 1 = ON	State of the digital input.
digital_output0	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible.
digital_output1	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible.
digital_output2	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible.
digital_output3	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible.
digital_output4	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible.
digital_output5	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible.
do0	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible. This is an abbreviated form of the corresponding digital output and not a further output.
do1	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible. This is an abbreviated form of the corresponding digital output and not a further output.
do2	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible. This is an abbreviated form of the corresponding digital output and not a further output..
do3	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible. This is an abbreviated form of the corresponding digital output and not a further output.
do4	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible. This is an abbreviated form of the corresponding digital output and not a further output.
do5	property	int	R / W	0 = OFF, 1 = ON	State of the digital output. Reading and writing are always possible. This is an abbreviated form of the corresponding digital output and not a further output.
gpio0	property	int	R / W	0 = OFF, 1 = ON	State of the GPIO, Attention! The *"setting"* of this property is only possible if the GPIO is configured as *"output"*, otherwise there will be an *IOError.*
gpio0_direction	property	int	R / W	0 = input, 1 = output	Configuration of the GPIO either as input (default) or as output.
gpio1	property	int	R / W	0 = OFF, 1 = ON	State of the GPIO, Attention! The *"setting"* of this property is only possible if the GPIO is configured as *"output"*, otherwise there will be an *IOError.*
gpio1_direction	property	int	R / W	0 = input, 1 = output	Configuration of the GPIO either as input (default) or as output.

Figura D.5: Funciones pertenecientes a la librería PPL Python [18].

PiXtend

Application-Note: PiXtend Python Library

name	type	data type	access	values	description
gpio2	property	int	R / W	0 = OFF, 1 = ON	State of the GPIO, Attention! The *"setting"* of this property is only possible if the GPIO is configured as *"output"*, otherwise there will be an *IOError*.
gpio2_direction	property	int	R / W	0 = input, 1 = output	Configuration of the GPIO either as input (default) or as output.
gpio3	property	int	R / W	0 = OFF, 1 = ON	State of the GPIO, Attention! The *"setting"* of this property is only possible if the GPIO is configured as *"output"*, otherwise there will be an *IOError*.
gpio3_direction	property	int	R / W	0 = input, 1 = output	Configuration of the GPIO either as input (default) or as output.
h0_dht11	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h0_dht22	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h1_dht11	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h1_dht22	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h2_dht11	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h2_dht22	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h3_dht11	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
h3_dht22	property	float	R	rel. humidity in %	Converted value from the sensor according to the specifications of the data sheet of the sensor.
hum_input0_raw	property	int	R	16-bit integer value from the sensor	This property supplies only the raw value of the sensor without conversion.
hum_input1_raw	property	int	R	16-bit integer value from the sensor	This property supplies only the raw value of the sensor without conversion.
hum_input2_raw	property	int	R	16-bit integer value from the sensor	This property supplies only the raw value of the sensor without conversion.
hum_input3_raw	property	int	R	16-bit integer value from the sensor	This property supplies only the raw value of the sensor without conversion.
pwm0	property	int	R / W	0 .. 65000	PWM 0 duty cycle.
pwm1	property	int	R / W	0 .. 65000	PWM 1 duty cycle.
pwm_ctrl_cs0	property	int	R / W	0 = OFF, 1 = ON	PWM clock select bit 0.

Figura D.6: Funciones pertenecientes a la librería PPL Python [18].

PiXtend

Application-Note: PiXtend Python Library

name	type	data type	access	values	description
pwm_ctrl_cs1	property	int	R / W	0 = OFF, 1 = ON	PWM clock select bit 1.
pwm_ctrl_cs2	property	int	R / W	0 = OFF, 1 = ON	PWM clock select bit 2.
pwm_ctrl_mode	property	int	R / W	0 = Servo Mode, 1 = PWM Mode	Operating mode of the PWM outputs.
pwm_ctrl_od0	property	int	R / W	0 = OFF, 1 = ON	Overdrive setting for PWM 0.
pwm_ctrl_od1	property	int	R / W	0 = OFF, 1 = ON	Overdrive setting for PWM 1.
pwm_ctrl_period	property	int	R / W	0 .. 65000	Frequency of PWM's.
relay0	property	int	R / W	0 = OFF, 1 = ON	State of relay 0.
relay1	property	int	R / W	0 = OFF, 1 = ON	State of relay 1.
relay2	property	int	R / W	0 = OFF, 1 = ON	State of relay 2.
relay3	property	int	R / W	0 = OFF, 1 = ON	State of relay 3.
serial_mode	property	bool	R / W	False = RS232, True = RS485	Define the mode of the serial interface on the *PiXtend* board.
servo0	property	int	R / W	0 .. 250	Unit for setting the motor position of the model-building servomotor.
servo1	property	int	R / W	0 .. 250	Unit for setting the motor position of the model-building servomotor.
t0_dht11	property	float	R	temperature in degrees Celsius, e.g. 9.0	Values from a DHT11 sensor connected to GPIO0.
t0_dht22	property	float	R	temperature in degrees Celsius, e.g. 9.5	Values from a DHT22 sensor connected to GPIO0.
t1_dht11	property	float	R	temperature in degrees Celsius, e.g. 9.0	Values from a DHT11 sensor connected to GPIO1.
t1_dht22	property	float	R	temperature in degrees Celsius, e.g. 9.5	Values from a DHT22 sensor connected to GPIO1.
t2_dht11	property	float	R	temperature in degrees Celsius, e.g. 9.0	Values from a DHT11 sensor connected to GPIO2.
t2_dht22	property	float	R	temperature in degrees Celsius, e.g. 9.5	Values from a DHT22 sensor connected to GPIO2.
t3_dht11	property	float	R	temperature in degrees Celsius, e.g. 9.0	Values from a DHT11 sensor connected to GPIO3.
t3_dht22	property	float	R	temperature in degrees Celsius, e.g. 9.5	Values from a DHT22 sensor connected to GPIO3.

