ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Automatización de un Sistema de Esclusas Empleando Tecnología del Internet de las Cosas Industrial (IIoT)

PROYECTO DE TITULACIÓN

Previo la obtención del Título de:

Magister en Automatización y Control

Presentado por:
Carlos Augusto Zúñiga Reyes

GUAYAQUIL - ECUADOR
Año: 2022

DEDICATORIA

El presente proyecto lo dedico a Dios que gracias a Él tengo la vida y a su madre que intercede por mí.

A mi esposa, Cinthia Pamela Vega Ravelo, mi compañera de vida, mi mejor amiga.

A mis padres, Carlos Zúñiga Daquilema y Sara Reyes Villón, los mejores padres que un ser humano podría desear.

A mis queridos hermanos Herdiz, Ronny y Karlita.

A mi suegra, Amarilis Ravelo, por siempre estar dispuesta a darme su ayuda y estima.

AGRADECIMIENTOS

Mi más sincero agradecimiento al PhD. Cesar Martín por su tutoría, al PhD. Dennys Paillacho, PhD. Efrén Herrera y al PhD. Douglas Plaza por la gestión de revisión y aprobación de mi proyecto de titulación. Un agradecimiento también a todos los profesores de la MACI promoción 9 por sus grandes aportes a mis conocimientos profesionales y al personal administrativo por su excelente trabajo.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; *Carlos Augusto Zúñiga Reyes* doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

CARLOS AUGUSTO ZUNIGA REYES

Firmado digitalmente por CARLOS AUGUSTO ZUNIGA REYES
Fecha: 2022.08.02 10:48:18 -05'00'

Carlos A. Zúñiga Reyes

COMITÉ EVALUADOR

CESAR ANTONIO MARTIN MORENO

Firmado digitalmente por CESAR ANTONIO MARTIN MORENO
Fecha: 2022.08.02 10:55:16 -05'00'

PhD. Cesar Martín

PROFESOR TUTOR

Firmado electrónicamente por:
DENNYS FABIAN PAILLACHO CHILUIZA

PhD. Dennys Paillacho

PROFESOR EVALUADOR

RESUMEN

El proyecto de titulación está enfocado a la automatización de un sistema de esclusas de 4 compuertas utilizando un frenado dinámico para cada compuerta, con la finalidad de poder contener el agua entre secciones. Se tomó de ejemplo el actual sistema de esclusas de la ciudad de Guayaquil que une el río Guayas con el Estero Salado, y en base a este sistema de esclusas se desarrolló un prototipo mediante el uso de tecnología de la industria 4.0. Como parte de las tecnologías de la industria 4.0 se usó el internet de las cosas industriales para el control inalámbrico, además de tecnologías de la industria 3.0 como el uso de controladores lógicos programables, interfaces humano máquina y bases de datos. Como parte de las pruebas finales se realizó un testeo de latencias de todas las comunicaciones para ver si es viable la implementación del Internet de las Cosas (IoT) en un proyecto industrial, además se realizó una prueba del frenado dinámico para la correcta compresión de los sellos mecánicos de las compuertas usando un controlador proporcional, integral y derivativo (PID), previamente se obtuvo la función de transferencia de la planta.

Palabras Clave: Proyecto Integrador, ESPOL, Esclusas, Automatización, Industria 4.0.

ABSTRACT

The titling project is focused on the automation of a 4-gate lock system using dynamic braking for each gate, in order to contain the water between sections. The current system of locks in the city of Guayaquil that connects the Guayas River with the Estero Salado was taken as an example, and a prototype was developed about that, through the use of industry 4.0 technology. As part of industry 4.0 technologies used the internet of industrial things for wireless control, in addition to industry 3.0 technologies such as the use of programmable logic controllers, human-machine interfaces, and databases. As part of the final tests, a latency test of all communications was carried out to see if the implementation of the Internet of Things (IoT) in an industrial project is feasible, in addition, dynamic braking tests were carried out for the correct compression of the mechanical seals of the gates using a proportional, integral and derivative controller (PID), previously the transfer function of the plant was obtained.

Keywords: *Integrative Project, ESPOL, Lock River, Automation, Industry 4.0.*

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
PLC	Programmable Logic Controller
VDC	Voltage Direct Current
VAC	Voltage Alternating Current
IIoT	Industrial Internet of Things
HMI	Human Machine Interface
SCADA	Supervisory Control and Data Acquisition
CEPAL	Comisión Económica para América Latina
IO	Internet de los Objetos
WSN	Wireless Sensor Networks
PC	Personal Computer
ERP	Enterprise Resource Planning
MES	Manufacturing Execution Systems
PID	Proportional, Integral, Derivative
DNO	Driver Noroeste
DNE	Driver Noreste
DSO	Driver Suroeste
DSE	Driver Noreste
MNO	Motor Noroeste
MNE	Motor Noreste
MSO	Motor Suroeste
MSE	Motor Noreste
SW	Switch de Flujo
TB	Terminal Block
SSW	Safety Switch
LT	Luz Terminal
IP	Internet Protocol Address
HTTP	Hypertext Transfer Protocol
LAD	Ladder
LAD	Totally Integrated Automation
DB	Data Blocks
OB	Organization Blocks

MCU	Microcontroller Unit
IDE	Integrated Development Environment
WIFI	Wireless Fidelity
CAD	Computer Aided Design
IETH	Industrial Ethernet
TCP	Transmission Control Protocol

SIMBOLOGÍA

no	Noroeste
ne	Noreste
so	Suroeste
se	Noreste
s	Segundos
ms	Milisegundos
º	Grados
mV	Milivoltio
r.p.m.	Revoluciones por minuto
r.p.s.	Revoluciones por segundo

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE PLANOS

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema

Guayaquil se ha caracterizado a lo largo de su historia como una ciudad puerto por excelencia ya que cuenta con diversos afluentes que interconectan el golfo de Guayaquil con el océano pacífico. Actualmente se busca realizar obras de infraestructura para potenciar su calidad de ciudad puerto y que la ubique de forma competitiva a nivel internacional, esto considerando que con la construcción del tercer juego de esclusas en el Canal de Panamá y el dragado al acceso al Puerto de Guayaquil se tendrá mayor movimiento de comercio internacional en la ruta Guayaquil-Panamá.

Como parte de estas obras de repotenciación se encuentra el proyecto de rehabilitación de las esclusas que fueron construido en 1962 y este canal operó con normalidad hasta 1970 donde su tráfico disminuyó ostensiblemente[1]. Las esclusas de Guayaquil permiten comunicar el río Guayas con el Estero Salado y permite la navegación de barcos. Una vez rehabilitado el canal de “Las Esclusas” podrán circular embarcaciones desde un metro de calado, hasta el Buque Escuela Guayas (4.40 metros de calado), además de otro tipo de flotas con un máximo de 12 metros de manga[2], ayudando con esto a la movilización de embarcaciones que van desde las empresas ubicadas en las riberas del río Guayas hasta los puertos ubicados en el estero salado y viceversa.

Actualmente la navegación tarda aproximadamente 12 horas entre el Puerto Marítimo de Guayaquil ubicado en el estero salado y el barrio del astillero ubicado en las riberas del río Guayas. Con las esclusas habilitadas se consigue la reducción del tiempo de navegación de las embarcaciones de 12 horas a 2 horas entre el puerto de Guayaquil y el Río Guayas[3] lo que implicaría un enorme ahorro de recursos para las embarcaciones. Además, la maniobrabilidad de navegación para el ingreso a Guayaquil desde el océano pacífico se facilita al ingresar por el río Guayas que, por el estero salado, ya que por el

[1] Informe Marítimo Portuario, las Esclusas de Guayaquil Nº12

[2] Fuente CAMAE

[3] Fuente obtenida de CAMAE

estero salado se debe tener en cuenta muchos factores como el nivel del agua en ciertos puntos del recorrido que podrían llegar a encallar a una embarcación que no conozca bien la zona o no posea equipos de medición de nivel adecuados o bien calibrados.

El proyecto de las esclusas se encuentra dividido en dos partes o dos hitos entregables, el primer hito ya fue entregado y en ese hito se incluyó un sistema de automatización de 4 compuertas de forma individual, esto significa que cada compuerta posee su propio PLC, variador de frecuencia y finales de carrera que la moverán en apertura o cierre para trabajos de mantenimiento. El segundo hito pretende agregar un control a cada compuerta desde un único controlador lógico, para que mediante regulación de la velocidad de los motores las compuertas puedan ir acelerando o frenando de forma automática hasta posicionarse en el ángulo exacto para una correcta compresión de los sellos mecánicos.

1.2 Justificación del problema

La propuesta de Titulación mediante la automatización del sistema de esclusas, apunta a poder diseñar un sistema que permita controlar la apertura y cierre de las compuertas de las esclusas obteniendo la compresión adecuada entre los sellos mecánicos que impiden el paso de agua entre secciones, este control se lo realizará mediante la regulación de la velocidad del motor. Al no poseer acceso a un motor de esclusa y su respectivo variador de frecuencia para realizar las pruebas de concepto, se ha optado por trabajar con el motor de la tarjeta TSC LAB (ver sección 1.4.4) entregada por la ESPOL que me permite de forma académica trabajar con un motor de voltaje continuo (VDC) y poder leer sus rpm.

Es necesario resaltar la importancia de que en un sistema de esclusas la compresión de sellado mecánico entre las compuertas es primordial para poder conseguir la nivelación correcta del agua entre secciones y así poder cumplir la finalidad que es cambiar al barco de posición en altura para que pueda ir del río Guayas al estero salado o viceversa, aclarando que el nivel entre el estero salado y del río Guayas puede llegar a tener hasta 1.6 metros de altura de diferencia. Si este sellado mecánico no se llega a dar de forma exacta se corre el riesgo de no poder nivelar el agua en la sección interior y crear corrientes que arrastren al barco hacia las compuertas ocasionando accidentes.

1.3 Objetivos

1.3.1 Objetivo General

Automatizar sistema de esclusas para la apertura y cierre de compuertas mediante el control de la velocidad del motor para mantener el correcto nivel de agua entre secciones utilizando tecnologías: IoT, comunicaciones inalámbricas y control industrial.

1.3.2 Objetivos Específicos

- Desarrollar una estructura de comunicación entre la tarjeta TSC LAB, el HMI y el controlador lógico programable en un servidor IoT
- Diseñar una interfaz de usuario para la visualización de datos, alarmas e inicio del proceso de apertura y cierre usando un HMI industrial.
- Desarrollar la lógica del control de la apertura y cierre de las compuertas e implementarlo en un controlador lógico programable
- Implementar un controlador de velocidad de motor de corriente continua usando Microcontroladores
- Usar una base de datos para registro de eventos y alarmas

1.4 Marco teórico

El marco teórico del proyecto de titulación está divido en 4 secciones las cuales tratan de abarcar la mayor cantidad de herramientas y conocimientos científicos usados para el desarrollo de la propuesta. Estas secciones comprenden los sistemas de monitoreo y control, sistemas de esclusas, instrumentos de las esclusas y por último el internet industrial de las cosas.

1.4.1 Sistemas SCADA

Los sistemas de monitoreo y control más completos que tenemos son los sistemas de supervisión, control y adquisición de datos (SCADA), para este proyecto de titulación usaremos un sistema SCADA. La definición de SCADA es descrito para el autor Aquilino como “cualquier software que permita el acceso a datos remotos de un proceso y permita, utilizando las herramientas de comunicación necesarias en cada caso, el control del mismo.” (Rodríguez Penin, 2007, pág. 9), mientras que para el autor Esteban Pérez

los sistemas SCADAS es un tipo de software que “permite ilustrar gráficamente los procesos productivos en pantalla y crear alarmas y advertencias en tiempo real, para el manejo confiado y pleno del proceso que se desea controlar” (Pérez-López, 2015, pág. 4).

Los sistemas SCADA se encuentran en el tercer nivel de la pirámide de automatización industrial como se indica en la figura 1.1

Figura 1.1 Pirámide de automatización industrial

(Festo Didactic SE, 2021, pág. 129)

Dentro de las prestaciones del sistema SCADA como herramienta destacamos las siguientes características las cuales son, monitorización, supervisión, adquisición de datos, visualización de alarmas y eventos, mando, grabación de acciones o recetas, seguridad de los datos, seguridad en los accesos y programación numérica.

Para el autor Aquilino la monitorización la podemos definir como “la representación de datos en tiempo real a los operadores de planta. Se leen los datos de los autómatas (temperatura, velocidades, detectores…)” (Rodríguez Penin, 2007, pág. 11) por otro lado tenemos que el mismo autor define la supervisión de la siguiente manera diferenciándola de la monitorización:

> Supervisión, mando y adquisición de datos de un proceso y herramientas de gestión para la toma de decisiones (mantenimiento predictivo, por ejemplo). Tienen además la capacidad de ejecutar programas que puedan supervisar y modificar el control establecido y, bajo ciertas condiciones, anular o modificar tareas asociadas a los autómatas. Evita una continua supervisión humana. (Rodríguez Penin, 2007, pág. 11)

1.4.2 Normativas

El diseño de los sistemas SCADA al igual que la mayoría de sistemas en el mundo, se rigen a ciertas normas y estándares internacionales que permiten reducir los efectos adversos de un mal diseño, desde el tema de ergonomía hasta el tema de protección de riesgo eléctrico. A continuación, se dará una breve explicación de ciertas normas puntuales que abarquen los conceptos a estandarizar en el proyecto de titulación, que son orientados al diseño del HMI como la parte intangible del sistema SCADA, y el diseño e instalación de máquinas para la evaluación y reducción de riesgos como parte tangible del sistema SCADA.

1.4.2.1 Norma ISA101

Esta norma está orientada al diseño HMI de sistemas SCADA, la revista infoPLC define la norma de la siguiente forma “La norma pretende proporcionar orientación para diseñar, construir, operar y mantener HMI efectivas que resulten más seguras, más eficaces y más eficiente en el control de un proceso, en todas las condiciones de funcionamiento.” (infoPLC, 2015)

1.4.2.2 Norma ISO12100:2010

La siguiente norma nos da las pautas para poder conseguir evaluar los riesgos y reducirlos. La parte evaluación de riesgos se define en la cláusula 5, cláusula que la revista Tecnical de España la resume de la siguiente manera “El fabricante de la máquina debe determinar los límites y el uso previsto de la máquina para poder identificar los peligros y calcular los riesgos a la hora de evaluar los riesgos, debe decidir si éstos se han reducido adecuadamente” (TECNICAL, pág. 4). En cambio, al referirnos a la

reducción de riesgos el mismo autor resume la cláusula 6 de la norma ISA12100:2010 con las siguientes palabras:

> Si los riesgos no se han reducido adecuadamente, debe considerar una reducción de riesgos llevando a cabo los tres pasos siguientes, en el orden establecido: Medidas de diseño inherentemente seguro (cláusula 6.2), Prevención y medidas de protección suplementaria (cláusula 6.3) y por último información útil (cláusula 6.4)

1.4.3 Sistemas de Esclusas

Para los sistemas se esclusas se suelen usar dos tipos de compuertas, las electromecánicas y las compuertas hidráulicas, para este proyecto de titulación se considera el uso de compuertas electromecánicas. El siguiente autor define el sistema de esclusas de la siguiente manera:

> Estructuralmente la esclusa es un canal que comunica los niveles de agua aguas arriba y aguas debajo de una presa, debiendo cumplir dos requisitos opuestos, primero que la esclusa se llene lo más rápidamente posible para no alterar el tráfico de buques y brazas, segundo que el llenado no sea tan rápido como para causar esfuerzos peligrosos en los cables de amarre de las embarcaciones, evitando que éstas choquen contra los muros o entre sí. (S. Escalante & Sivori, 2018, pág. 48)

En la figura 1.2 se puede analizar cómo es el funcionamiento de una esclusa de manera gráfica.

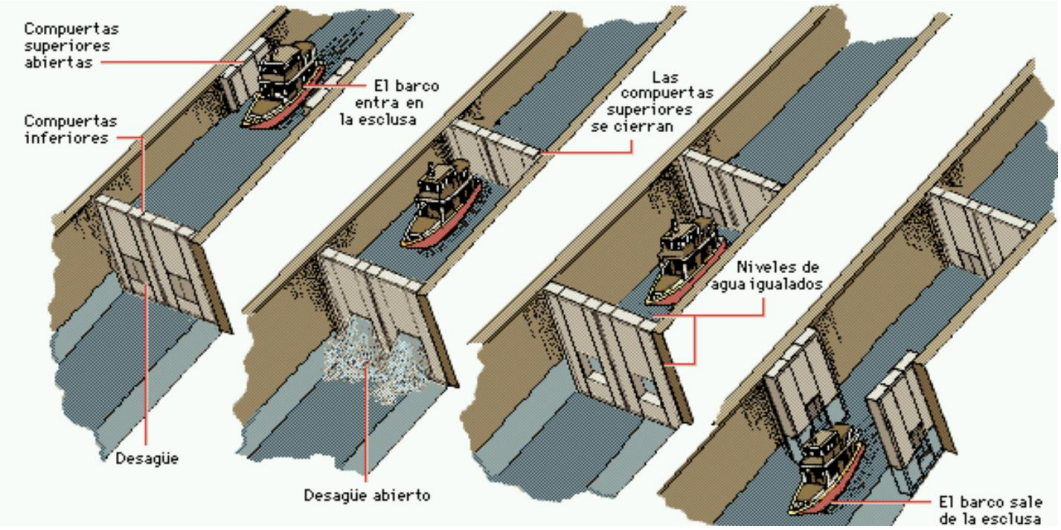


Figura 1.2 Sistema de Esclusas

(S. Escalante & Sivori, 2018, pág. 49)

1.4.4 Industria 4.0

Como parte del proyecto de titulación se manejarán herramientas propias de la industria 4.0, pero para esto debemos entender que es la industria 4.0. La CEPAL da una explicación de cuando surge la industria 4.0 y que es específicamente la llamada industria 4.0, explica lo siguiente:

> La denominada Industria 4.0 surge en los países desarrollados en la segunda década de los años 2000 como respuesta de política industrial frente a una nueva fase en la revolución de las tecnologías de la información y comunicación. El tipo de proceso productivo que se desarrolla y expande desde entonces se basa en los llamados sistemas “ciberfísicos”, en los cuales los procesos de producción (sean físicos o biológicos) son controlados o monitoreados por algoritmos integrados a Internet. Los nuevos sistemas ciberfísicos se apoyan en la modelización digital de los procesos de producción y en el intercambio de datos generados en el propio proceso de fabricación. Este intercambio puede darse entre productos y máquinas (por ejemplo, mediante sensores que facilitan la trazabilidad de la producción), entre distintas máquinas (por ejemplo, entre máquinas y robots que operan en proximidad) o entre diferentes actores de la cadena de producción (por ejemplo, entre las empresas y sus clientes mediante plataformas de gestión de compras y entregas que facilitan el control de inventarios en tiempo real y la gestión de abastecimiento). (Erbes, Gutman, Lavarello, & Verónica, 2019, pág. 7)

1.4.4.1 IoT

Como parte de la industria 4.0 se encuentra el IIoT, pero antes de entender su definición es necesario entender que es el IoT o Internet de las Cosas y como se usó este concepto a la industria. Para Moisés Barrio el Internet de las Cosas (IoT) se define de la siguiente manera:

> El «Internet de las Cosas» (Internet of Things, habitualmente referido por sus siglas inglesas IoT), también denominado por algunos «Internet de los Objetos» (IO) es un supra concepto que caracteriza la próxima gran transformación en la evolución de Internet1: su expansión más allá de la comunicación entre las personas, o entre las personas y el contenido digital, que ahora se extiende a miles de millones de objetos cotidianos. Los sistemas IoT implican la adquisición de datos de sensores y la entrega de órdenes a dispositivos que interactúan o forman parte del mundo real. También reconocen eventos y cambios, y pueden reaccionar de forma autónoma y apropiada. El Internet de las Cosas se equipará a menudo con electrodomésticos y bienes de consumo, como las ropas tecnológicas (wearables) o los coches inteligentes. Por lo tanto, muchas de las preocupaciones iniciales se han centrado en los productos de consumo. (Moisés Barrio, 2018, pág. 13)

Para otros autores el Internet de las Cosas es una “compleja red que conecta millones de dispositivos y personas en una infraestructura de multi.protocolo y multi-plataforma, la visión principal” (Liñán Colina, Vives, Bagula, Zennaro, & Pietrosemoli, 2015, pág. 2) además la visión principal del Internet de las cosas es la “creación de un mundo inteligente donde lo real, lo digital y lo virtual converjan para crear un entorno inteligente que proporcione más inteligencia a la energía, la salud, el transporte, las ciudades, la industria, los edificios y muchas otras áreas” (Liñán Colina, Vives, Bagula, Zennaro, & Pietrosemoli, 2015, pág. 2)

La arquitectura de capas del IoT se puede entender fácilmente con la siguiente imagen que se muestran en la figura 1.3

Figura 1.3 Arquitectura de Capas de IoT

(Benítez Machado, Anías Calderón, & Plasencia Moreno, 2016)

Por lo tanto, al analizar las distintas definiciones de los autores mencionados, se llega a la conclusión de que el internet de las cosas une los dos mundos que rigen nuestra vida, el real y el digital, para poder facilitar la vida de las personas.

1.4.4.2 IIoT

Por otro lado, tenemos que el IIoT o Industrial Internet de las Cosas es implementar los conocimientos de la sección 1.4.3.1 pero implementados en la industria, entendiendo que para que sea aplicado en la industria existen normas como las que ya fueron detalladas en la sección 1.4.2.

Para que este concepto sea posible se necesita una red de sensores inalámbricos o WSN que los autores lo definen como:

> Red que se auto-configura, formada de pequeños nodos sensores (llamados motas: mote en inglés denota el pequeño tamaño, como el de una mota de polvo: N del T) que se comunican entre ellos por señales de radio, y desplegados en gran cantidad para percibir el mundo físico. Los nodos sensores son básicamente pequeños computadores con funciones extremadamente básicas. Consisten de una unidad de procesamiento con capacidad de computación restringida, una memoria limitada, un dispositivo de comunicación de radio, una fuente de

alimentación, y uno o más sensores. (Liñán Colina, Vives, Bagula, Zennaro, & Pietrosemoli, 2015, pág. 4)

1.4.5 Tarjeta TSC LAB

La tarjeta TSC LAB es una tarjeta de código abierto. Cumple la función de un mini laboratorio de prueba, esta tarjeta posee sensores de temperatura, lector óptico, resistencias de calor y un motor DC. Para más información revisar la fuente bibliográfica, su procesador se encuentra en la tarjeta ESP-32 embebida en la tarjeta TSC LAB (Asanza & Chica, 2021)

1.4.6 Modelo Matemático

El modelo matemático puede definirse como “una metodología mediante la cual se generan modelos matemáticos para tratar de predecir el comportamiento de un sistema real. Dependiendo del tipo de modelo matemático que se formule, se requieren tipos de análisis particular.” (Cardona, Leal, & Ustariz, 2020, pág. 111)

Referente al modelado matemático se puede decir lo siguiente:

> La metodología de modelado utilizando modelos matemáticos de caja blanca y negra tienen los siguientes aspectos coincidentes: Permiten obtener soluciones aproximadas para hacer predicciones sobre el funcionamiento de los sistemas, se utilizan técnicas estadísticas para la validación de los resultados y es necesario utilizar software. En los modelos de caja negra no es necesario conocer cómo funciona exactamente un sistema, a diferencia de los modelos de caja blanca, en los cuales es fundamental la comprensión de las leyes o principios físicos que rigen el comportamiento y la estructuración del mismo. Otra diferencia esencial, es el conocimiento matemático que se requiere para resolver cada tipo de modelo, puesto que en los modelos de caja blanca resultan ecuaciones mucho más complejas que en los modelos de caja negra. (Cardona, Leal, & Ustariz, 2020, pág. 111)

El modelo Box & Jenkin es un modelo matemático que se define como:

> Construye un modelo de una serie temporal, para explicar su estructura y predecir la evolución de esta serie en el futuro. Una serie temporal se puede considerar como un conjunto de observaciones (datos), de una variable, tomados en intervalos regulares de tiempo. En particular, la metodología Box & Jenkins es un procedimiento de análisis estadístico para ajustar a una serie un tipo especial de modelos, denominados ARIMA (Autorregresive Integrated Moving Average). (Fournies, 2015, pág. 4)

CAPÍTULO 2

2. DISEÑO METODOLÓGICO

2.1 Diseño Conceptual de Sistema de Esclusas

El presente proyecto de titulación tiene como objetivo el diseño y desarrollo de un sistema de automatización de un sistema de esclusas; considerando que existen diversos tipos de sistemas de esclusas que poseen a su vez diversos tipos de tecnologías como electromecánica, hidráulica, neumática, se decide realizar un diseño conceptual que aplique a un tipo de sistema de esclusa en particular. Se toma de referencia el sistema de esclusas que une el río Guayas con el estero salado ya que este tipo de esclusa es de los más usados mundialmente, usando 4 compuertas y tecnología electromecánica, todo esto con la finalidad de poder obtener una arquitectura conceptual que luego permita llegar a la ingeniería al detalle. El proyecto de titulación podría además ser implementado en las esclusas de Guayaquil.

Para una mayor comprensión del sistema se divide en dos partes, la primera parte representa la arquitectura del modo manual de las compuertas de forma individual como se indica en la figura 2.2 del Plano 2 (véase apéndice B) y la segunda parte representa la arquitectura en modo automático de todo el recinto de compuertas de la esclusa como se indica en la figura 2.1 del Plano 1 (véase apéndice A). El desarrollo del prototipo es basado en el modo automático.

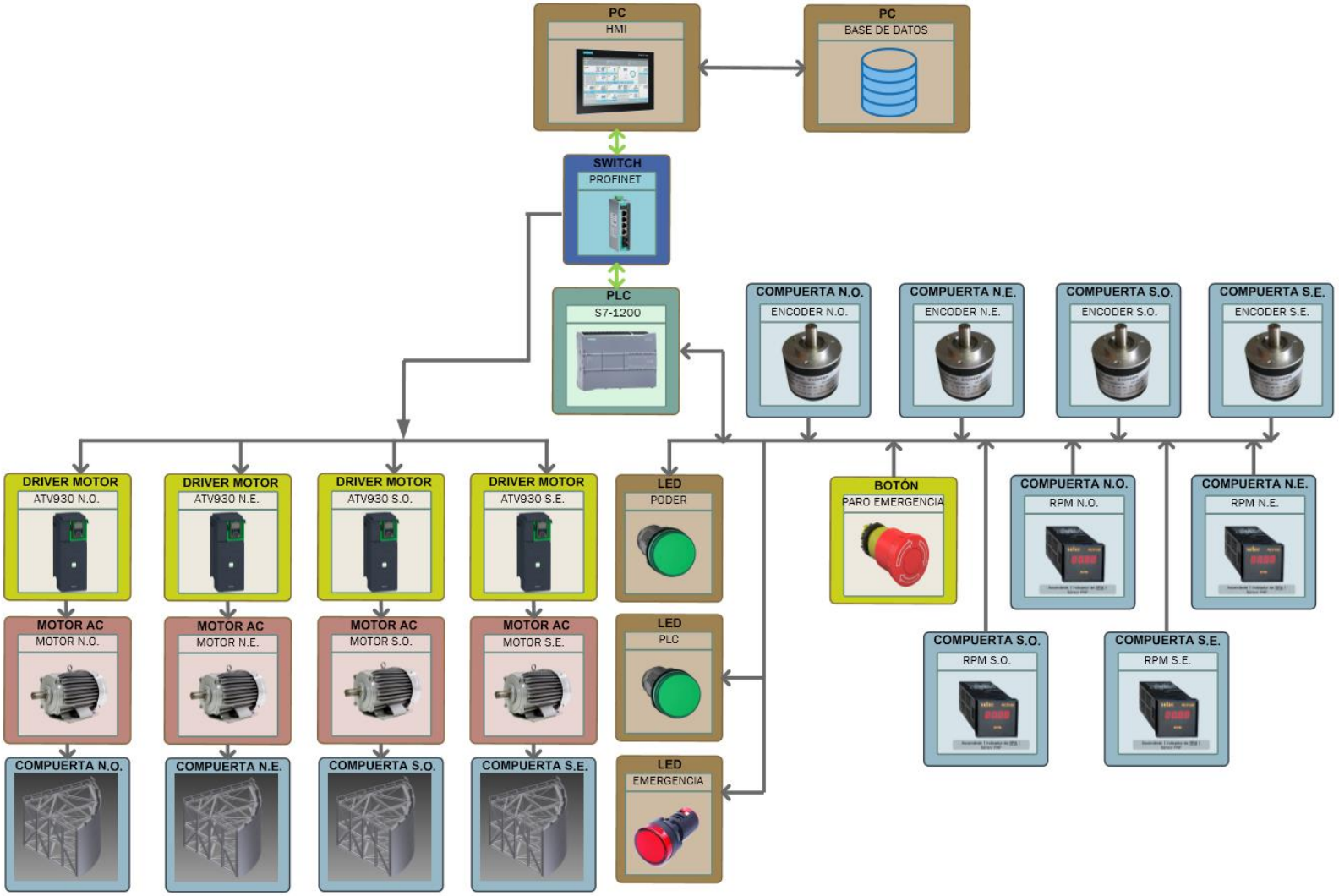


Figura 2.1 Diseño conceptual de esclusas modo automático

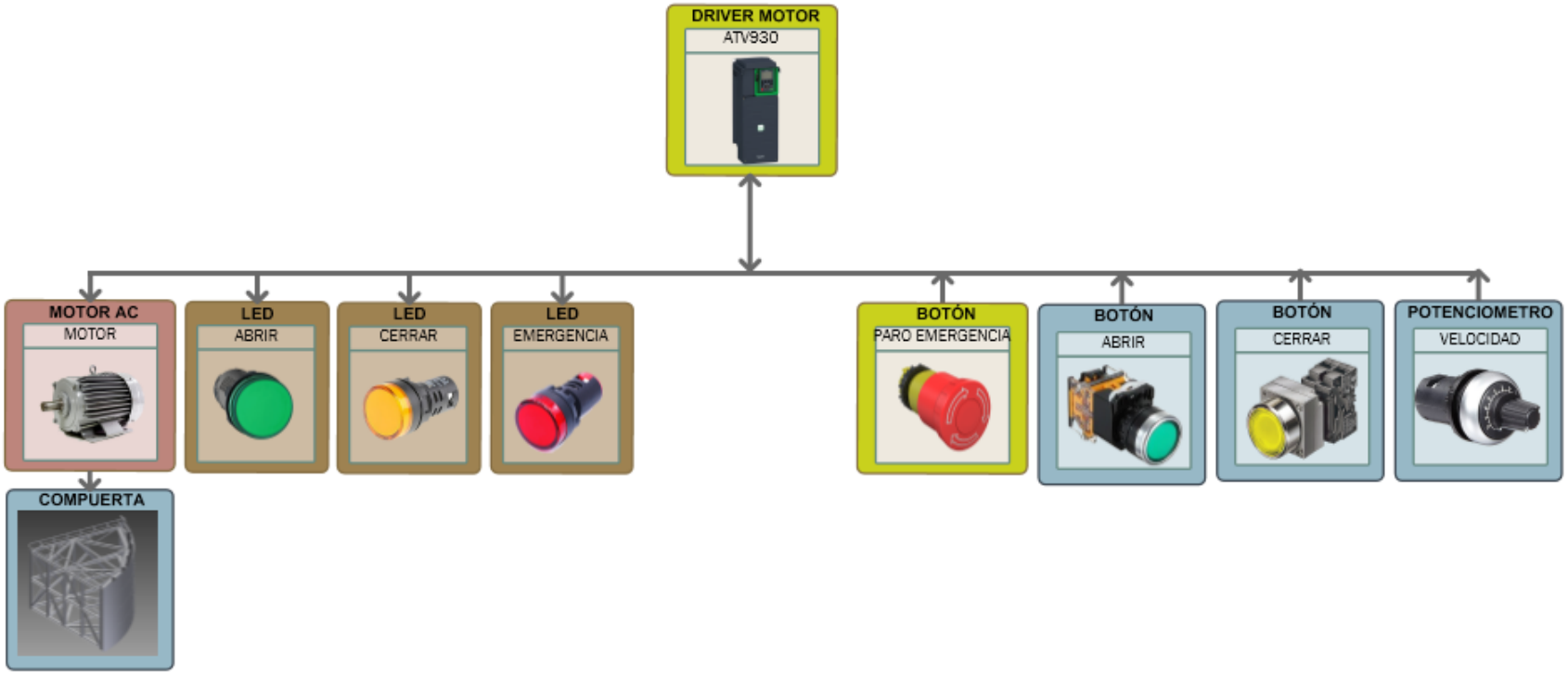


Figura 2.2 Diseño conceptual de esclusas modo manual por Compuerta

Una vez que se obtiene la arquitectura conceptual para el modo manual de cada compuerta individual y el modo automático de todo el recinto de compuertas se procede a definir la ubicación de los sensores, actuadores, PLC y HMI como se indica en la figura 2.3 en donde se puede observar 4 casetas, cada compuerta tiene una caseta en donde se encuentran los sensores, actuadores y el controlador del motor, mientras que por otro lado existe una caseta principal donde se encuentra el panel PC, PLC y el switch ethernet.

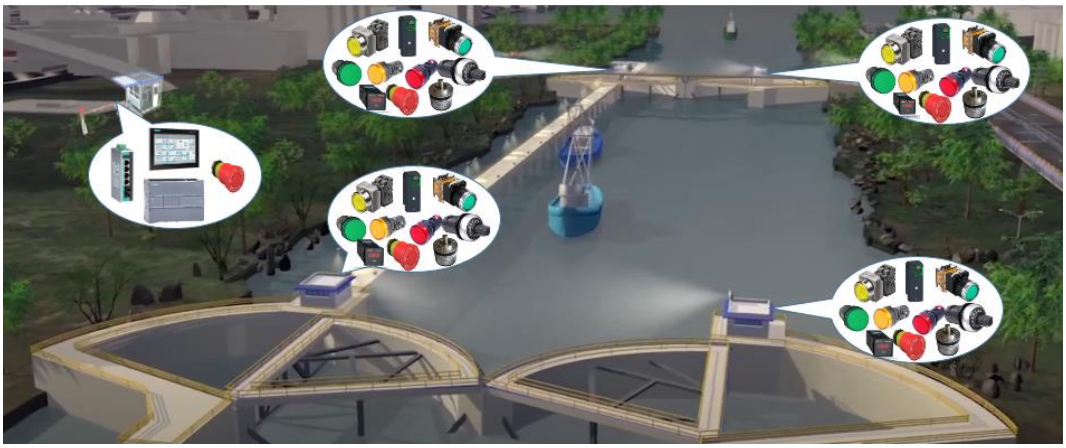


Figura 2.3 Ubicación de sensores y actuadores

Con la arquitectura conceptual ya se puede definir en qué nivel de automatización industrial se encuentra, en este caso en particular está en el nivel 3 que es de supervisión, basado en los cinco niveles de la automatización industrial como se muestra en la figura 2.4

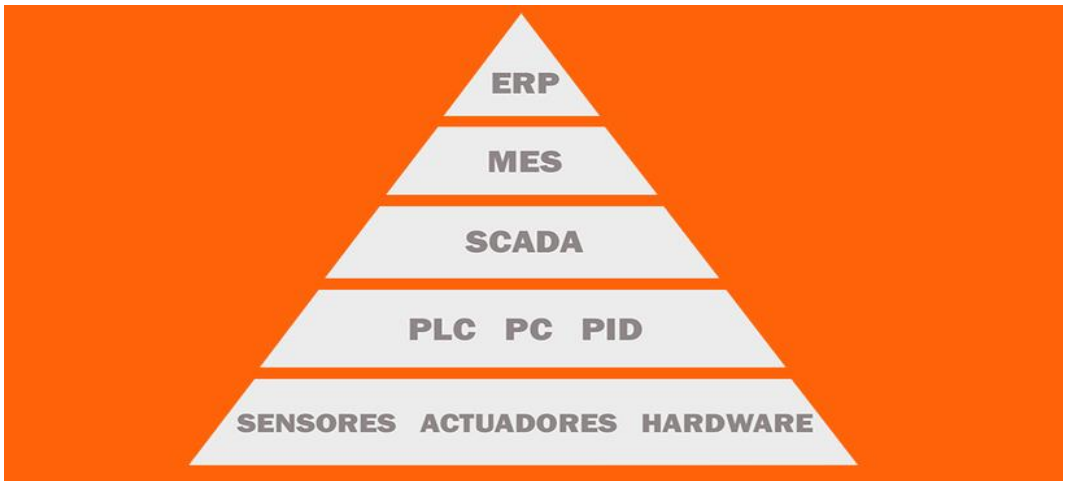


Figura 2.4 Niveles de la automatización industrial

(SEIKA, 2019)

2.1.1 Nivel Operativo o de Campo

Para las 5 casetas del sistema de esclusas, 4 de las compuertas y 1 de la garita principal se tiene en este nivel sensores y actuadores propios del sistema de esclusa que se detallan a continuación:

Actuadores:

- 4 motores
- 15 leds de aviso

Sensores:

- 4 tacómetros
- 4 encoders absolutos
- 5 botón de emergencia
- 4 potenciómetros de velocidad

Cada motor mueve una compuerta en específico mediante el uso de una caja reductora como se muestra en la figura 2.5 que permite maximizar el torque a cambio de una velocidad de movimiento lenta, muy práctico cuando se trata de movimientos de objetos de gran tamaño, así se tiene mayor maniobrabilidad en caso de errores. La relación de las vueltas del motor con los grados de la compuerta es de "**1 vuelta motor / 0.0296 grados compuerta".** El motor a utilizarse debe girar a una velocidad máxima de 1200 rpm por limitaciones mecánicas, el motor instalado en las esclusas es de 1180 rpm, usar más velocidad de rotación del motor podría ocasionar daños en el sistema de engranajes de la caja reductora.

Figura 2.5 Sistema mecánico de movimiento

En la vida real este tipo de sistema es muy usado para la apertura y cierre de compuertas, de puentes, etc. Un ejemplo real es el sistema de apertura y cierra de las compuertas de las esclusas de Guayaquil como se puede ver en la figura 2.6.

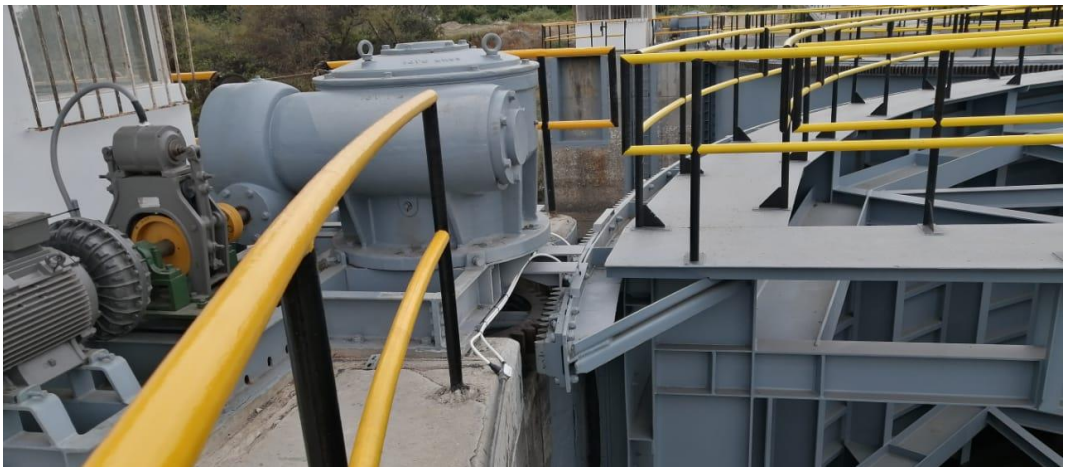


Figura 2.6 Caja reductora esclusa de Guayaquil

Los 15 leds tienen 5 finalidades, 2 leds son de color verde ubicados en la garita principal indican que existe poder y el otro que el PLC está operativo, 4 leds de color verde ubicados en las 4 casetas de compuertas indican que se está abriendo la compuerta respectivamente, 4 leds de color amarillo ubicados en las 4 casetas de compuertas indican que se está cerrando la compuerta respectivamente, mientras que 5 leds son de color rojo ubicados en las 5 casetas e indica que existe alguna alarma activada debido alguna anomalía en el proceso.

Por otro lado, tenemos los sensores, dentro de los cuales se mencionan 4 tacómetros que indican la velocidad en rpm o rps de los 4 motores respectivamente, 4 encoders absolutos que indican la posición de las compuertas y 5 botones de emergencia ubicados en las 5 casetas respectivamente para paralizar el proceso cuando es activado.

2.1.2 Nivel de Control:

En el nivel de la arquitectura se encuentran los siguientes controladores:

- 1 PLC
- 4 controladores de motor (Variadores de frecuencia)

El PLC es el encargado del control total de todo el sistema de esclusas, se encuentra en la garita principal, este a su vez se conecta con los 4 controladores de motor ubicados en las 4 casetas respectivamente, mediante protocolos de comunicación industrial conocidos por donde se envía la frecuencia de velocidad del movimiento del motor para un control de la velocidad mediante rpm, con la finalidad de obtener precisión a la hora de abrir o cerrar las compuertas.

2.1.3 Nivel de Supervisión:

En el nivel de supervisión se encuentra lo siguiente:

- HMI
- Base de datos

EL HMI sirve para la visualización del estado de nuestros sensores, para activación de procesos, visualización de alarmas mientras que la Base de Datos permite el almacenamiento de eventos y alarmas configuradas desde el HMI. Todo el nivel de supervisión se encuentra solo en la garita principal.

2.2 Diseño Conceptual de Prototipo

El proyecto de titulación se centra específicamente en el diseño de un prototipo para el modo automático de todo el recinto de compuertas de las esclusas, por lo tanto, la

ingeniería desarrollada en las siguientes secciones del capítulo 2 se fundamentan de la arquitectura mostrada en la figura 2.1. Debido a que no se posee un sistema de esclusas con el cual poder realizar el testeo real, se procede a realizar una nueva arquitectura para el prototipo, cambiando la parte de los sensores y actuadores. Para el prototipo se utiliza componentes académicos como se muestra en la figura 2.7 del PLANO 3 (véase apéndice C). Además, al no poseer encoders absolutos para el prototipo se decide calcular la posición en grados de la compuerta mediante la relación de 1 vuelta del motor equivale a 0.0296 grados de movimiento de la compuerta obteniendo cada vuelta con un lector óptico conectado a una interrupción del MCU de la tarjeta.

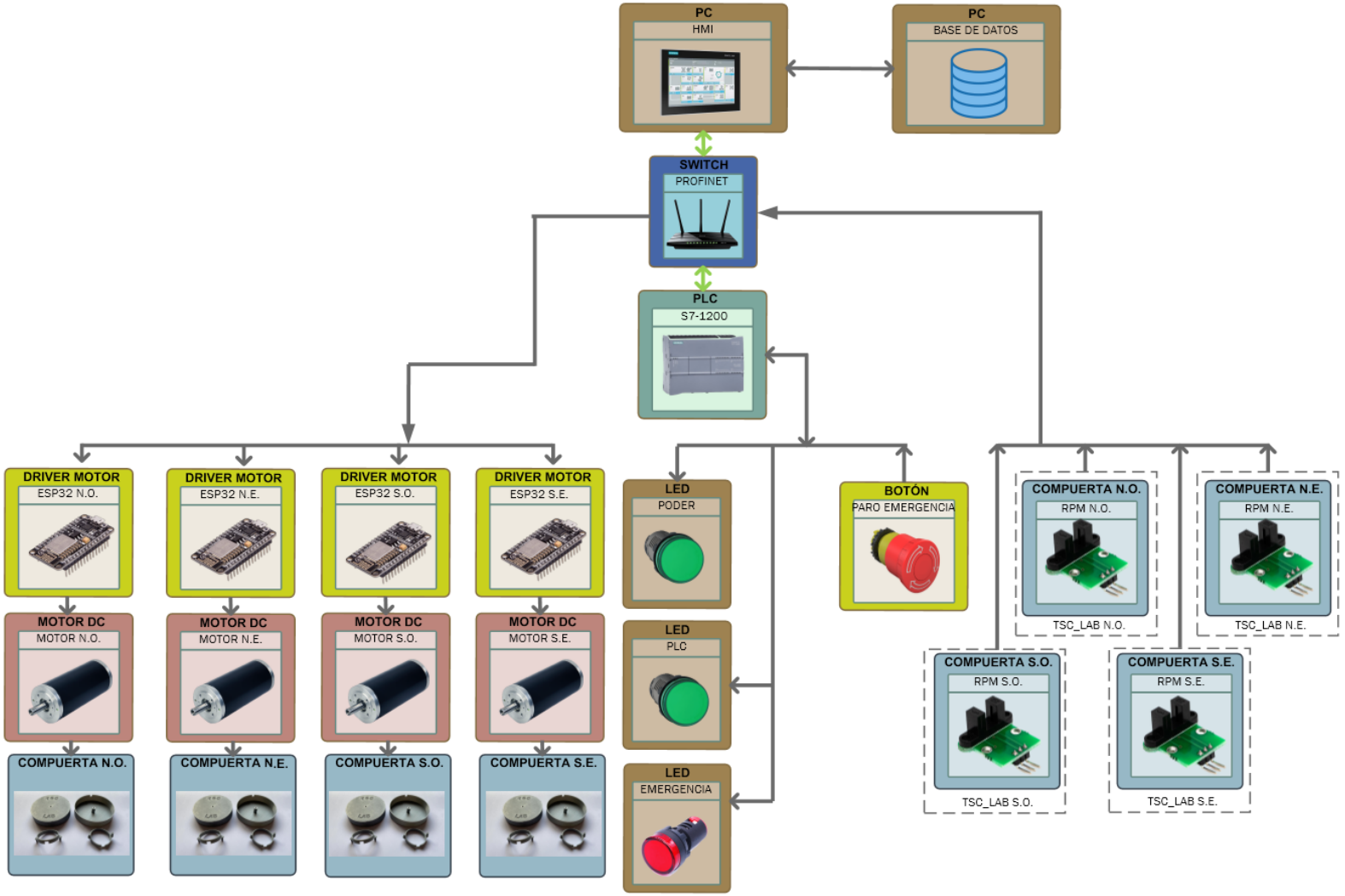


Figura 2.7 Arquitectura conceptual de prototipo

Como se puede observar en la figura 2.7, se cambia componentes como el switch industrial por un switch comercial, el tacómetro industrial se cambia por un tacómetro académico, motores VAC de gran potencia por motores VDC de menor potencia, pero de igual rpm, los controladores de motores se cambian por sistemas embebidos en los cuales se programa un controlador PID para simular un controlador industrial real.

Tanto el sistema embebido que sirve como controlador, el motor y el tacómetro se encuentran integrados en una tarjeta de prueba académica llamada TSC_LAB, en la figura 2.8 se muestra la tarjeta.

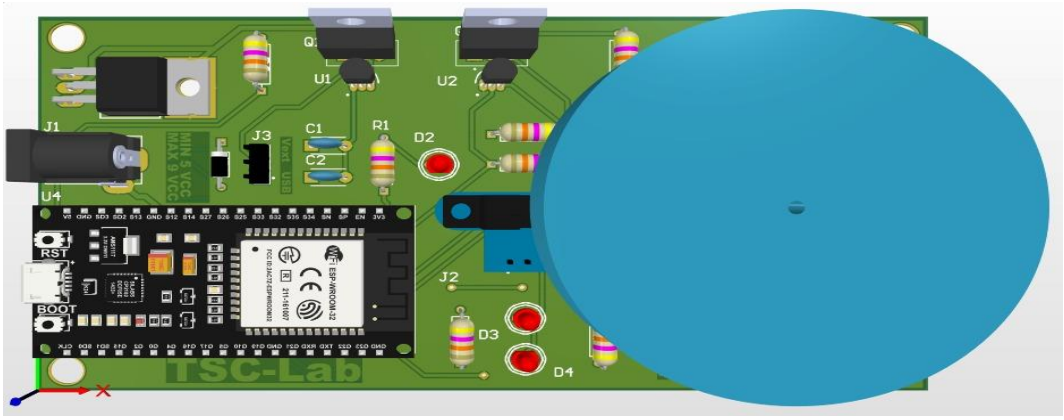


Figura 2.8 Tarjeta TSC_LAB

(Asanza & Chica, 2021)

Sobre el motor se procede a instalar un objeto con la finalidad de agregar peso consiguiendo generar perturbaciones que asemejen a las perturbaciones ambientales que están presentes en los recintos de esclusas y que influyan en la apertura y cierre de las compuertas, dicho de otra manera, conseguir afectar la velocidad de los motores para ver como actúa el controlador PID ante eventos no deseados.

2.3 Diseño Básico de Prototipo

Se desarrolla una ingeniería básica con la finalidad de definir de una forma precisa la cantidad de dispositivos y componentes a usarse, el tipo de poder a usar para cada dispositivo, además de los tipos de comunicaciones que se usan para la implementación, para una mejor comprensión se usan acrónimos que están presente en el PLANO 3 y que luego se usarán en la ingeniería al detalle en la sección 2.4. En la figura 2.9 del PLANO 4 (véase apéndice D) se observa la arquitectura básica del prototipo.

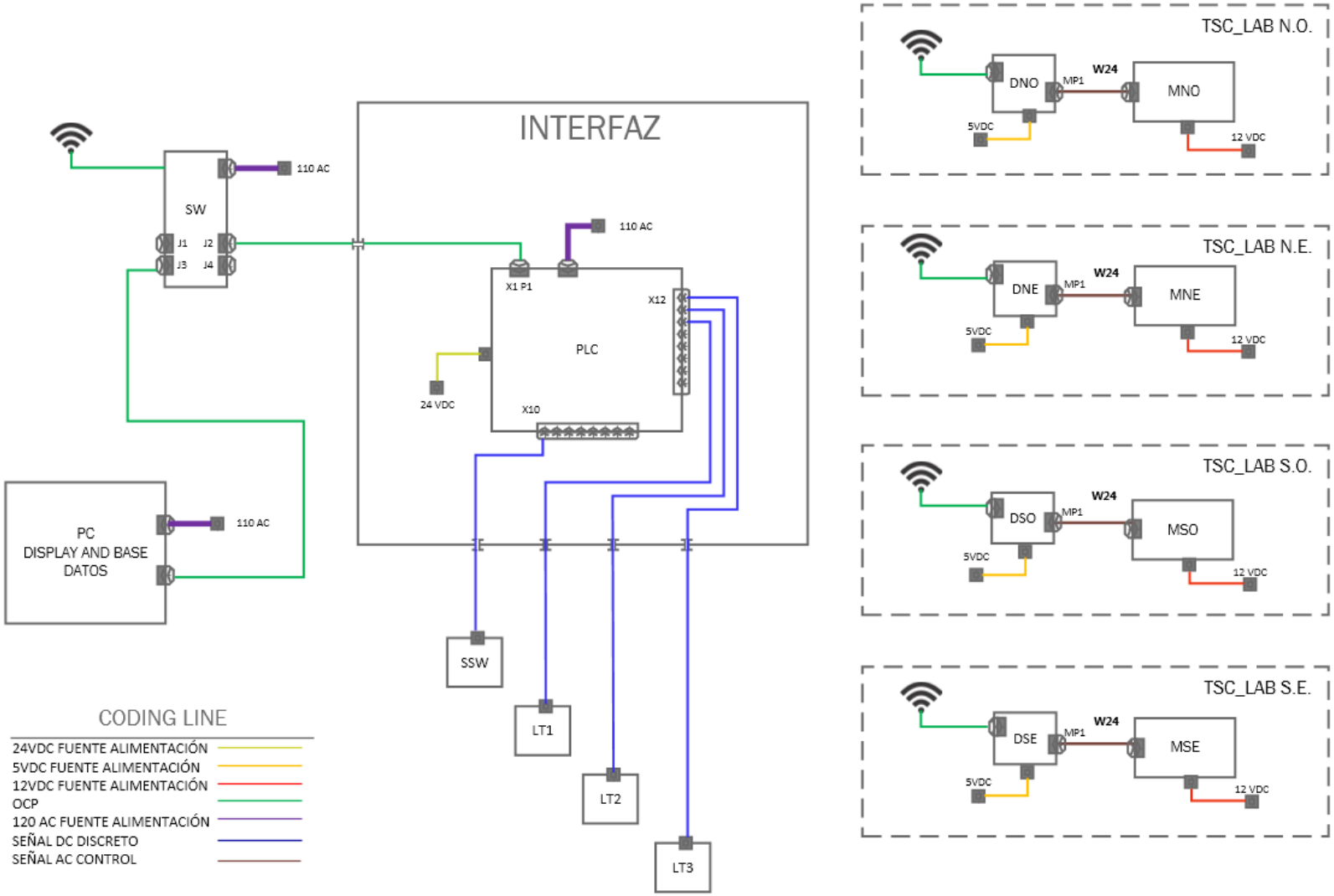


Figura 2.9 Arquitectura básica de prototipo

Se diseña una interfaz en donde se encuentra el PLC, en esta interfaz además se encuentra un bloque de borneras que distribuyen las señales y poderes a los distintos dispositivos, sensores y actuadores de la arquitectura básica.

Como se muestra en la figura 2.10 el switch tiene al menos dos puertos ethernet libres para poder usarse, en este caso un puerto se usa para conectar el PLC y el otro para conectar el Panel PC o PC, en este caso se trabaja con una PC de escritorio. El switch funciona a 110 VAC. Además, este switch tiene conexión wifi para el acceso de las tarjetas TSC LAB.

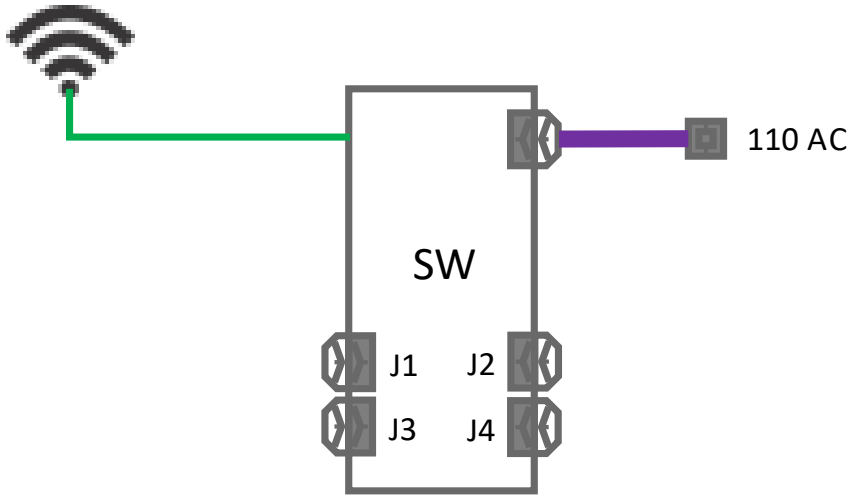


Figura 2.10 Switch

Por otro lado, la interfaz se conecta con un botón de parada (SSW), una luz verde de aviso de poder (LT1), una luz verde de aviso de PLC operativo (LT2) y una luz roja de alarma (LT3), el SSW se conecta al puerto X10 de entradas del PLC, mientras que las 3 luces se conectan al puerto X12 del PLC como se muestra en la figura 2.11. El PLC se energiza con 110 VAC y también recibe 24 VDC para energizar las luces.

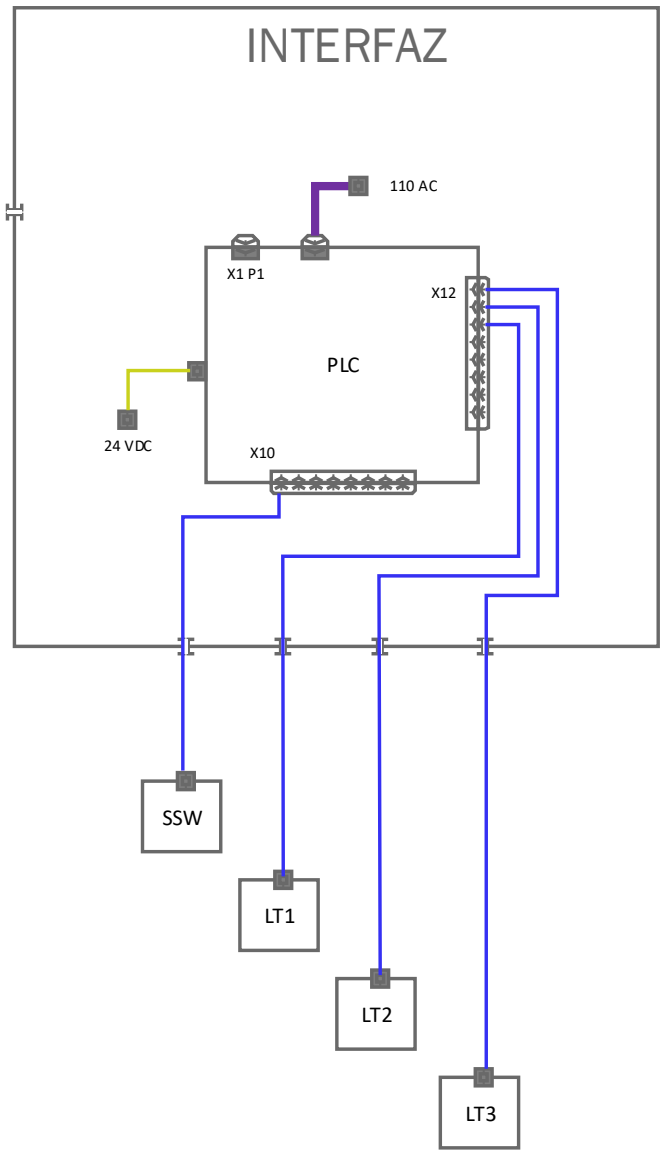


Figura 2.11 Interfaz, luces y botón emergencia

Tenemos también las tarjetas TSC LAB; cada tarjeta representa el controlador del motor, el motor y el tacómetro. Para el caso de la figura 2.12 se indica la compuerta Noroeste. Se alimenta de 12 VDC, que distribuye al motor y a un convertidor de 12VDC a 5VDC para poder energizar el sistema embebido, en este caso la tarjeta ESP 32

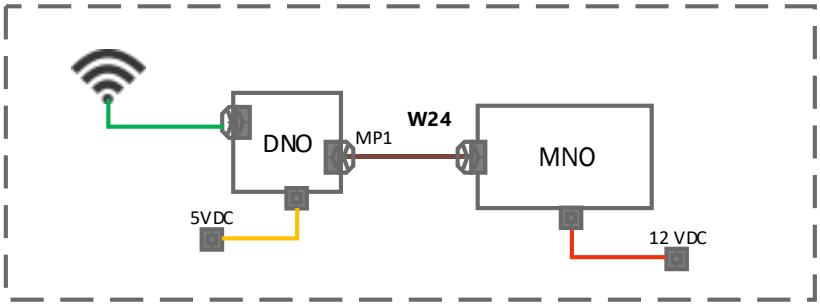


Figura 2.12 Motor, controlador y tacómetro de compuerta Noroeste

Y por último tenemos la PC que sirve para alojar el HMI y la base de datos, se detalla en la figura 2.13 del PLANO 3. Se alimenta de 110 VAC.

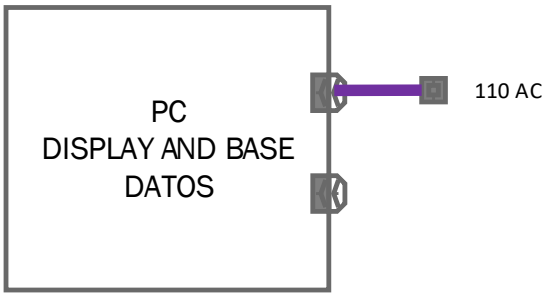


Figura 2.13 Panel PC

2.4 Diseño al Detalle

2.4.1 HARWARE

En la sección 2.2 se desarrolla la ingeniería conceptual mientras que en la sección 2.3 desarrolla una ingeniería básica que ayuda a la definición de componentes, comunicaciones y poderes, ahora en la sección 2.4 se procede a realizar la ingeniería al detalle en donde se explica con claridad como irá interconectado cada componente del prototipo del sistema de esclusas, de forma interna como se muestra en la figura 2.14 (véase apéndice E) y de forma externa como se muestra en la figura 2.15 del PLANO 4 (véase apéndice F).

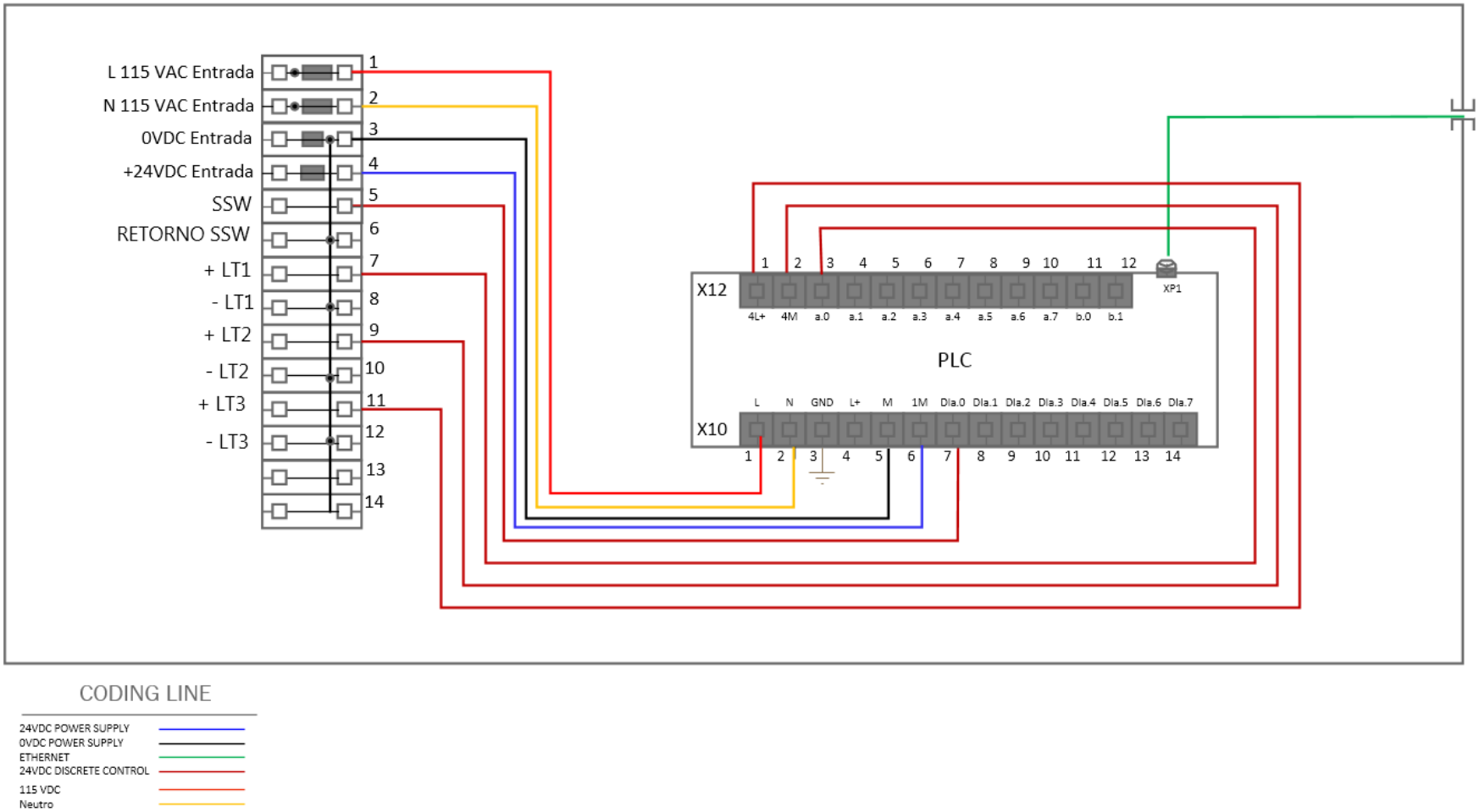


Figura 2.14 Ingeniería al detalle interior

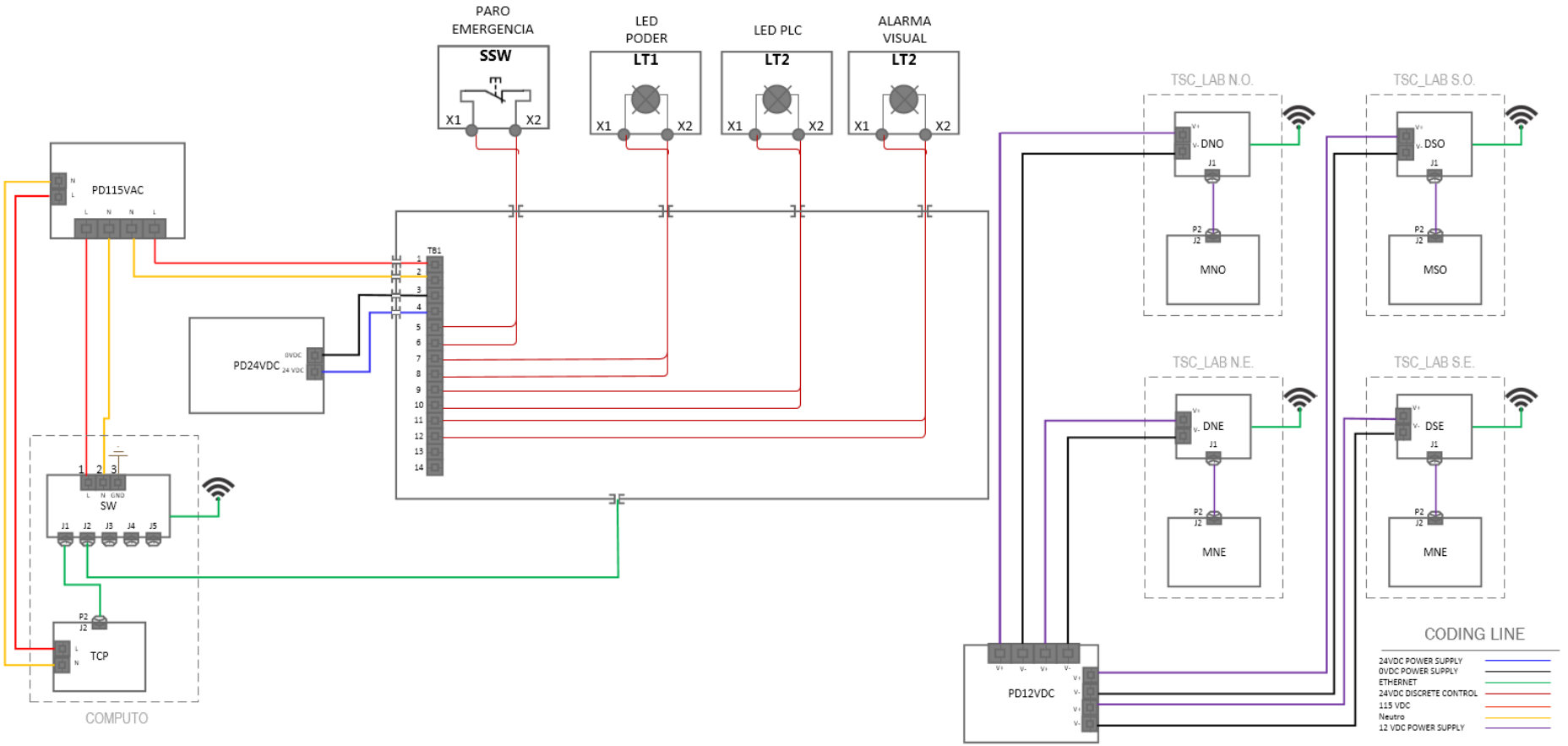


Figura 2.15 Ingeniería al detalle exterior

Para el diseño al detalle es necesario la selección de marca y modelo de equipo, ya que es necesario para definir en qué conector, terminal o bornera van a conectarse cada uno de los componentes. Por lo tanto, la lista de materiales se detalla en la tabla 2.1.