Figura D.7: Funciones pertenecientes a la librería PPL Python [18].

PiXtend

Application-Note: PiXtend Python Library

name	type	data type	access	values	description
temp_input0_raw	property	int	R	16-bit integer value from the sensor	Direct raw value from the sensor without conversion.
temp_input1_raw	property	int	R	16-bit integer value from the sensor	Direct raw value from the sensor without conversion.
temp_input2_raw	property	int	R	16-bit integer value from the sensor	Direct raw value from the sensor without conversion.
temp_input3_raw	property	int	R	16-bit integer value from the sensor	Direct raw value from the sensor without conversion.
uc_board_version	property	int	R	12 = 1.2.x, 13 = 1.3.x	Hardware version of the *PiXtend* board, e.g. Ver. 1.2.x which can not read back the digital outputs and relays states.
uc_control	property	int	R	0 = Man. Mode, 16 = Auto Mode	Setting the operating mode of the micro-controller on the *PiXtend* board.
uc_fw_version	property	int	R	1, 2, 3, 4, 5 equates firmware version	Firmware version of the micro-controller on the *PiXtend* board.
uc_status	property	int	R	0 = Init, 1 = Run	State of the micro-controller.
use_fahrenheit	property	bool	R / W	False = °C, True = °F	The temperatures from T0 to T3 are given in °C by default, if this value is "*True*", you get °F.

Figura D.8: Funciones pertenecientes a la librería PPL Python [18].

Bibliografía

[1] D. J. Saa Zamorano *y otros*, "Análisis de la industria 4.0 en latinoamérica y países desarrollados," 2021.

[2] M. X. López Flores, "Industria 4.0 para la monitorización de un proceso industrial," Master's thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas . . . , 2019.

[3] C. B. Y., J. M. I., J. G. B., F. A., y M. L., "El entorno de la industria 4.0: Implicaciones y perspectivas futuras," *Conciencia Tecnológica*, 2017. [En línea]. Disponible: https://www.redalyc.org/articulo.oa?id=94454631006

[4] A. P. T. Guacapiña, "Sistema de monitorización para la industria 4.0. un enfoque basado en sistemas ciberfísicos," 2018. [En línea]. Disponible: http://oa.upm.es/52623/

[5] 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," *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, vol. 2020, pp. 667–680, 2 2020.

[6] M. Casalet, "La digitalización industrial un camino hacia la gobernanza colaborativa estudios de caso la digitalización industrial un camino hacia la gobernanza colaborativa estudios de casos," 2018. [En línea]. Disponible: www.cepal.org/es/suscripciones

[7] D. T. Basantes Montero, "Sistema de manufactura flexible orientado a industria 4.0," Master's thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas . . . , 2019.

[8] C. S. Chiliquinga Rodríguez, "Internet de las cosas basado en redes definidas por software," B.S. thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas . . . , 2020.

[9] D. R. Díaz, "Diseño e implementación de una celda automatizada con robótica colaborativa," 7 2021.

[10] P. F. S. d. Melo, "Dispositivo de controle para a indústria 4.0 baseado no rami 4.0 e opc ua," 2020.

[11] C. A. Montilla y J. F. Arroyave, "Simulación de operación de celdas de manufactura flexible fmc, utilizando las redes de petri," *Scientia et technica*, vol. 1, num. 34, 2007.

[12] "Neumática electroneumática fundamentos." [En línea]. Disponible: www.festo-didactic.com

[13] D. Bruckner, M.-P. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, y T. Sauter, "An introduction to opc ua tsn for industrial communication systems," *Proceedings of the IEEE*, vol. 107, pp. 1121–1131, 2019.

[14] S. Grüner, J. Pfrommer, y F. Palm, "Restful industrial communication with opc ua," *IEEE Transactions on Industrial Informatics*, vol. 12, p. 1, 2 2016.

[15] H. Boyes, B. Hallaq, J. Cunningham, y T. Watson, "The industrial internet of things (iiot): An analysis framework," *Comput. Ind.*, vol. 101, pp. 1–12, 2018.

[16] D. R. C. Silva, G. M. B. Oliveira, I. Silva, P. Ferrari, y E. Sisinni, "Latency evaluation for mqtt and websocket protocols: an industry 4.0 perspective," 2018, pp. 1233–1238.

[17] (2021) Raspberry pi 3 model b. [En línea]. Disponible: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

[18] (2021) Pixtend v1.3 downloads. [En línea]. Disponible: https://www.pixtend.de/downloads/downloads-v1-3/