Tabla 2.1 Lista de materiales

Nº	Ítem	Cantidad	Marca	Modelo	Características
1	PLC	1	SIEMENS	S7-1200	CPU 1212C, 110 VAC, Relés
2	Switch	1	CALEX	AAA	Switch Router Wifi, 110 VAC
3	Led rojo	1	N.A.	N.A.	Led rojo de diodo para PC de 5 VDC
4	Led verde	2	N.A.	N.A.	Led verde de diodo para PC de 5 VDC
5	PC	1	ASRock	ASRock Gamer	PC de escritorio
6	Sistema Embebido	1	ESP	ESP-32	Integrado en la TSC_LAB
7	Motor VDC	1	N.A.	N.A.	Integrado en la TSC_LAB
8	Tacómetro	1	N.A.	N.A.	Integrado en la TSC_LAB
9	Paro Emergencia	1	N.A.	N.A.	Pulsador genérico
10	Bornera Simple	10	Legrand	37160	Bornera paso 5, 2,5 mm2, 1P
11	Bornera de tierra	2	Legrand	37171	Bornera para tierra, paso 6, 4 mm2, 1P
12	RIEL DIN	1	N.A.	N.A.	Riel de soporte de bornera de 35mmx10cm

2.4.2 SOFTWARE

Como parte del desarrollo se tuvo que realizar el diseño y configuración de la parte intangible del proyecto de titulación que es el software, el software fue fundamental para poder culminar la implementación de todo el diseño.

La arquitectura del software es donde se plasma los tipos de protocolos a usarse entre las comunicaciones, para el proyecto de titulación en mención se usa diversos tipos de comunicaciones, entre industriales y comerciales, eso sí, manteniendo los niveles de seguridad estandarizados. En la figura 2.16 del PLANO 5 (véase apéndice G) se puede observar la arquitectura del software junto con la explicación de los distintos protocolos de comunicación y las distintas configuraciones de IP. Se requiere programar en 5 lenguajes de programación distintos para culminar el software del proyecto de titulación, dichos lenguajes de programación son lenguaje C, M, Javascript, Ladder, Basic Script.

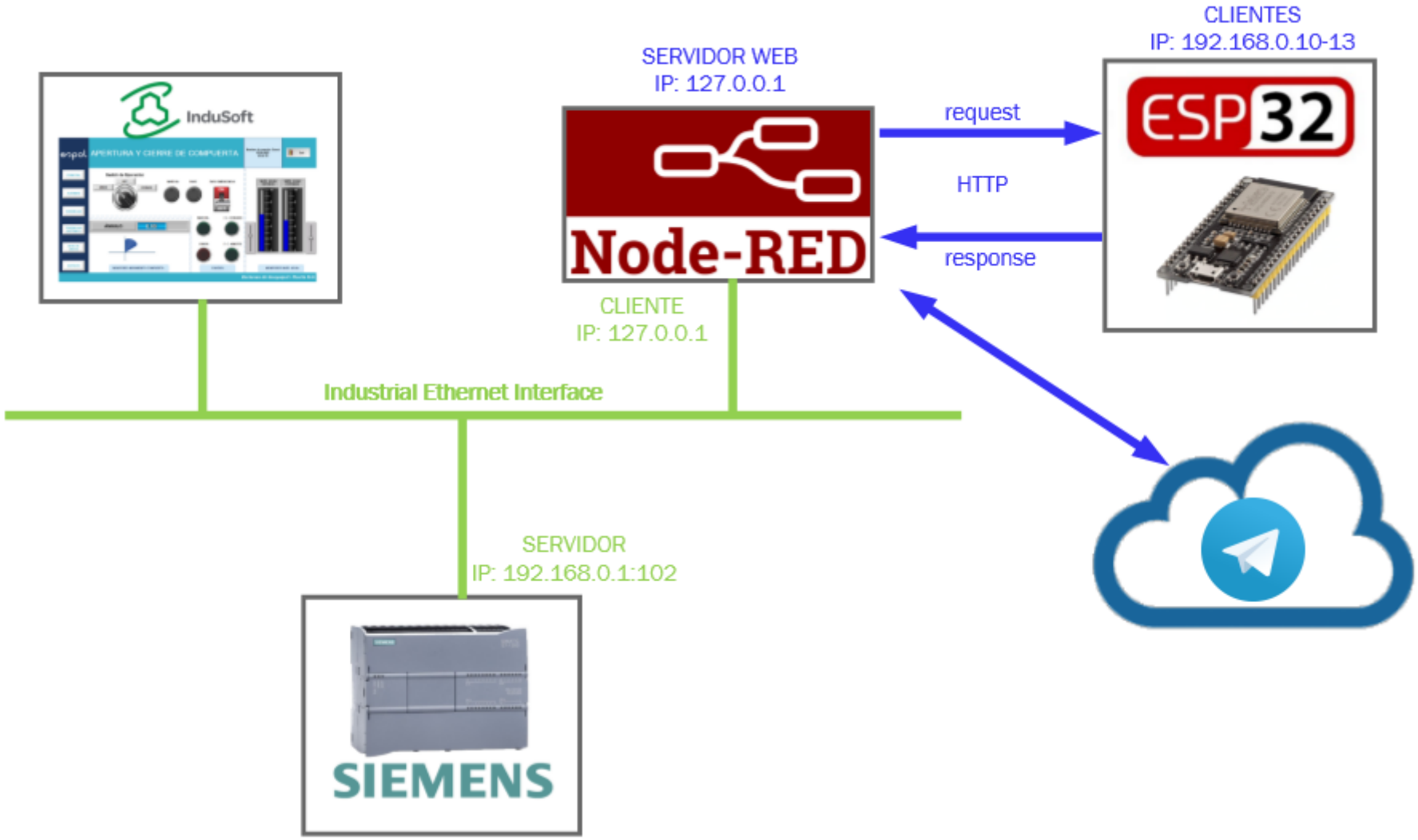


Figura 2.16 Arquitectura de Software

La arquitectura de software trabaja en un protocolo de comunicación por medio de Ethernet Industrial. La variante para el prototipo es que el controlador del motor, en este caso las tarjetas TSC LAB, no tienen posibilidad de conectarse directamente a un protocolo industrial por lo que es necesario usar una interface como NODE RED para convertir el protocolo HTTP a Ethernet Industrial. En la tabla 2.2 se detalla la configuración de las IP.

Tabla 2.2 Lista de IP

Nº	Ítem	IP	Comunicación
1	HMI INDUSOFT	192.168.0,4	Industrial ETH
2	SIEMENS S7 1200	192.168.0.1	Industrial ETH
3	ESP 32 N.O.	192.168.0.10	HTTP
4	ESP 32 N.E.	192.168.0.11	HTTP
5	ESP 32 S.O.	192.168.0.12	HTTP
6	ESP 32 S.E.	192.168.0.13	HTTP
7	NODE RED	127.0.0.1	HTTP
8	NODE RED	127.0.0.1	Industrial ETH

Por lo tanto, la arquitectura de software se divide en tres comunicaciones, la primera indicada en la figura 2.17 muestra la comunicación del HMI de Indusoft con el PLC S7-1200 y el NODE RED; siendo el servidor el PLC S7 1200 y los dos clientes el HMI y el NODE RED.

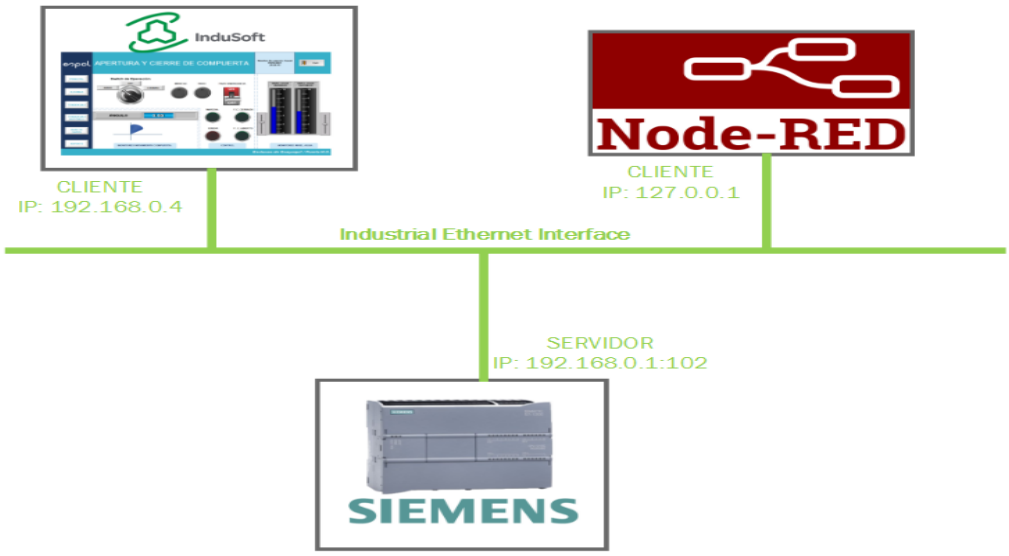


Figura 2.17 Comunicación Ethernet Industrial

La segunda comunicación está desarrollada en protocolo HTTP y ayuda a comunicar entre el NODE RED y las Tarjetas TSC_LAB como se indica en la figura 2.18.

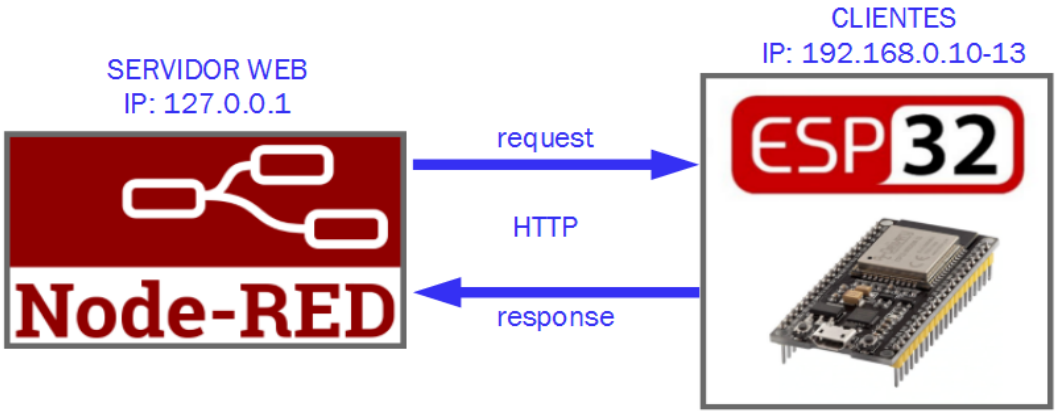


Figura 2.18 Comunicación HTTP

La tercera comunicación es entre NODERED y Telegram como se indica en la figura 2.19

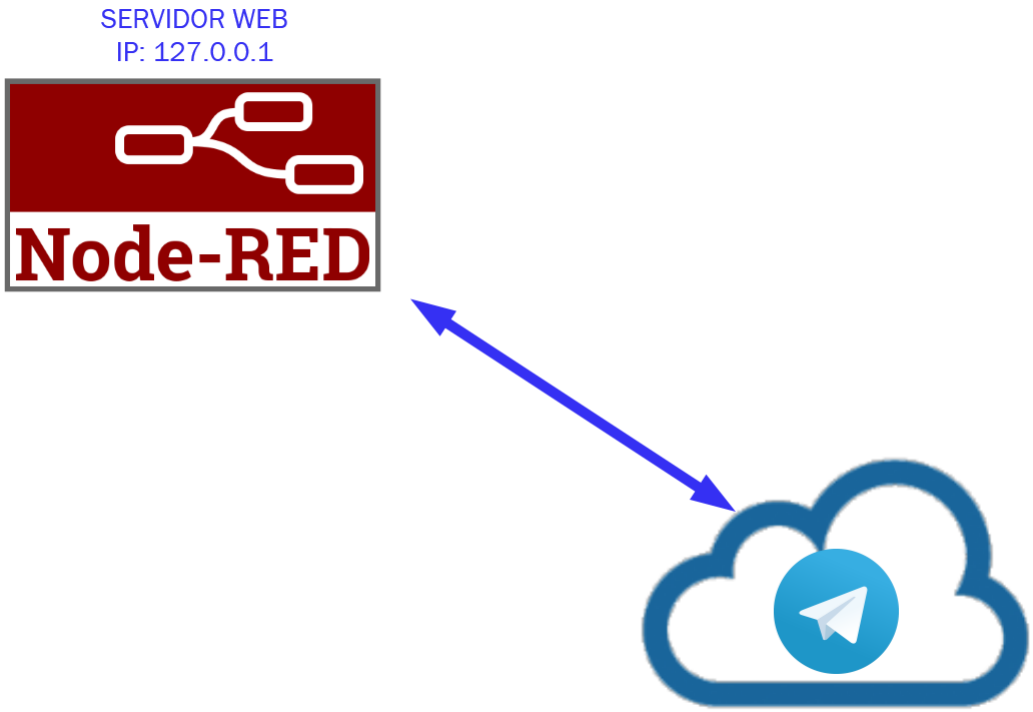


Figura 2.19 Comunicación TCP/IP

En las siguientes subsecciones se explica a detalle el software del HMI, PLC, ESP-32 y el NODE-RED.

2.4.2.1 HMI

El desarrollo del HMI se lo realiza usando el software INDUSOFT WEB STUDIO 8.1 con licencia estudiantil, en el cual se muestra una pantalla principal para indicar el estado de las 4 compuertas, los grados de apertura de cada una como se indica en la figura 2.20.

Figura 2.20 Pantalla Principal HMI

Se tiene la opción de mandar abrir o cerrar las compuertas siempre en pares, ya sea las 2 compuertas del este o las 2 compuertas del oeste. En la parte izquierda se tiene 5 botones, el primero lleva a la pantalla principal mostrada en la figura 2.20, el segundo botón es de alarmas y enseña las alarmas que se activan cuando se presiona el botón de emergencia como se indica en la figura 2.21.

Figura 2.21 Pantalla de Alarmas HMI

El tercer botón muestra tendencias de la información recibida de los ángulos de las compuertas, eso significa que se verá el cambio de ángulos ir de 0 grados a 90 grados cuando se abre y luego ir de 90 grados a 0 grados cuando se cierra como se indica en la figura 2.22.

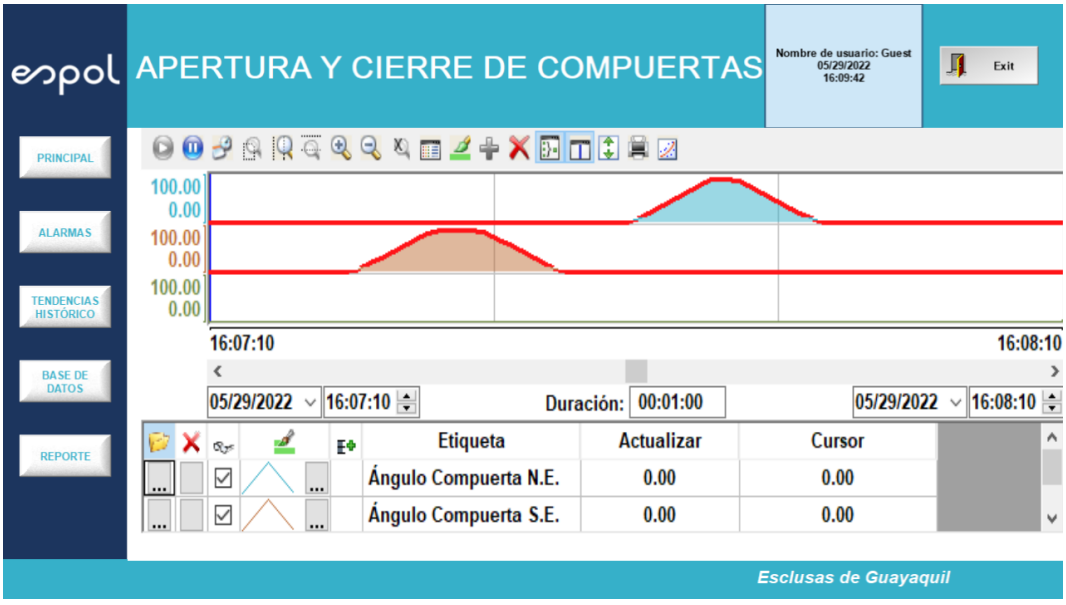


Figura 2.22 Pantalla de Tendencias HMI

Tenemos el cuarto botón que es de la base datos, son los datos de activación de marcha y stop que se van guardando desde el inicio de puesta en operación, en la figura 2.23 se puede observar esta pantalla.

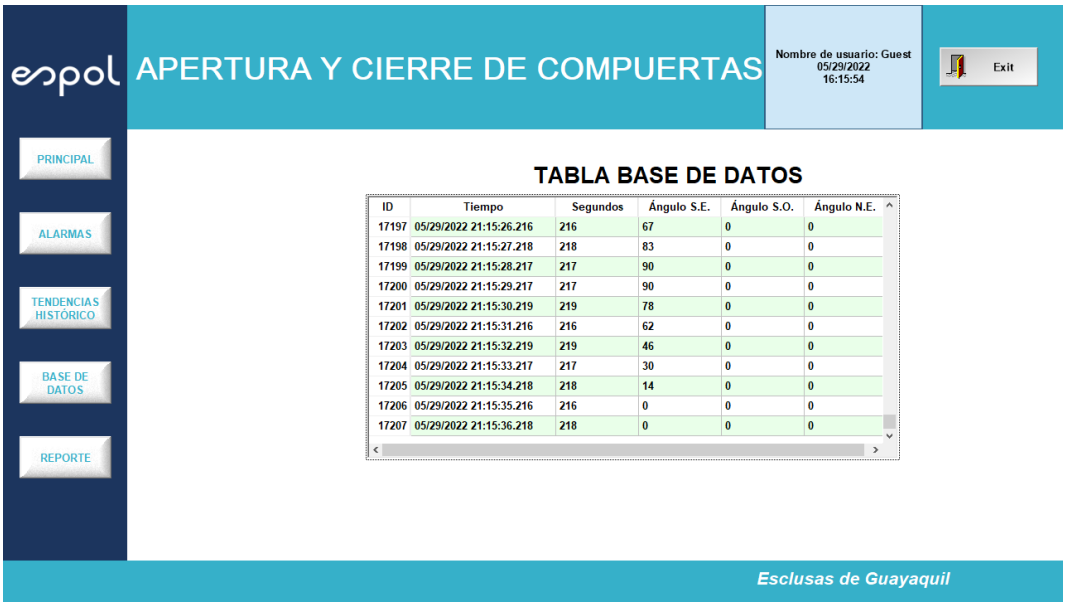


Figura 2.23 Pantalla de Base de Datos HMI

Por último, tenemos el botón de reporte que nos ayuda a generar un reporte en PDF en caso de tener que enviar reportes a supervisores como se detalla en la figura 2.24.

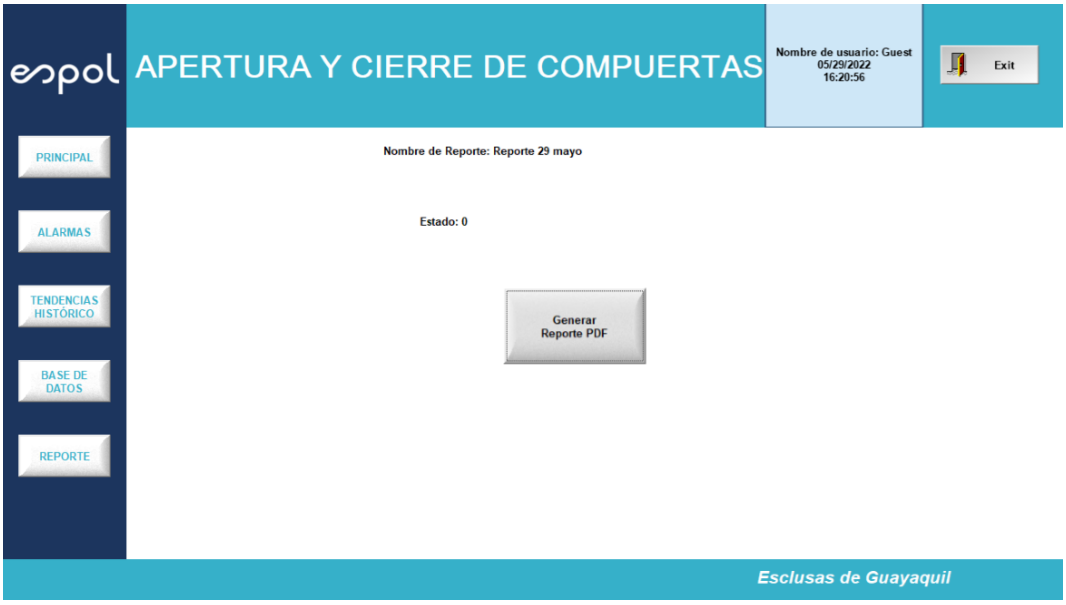


Figura 2.24 Pantalla generación de reporte

En la figura 2.25 se muestra la estructura de cómo se ven los reportes luego de generarlos desde el HMI.

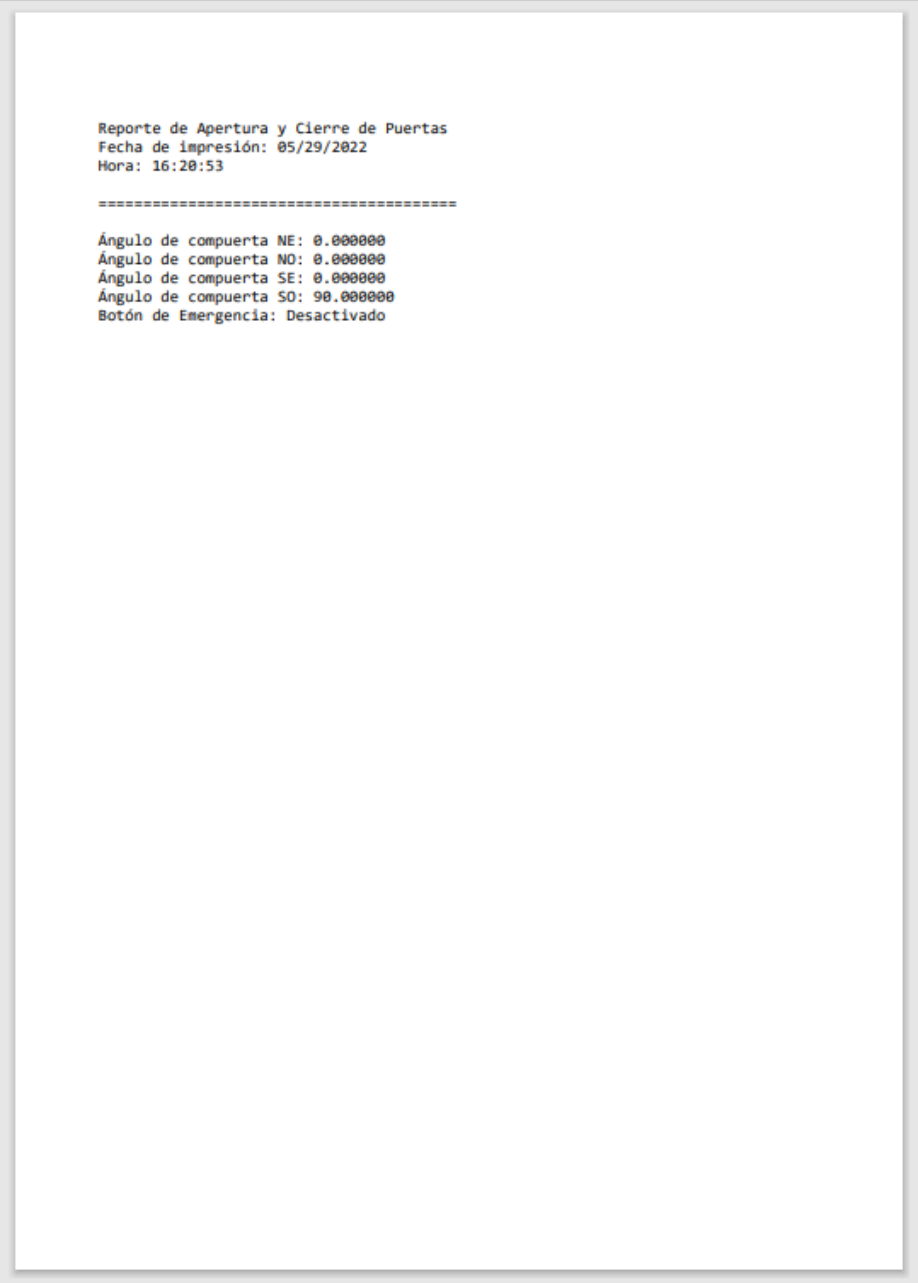


Figura 2.25 Reporte en PDF

En la tabla 2.3 se muestra todas las variables usadas en el desarrollo del HMI, explicando su nombre, identificador, origen, magnitud, tipo, flujo variable, rango, alarma, tipo de alarma en caso de poseer alarma, históricos y tiempo de actualización de históricos en caso de tener histórico.

Tabla 2.3 Lista de Variables HMI

Variable	Identificador	Origen / Dirección	Magnitud	Tipo	Flujo de Variable	Rango	Alarma (S/N)	Tipo de Alarma	Histórico	Histórico T. Actualización
Modo de Operación	bt_mod_operación	Switch que al tomar valor 0 entra en modo abrir, si es 1 en modo cerrar y si es 2 en modo Manual		Digital	Salida	0,1,2	N		N	
Compuertas	bt_compuerta_e_o	Switch que al tomar valor 0 activa las compuertas del este, si es 1 activa las compuertas del oeste		Digital	Salida	0,1,2	N		N	
Marcha	bt_marcha	Si se pulsa manda a la marcha las compuertas		Digital	Salida	0/1	N		N	
Paro	bt_paro	Si se pulsa manda a parar la compuerta		Digital	Salida	0/1	N		N	
Indicador de Marcha	led_marcha	Led de visualización de marcha en el HMI		Digital	Interna	0/1	N		N	
Ángulo de posición N.O.	ang_no	Proceso/chanel1/device1/k0102	Grados	Analógica	Entrada	0-70	S	HiHi= 91, Hi=90	S	1000 ms
Ángulo de posición N.E.	ang_ne	Proceso/chanel1/device1/k0100	Grados	Analógica	Entrada	0-70	S	HiHi= 91, Hi=90	S	1000 ms
Ángulo de posición S.O.	ang_so	Proceso/chanel1/device1/k0101	Grados	Analógica	Entrada	0-70	S	HiHi= 91, Hi=90	S	1000 ms
Ángulo de posición S.E.	ang_se	Proceso/chanel1/device1/k0103	Grados	Analógica	Entrada	0-70	S	HiHi= 91, Hi=90	S	1000 ms

Variable	Identificador	Origen / Dirección	Magnitud	Tipo	Flujo de Variable	Rango	Alarma (S/N)	Tipo de Alarma	Histórico	Histórico T. Actualización
Paro Emergencia	emergencia	Señal de paro de emergencia al estar en 1		Digital	Entrada	0/1	N		S	10 ms
Fin de carrera cerrado	fin_cerrado	Proceso/chanel1/device1/k0104		Digital	Entrada	0/1	N		S	10 ms
Fin de carrera abierto	fin_abierto	Proceso/chanel1/device1/k0105		Digital	Entrada	0/1	N		S	10 ms
Ángulo máximo de compuerta	lim_abierto	Definido en 70 grados	Grados	Analógica	Interna	70	N		N	
Ángulo mínimo de compuerta	lim_cerrado	Definido en 0 grados	Grados	Analógica	Interna	0	N		N	
Indicador de apertura total	var_abierto	Bandera es 1 si está abierto totalmente la compuerta		Digital	Interna	0/1	N		N	
Indicador de cierre total	var_cerrado	Bandera es 1 si está cerrado totalmente la compuerta		Digital	Interna	0/1	N		N	
Bandera de movimiento	bandera_marcha	Indica si se está moviendo la compuerta		Digital	Interna	0/1	N		N	
Reconocimiento de alarmas	ack_reco	Reconoce las alarmas		Digital	Interna	0/1	N		N	

Para el HMI se desarrolla dos tipos de códigos de programación, el primero para el script principal (véase apéndice H) y el segundo para script gráfico (véase apéndice I)

2.4.2.2 NODE-RED

El NODE_RED como fue explicado en secciones anteriores se lo utiliza para poder comunicar la tarjeta TSC_LAB con el PLC y a su vez esta conexión permite por medio del PLC estar conectado al HMI. Es un software de código abierto. Este software posee tres comunicaciones separadas en 6 ventanas o flows; la primera comunicación es mediante el uso de HTTP para comunicar la tarjeta TSC_LAB con el NODE_RED como se indica en la figura 2.26, como son cuatro tarjetas entonces cuatro flows corresponden a la comunicación vía HTTP de las 4 compuertas respectivamente, podemos observar que mediante la función GET obtengo información desde NODERED, mientras que con la función POST envío información de la tarjeta hacia el NODERED (véase apéndice J).

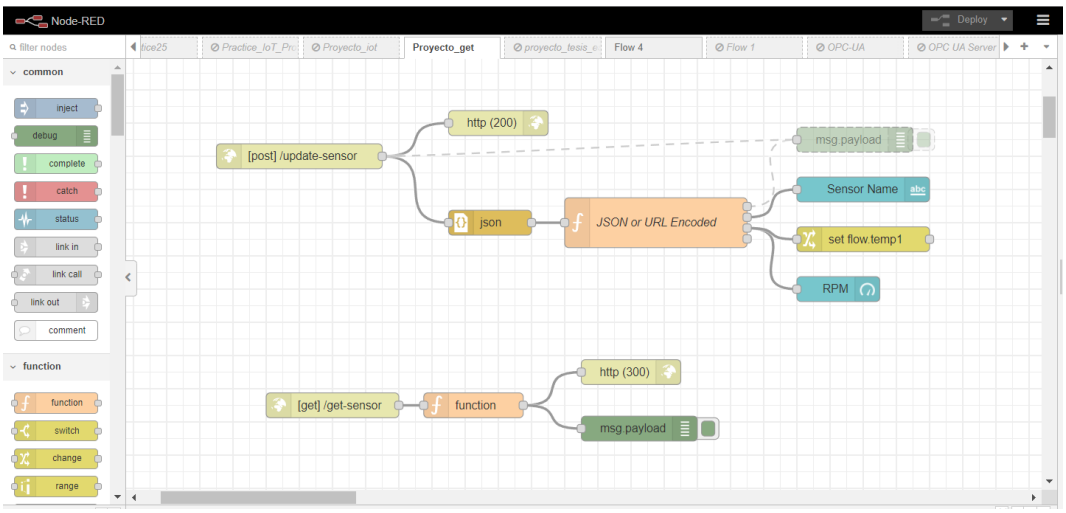


Figura 2.26 Flow de Conexión HTTP compuerta S.E.

La segunda comunicación es entre el NODE_RED y el PLC, para esta implementación usaremos un flow independiente donde se configure la comunicación como se muestra en la figura 2.27 (véase apéndice K).

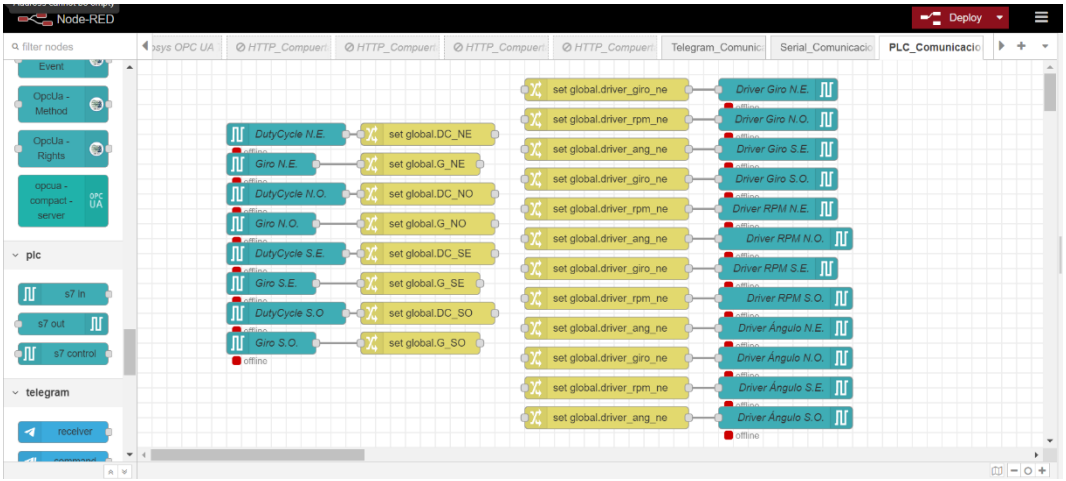


Figura 2.27 Flow de Conexión Ethernet Industrial

Por último, tenemos la tercera comunicación que es entre el NODERED y TELEGRAM para el monitoreo y control del inicio del proceso como se indica en la figura 2.28 (véase apéndice L).

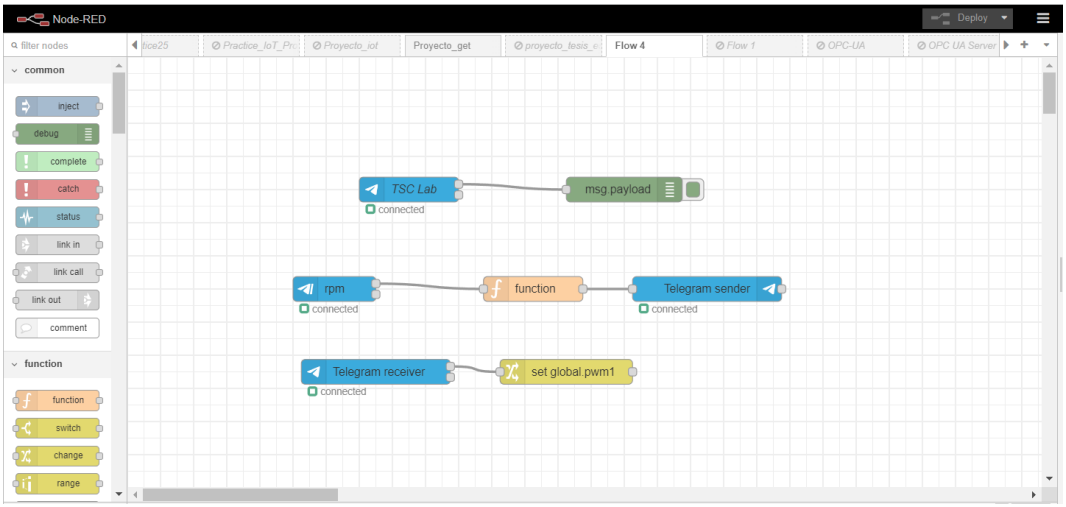


Figura 2.28 Flow de Conexión con Telegram

También es necesario el desarrollo de una versión cableada vía serial entre la tarjeta TSC_LAB y el NODERED, esto se lo hizo para ser usado en las primeras pruebas del frenado dinámico, con la finalidad de poder conseguir el frenado dinámico deseado, descartando un posible escenario de mal funcionamiento del control del PID por problemas de comunicación HTTP, ya una vez testeado el frenado dinámico vía serial se procede a testear vía inalámbrica con protocolo HTTP.

Esta implementación alámbrica generó una variación en los flows del NODERED para la comunicación con la tarjeta TSC_LAB. La transmisión de la tarjeta se muestra en la figura 2.29 (véase apéndice M) y la recepción de la tarjeta se muestra en la figura 2.30 (véase apéndice N).

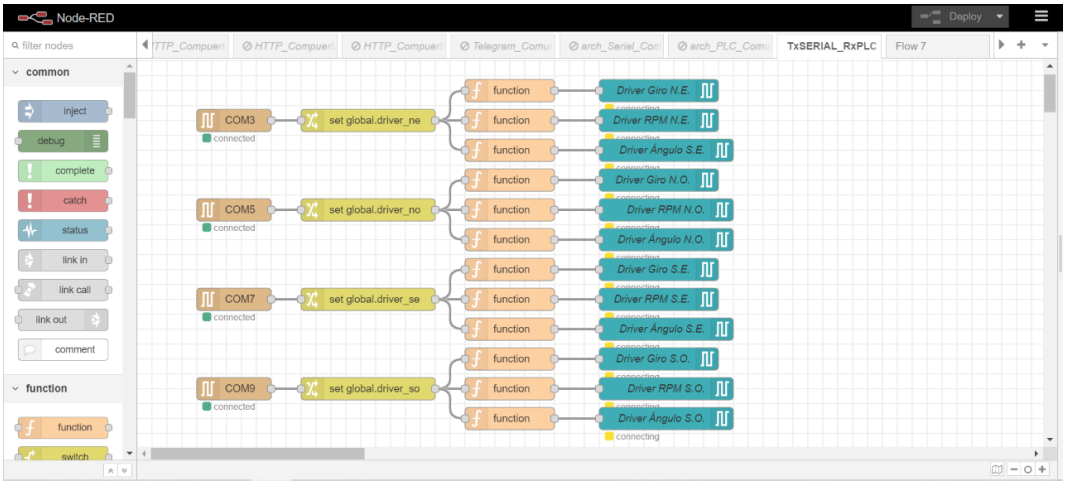


Figura 2.29 Flow de Tx Serial y Rx PLC

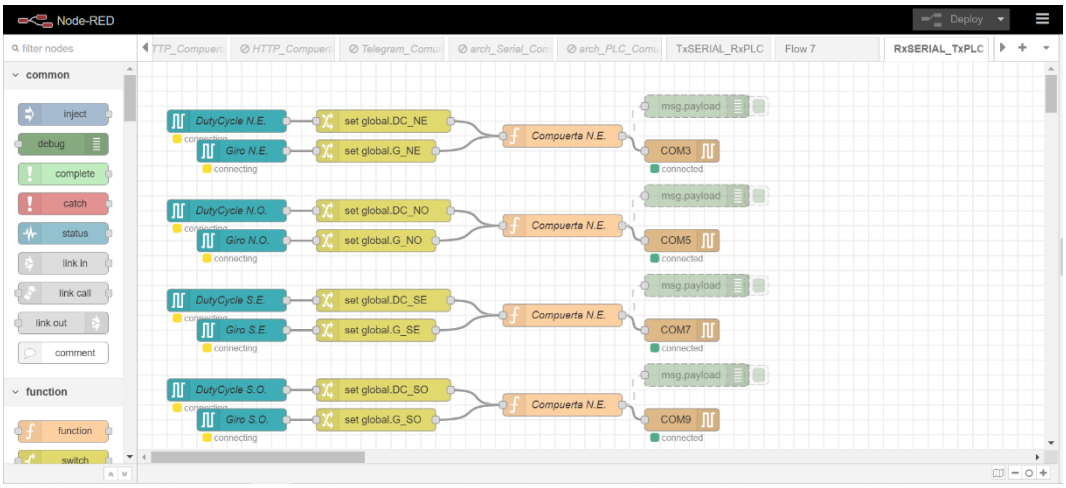


Figura 2.30 Flow de Rx Serial y Tx PLC

2.4.2.3 PLC

Como parte del control tenemos el PLC, en este caso es un PLC SIEMENS S7 1200 con procesador 1212C.

En la sección 2.1 se indica que el PLC es el encargado de controlar de manera conjunta la apertura y cierre de las 4 compuertas, por medio de la comunicación de los respectivos

controladores de los motores. La lógica de funcionamiento para una compuerta se detalla en el diagrama de flujo de la figura 2.31, y esa lógica se programa en TIA PORTAL (véase apéndice U), proceso que inicia con la señal “START” enviada desde el HMI, junto con el envío de esta señal desde el HMI también se indica que compuertas van actuar en el proceso y si es una operación de apertura o cierre.

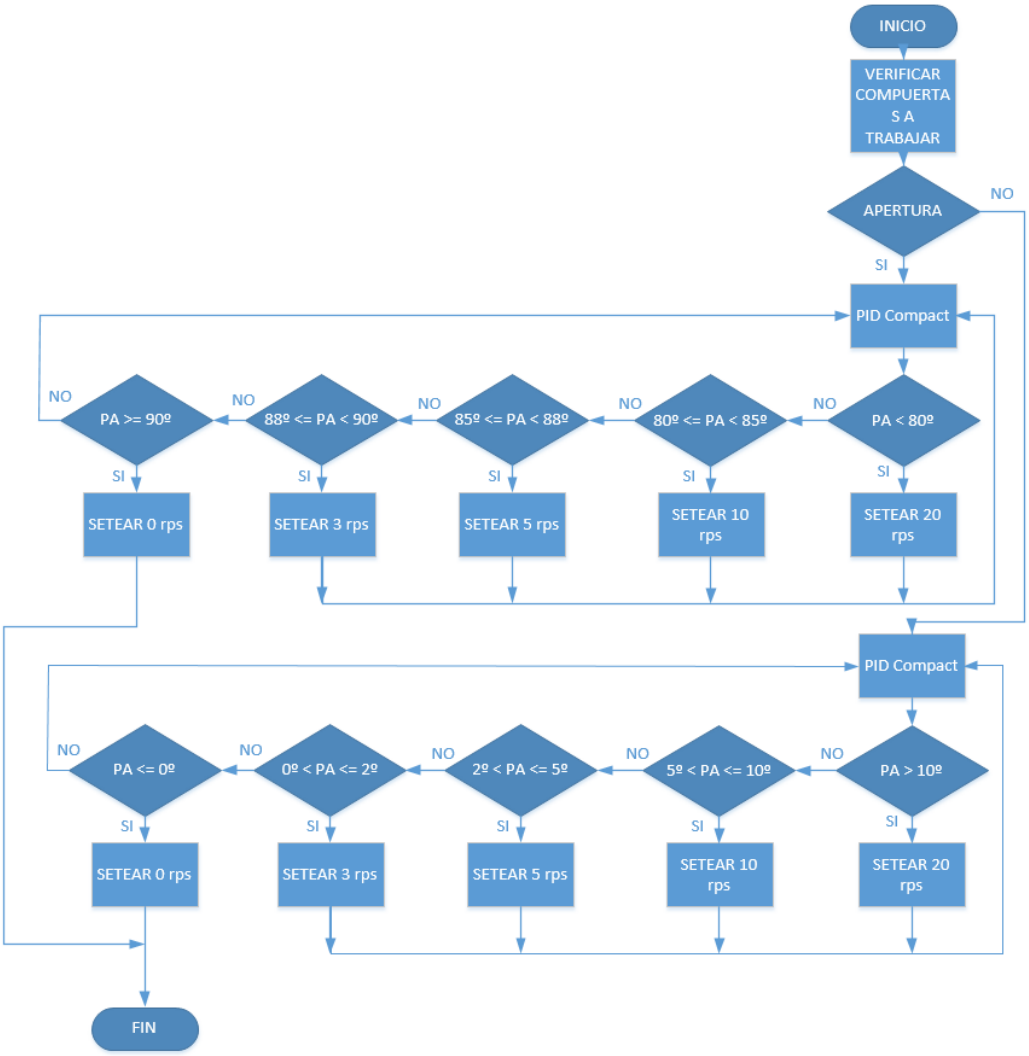


Figura 2.31 Diagrama de Flujo del PLC

Para el caso de la apertura, una vez iniciado el proceso el PLC verifica la posición angular de las dos compuertas recibidas desde la tarjeta TSC_LAB, si esta posición angular es menor que 80 grados se configura la velocidad en 20 rps, el equivalente a 1200 rpm que es la velocidad máxima con la cual puede girar el motor para no dañar el sistema de engranaje, a su vez el controlador PID Compact tratará de mantener la velocidad en 20 rps enviando un DutyCycle a la tarjeta TSC_LAB entre 0 a 255. En el caso de que la

posición angular sea mayor o igual a 80 grados y menor a 85 grados entonces la velocidad se configura en 10 rps el controlador PID Compact actuará para mantener constante la velocidad mediante el envío del DutyCycle. Luego se repite lo mismo, si la posición angular es mayor o igual a 85 y menor a 88 se configura la velocidad en 5 rps y si la posición angular es mayor o igual a 88 y menor a 90 se configura la velocidad en 3 rps, por último, cuando la posición angular sea igual o mayor a 90 grados entonces la velocidad se configura en 0 rps.

Para el caso del cierre, una vez iniciado el proceso el PLC verifica la posición angular de las dos compuertas recibidas desde la tarjeta TSC_LAB, si esta posición angular es menor que 10 grados se configura la velocidad en 20 rps, el equivalente a 1200 rpm que es la velocidad máxima con la cual puede girar el motor para no dañar el sistema de engranaje, a su vez el controlador PID Compact tratará de mantener la velocidad en 20 rps enviando un DutyCycle a la tarjeta TSC_LAB entre 0 a 255. En el caso de que la posición angular sea menor o igual a 10 grados y mayor a 5 grados entonces la velocidad se configura en 10 rps y el controlador PID Compact actuará para mantener constante la velocidad mediante el envío del DutyCycle. Luego se repite lo mismo, si la posición angular es menor o igual a 5 y mayor a 2 se configura la velocidad en 5 rps y si la posición angular es menor o igual a 2 y mayor a 0 se configura la velocidad en 3 rps, por último, cuando la posición angular sea igual o menor a 0 grados entonces la velocidad se configura en 0 rps.

Se utilizó una función cíclica donde se realiza 4 controles PID cada 10 ms de las 4 compuertas, recordando que solo se abren dos compuertas a la vez, las puertas oeste o las puertas este, solo dos PID están habilitados por operación de apertura o cierre, mientras que los otros dos restantes están deshabilitados. El controlador PID cumple la función de mantener la velocidad del motor seteado en el valor que arroja el PID compact, ya que al recibir de retroalimentación la velocidad del motor en rps entonces este procede a calibrar la velocidad hasta posicionarlas en el valor de velocidad deseado mediante el envío del DutyCycle a la tarjeta TSC_LAB. El DutyCycle puede tomar un valor comprendido entre 0 a 255.

En la figura 2.32 se detalla las variables del DB1 usadas para la transmisión de datos desde el PLC al NODERED.

Figura 2.32 DB1 para envío a NODERED

En la figura 2.33 se detalla las variables del DB2 usadas para la recepción de datos del NODERED al PLC.

Figura 2.33 DB2 para recepción de NODERED

En la figura 2.34 se detalla las variables del DB3 usadas para la transmisión de datos desde el PLC al HMI.

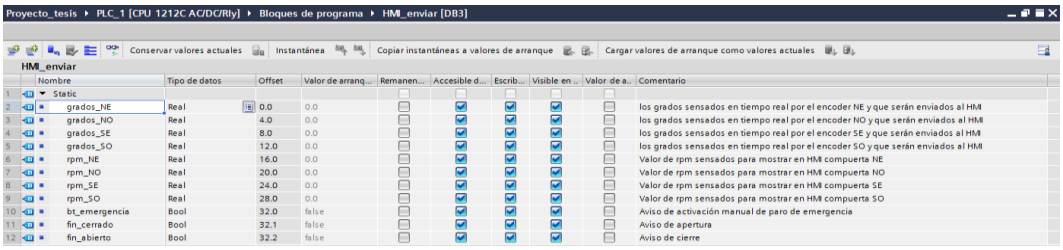


Figura 2.34 DB3 para envío a HMI

En la figura 2.35 se detalla las variables del DB4 usadas para la recepción de datos del HMI al PLC.

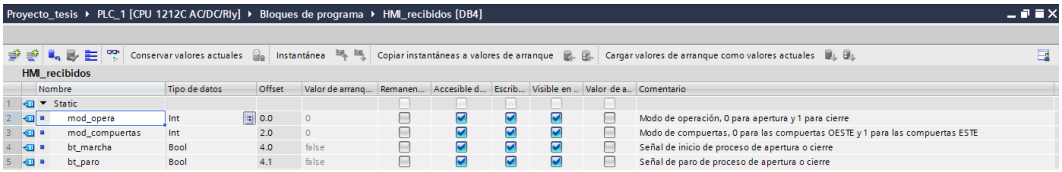


Figura 2.35 DB4 para recepción de HMI

2.4.2.4 TSC_LAB

Para el proyecto de titulación se usa dos tarjetas TSC_LAB para simular dos compuertas. La tarjeta TSC_LAB posee un MCU en la tarjeta ESP-32, estas tarjetas son programadas en lenguaje C, usando de referencia el IDE de arduino. El funcionamiento de esta tarjeta se detalla en el diagrama de flujo de la figura 2.36:

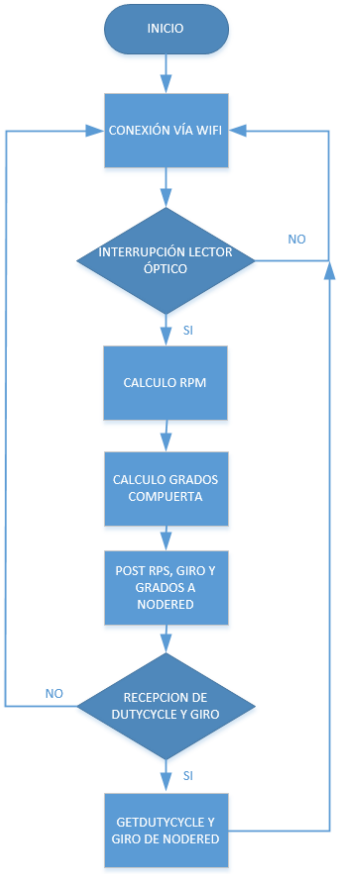


Figura 2.36 Diagrama de flujo de TSC_LAB

Como se indica en el diagrama de flujo, se desarrolla un código (véase apéndice O) para que la tarjeta al iniciar lo primero que haga sea conectarse a la red vía WIFI, una vez conectado comienza a calcular las rpm del motor y a su vez también comienza a calcular el desplazamiento de la compuerta en grados, este desplazamiento se obtiene al activarse la interrupción del MCU conectada al lector óptico, el cual en la variable posición angular se va acumulando los grados de desplazamiento, luego la tarjeta TSC_LAB espera hasta que el NODERED envíe el DutyCycle que proviene del PLC,

esta consulta se realiza cada que existe un cambio en los datos enviados desde el PLC al NODERED tomando 1000 ms de duración para que exista el cambio. Por último, los datos de velocidad rpm, giro y los grados de posición de la compuerta son enviados desde la tarjeta TSC_LAB hasta el NODERED cada 1000 ms usando una lógica de ejecución de procesos en paralelo de RTOS.

La velocidad de procesamiento del microcontrolador de la tarjeta ESP-32 es de 240 Mhz, y un comando de lectura por pin le toma realizarlo en 302 nanosegundos, el motor usado es de 5000 rpm, lo que significa que le toma 12 milisegundos dar una vuelta. Con esto podemos ver que todas las vueltas del motor son contabilizadas sin ningún inconveniente.

Se utiliza dos ServerName para cada tarjeta, un ServerName para el get y uno para el post respectivamente. A continuación, un ejemplo para la compuerta Noreste:

GET: http://192.168.0.8:1880/get-sensor
POST: http://192.168.0.8:1880/update-sensor

Al igual que se hizo con el NODERED, se desarrolla un código para la comunicación serial con el NODERED, esto fue explicado en la sección 2.4.2.2. El código solo cambia el tipo de comunicación, mas no cambia el tipo de lógica de funcionamiento (véase apéndice P).

Al ser una comunicación serial se procede a realizar un protocolo de comunicación propio para la transmisión Tx y para la recepción Rx.

El protocolo de transmisión desde la tarjeta TSC_LAB está conformado por una trama de 7 palabras, donde la palabra 1 es de inicio, la palabra 2 es si la compuerta está abriendo o cerrando, las palabras 3, 4 y 5 representan la posición de los grados de la compuerta codificados hexadecimalmente en 3 Bytes, el receptor, en este caso el NODERED, recibe los 3 Bytes que representan un valor hexadecimal, y ese valor se multiplica por 0.0296 para obtener la posición de la compuerta en grados reales, luego tenemos la palabra 6 que es el valor rps medido en ese instante y por último tenemos la

palabra 7 que es de fin de la trama. En la tabla 3.1 se detalla la trama del protocolo para la transmisión desde la tarjeta TSC_LAB.

Tabla 2.4 Trama de protocolo de transmisión

# Palabra	Rango de Valores ASCII (DEC)	Descripción
1	36	Inicio del mensaje, se visualiza con el símbolo '$'
2	65, 66	En apertura (65) o cierre (66) de la compuerta
3	48-57, 65-70	Un número hexadecimal de 0 a la F. Bit más significativo
4	48-57, 65-70	Un número hexadecimal de 0 a la F
5	48-57, 65-70	Un número hexadecimal de 0 a la F. Bit menos significativo
6	0-84	Valor de revoluciones por segundo (rps)
7	64	Fin del mensaje, se visualiza con el símbolo '@'

El protocolo de recepción de la tarjeta TSC_LAB está conformado por una trama de 6 palabras, donde la palabra 1 es de inicio, la palabra 2 da la orden de si se debe abrir o cerrar la compuerta, la palabra 3, 4 y 5 indica el valor de DutyCycle del motor y por último tenemos la palabra 6 que es de fin de la trama. En la tabla 3.2 se detalla la trama del protocolo para la recepción desde la tarjeta TSC_LAB.

Tabla 2.5 Trama de protocolo de recepción

# Palabra	Rango de Valores ASCII (DEC)	Descripción
1	36	Inicio del mensaje, se visualiza con el símbolo '$'
2	65, 66	Modo de operación apertura (65) o cierre (66) de la compuerta
3	48-57	Un número decimal de 0 al 9. Bit más significativo

# Palabra	Rango de Valores ASCII (DEC)	Descripción
4	48-57	Un número decimal de 0 al 9
5	48-57	Un número decimal de 0 al 9. Bit menos significativo
6	64	Fin del mensaje, se visualiza con el símbolo '@'

Un ejemplo de cómo se ve la comunicación analizada desde un terminal se detalla en la figura 2.37, donde una trama de transmisión está encerrada en un cuadrado de color azul y una trama de recepción está encerrada en un cuadro de color verde. Los datos mostrados son recopilados de una comunicación entre la tarjeta TSC_LAB y el NODERED, considerando el NODERED como transmisor, en una operación de apertura de compuerta, los valores de cada carácter están en decimal. Se usa el software DockLight con licencia de prueba para la captura de las tramas.

```
02/06/2022 13:52:51.708 [TX] - 036 065 050 050 053 078 064
02/06/2022 13:52:51.725 [RX] - 036 065 048 048 049 084 064 036 065
048 048 050 084 064 036 065 048 048 051 084 064 036 065 048 048 052
084 064 036 065 048 048 053 084 064 036 065 048 048 054 084 064 036
065 048 048 055 084 064 036 065 048 048 056 084 064 036 065 048 048
057 084 064 036 065 048 048 065 084 064 036 065 048 048 066 084 064
036 065 048 048 067 084 064 036 065 048 048 068 084 064 036 065 048
048 069 084 064 036 065 048 048 070 084 064 036 065 048 049 048 084
064 036 065 048 049 049 084 064 036 065 048 049 050 084 064 036 065
048 049 051 084 064 036 065 048 049 052 084 064 036 065 048 049 053
084 064 036 065 048 049 054 084 064 036 065 048 049 055 084 064 036
065 048 049 056 084 064 036 065 048 049 057 084 064 036 065 048 049
065 084 064 036 065 048 048 049 084 064 036 065 048 048 050 084 064
036 065 048 048 051 084 064 036 065 048 048 052 084 064 036 065 048
048 053 084 064 036 065 048 048 054 084 064 036 065 048 048 055 084
064 036 065 048 048 056 084 064 036 065 048 048 057 084 064 036 065
048 048 065 084 064 036 065 048 048 066 084 064 036 065 048 048 067
084 064 036 065 048 048 068 084 064 036 065 048 048 069 084 064 036
065 048 048 070 084 064 036 065 048 049 048 084 064 036 065 048 049
049 084 064 036 065 048 049 050 084 064 036 065 048 049 051 084 064
036 065 048 049 052 084 064 036 065 048 049 053 084 064 036 065 048
```

Figura 2.37 Tramas comunicación TSC_LAB y NODERED

Tratando de simular una perturbación o condiciones reales, se optó por agregar un peso adicional al motor de una tarjeta, esto considerando que en la vida real las dos compuertas no tienen las mismas condiciones en el entorno, por lo que algunas por suciedad o por rieles desgastados hace que el motor de una compuerta se esfuerce un poco más que el motor de la otra compuerta. Por lo tanto, se procede a diseñar en un

software CAD una pieza que agregue un peso adicional además de pequeñas ventosas que causen algo de resistencia al movimiento como se indica en la figura 2.35. El archivo generado es stl.

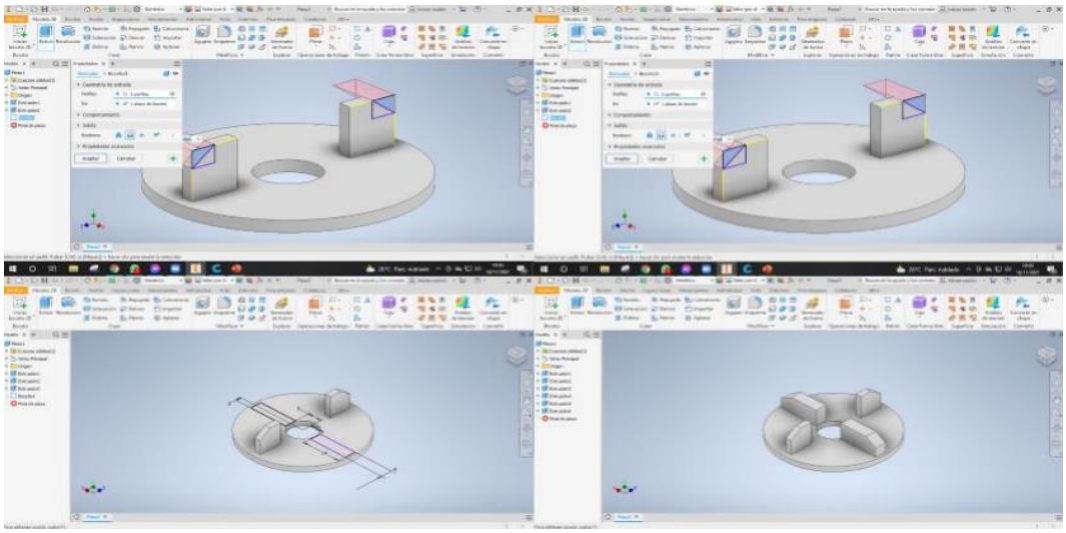


Figura 2.38 Diseño de pieza mecánica

Se utiliza luego un software de código abierto para impresión en 3D, para este caso el software es CURA v15.04 que trabaja bien con la impresora 3D personal ANYCUBIC KOSSEL y que puede convertir un archivo STL a un archivo legible para la impresora. El diseño queda de la siguiente manera como se muestra en la figura 2.36.

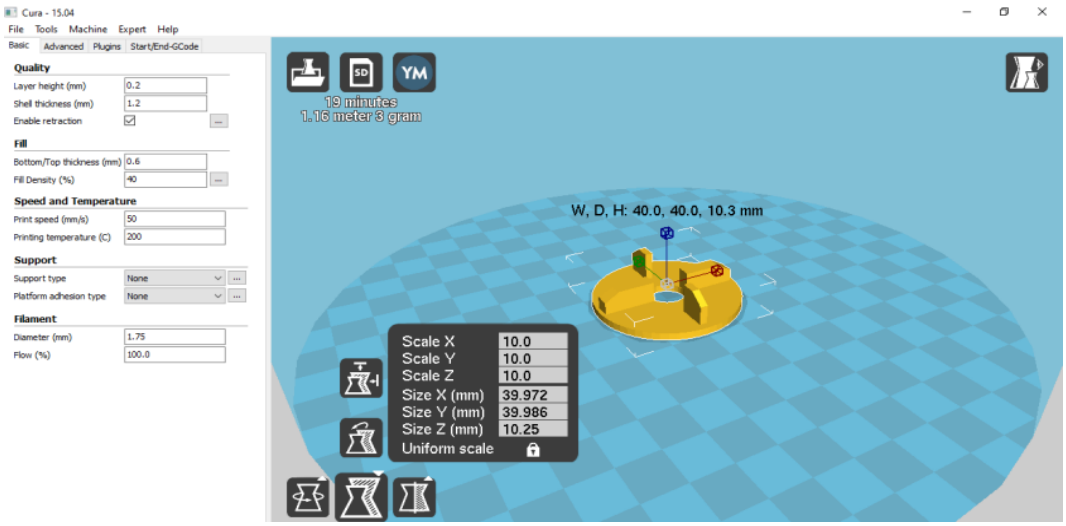


Figura 2.39 Preparación para impresión 3D

Y por último se procede a imprimir en 3D usando la impresora de uso personal ANYCUBIC KOSSEL como se indica en la figura 2.37.

Figura 2.40 Impresión 3D

2.5 Modelamiento Matemático

Es necesario modelar matemáticamente la planta para poder trabajar con ella en la implementación de un controlador. Por lo tanto, se procede a trabajar en un modelamiento matemático de caja negra.

2.5.1 Recolección de datos en lazo abierto

Se realiza la captura de datos de la planta con la finalidad de luego poder realizar la identificación de la planta, para esta prueba se desarrolla un código en la tarjeta TSC_LAB (véase apéndice Q) que permita acelerar el motor por 13 segundos y luego desacelerar el motor por otros 13 segundos de forma cíclica, esto se realizó con la finalidad de ver la respuesta del motor ante un tren de impulsos hasta llegar a la aceleración máxima de 5000 rpm y luego ver como se desacelera hasta llegar a 0 rpm. Usando el software de código abierto arduino, se procede a verificar el comportamiento de los datos del motor con el serial plotter como se puede observar en la figura 3.14.

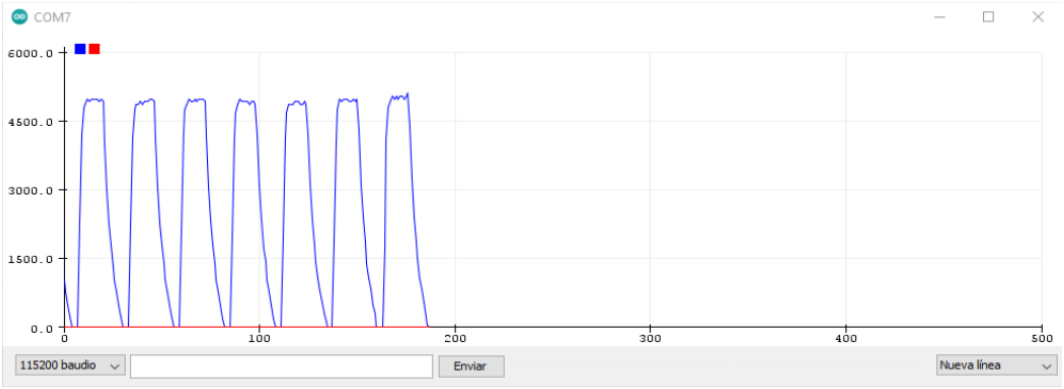


Figura 2.41 Serial Ploter

Luego de haber verificado visualmente el comportamiento del motor, se procede a capturar los datos de la velocidad en rpm, estos datos capturados se guardan como extensión ".csv" para su posterior uso en matlab. En la figura 3.15 se puede observar las velocidades en rpm en la primera columna y el pulso en la segunda columna.

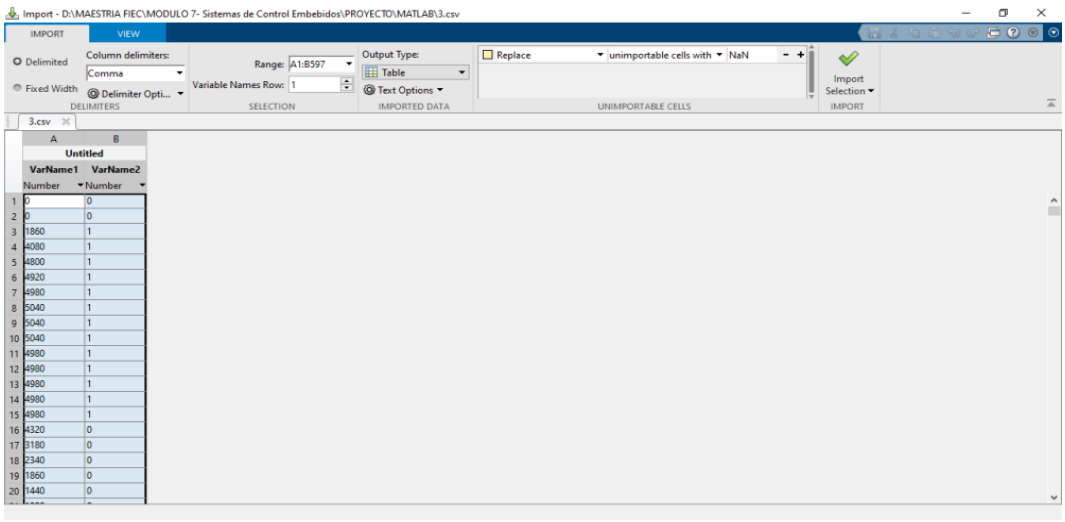


Figura 2.42 Datos capturados

Se realiza la captura de datos, para esto caso son 597 datos, se midió a una tasa de 1 dato por segundo.

2.5.2 Identificación de la Planta

Usando el software MATLAB se procede a exportar y graficar los datos capturados mencionados en la sección 3.2.1, para esta prueba se necesita programar un código en MATLAB (véase apéndice R). Una vez compilado el código en MATLAB se obtiene la

señal de velocidad en rpm vs el tiempo como se muestra en la figura 3.16. Esto genera los datos IN y OUT que corresponden al tiempo y a la velocidad en rpm de velocidad respectivamente.

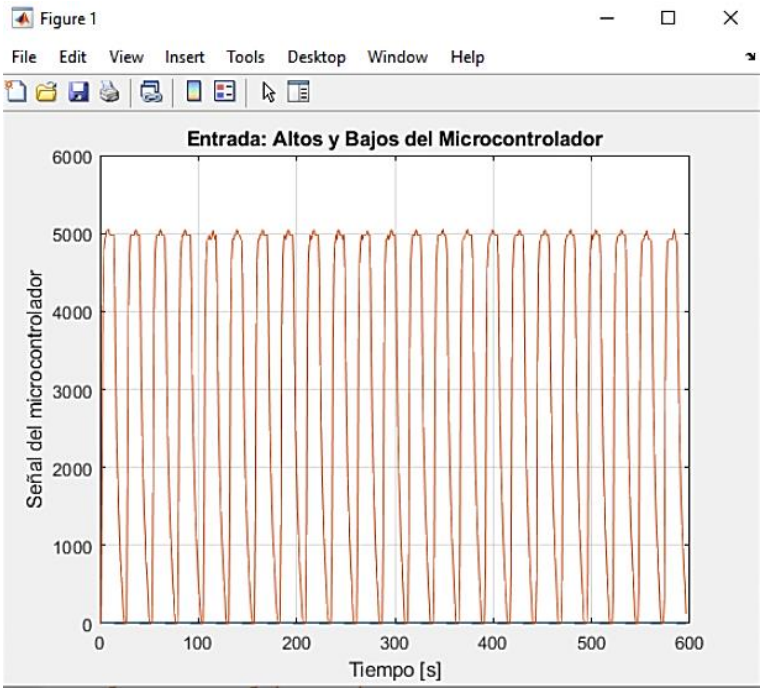


Figura 2.43 Datos en MATLAB

Con los datos IN y OUT generados, se procede abrir la interfaz de usuario gráfica “System Identification” para encontrar el modelo más apropiado de la planta usando el modelo de caja negra. Se importa en dominio de tiempo los valores de IN y OUT como se indica en la figura 3.17.

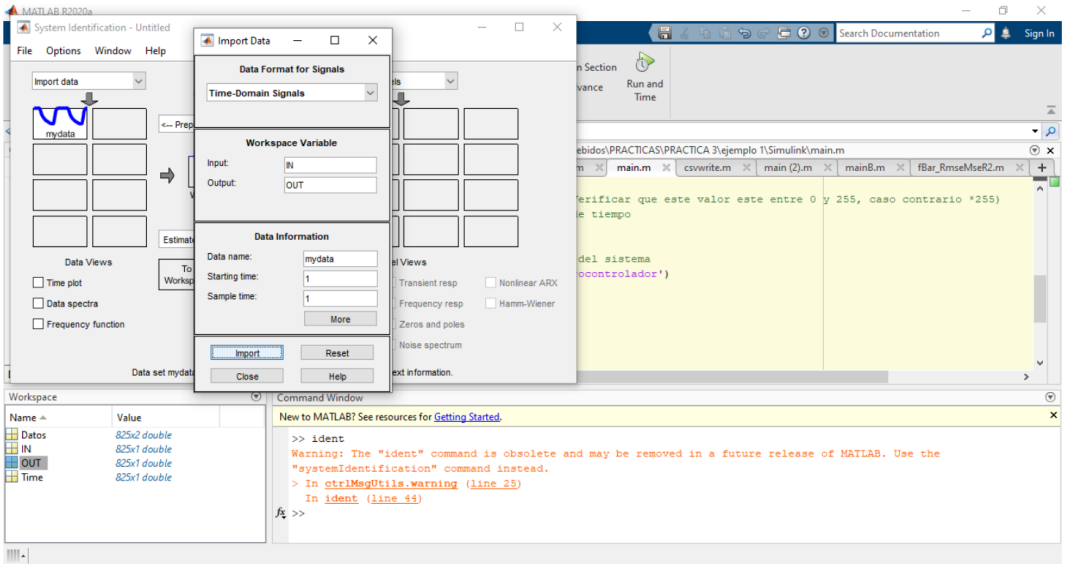


Figura 2.44 Interfaz de Usuario Gráfica “System Identification”

Ya importados los datos se procede a usar la opción “Select range” para crear los datos para estimación y para validación, siguiendo una relación de 80% de los datos para estimación y 20% de los datos para validación como se indica en la figura 3.18.

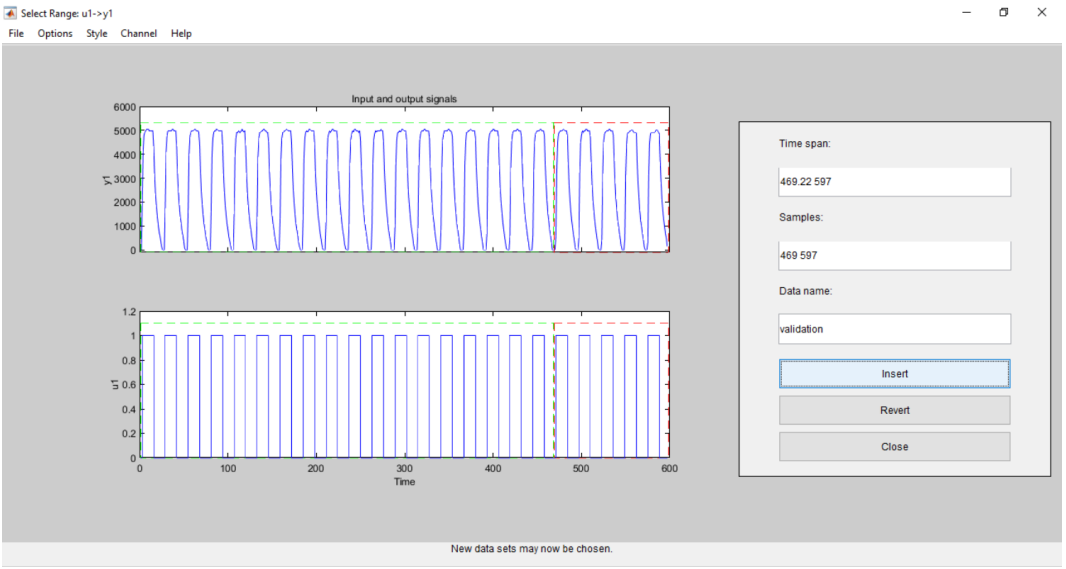


Figura 2.45 Select Range

Con el fin de encontrar el mejor modelo de la planta, se procede a testear diversos métodos, de los cuales el Box Jenkins cumple de mejor forma basado en el criterio de parsimonia, estos diversos modelos se pueden observar en la figura 3.19, en donde se puede ver su gráfica y sus fits.

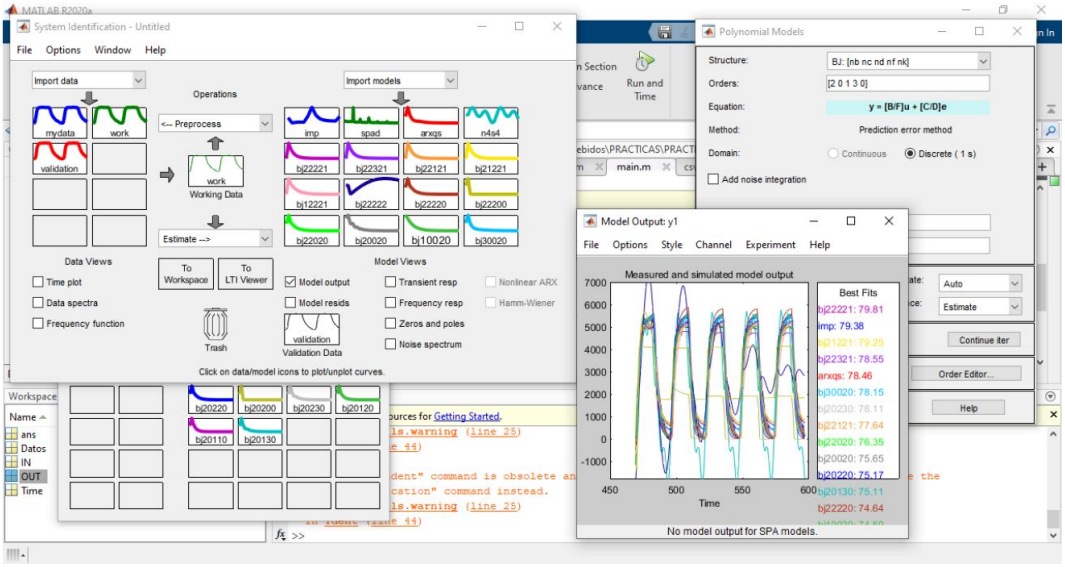


Figura 2.46 Modelos de Plantas

Después de revisar los distintos modelos, el modelo bj20120 es el más idóneo basado en distintos criterios como respuesta a un pulso, entre otros criteros. Además, fue un modelo con unos de los fits más alto (70.18) y que además se asemejaba a la planta real, basado en los datos capturados, esto se puede observar en la figura 3.20.

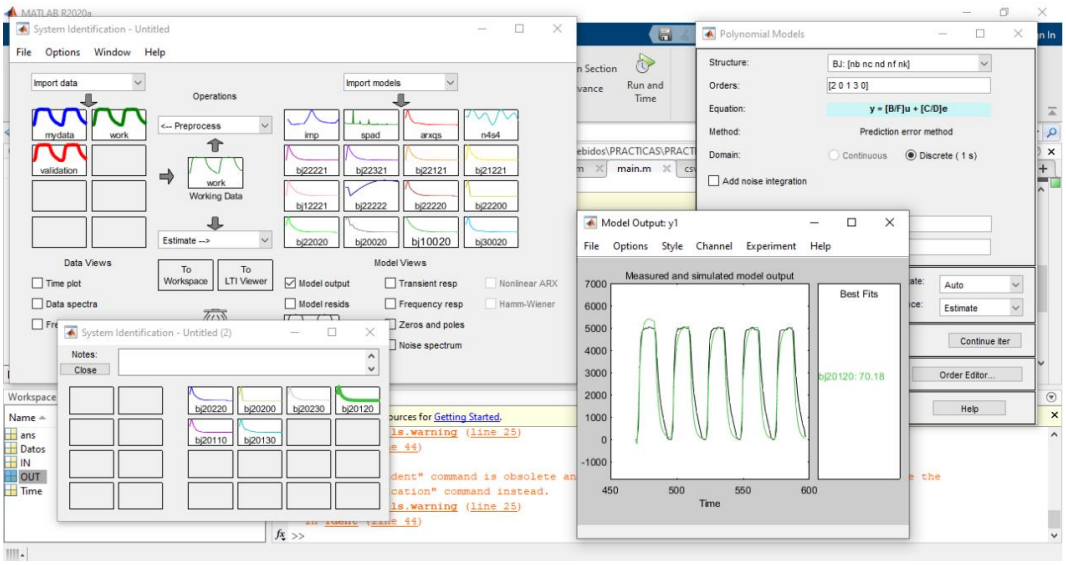


Figura 2.47 Modelo BJ20120

Ya con el modelo de la planta se procede a programar en MATLAB un código (véase apéndice S) que permita encontrar la función de transferencia “G” y luego esta misma función mostrarla con numerador y denominador. Al compilar entonces se obtiene la

función de transferencia con numerador y denominador como se muestra en la figura 3.21.

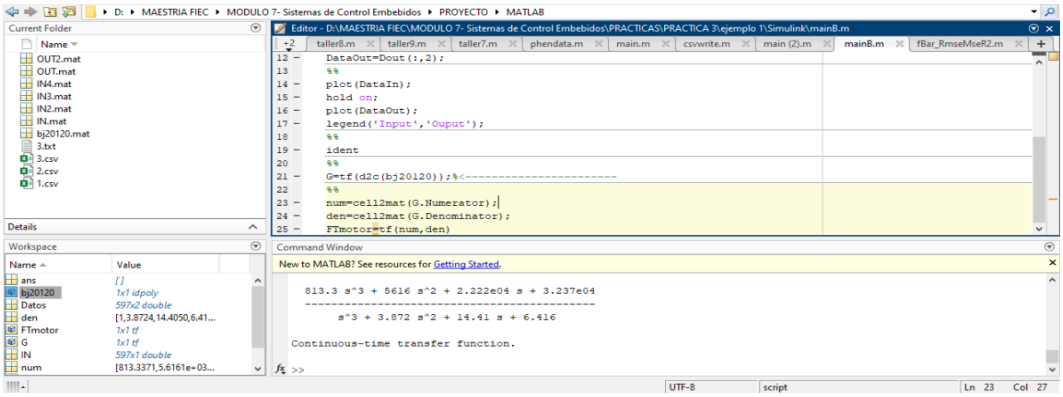


Figura 2.48 Función de transferencia

La función de transferencia es la siguiente:

$$G = \frac{8.13s^3 + 5616s^2 + 2.222e04s + 3.237e04}{s^3 + 3.872s^2 + 14.41s + 6.416}$$

Considerando que es muy importante encontrar una función de transferencia que esté acorde a la planta, se procede a verificar la función de transferencia con la planta, usando los datos capturados de la planta. Se grafica la función de transferencia G y los datos capturados en una sola gráfica y se puede observar que tenían similitud como se indica en la figura 3.22.

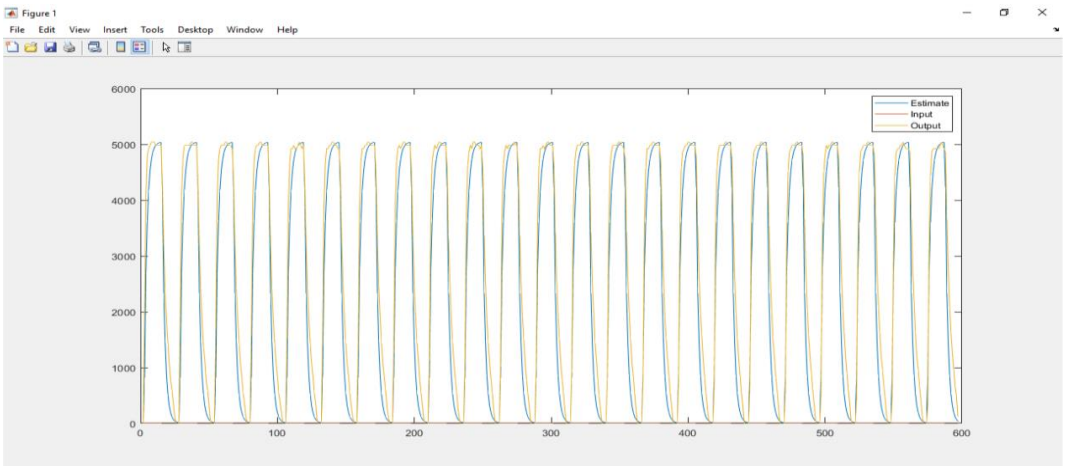


Figura 2.49 Comparación de planta con G

2.6 Controlador PID

Para obtener los valores kp, kd y ki se procede a usar las herramientas que nos brinda MATLAB; en este caso se usa dos, el pidtune y el PID tuner. Al aplicar el pidtune a la función de transferencia da los valores kp = 0.000109, ki = 0.000416 y kd = 0 como se indica en la figura 3.23.

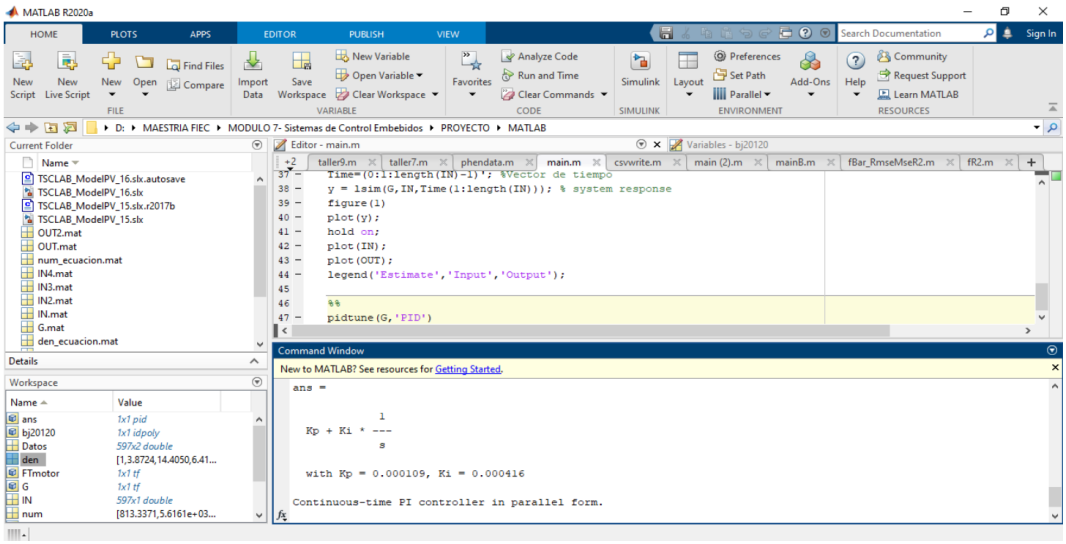


Figura 2.50 Herramienta pidtune

Mientras que al usar la herramienta PID TUNER se obtiene otros valores, los cuales son kp = 1.8018e-05, ki = 6.5313e-05, kd = 0 como se indica en la figura 3.24;

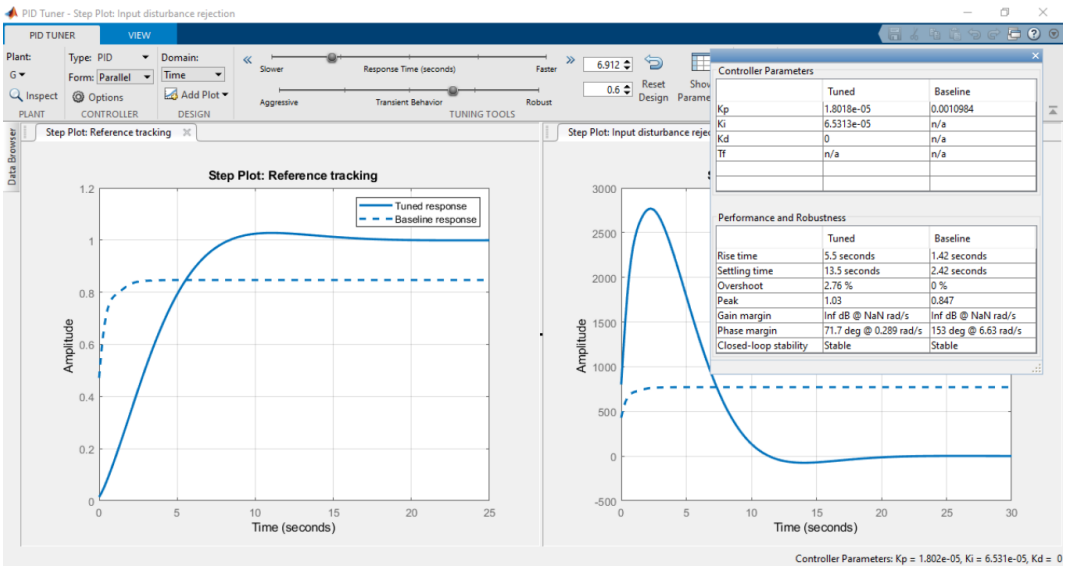


Figura 2.51 Herramienta PID TUNER

Se decide entonces escoger los valores de kp, kd y ki obtenidos mediante el uso del pidtune para el diseño del controlador PID a usarse, luego esos valores serán usados en el PLC. La explicación de por qué la variable derivativa dio 0 (kd =0) es porque para los motores dc la variable integradora puede obtener buena respuesta en tiempo y en error.

CAPÍTULO 3

3. RESULTADO Y ANÁLISIS

Este capítulo se lo divide en dos partes, la primera es el tema de las comunicaciones de todo el sistema, analizando tiempos de envío, tiempos de comunicación, tiempos de refrescamiento, entre otros temas, mientras que la segunda parte se centra en el análisis del frenado dinámico implementado, constatando su correcto funcionamiento. La sección de las comunicaciones es la 3.1 y la sección del frenado dinámico es la 3.2. En la figura 3.1 se puede observar el laboratorio implementado para las pruebas.

Figura 3.1 Laboratorio de prueba

3.1 Comunicaciones

Como parte de las pruebas del proyecto de titulación se realiza un análisis de las comunicaciones entre la red, con el fin de encontrar tiempos de refrescamiento de información, latencias, mejoras en la comunicación y entender que tan factible es el uso del internet de las cosas cuando se habla de comunicación inalámbrica industrial.

3.1.1 Comunicación entre NODE-RED y PLC

La comunicación entre el NODE-RED y el PLC se realiza mediante el uso de ethernet industrial como se detalla en la sección 2.4.2 y de forma gráfica en la figura 2.14. En el NODE-RED se usa un tiempo de refrescamiento (Cycle time) de 2 ms y un tiempo de espera (Timeout) de 1500 ms como se muestra en la figura 3.2. El tiempo de refrescamiento de las variables se configura en 2 ms, tiempo suficiente para ejecutar los procesos internos del controlador lógico programable. El valor del Cycle Time del NODE-RED depende directamente del Cycle Time del PLC, ya que no es recomendable poner un valor de Cycle Time menor al del PLC, si se hiciera eso el NODE-RED está consultando por variables que aún no tienen el tiempo de actualizarse, por eso motivo el Cycle Time del NODE-RED es mayor al PLC, que tan mayor será dependerá del proceso; por ejemplo no es lo mismo un proceso que reciba señales de temperatura cada 30 segundos que un proceso de control que reciba señales de velocidad en rpm cada 100 ms.

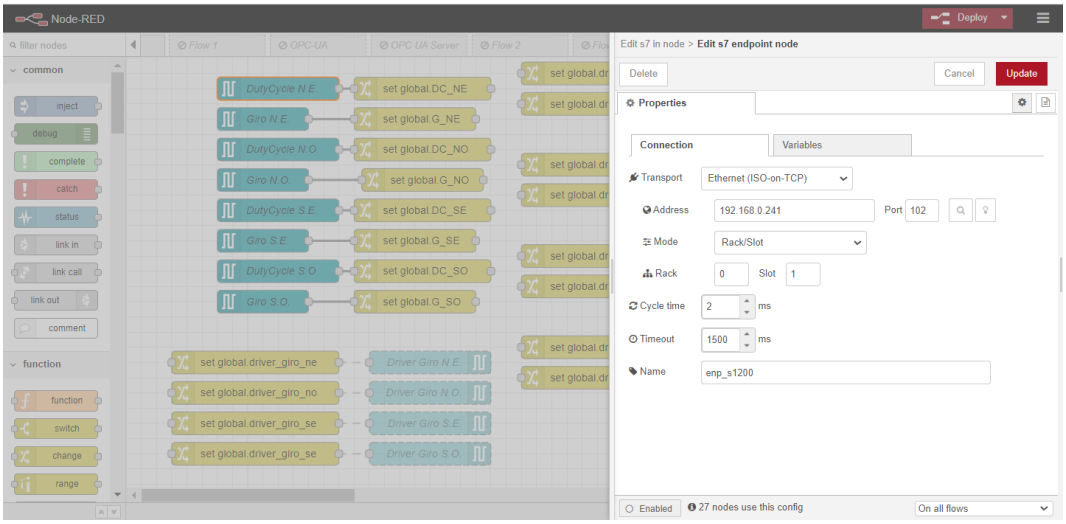


Figura 3.2 Tiempo de refrescamiento NODE_RED IETH

Usando la herramienta “Online & Diagnostics” del TIA PORTAL, se procede a calcular el cycle time del PLC para saber cuánto es el tiempo de refrescamiento de las variables del programa y así poder jugar con los tiempos de refrescamiento de las conexiones entre dispositivos, considerando que tanto el NODERED como el HMI de INDUSOFT preguntan al PLC como clientes. Entonces el Cycle time da 1 ms promedio usando el software como herramienta de medición, simplemente se analiza el tiempo de respuesta

de un variable y con eso se captura el tiempo de actualización, el resultado de la captura del tiempo se muestra en la figura 3.3.

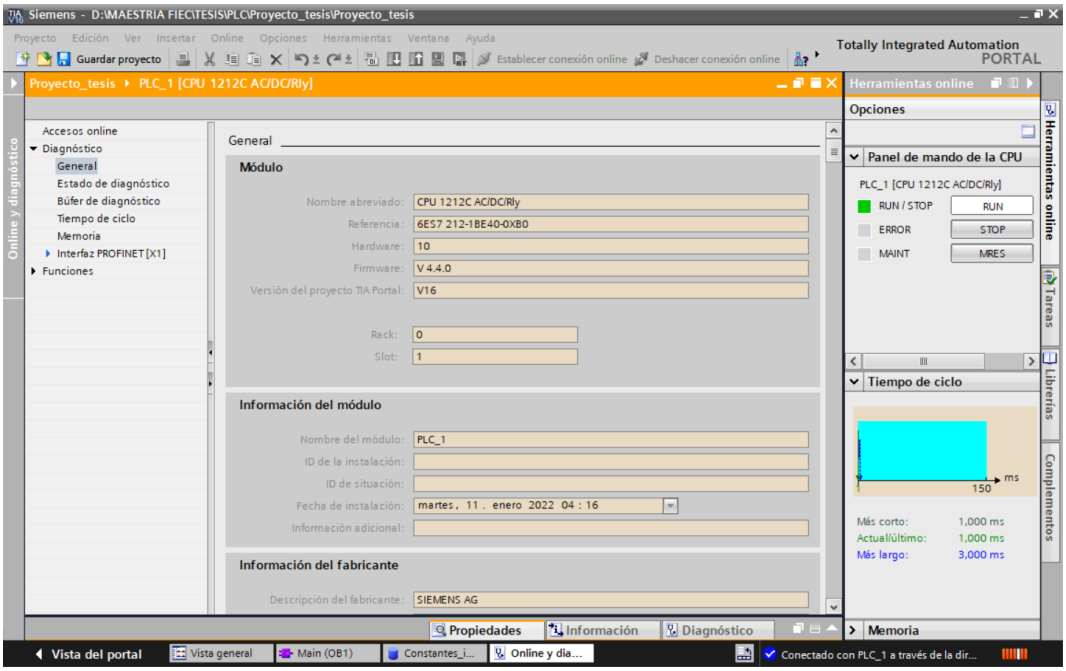


Figura 3.3 Cycle Time del PLC

Observando más detalladamente se observa que dio como resultado un Cycle time de hasta 3 ms máximo, aunque el promedio es de 1 ms. Se detalla de mejor forma en la figura 3.4

Figura 3.4 Cycle Time en valores

Se realiza un test para el análisis de tráfico entre el PLC y el NODE-RED, para este test se usa el software wireshark que es gratis y de código abierto. Los resultados arrojaron que sí existe comunicación TCP/IP y que el tiempo de lectura de variable estaba en los 3 ms como se indica en la figura 3.5. Podemos observar el mensaje ACK.

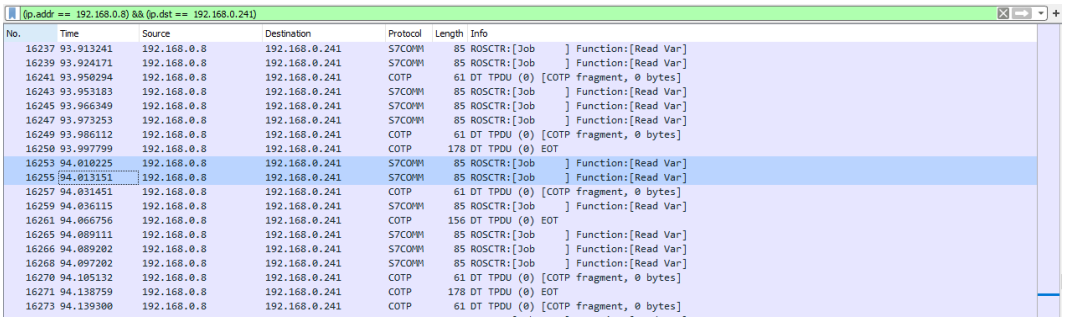


Figura 3.5 Análisis Wireshark PLC y NODE-RED

Por último, una vez se realiza la conexión y verificación mediante wireshark, se procede a realizar un debug en el NODERED para ver la actualización de las variables y su tiempo de actualización como se indica en la sección de DEBUG en la figura 3.6.

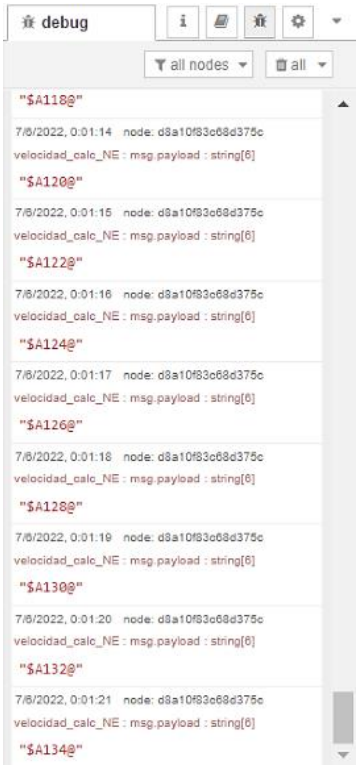


Figura 3.6 Debug actualización variables NODE-RED

3.1.2 Comunicación entre NODE-RED y TSC_LAB

Para la sección 3.1.2 se analiza para una sola tarjeta TSC_LAB en comunicación con el NODERED. Para el caso del NODE-RED se tiene un protocolo HTTP, en el cual se realiza GET y POST, explicado de forma gráfica en la figura 3.7, GET para obtener datos desde el NODE-RED hacia la tarjeta TSC_LAB y el POST para obtener datos desde la tarjeta TSC_LAB hacia el NODE-RED, por lo tanto, su análisis será distinto. Tanto el POST como el GET son implementados en el código de programación a ejecutarse a 1000 ms.

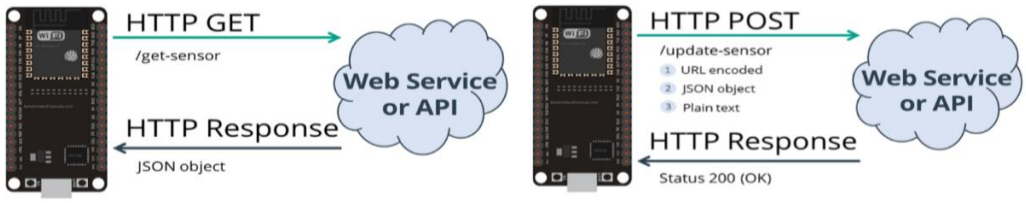


Figura 3.7 HTTP GET y HTTP POST

(Steph, 2020)

Igual que se hizo en la sección 3.1.1 se procede a realizar un análisis de tráfico entre el NODE-RED y la tarjeta TSC_LAB. Los resultados también arrojaron que existe comunicación TCP/IP y que el tiempo de comunicación es 2 ms como se indica en la figura 3.8.

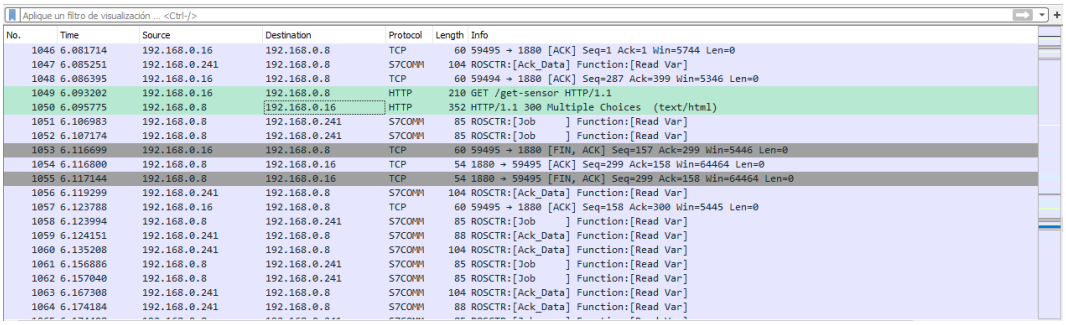


Figura 3.8 Análisis Wireshark TSC_LAB y NODE-RED

Por último, se procede a realizar un debug de forma independiente para el GET y para el POST como se indican en la figura 3.9 y 3.10 respectivamente. Para el caso de análisis del GET se activa el msg.payload del GET y se desactiva el msg.payload del POST, lo mismo sucede para el caso de análisis del POST donde se hace de forma invertida.

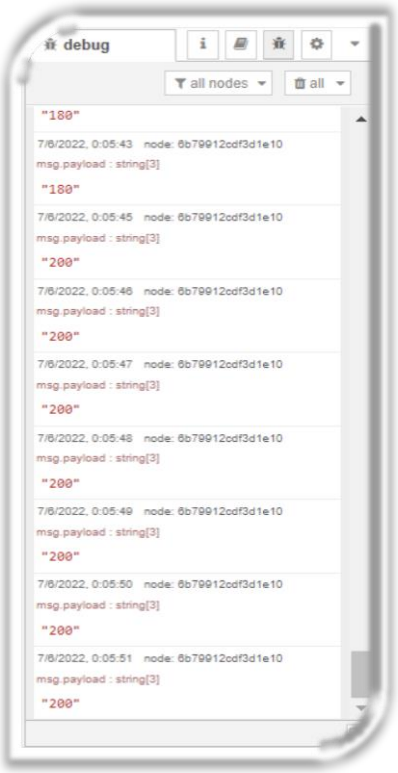


Figura 3.9 Debug HTTP GET

Figura 3.10 Debug HTTP POST

3.1.3 Comunicación entre HMI y PLC

Y por último tenemos la comunicación entre el HMI y el PLC, esta conexión es similar a la conexión de la sección 3.1.1, en donde se establece que variables se van a usar del PLC y con cuales se van a unir en el HMI, además se debe especificar si se desea escribir y leer en esa variable o solamente una de las dos opciones. Esta configuración se puede apreciar en la figura 3.11. Para este caso no es necesario configurar el puerto del PLC en la conexión del HMI.

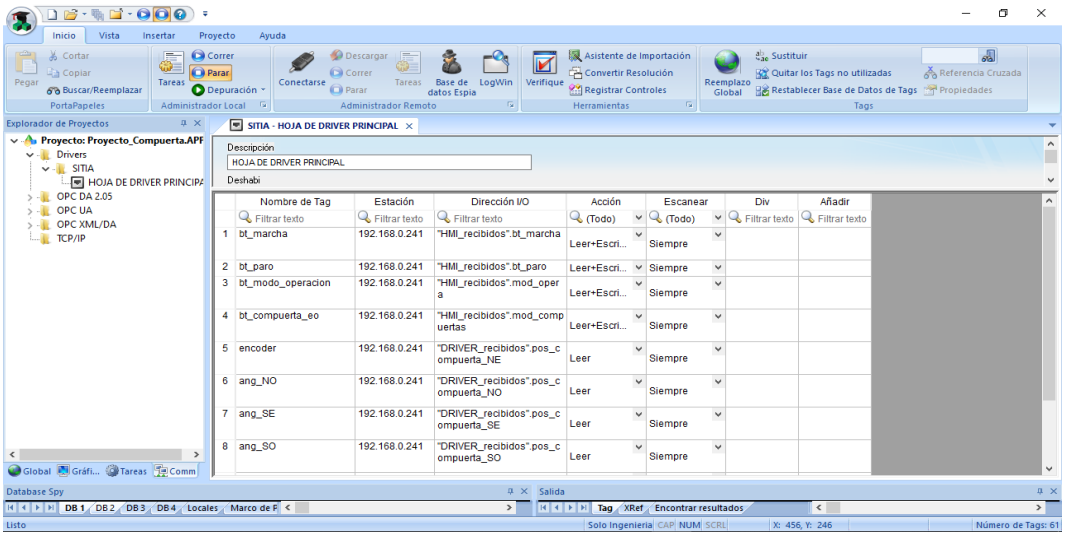


Figura 3.11 Configuración IETH HMI PLC

Luego se procede a verificar la conexión entre el HMI y el PLC realizando un análisis de tráfico mediante el uso del wireshark. Los resultados también arrojaron que existe comunicación TCP/IP y que el tiempo de comunicación es 0.9 ms como se indica en la figura 3.12.

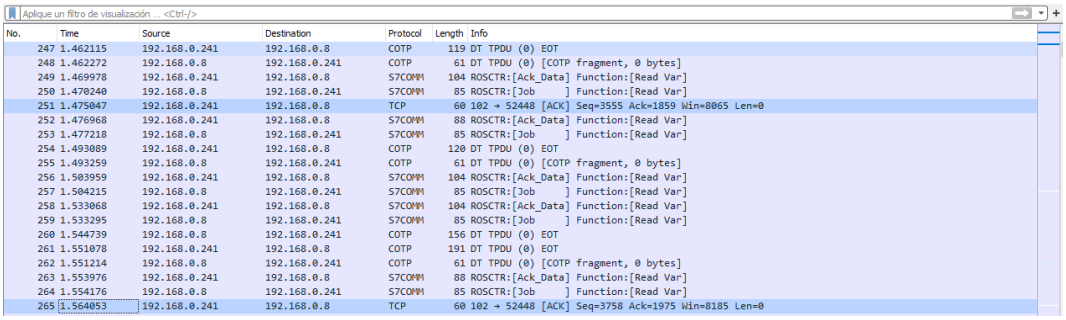


Figura 3.12 Análisis Wireshark HMI y PLC

Y al igual que en las anteriores secciones, se verifica en el HMI la conexión existente entre el HMI y el PLC, este se puede verificar viendo la interconexión entre los tags tanto del PLC como del HMI, en la figura 3.13 se puede verificar el estatus de OK en la conexión.

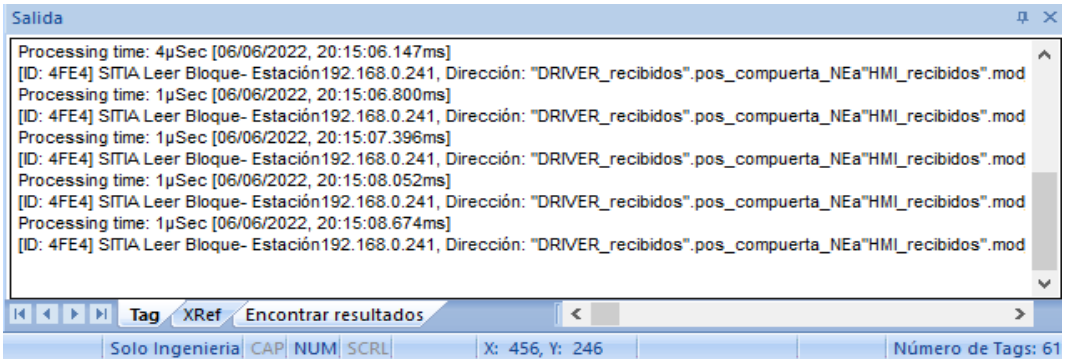


Figura 3.13 Debug conexión HMI y PLC

3.1.4 Comunicación del Sistema

En las secciones anteriores del capítulo 3 se procedió a probar las conexiones entre módulos las cuales dieron resultados de que si existía conexión. Luego de los resultados obtenidos se procedió a realizar la comunicación de todo el sistema trabajando en condiciones normales, en nuestro caso en una operación de apertura o de cierre de las compuertas. Los resultados son los deseados y es factible realizar el frenado dinámico esperado usando las comunicaciones detalladas en el capítulo 2.

Con las comunicaciones testeadas y funcionando de manera correcta se procede al testeo del frenado dinámico, el cual se define en la sección 3.2.

3.2 Frenado dinámico

Para el proyecto de titulación el frenado dinámico está basado en un controlador PID que ayuda a mantener la velocidad del motor en un valor definido por el PLC como se explicó en la sección 2.4.2.3 y para esta implementación se requiere verificar el funcionamiento de la planta en condiciones ideales y luego verificar el funcionamiento mediante el uso del PLC. Para trabajar en condiciones ideales se usa el software MATLAB como software de prueba, con la licencia estudiantil.

3.2.1 Controlador en Lazo Cerrado

Una vez calculados los valores kp, kd y ki en la sección 2.6 se procede a testear el controlador para ver su comportamiento, para esto se usa primero el MATLAB como prueba en condiciones ideales y luego se procede a testearlo en el PLC; las pruebas de respuesta escalón y respuesta a perturbación se realizan en MATLAB y se indican en las secciones 3.2.4.1 y 3.2.4.2.

3.2.1.1 Respuesta Escalón

Para realizar la respuesta escalón de nuestra planta usando el controlador PID obtenido en la sección 3.2.3 se necesita desarrollar en simulink la planta como se indica en la figura 3.25, esta planta es en lazo cerrado y un bloque está reservado para el controlador PID.

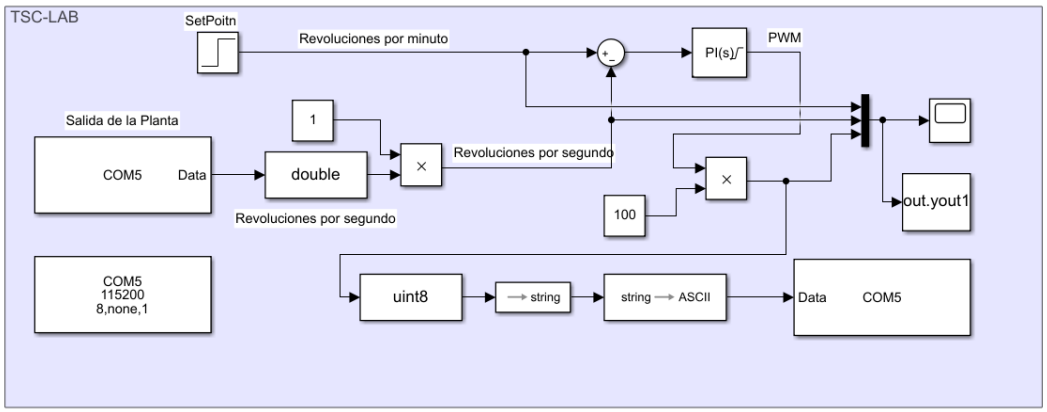


Figura 3.14 Planta Lazo Cerrado

En el bloque PID vamos a ingresar los valores de kp, kd y ki obtenidos, como se detalla en la figura 3.26.

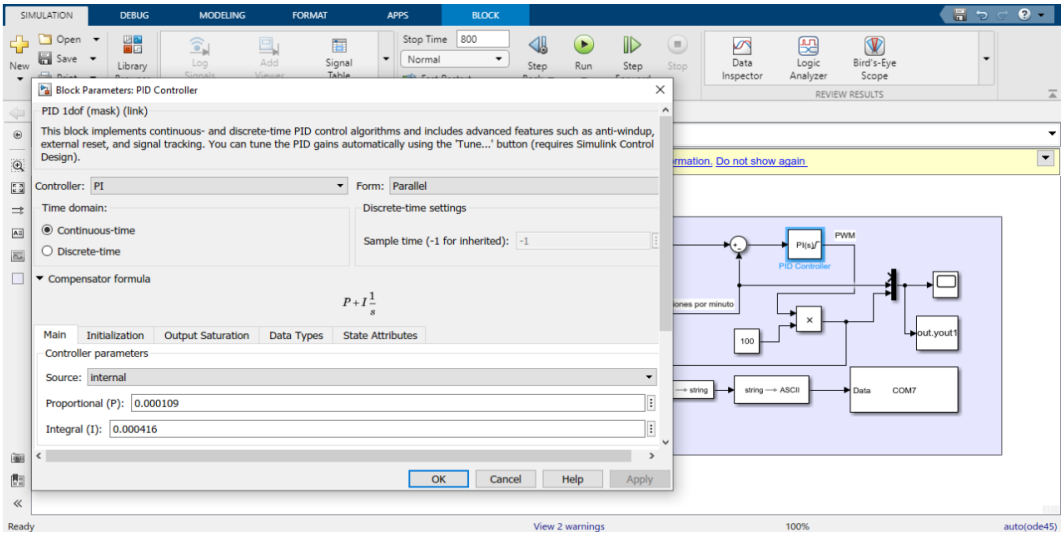


Figura 3.15 Controlador PID valores

Antes de simular se debe compilar un nuevo código en la tarjeta TSC_LAB (véase apéndice T) para que esta pueda ser compatible a la planta en lazo cerrado, la diferencia es que esta planta envía retroalimentación de la velocidad del motor en rps.

Primero se prueba en la planta de lazo cerrado de simulink dejando las revoluciones por segundo (rps) en 70 como se indica en la figura 3.27 y el PID debe conseguir estabilizar la velocidad del motor en esa frecuencia.

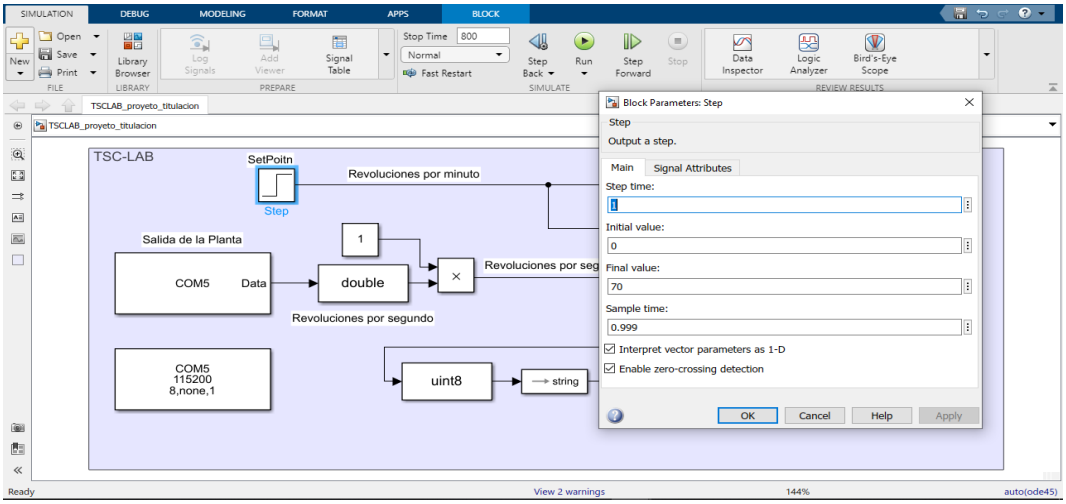


Figura 3.16 Set point de 70 rps

Al compilar se puede observar que el controlador si actua llevando al motor a mantenerse en las 70 rps como se indica en la figura 3.28. La línea azul son las rps medidas por el tacómetro, mientras que la línea de color rojo es el controlador PID, la línea amarilla es el setpoint que para este caso es 70 rps.

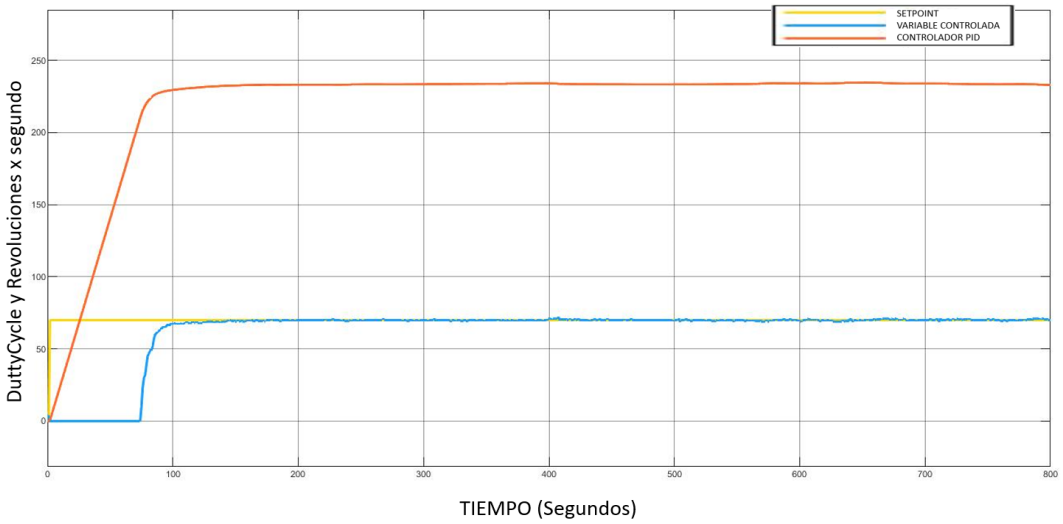


Figura 3.17 Controlador PID actuando

Es necesario indicar que el motor debe superar el valor del DutyCycle en 200 para que el motor pueda arrancar, por lo tanto, el motor comienza a funcionar en el segundo 72.509. El sistema es sobreamortiguado con un valor máximo de 72 rps transcurrido 407.6 segundos de inicio de proceso. También se puede obtener un tiempo de estabilización de 69.148 segundos desde el momento que entra en funcionamiento el motor. En la figura 3.29 se detalla las mediciones.

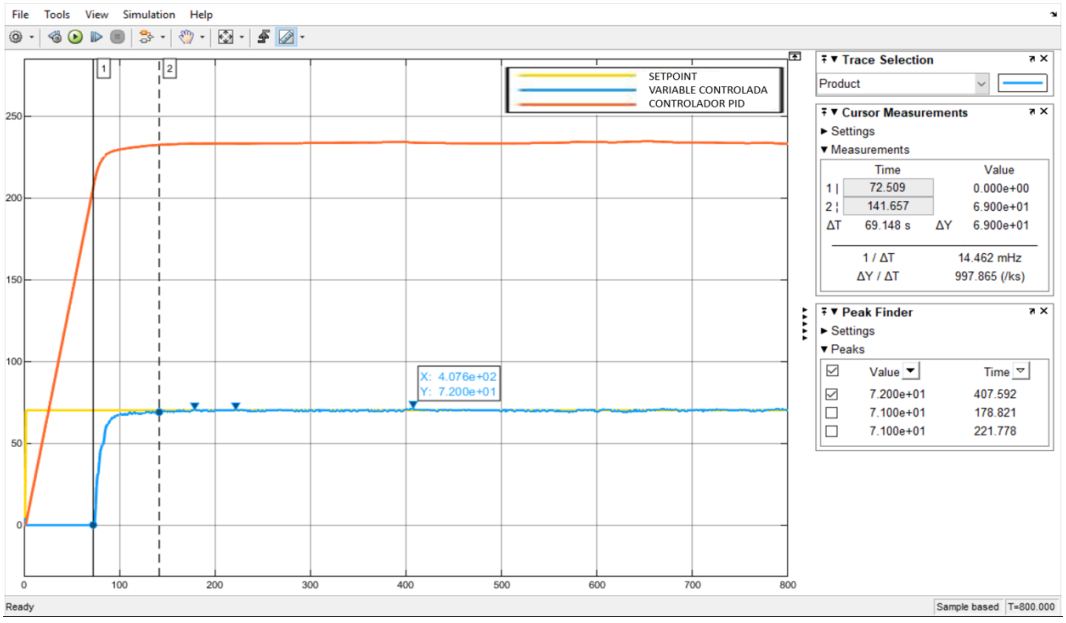


Figura 3.18 Controlador PID valores

3.2.1.2 Respuesta a frenado dinámico

Una vez que fue probado el controlador PID y analizando que responde de la forma esperada se procede a usar el mismo controlador en el PLC, para esto se ingresa las constantes kd, kp y ki en los parámetros del PID como se indica en la figura 3.30. El PID compact del TIA PORTAL indica que se debe ingresar los parámetros en tiempo continuo que para este proyecto fueron los que se obtuvieron.

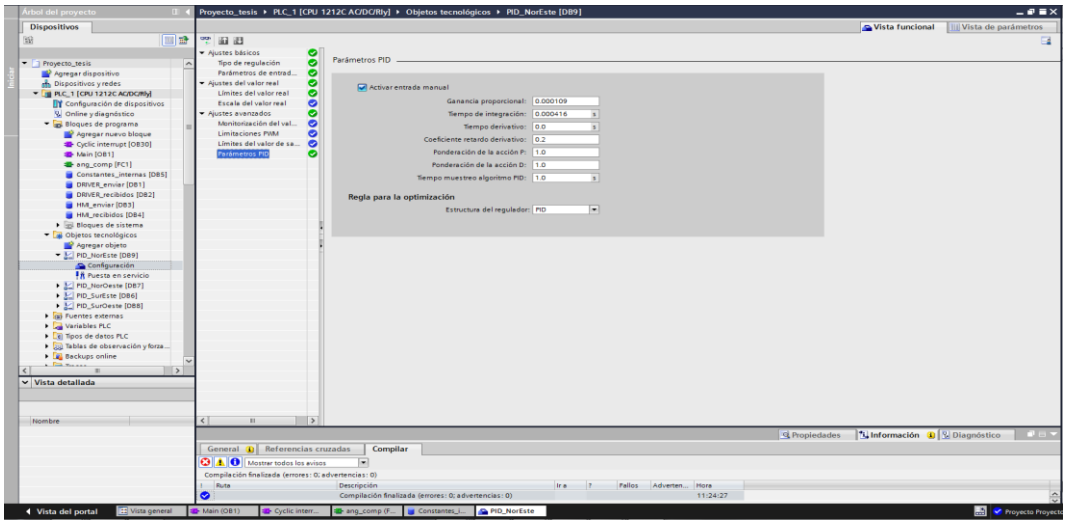


Figura 3.19 PID Compact kp, ki, kd

Al igual que se hizo en MATLAB se procede a testear el controlador PID para ver su comportamiento y se obtuvo un comportamiento similar al ya testeado en MATLAB como se indica en la figura 3.31 en donde se configura las rps en 33. Se puede ver como se estabiliza en 33 rps.

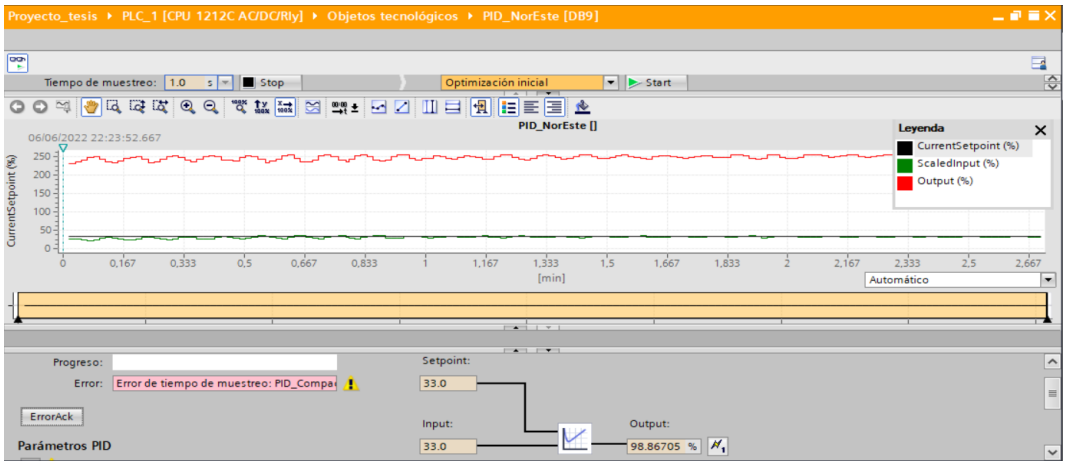


Figura 3.20 PID Compact kp, ki, kd

Una vez testeado el PID se procede a verificar el funcionamiento de todo el sistema, en apertura y cierre de compuertas. Para evaluar todo el funcionamiento de la compuerta se usa las compuertas sureste y noreste, al dar inicio a la apertura se puede ver que las compuertas inician en 0 grados y transcurrido el tiempo de apertura automáticamente se detienen en 90.10 grados para la compuerta sureste y 90.69 grados para la compuerta noreste como se detalla en el HMI de la figura 3.32 y la base de datos DB2 del PLC de la figura 3.33. Los grados máximos de tolerancia para una correcta compresión son 90 grados +-1 grado de tolerancia por lo que está dentro del margen de correcto funcionamiento.

Figura 3.21 Finalización de apertura compuertas este HMI

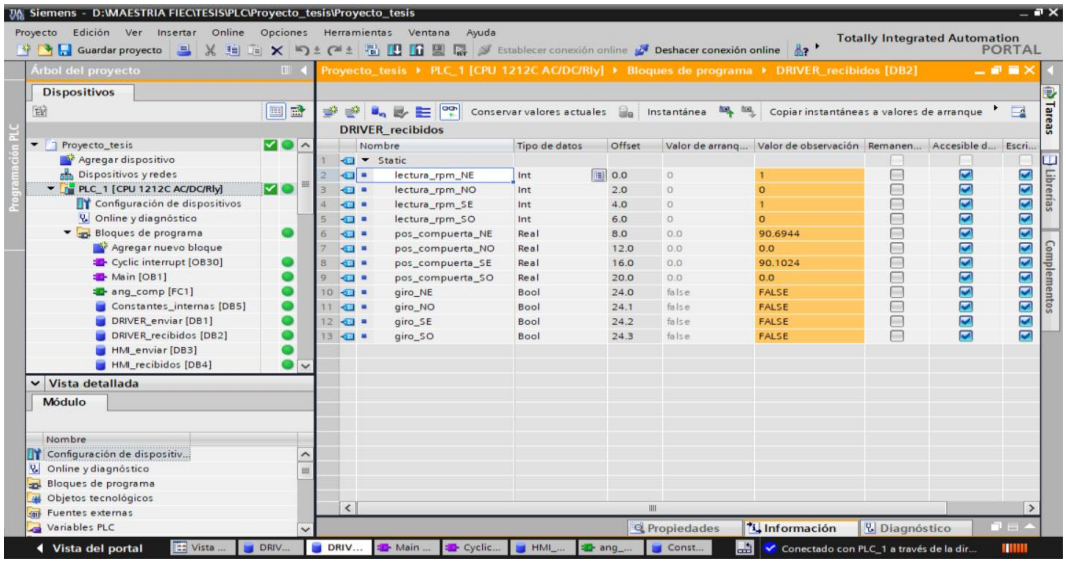


Figura 3.22 Finalización de apertura compuertas este PLC

Así se comprueba el correcto funcionamiento del frenado dinámico de las compuertas para una correcta compresión de los sellos mecánicos, todo esto mediante la automatización de la apertura.

También se procede a verificar el cierre de las compuertas sureste y noreste, desde la misma posición que se encontraban, 90.10 grados para la compuerta sureste y 90.69 grados para la compuerta noreste. Luego de iniciado el proceso de cierre, las compuertas se detienen automáticamente en 0.53 grados para la compuerta sureste y 0.50 grados para la compuerta noreste como se detalla en el HMI de la figura 3.34 y la base de datos DB2 del PLC de la figura 3.35.

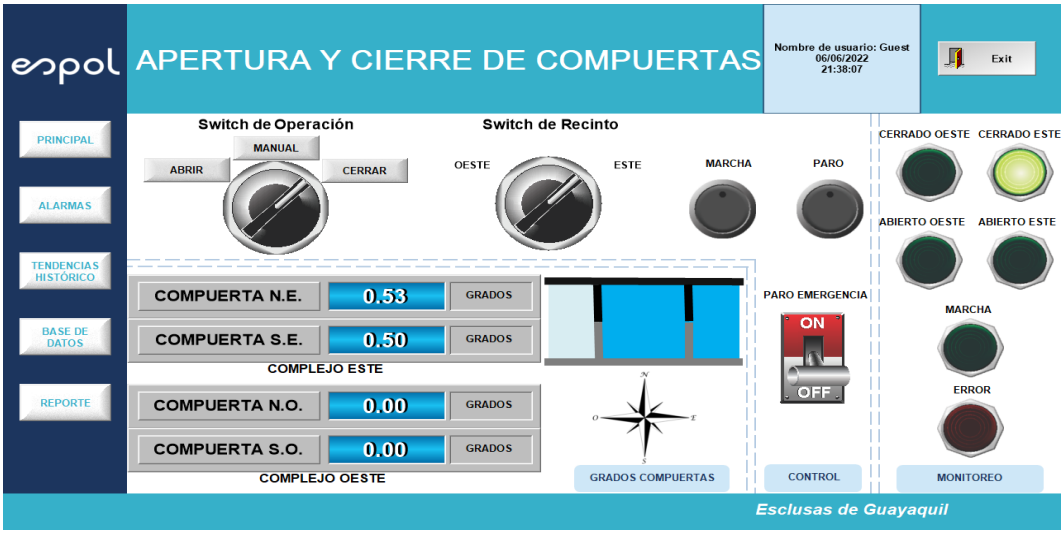


Figura 3.23 Finalización de cierre compuertas este HMI

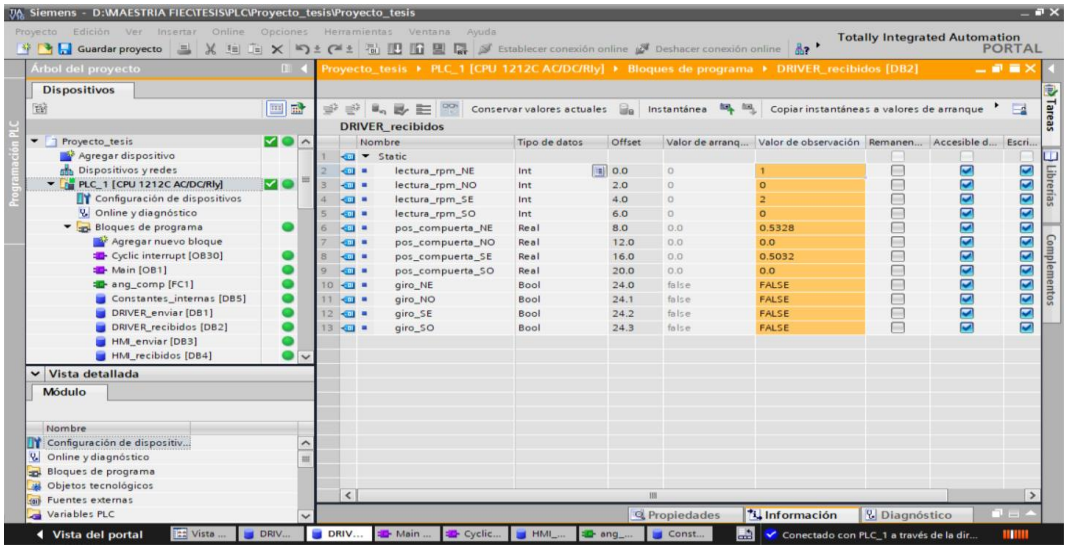


Figura 3.24 Finalización de cierre compuertas este PLC

Es entonces que con estas pruebas se comprueba el correcto funcionamiento del frenado dinámico de las compuertas para una correcta compresión de los sellos mecánicos, todo esto mediante la automatización del cierre.

3.2.1.3 Monitoreo y Control en la nube

Como parte del proyecto de titulación se tiene implementado el monitoreo y control desde la nube, en este caso mediante el uso de la aplicación de Telegram, en la cual se recibe las rpm medidas al momento del proceso de apertura o cierre de la compuerta y también se puede controlar la velocidad de apertura de las dos compuerta enviando el valor de rpm que queremos fijar como se indica en la figura 3.36.

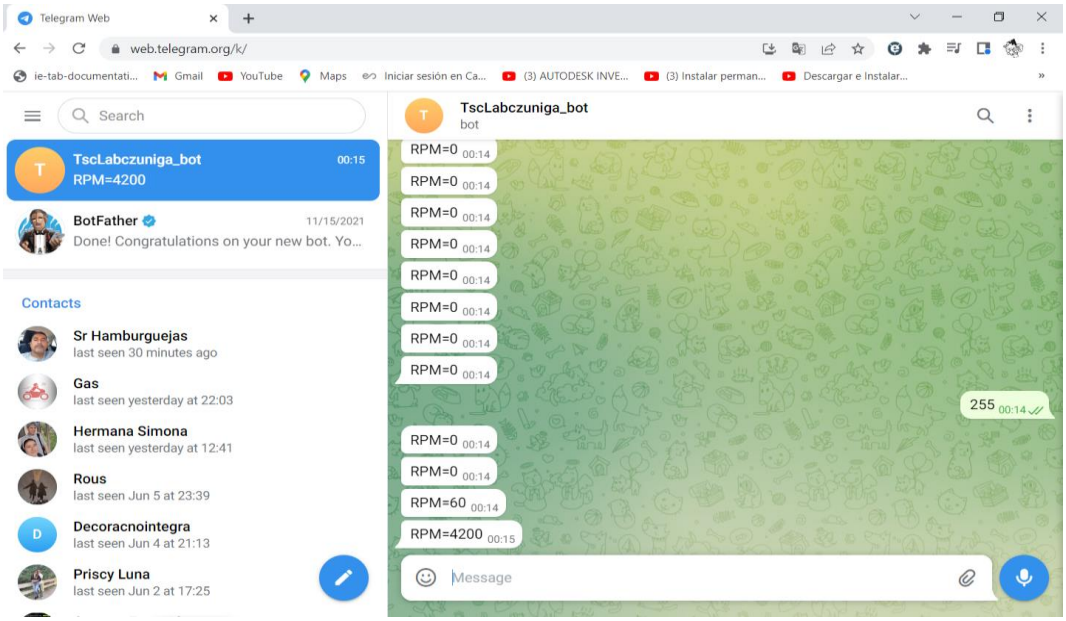


Figura 3.25 Consulta de posición angular de compuerta Telegram

3.3 Estudio económico

Considerando de ejemplo la esclusa de Guayaquil se puede concluir; que debido a que gran parte de la infraestructura ya está implementada (como es el caso del plc, variadores de frecuencia, botoneras, leds, motores y caja reductora), es rentable implementar la solución propuesta en este proyecto de titulación ya que los gastos de inversión estarían orientados a la compra de los componentes de la interfaz de control, en este caso del Panel PC a utilizarse para la visualización del HMI, los tacómetros y los encoders absolutos entre otros materiales y accesorios.

El diseño de la interfaz de control ubicado en la caseta principal implicaría una inversión de aproximadamente US $5600 como se detalla en la tabla 3.1. El presupuesto es calculado con valores referenciales de mercado local usando cotizaciones actualizadas.

Tabla 3.1 Presupuesto de Interfaz de Control

Nº	Ítem	Cantidad	Precio u.	Total
1	PLC	1	900,00	900,00
2	Switch	1	500,00	500,00
3	Led rojo	1	20,00	20,00
4	Led verde	2	20,00	40,00
5	PC	1	3000,00	3000,00
6	Encoder absoluto	1	900,00	900,00
7	Paro Emergencia	1	50,00	50,00
8	Bornera Simple	10	3,00	30,00
9	Bornera de tierra	2	5,00	10,00
10	RIEL DIN	1	5,00	5,00
11	Accesorios	1	150,00	150,00
TOTAL (US $)				**5605,00**

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Las conclusiones se dividen en dos secciones, la primera en comunicaciones y la segunda en frenado dinámico.

4.1.1 Comunicaciones

- La comunicación inalámbrica es válida para sensores/actuadores siempre y cuando los dispositivos tengan las capacidades hardware necesarias para funcionar correctamente en el manejo de las velocidades de transmisión, así se evita alguna pérdida de información que pueda perjudicar el correcto funcionamiento del sistema. Para el caso de este proyecto las capacidades hardware de los dispositivos son válidas para la comunicación.
- El servidor IoT implementado en este proyecto cumplió correctamente con su función de la comunicación de todos los dispositivos, no hubo pérdidas de información y además el servidor IoT permite que este proyecto pueda ser controlado remotamente desde cualquier parte del mundo. El servidor fue testeado en comunicaciones con velocidades de transmisión de tramas de hasta 0.5 ms y respondió sin ningún problema.

4.1.2 Frenado Dinámico

- En este proyecto se obtuvo la función de transferencia de la planta utilizando técnicas de identificación de sistemas que utilizan mediciones reales del proceso, y que demostró ser un método efectivo y rápido para el diseño de los controladores
- Para un controlador PID de un motor DC la constante derivativa no es tan relevante como si lo es la constante integradora, ya que esta última puede obtener mejoras en tiempo de respuesta y error estacionario por si sola.

- Mientras más elementos se consideren en el sistema, estos contribuirán a la dinámica del mismo, por lo tanto, esto conllevará a que existan más polos, aunque algunos de ellos pueden no ser dominantes.
- El control de frenado dinámico puede ser implementado en cualquier tipo de esclusas con compuertas electromecánicas siempre y cuando se pueda controlar la velocidad del motor, se reciba la posición de la compuerta y la velocidad en revoluciones por minuto.
- El motor de la tarjeta TSC_LAB pudo ser controlado en velocidad de forma deseada, aunque este motor no sea de características industriales ni posea certificaciones, demostrando que el controlador PID obtenido funciona correctamente incluso con equipos de gama baja.

4.2 Recomendaciones

- Es necesario implementar el sistema usando encoders absolutos para la obtención de los grados de apertura de la compuerta, es la forma más fiable de obtener los grados.
- Dentro de todo proyecto se pone a consideración también la implementación de algún sistema de redundancia, siempre y cuando esté acorde al presupuesto del proyecto; un buen sistema de redundancia para este sistema es ubicar finales de carreras en los extremos de las compuertas en cierre y apertura, lo cual implicaría la compra de 2 finales de carrera para cada compuerta dando un total de 8 finales de carrera, esto para evitar que de llegar a dañarse el sistema de frenado dinámico estos finales de carrera frenen de forma abrupta las compuertas, no es lo recomendable pero sirve como una protección de último recurso.
- Tratar siempre de reducir los tiempos en la comunicación, esto ayudará a que en caso de existir latencia en la comunicación se reduzca el impacto en el funcionamiento del sistema.
- Evitar usar encoders incrementales o velocidad para obtener la posición angular de la compuerta, ya que esta forma requiere una periódica calibración y tener que encerar el contador cada cierto tiempo.
- Para el prototipo, al momento de contabilizar la vuelta del motor es mejor realizarla mediante una interrupción para evitar alguna perdida teniendo

también en consideración la velocidad de procesamiento en Hz del microcontrolador.

- Procurar usar RTOS al momento de ejecutar procesos críticos en tiempo en el microcontrolador, con esto se puede estar seguro que se ejecutarán en el tiempo deseado.

BIBLIOGRAFÍA

5. BIBLIOGRAFÍA

Asanza, V., & Chica, K. (7 de Julio de 2021). *TSC LAB*. Obtenido de https://tsc-lab.blogspot.com/p/data-sheet.html?spref=tw

Benítez Machado, D., Anías Calderón, C., & Plasencia Moreno, L. (2016). Propuesta de arquitectura para Internet de las Cosas. *ResearchGate*, 3.

Cardona, J., Leal, J., & Ustariz, J. (diciembre de 2020). SCIELO. En *Modelado matemático de caja blanca y negra en educación en ingeniería* (pág. 118). Bogotá: Formación Universitaria. Obtenido de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-50062020000600105

Erbes, A., Gutman, G., Lavarello, P., & Verónica, R. (2019). *Industria 4.0 - Oportunidades y desafíos para el desarrollo productivo de la provincia de Santa Fe.* Santiago: Naciones Unidas.

Festo Didactic SE. (2021). *Automatización industrial - Sistemas de aprendizaje para la formación técnica.* Alemania: Festo Didactic SE.

Fournies, A. (2015). Modelos ARMA y Box and Jenkins. Valparaíso.

infoPLC. (18 de noviembre de 2015). *infoplc*. Obtenido de https://www.infoplc.net/actualidad-industrial/item/102902-isa101-hmi#:~:text=La%20norma%20pretende%20proporcionar%20orientaci%C3%B3n,todas%20las%20condiciones%20de%20funcionamiento.

Liñán Colina, A., Vives, A., Bagula, A., Zennaro, M., & Pietrosemoli, E. (2015). *Internet de las Cosa.* ICTP.

Moisés Barrio, A. (2018). *Internet de las cosas.* Madrid: REUS.

Pérez-López, E. (2015). Los sistemas SCADA en la automatización industrial. *Revista Tecnológica en Marcha*, 14.

Rodríguez Penin, A. (2007). *Sistemas SCADA Guía Práctica.* Marcombo.

S. Escalante, R., & Sivori, G. (2018). *Diseño de vías navegables.* ESCUELA DE GRADUADOS EN INGENIERIA PORTUARIA.

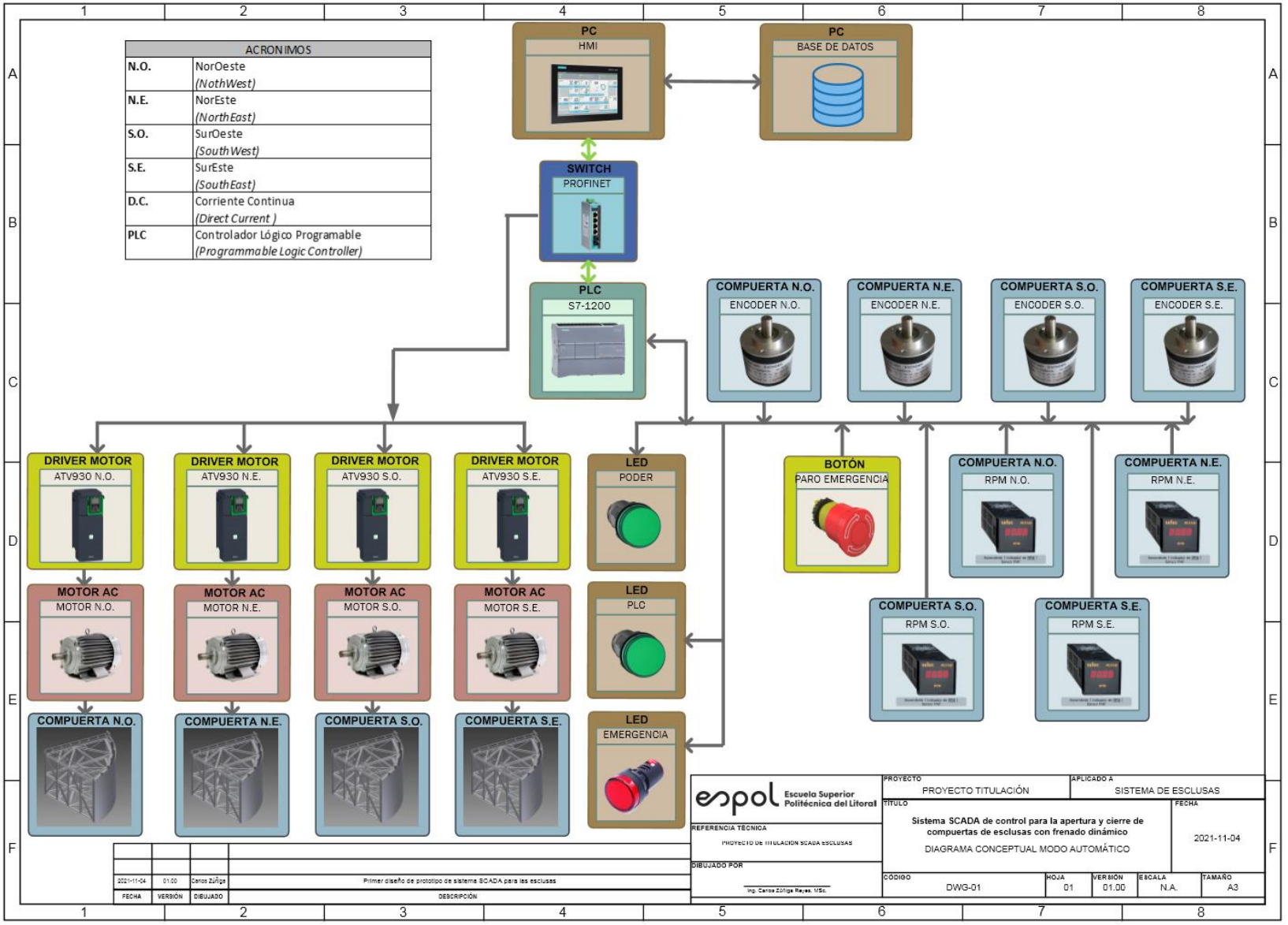
SEIKA. (3 de agosto de 2019). *SEIKA*. Obtenido de https://www.seika.com.mx/5-niveles-de-la-automatizacion-industrial/

Steph. (8 de abril de 2020). *RANDOM NERD TUTORIALS*. Obtenido de https://randomnerdtutorials.com/esp32-http-get-post-arduino/

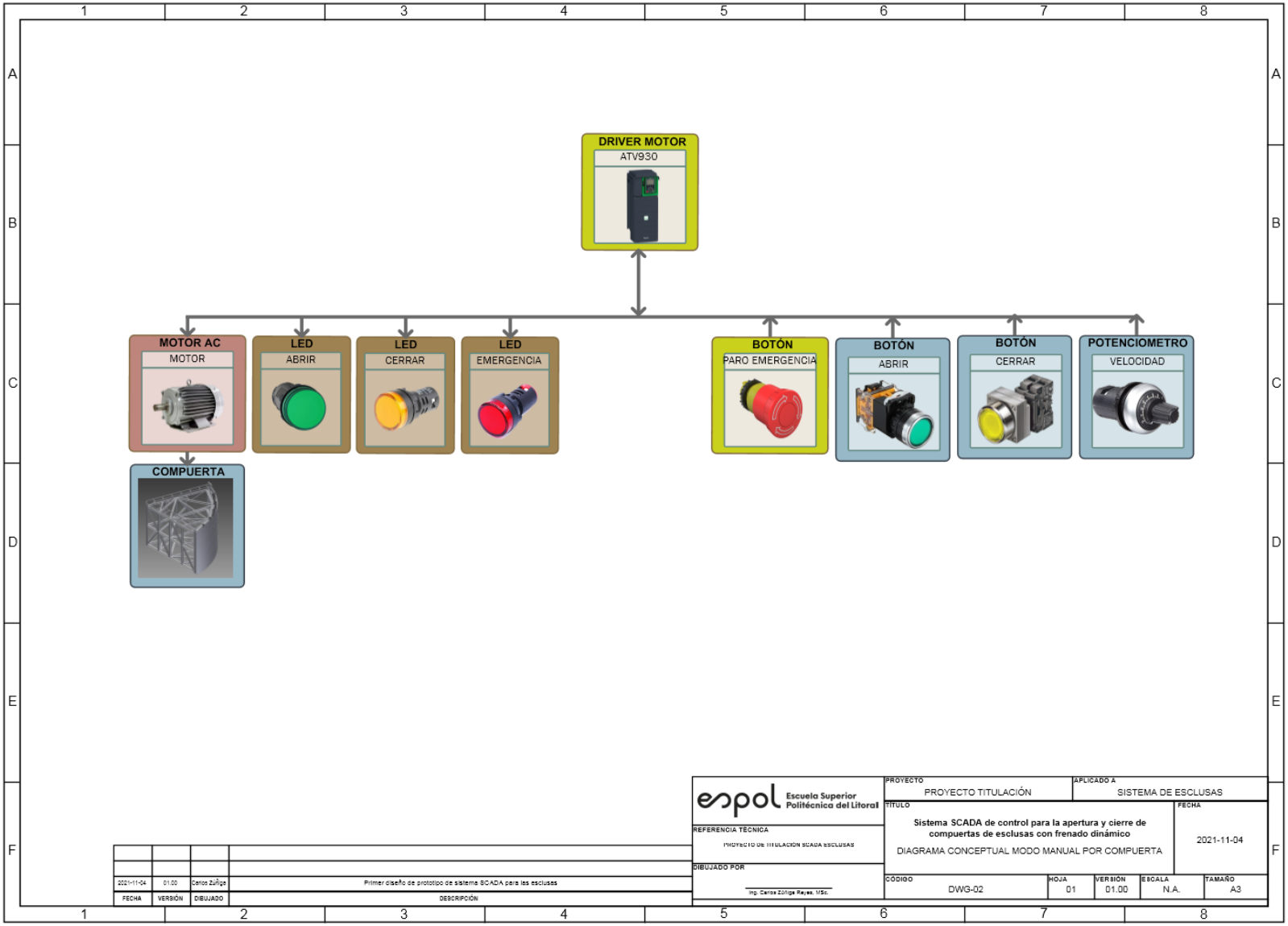
TECNICAL. (s.f.). Guía básica sobre circuitos neumáticos de seguridad para satisfacer la norma ISO 13849. 35. MANRESA, España.

APÉNDICES

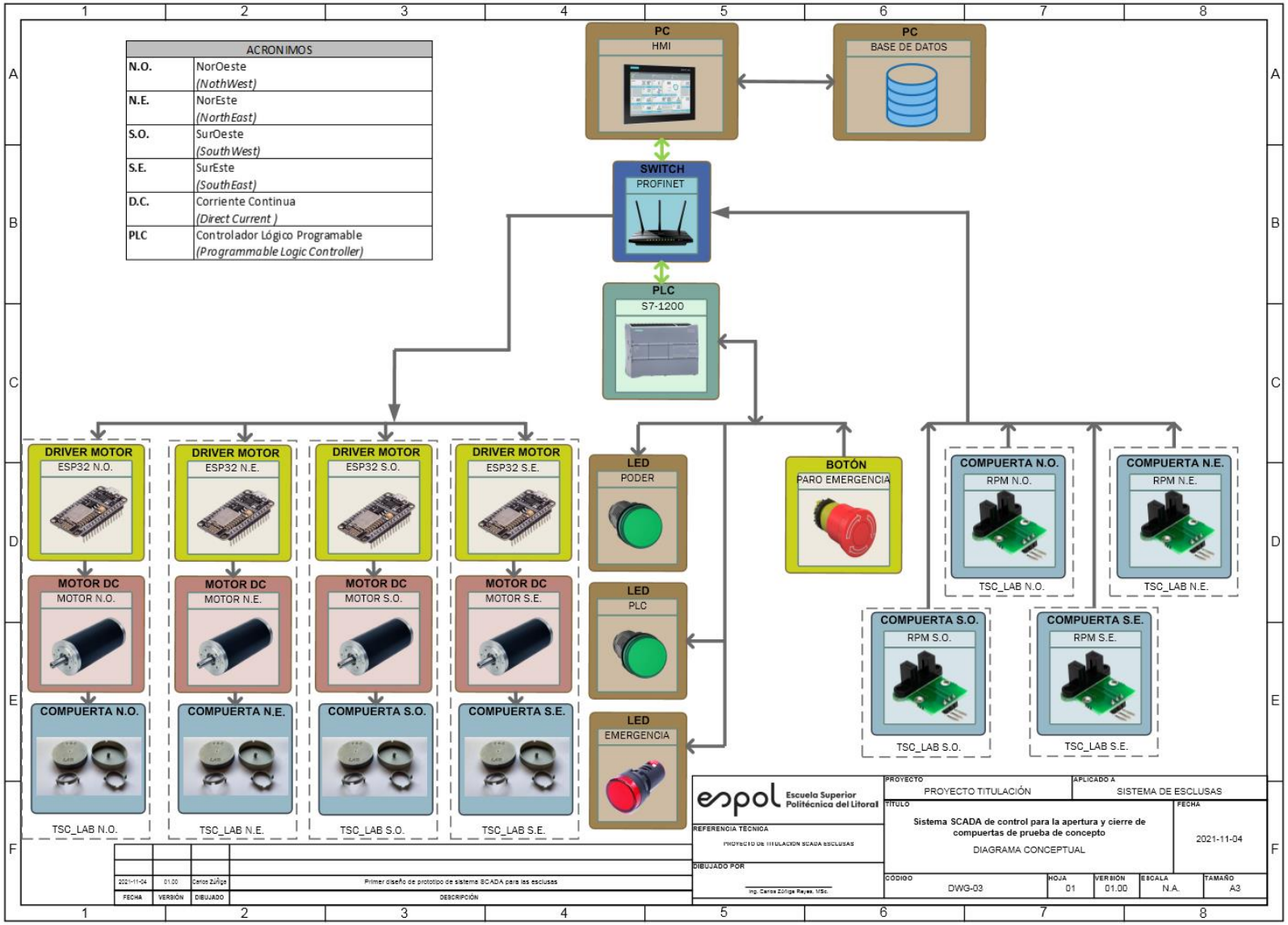
APÉNDICE A. PLANO 1 Diagrama Conceptual Modo Automático



APÉNDICE B. PLANO 2 Diagrama Conceptual Modo Manual

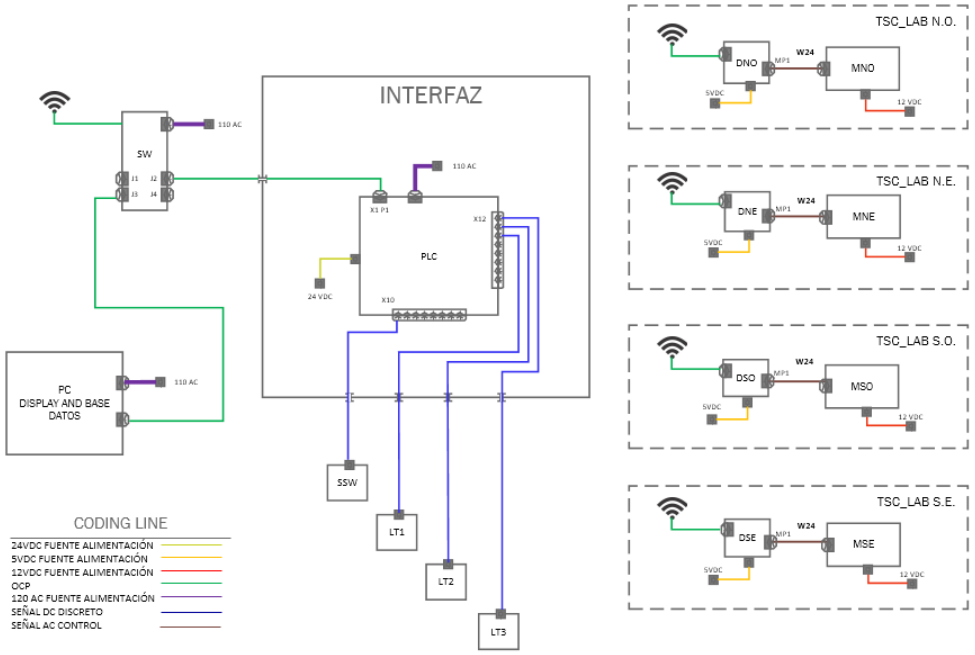


APÉNDICE C. PLANO 3 Diagrama Conceptual de Prototipo



APÉNDICE D. PLANO 4 Diagrama de Bloques de Prototipo

ACRONIMOS	
DNO	Driver NorOeste
DNE	Driver NorEste
DSO	Drive SurOeste
DSE	Drive SurEste
MNO	Motor NorOeste
MNE	Motor NorEste
MSO	Motor SurOeste
MSE	Motor SurEste
SSW	Switch de seguridad *(Safety Switch)*
LT	Luz terminal
PLC	Controlador Lógico Programable *(Programmable Logic Controller)*
SW	Switch de Flujo *(Flow Switch)*
TB	Bloque de Terminales *(Terminal Block)*



2021-11-04	01.00	Carlos Zúñiga	Primer diseño de prototipo de sistema SCADA para las esclusas
FECHA	VERSIÓN	DIBUJADO	DESCRIPCIÓN

espol Escuela Superior Politécnica del Litoral

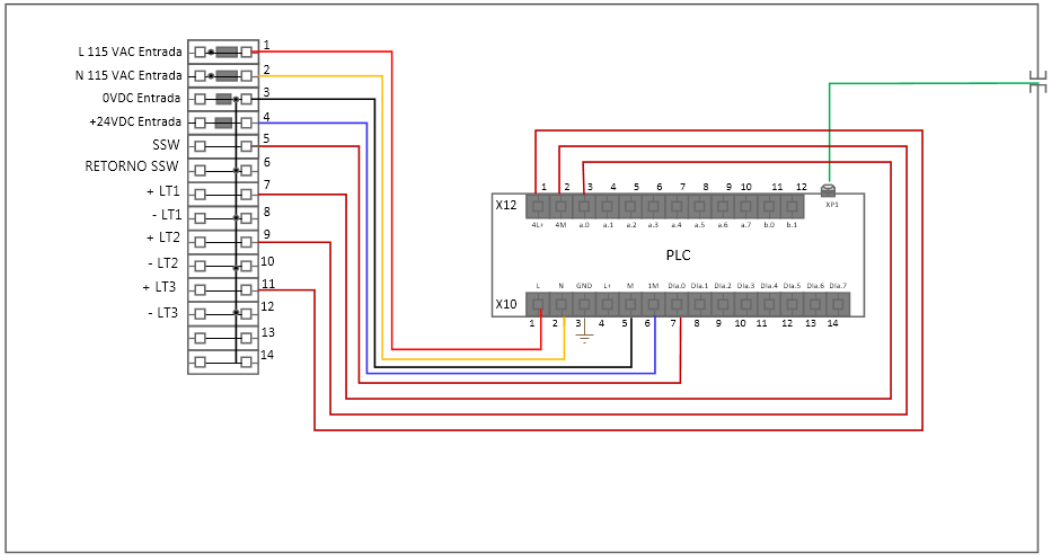
REFERENCIA TÉCNICA: PROYECTO DE TITULACIÓN SCADA ESCLUSAS

DIBUJADO POR: Ing. Carlos Zúñiga Reyes, MSc.

PROYECTO	PROYECTO TITULACIÓN
APLICADO A	SISTEMA DE ESCLUSAS
TÍTULO	Sistema SCADA de control para la apertura y cierre de compuertas de esclusas con frenado dinámico DIAGRAMA BLOQUES
FECHA	2021-11-04
CÓDIGO	DWG-04
HOJA	01
VERSIÓN	01.00
ESCALA	N.A.
TAMAÑO	A3

APÉNDICE E. PLANO 5 Diagrama de Conexiones Interior de Prototipo

INTERFAZ DE CONTROL - INTERIOR



ACRONIMOS	
DNO	Driver NorOeste
DNE	Driver NorEste
DSO	Drive SurOeste
DSE	Drive SurEste
MNO	Motor NorOeste
MNE	Motor NorEste
MSO	Motor SurOeste
MSE	Motor SurEste
PLC	Controlador Lógico Programable *(Programmable Logic Controller)*
SW	Switch de Flujo *(Flow Switch)*
TB	Bloque de Terminales *(Terminal Block)*

CODING LINE

24VDC POWER SUPPLY
0VDC POWER SUPPLY
ETHERNET
24VDC DISCRETE CONTROL
115 VOC
Neutro

FECHA	VERSIÓN	DIBUJADO	DESCRIPCIÓN
2021-11-04	01.00	Carlos Zúñiga	Primer diseño de prototipo de sistema SCADA para las esclusas

espol Escuela Superior Politécnica del Litoral

REFERENCIA TÉCNICA: PROYECTO DE TITULACIÓN SCADA ESCLUSAS

DIBUJADO POR: Ing. Carlos Zúñiga Reyes, MSc.

PROYECTO	APLICADO A
PROYECTO TITULACIÓN	SISTEMA DE ESCLUSAS

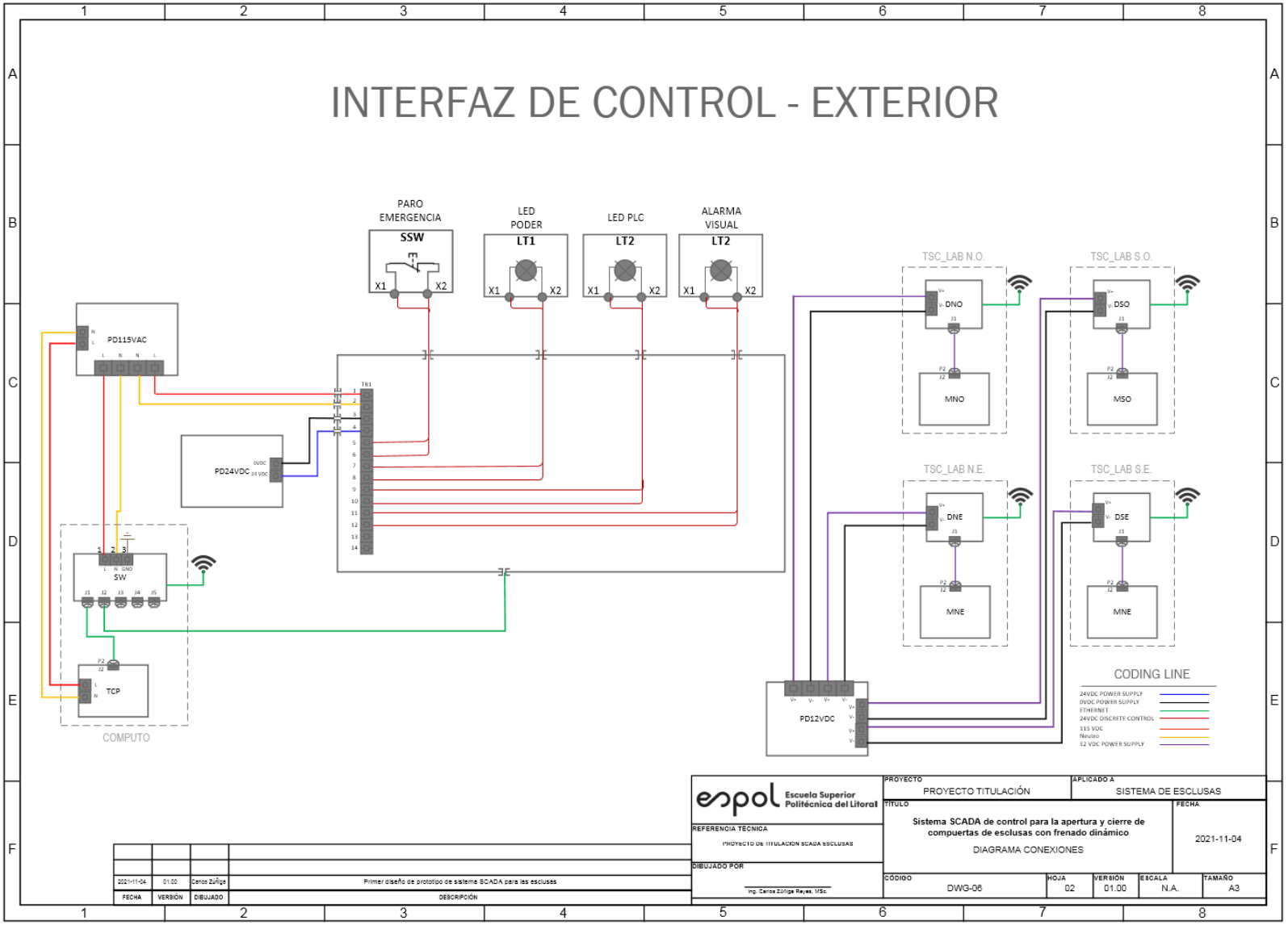
TÍTULO: Sistema SCADA de control para la apertura y cierre de compuertas de esclusas con frenado dinámico

DIAGRAMA CONEXIONES

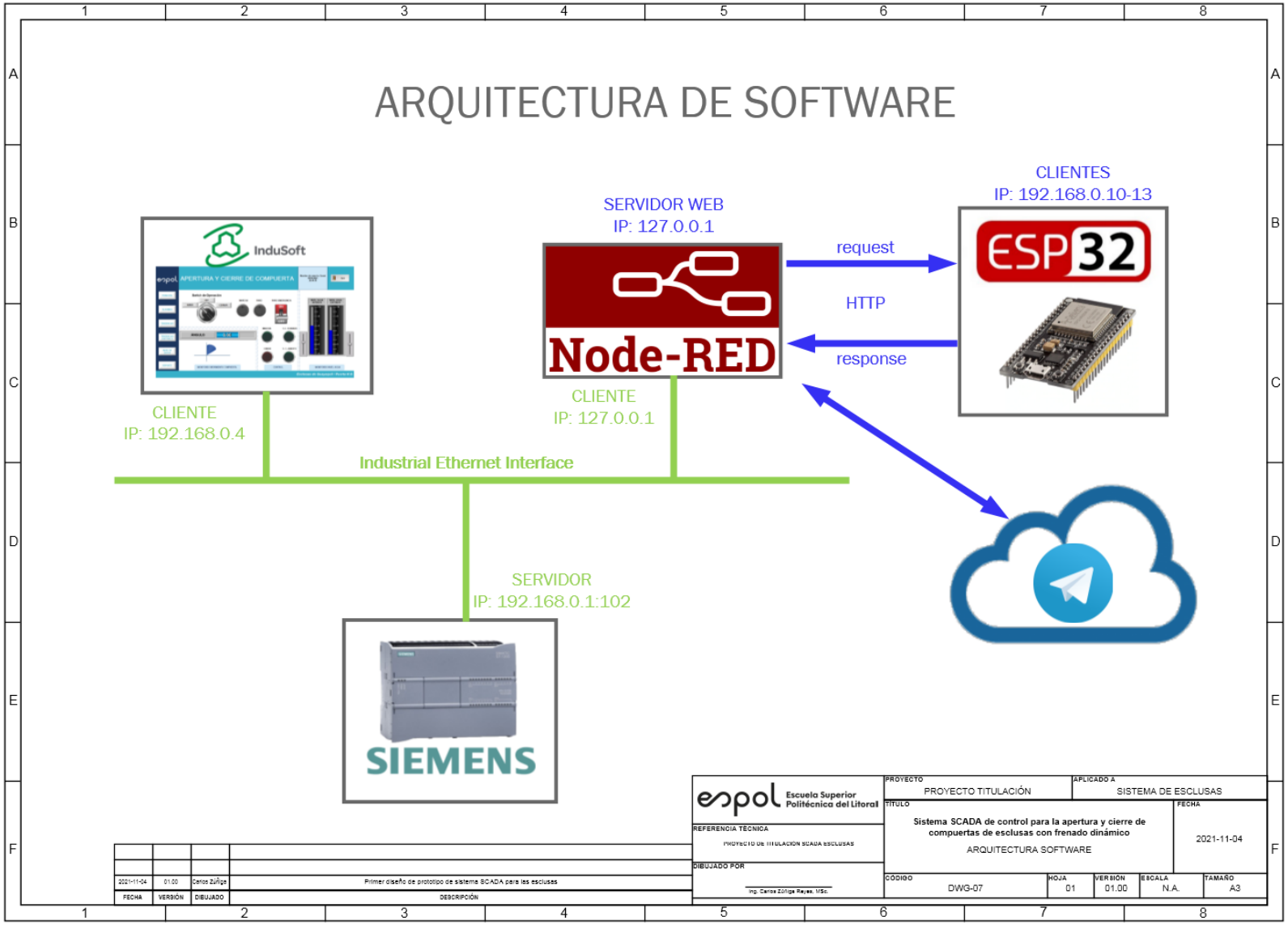
FECHA: 2021-11-04

CÓDIGO	HOJA	VERSIÓN	ESCALA	TAMAÑO
DWG-05	01	01.00	N.A.	A3

APÉNDICE F. PLANO 6 Diagrama de Conexiones Exterior de Prototipo



APÉNDICE G. PLANO 7 Diagrama de Arquitectura de Software



APÉNDICE H. Scripts HMI principal

```
'Las variables disponibles sobre esta pantalla pueden ser declaradas e inicializadas aquí.


'Los Procedimientos disponibles sobre esta pantalla pueden ser implementados aquí


'Este procedimiento es ejecutado solo una vez cuando esta pantalla es abierta.
Sub Screen_OnOpen()

End Sub

'Este procedimiento es ejecutado continuamente mientras esta pantalla está abierta.
Sub Screen_WhileOpen()

	If $EMERGENCIA = 0 Then 'Se pregunta si esta activado el botón de emergencia


	If $bt_compuerta_eo Then

			If $bt_marcha Then
				$bandera_marcha = 1
				$bandera_marcha_SE = 1
			End If

			If $bt_paro Then
				$bandera_marcha = 0
				$bandera_marcha_SE = 0
			End If


				Select Case $bt_modo_operacion
					Case 0

	If $bandera_marcha And $encoder < $lim_abierto_min And Not $var_abierto Then
		$encoder = $encoder + 1
		$led_marcha = 1
		$fin_cerrado = 0
		End If

	If $bandera_marcha_SE And $ang_SE < $lim_abierto_SE_min And Not $var_abierto_SE Then
		'$ang_SE = $ang_SE + 1
		$led_marcha = 1
		$fin_cerrado_SE = 0
	End If

	If $bt_paro Then
		$led_marcha = 0
	End If

	If $encoder >= $lim_abierto_min And $encoder <= $lim_abierto_max Then
		$var_abierto = 1
		$bandera_marcha = 0



		Else
		$var_abierto = 0
		$fin_abierto = 0
```

```
            End If

        If $ang_SE >= $lim_abierto_SE_min And $ang_SE <= $lim_abierto_SE_max Then
            $var_abierto_SE = 1
            $bandera_marcha_SE = 0


        Else
        $var_abierto_SE = 0
        $fin_abierto_SE = 0
        End If


If $ang_SE >= $lim_abierto_SE_min And $ang_SE <= $lim_abierto_SE_max And $encoder >=
$lim_abierto_min And $encoder <= $lim_abierto_max Then
            $bt_paro = 0
            $fin_abierto = 1
            $fin_abierto_SE = 1
            $led_marcha = 0
            $sec_centro = 3

            End If



            Case 2

            Case 1

        If $bandera_marcha And $encoder > $lim_cerrado_min And Not $var_cerrado Then
            '$encoder = $encoder - 1
            $led_marcha = 1
            $fin_abierto = 0
            End If

        If $bandera_marcha_SE And $ang_SE > $lim_cerrado_SE_min And Not $var_cerrado_SE Then
                '$ang_SE = $ang_SE - 1
                    $led_marcha = 1
                $fin_abierto_SE = 0
                End If


            If $bt_paro Then
            $led_marcha = 0
        End If



        If $encoder >= $lim_cerrado_max And $encoder <= $lim_cerrado_min Then
                            $var_cerrado = 1
                            $bandera_marcha = 0



                                        Else
                                            $var_cerrado = 0
                                            $fin_cerrado = 0
                                        End If
```

```
If $ang_SE >= $lim_cerrado_SE_max And $ang_SE <= $lim_cerrado_SE_min Then
                                        $var_cerrado_SE = 1
        $bandera_marcha_SE = 0

                                        Else
                                        $var_cerrado_SE = 0
                                        $fin_cerrado_SE = 0
                                End If

        If $ang_SE >= $lim_cerrado_SE_max And $ang_SE <= $lim_cerrado_SE_min And $encoder >=
$lim_cerrado_max And $encoder <= $lim_cerrado_min Then
                                        $bt_paro = 0
                                        $fin_cerrado = 1
                                        $fin_cerrado_SE = 1
                                        $led_marcha = 0
                                        End If

                                End Select

        End If


                If $bt_compuerta_eo = 0 Then

                        If $bt_marcha Then
                        $bandera_marcha_NO = 1
                        $bandera_marcha_NE = 1
                End If

                If $bt_paro Then
                        $bandera_marcha_NO = 0
                        $bandera_marcha_NE = 0
                End If


                        Select Case $bt_modo_operacion
                                Case 0

        If $bandera_marcha_NO And $ang_NO < $lim_abierto_NO_min And Not $var_abierto_NO Then
                                                '$ang_NO = $ang_NO + 1
                                                $led_marcha = 1
                                                $fin_cerrado_NO = 0
                                        End If

        If $bandera_marcha_NE And $ang_NE < $lim_abierto_NE_min And Not $var_abierto_NE Then
                                                '$ang_NE = $ang_NE + 1
                                                $led_marcha = 1
                                                $fin_cerrado_NE = 0
                                        End If

                                        If $bt_paro Then
                                                $led_marcha = 0
                                        End If

        If $ang_NO >= $lim_abierto_NO_min And $ang_NO <= $lim_abierto_NO_max Then
                                                $var_abierto_NO = 1
                                                $bandera_marcha_NO = 0
```

```
                                        Else
                                                $var_abierto_NO = 0
                                                $fin_abierto_NO = 0
                                        End If

                If $ang_NE >= $lim_abierto_NE_min And $ang_NE <= $lim_abierto_NE_max Then
                                                $var_abierto_NE = 1
                                                $bandera_marcha_NE = 0


                                        Else
                                                $var_abierto_NE = 0
                                                $fin_abierto_NE = 0
                                        End If

                                        If $ang_NE >= $lim_abierto_NE_min And $ang_NE <=
$lim_abierto_NE_max And $ang_NO >= $lim_abierto_NO_min And $ang_NO <= $lim_abierto_NO_max
Then
                                                $bt_paro = 0
                                                $fin_abierto_NO = 1
                                                $fin_abierto_NE = 1
                                                $led_marcha = 0
                                                $sec_centro = 2

                                        End If


                                Case 2

                                Case 1

If $bandera_marcha_NO And $ang_NO > $lim_cerrado__NO_min And Not $var_cerrado_NO Then
                                                '$ang_NO = $ang_NO - 1
                                                $led_marcha = 1
                                                $fin_abierto_NO = 0
                                        End If

If $bandera_marcha_NE And $ang_NE > $lim_cerrado_NE_min And Not $var_cerrado_NE Then
                                                '$ang_NE = $ang_NE - 1
                                                $led_marcha = 1
                                                $fin_abierto_NE = 0
                                        End If


                                        If $bt_paro Then
                                                $led_marcha = 0
                                        End If


                If $ang_NO >= $lim_cerrado_NO_max And $ang_NO <= $lim_cerrado_NO_min Then
                                                $var_cerrado_NO = 1
                                                $bandera_marcha_NO = 0


                                                Else
                                                $var_cerrado_NO = 0
                                                $fin_cerrado_NO = 0
                                        End If
```

```
        If $ang_NE >= $lim_cerrado_NE_max And $ang_NE <= $lim_cerrado_NE_min Then
                                                $var_cerrado_NE = 1
                                                $bandera_marcha_NE = 0


                                                Else
                                                $var_cerrado_NE = 0
                                                $fin_cerrado_NE = 0
                                        End If

        If $ang_NE >= $lim_cerrado_NE_max And $ang_NE <= $lim_cerrado_NE_min And $ang_NO >=
$lim_cerrado_NO_max And $ang_NO <= $lim_cerrado_NO_min Then
                                                $bt_paro = 0
                                                $fin_cerrado_NO = 1
                                                $fin_cerrado_NE = 1
                                                $led_marcha = 0
                                                End If

                                End Select


                End If


        Else
                                        $led_marcha = 0

        End If


End Sub

'Este procedimiento es ejecutado solo una vez cuando esta pantalla es cerrada.
Sub Screen_OnClose()

End Sub
```

APÉNDICE I. Scripts HMI imagenes

```
'Variables con alcance global pueden ser declaradas e inicializadas aquí.


'Los procedimientos con alcance local pueden ser implementados aquí.


'Este procedimiento es ejecutado solo una vez cuando el módulo grafico es iniciado.
Sub Graphics_OnStart()

$sec_centro = 2

'$encoder = 0
'$ang_SE = 0
'$ang_NO = 0
'$ang_NE = 0

$nivel_exterior = 5
$nivel_interior = 6

$lim_abierto_min = 90
$lim_abierto_max = 91
$lim_abierto_SE_min = 90
$lim_abierto_SE_max = 91
$lim_abierto_NO_min = 90
$lim_abierto_NO_max = 91
$lim_abierto_NE_min = 90
$lim_abierto_NE_max = 91


$lim_cerrado_min = 1
$lim_cerrado_max = 0
$lim_cerrado_SE_min = 1
$lim_cerrado_SE_max = 0
$lim_cerrado_NO_min = 1
$lim_cerrado_NO_max = 0
$lim_cerrado_NE_min = 1
$lim_cerrado_NE_max = 0


$var_abierto = 0
$var_abierto_SE = 0
$var_abierto_NO = 0
$var_abierto_NE = 0


$var_cerrado = 0
$var_cerrado_SE = 0
$var_cerrado_NO = 0
$var_cerrado_NE = 0

$bandera_marcha = 0
$bandera_marcha_SE = 0
$bandera_marcha_NO = 0
$bandera_marcha_NE = 0


End Sub
```

```
'Este procedimiento es ejecutado continuamente mientras el módulo grafico esta ejecutándose.
Sub Graphics_WhileRunning()

End Sub

'Este procedimiento es ejecutado solo una vez cuando el módulo grafico es cerrado.
Sub Graphics_OnEnd()

End Sub
```

APÉNDICE J. NODERED: Comunicación Http Nodered


```
[
    {
        "id": "f033cffa3119fac6",
        "type": "tab",
        "label": "Proyecto_get",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "c9c6d0568bc544a0",
        "type": "http response",
        "z": "f033cffa3119fac6",
        "name": "",
        "statusCode": "200",
        "headers": {},
        "x": 560,
        "y": 360,
        "wires": []
    },
    {
        "id": "e97834103a87131e",
        "type": "json",
        "z": "f033cffa3119fac6",
        "name": "",
        "property": "payload",
        "action": "obj",
        "pretty": true,
        "x": 550,
        "y": 480,
        "wires": [
            [
                "6da1ec03df8e10f3"
            ]
        ]
    },
    {
        "id": "6575e11d47044c85",
        "type": "debug",
        "z": "f033cffa3119fac6",
        "d": true,
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 990,
        "y": 380,
        "wires": []
    },
    {
        "id": "cc8d9beb4314693f",
        "type": "ui_text",
        "z": "f033cffa3119fac6",
        "group": "2b7ac01b.fc984",
        "order": 1,
```

```
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Sensor Name",
        "format": "{{msg.payload}}",
        "layout": "row-spread",
        "className": "",
        "x": 1000,
        "y": 440,
        "wires": []
    },
    {
        "id": "d95dfe4554754133",
        "type": "ui_gauge",
        "z": "f033cffa3119fac6",
        "name": "",
        "group": "2b7ac01b.fc984",
        "order": 2,
        "width": 0,
        "height": 0,
        "gtype": "gage",
        "title": "RPM",
        "label": "ºC",
        "format": "{{value*60}}",
        "min": 0,
        "max": "50",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "className": "",
        "x": 970,
        "y": 560,
        "wires": []
    },
    {
        "id": "6da1ec03df8e10f3",
        "type": "function",
        "z": "f033cffa3119fac6",
        "name": "JSON or URL Encoded",
        "func": "var msg0 = { payload: msg.payload.api_key };\nvar msg1 = { payload:
msg.payload.sensor };\nvar msg2 = { payload: msg.payload.value1 };\nvar msg3 = { payload:
msg.payload.value2 };\n\n\nreturn [msg0, msg1, msg2, msg3];",
        "outputs": 4,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 750,
        "y": 480,
        "wires": [
            [
                "6575e11d47044c85"
            ],
            [
                "cc8d9beb4314693f"
            ],
            [
                "d95dfe4554754133",
                "053e0e3234cb33ef"
            ],
            []
```

```
    ]
},
{
    "id": "d16b87a040409392",
    "type": "http in",
    "z": "f033cffa3119fac6",
    "name": "",
    "url": "update-sensor",
    "method": "post",
    "upload": false,
    "swaggerDoc": "",
    "x": 320,
    "y": 400,
    "wires": [
        [
            "c9c6d0568bc544a0",
            "e97834103a87131e",
            "6575e11d47044c85"
        ]
    ]
},
{
    "id": "053e0e3234cb33ef",
    "type": "change",
    "z": "f033cffa3119fac6",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "temp1",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 1000,
    "y": 500,
    "wires": [
        [
            "9b28b814155dfe3f"
        ]
    ]
},
{
    "id": "93c8bcedbfb38ed1",
    "type": "http in",
    "z": "f033cffa3119fac6",
    "name": "",
    "url": "/get-sensor",
    "method": "get",
    "upload": false,
    "swaggerDoc": "",
    "x": 360,
    "y": 700,
    "wires": [
        [
            "f7dc09b9dd5b88e6"
        ]
    ]
},
```

```
{
    "id": "438a619cded815e0",
    "type": "http response",
    "z": "f033cffa3119fac6",
    "name": "",
    "statusCode": "300",
    "headers": {},
    "x": 720,
    "y": 660,
    "wires": []
},
{
    "id": "f7dc09b9dd5b88e6",
    "type": "function",
    "z": "f033cffa3119fac6",
    "name": "",
    "func": "var duty = global.get(\"pwm1\")\nmsg.payload= duty\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 530,
    "y": 700,
    "wires": [
        [
            "438a619cded815e0",
            "6b79912cdf3d1e10"
        ]
    ]
},
{
    "id": "6b79912cdf3d1e10",
    "type": "debug",
    "z": "f033cffa3119fac6",
    "d": true,
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 730,
    "y": 728,
    "wires": []
},
{
    "id": "9b28b814155dfe3f",
    "type": "debug",
    "z": "f033cffa3119fac6",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 1200,
    "y": 500,
    "wires": []
},
```

```
    {
        "id": "e62d20ae0c19f592",
        "type": "function",
        "z": "f033cffa3119fac6",
        "name": "",
        "func": "var pwm = global.get(\"temp1\")\nmsg.payload={\n    \"chatId\": 2110234424,\n
\"type\": \"message\",\n    \"content\": \"RPM=\" + pwm*60\n}\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 1020,
        "y": 680,
        "wires": [
            [
                "c8be545249e147c9"
            ]
        ]
    },
    {
        "id": "c8be545249e147c9",
        "type": "telegram sender",
        "z": "f033cffa3119fac6",
        "name": "",
        "bot": "12cd4f431de2cd7e",
        "haserroroutput": false,
        "outputs": 1,
        "x": 1230,
        "y": 680,
        "wires": [
            []
        ]
    },
    {
        "id": "2b7ac01b.fc984",
        "type": "ui_group",
        "name": "SENSORS",
        "tab": "99ab8dc5.f435c",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "12cd4f431de2cd7e",
        "type": "telegram bot",
        "botname": "TscLabczuniga_bot",
        "usernames": "",
        "chatids": "",
        "baseapiurl": "",
        "updatemode": "polling",
        "pollinterval": "300",
        "usesocks": false,
        "sockshost": "",
        "socksport": "6667",
        "socksusername": "anonymous",
        "sockspassword": "",
        "bothost": "",
        "botpath": "",
        "localbotport": "8443",
        "publicbotport": "8443",
        "privatekey": "",
        "certificate": "",
        "useselfsignedcertificate": false,
```

```
        "sslterminated": false,
        "verboselogging": false
    },
    {
        "id": "99ab8dc5.f435c",
        "type": "ui_tab",
        "name": "HTTP",
        "icon": "dashboard",
        "order": 1,
        "disabled": false,
        "hidden": false
    }
]
```

APÉNDICE K. NODERED: Comunicación Ethernet Industrial


```
[
    {
        "id": "2580397843b11816",
        "type": "tab",
        "label": "arch_PLC_Comunicacion",
        "disabled": true,
        "info": "",
        "env": []
    },
    {
        "id": "22037276f4d22be1",
        "type": "change",
        "z": "2580397843b11816",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "DC_NE",
                "pt": "global",
                "to": "",
                "tot": "str"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 390,
        "y": 100,
        "wires": [
            []
        ]
    },
    {
        "id": "bf552e909ea519fc",
        "type": "s7 in",
        "z": "2580397843b11816",
        "endpoint": "8eb988afa28ab4f1",
        "mode": "single",
```

```
    "variable": "velocidad_calc_NE",
    "diff": true,
    "name": "DutyCycle N.E.",
    "x": 200,
    "y": 100,
    "wires": [
        [
            "22037276f4d22be1"
        ]
    ]
},
{
    "id": "f79fda98c32bae57",
    "type": "s7 in",
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "enable_NE",
    "diff": true,
    "name": "Giro N.E.",
    "x": 180,
    "y": 140,
    "wires": [
        [
            "f4fef4d978448e5f"
        ]
    ]
},
{
    "id": "b7d223bf1853cbf3",
    "type": "s7 in",
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "velocidad_calc_NO",
    "diff": true,
    "name": "DutyCycle N.O.",
    "x": 200,
    "y": 180,
    "wires": [
        [
```

```
            "438700c4053ecb13"
        ]
    ]
},
{
    "id": "abb30f2c1314a20c",
    "type": "s7 in",
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "enable_NO",
    "diff": true,
    "name": "Giro N.O.",
    "x": 180,
    "y": 220,
    "wires": [
        [
            "508f6dc701c45155"
        ]
    ]
},
{
    "id": "1be4d8a2a8598f32",
    "type": "s7 in",
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "velocidad_calc_SE",
    "diff": true,
    "name": "DutyCycle S.E.",
    "x": 200,
    "y": 260,
    "wires": [
        [
            "f3d5245129fb70c6"
        ]
    ]
},
{
    "id": "bdaffee31ea81e69",
    "type": "s7 in",
```

```
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "enable_SE",
    "diff": true,
    "name": "Giro S.E.",
    "x": 180,
    "y": 300,
    "wires": [
        [
            "d551b30c5252a9ef"
        ]
    ]
},
{
    "id": "e071a5f5256fb913",
    "type": "s7 in",
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "velocidad_calc_SO",
    "diff": true,
    "name": "DutyCycle S.O",
    "x": 200,
    "y": 340,
    "wires": [
        [
            "b1863a6db3e8e648"
        ]
    ]
},
{
    "id": "41ba4a86df40d955",
    "type": "s7 in",
    "z": "2580397843b11816",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "enable_SO",
    "diff": true,
    "name": "Giro S.O.",
    "x": 180,
```

```
    "y": 380,
    "wires": [
        [
            "241171a58485094a"
        ]
    ]
},
{
    "id": "438700c4053ecb13",
    "type": "change",
    "z": "2580397843b11816",
    "name": "",
    "rules": [
        {
            "t": "set",
```

APÉNDICE L. NODERED: Comunicación Telegram


```
[
    {
        "id": "82fca4b4d5757e53",
        "type": "tab",
        "label": "Flow 4",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "2edb2823c25e6b3d",
        "type": "telegram receiver",
        "z": "82fca4b4d5757e53",
        "name": "TSC Lab",
        "bot": "12cd4f431de2cd7e",
        "saveDataDir": "",
        "filterCommands": false,
        "x": 340,
        "y": 160,
        "wires": [
            [
                "cc87101ee5b4b7cb"
            ],
            []
        ]
    },
    {
        "id": "cc87101ee5b4b7cb",
        "type": "debug",
        "z": "82fca4b4d5757e53",
        "d": true,
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 600,
        "y": 160,
```

```
        "wires": []
    },
    {
        "id": "f2c7022fcf910782",
        "type": "function",
        "z": "82fca4b4d5757e53",
        "name": "",
        "func": "var pwm = global.get(\"temp1\")\nmsg.payload={\n    \"chatId\": 2110234424,\n
\"type\": \"message\",\n    \"content\": \"RPM=\" + pwm*60\n}\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 490,
        "y": 280,
        "wires": [
            [
                "421c021663af9f77"
            ]
        ]
    },
    {
        "id": "421c021663af9f77",
        "type": "telegram sender",
        "z": "82fca4b4d5757e53",
        "name": "",
        "bot": "12cd4f431de2cd7e",
        "haserroroutput": false,
        "outputs": 1,
        "x": 700,
        "y": 280,
        "wires": [
            []
        ]
    },
    {
        "id": "62898056710e0fbf",
        "type": "telegram receiver",
        "z": "82fca4b4d5757e53",
        "name": "",
        "bot": "12cd4f431de2cd7e",
        "saveDataDir": "",
        "filterCommands": false,
        "x": 300,
        "y": 380,
```

```
    "wires": [
        [
            "113e66b59e0fa613"
        ],
        []
    ]
},
{
    "id": "113e66b59e0fa613",
    "type": "change",
    "z": "82fca4b4d5757e53",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "pwm1",
            "pt": "global",
            "to": "payload.content",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 530,
    "y": 380,
    "wires": [
        []
    ]
},
{
    "id": "0cf245657c9614d7",
    "type": "telegram command",
    "z": "82fca4b4d5757e53",
    "d": true,
    "name": "",
    "command": "rpm",
    "description": "",
    "registercommand": false,
    "language": "",
    "scope": "default",
    "bot": "12cd4f431de2cd7e",
    "strict": false,
    "hasresponse": true,
```

```
        "useregex": false,
        "removeregexcommand": false,
        "outputs": 2,
        "x": 250,
        "y": 280,
        "wires": [
            [
                "f2c7022fcf910782"
            ],
            []
        ]
    },
    {
        "id": "12cd4f431de2cd7e",
        "type": "telegram bot",
        "botname": "TscLabczuniga_bot",
        "usernames": "",
        "chatids": "",
        "baseapiurl": "",
        "updatemode": "polling",
        "pollinterval": "300",
        "usesocks": false,
        "sockshost": "",
        "socksport": "6667",
        "socksusername": "anonymous",
        "sockspassword": "",
        "bothost": "",
        "botpath": "",
        "localbotport": "8443",
        "publicbotport": "8443",
        "privatekey": "",
        "certificate": "",
        "useselfsignedcertificate": false,
        "sslterminated": false,
        "verboselogging": false
    }
]
```

APÉNDICE M. NODERED: Tx TSC_LAB y Rx PLC


```
[
    {
        "id": "535099e28c53be04",
        "type": "tab",
        "label": "TxSERIAL_RxPLC",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "8846a01ea049f9e0",
        "type": "serial in",
        "z": "535099e28c53be04",
        "name": "",
        "serial": "92520c6ed17408f7",
        "x": 130,
        "y": 120,
        "wires": [
            [
                "dad14518915b02c2",
                "45022f7300e5a55a"
            ]
        ]
    },
    {
        "id": "dad14518915b02c2",
        "type": "change",
        "z": "535099e28c53be04",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "driver_ne",
                "pt": "global",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 310,
        "y": 120,
        "wires": [
            [
                "a7f68421818ee2be",
                "e0c8f88386243abb",
                "18a2c002f170d5fc"
            ]
        ]
    },
    {
        "id": "a7f68421818ee2be",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var num_rpm_ne = global.get(\"driver_ne[5]\");\nglobal.set(\"driver_rpm_ne\",
num_rpm_ne.charCodeAt(0));\n\nmsg.payload= global.get(\"driver_rpm_ne\");\n\n\nreturn
msg;\n\n\n",
        "outputs": 1,
```

```
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 120,
        "wires": [
            [
                "49c36fedbafbfef4",
                "4de86dae80d1f566"
            ]
        ]
    },
    {
        "id": "e0c8f88386243abb",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var dgyro_ne\nif (global.get(\"driver_ne[1]\") === 'A')\n{\n    dgyro_ne =
true;\n}\nelse \n{\n    dgyro_ne = false;\n}\nglobal.set(\"driver_giro_ne\",
dgyro_ne);\n\nmsg.payload= global.get(\"driver_giro_ne\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 80,
        "wires": [
            [
                "72a5aefbc23443d0",
                "d8fdd77c1d813634"
            ]
        ]
    },
    {
        "id": "18a2c002f170d5fc",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var angulo_ne = global.get(\"driver_ne[2]\") + global.get(\"driver_ne[3]\") +
global.get(\"driver_ne[4]\");\nvar num_ang_ne = parseInt(\"0x\"+ angulo_ne) *
0.0296;\nglobal.set(\"driver_ang_ne\", num_ang_ne);\n\nmsg.payload=
global.get(\"driver_ang_ne\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 160,
        "wires": [
            [
                "af2658d8849b2d2d",
                "055c2d06ed5227a1"
            ]
        ]
    },
    {
        "id": "72a5aefbc23443d0",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "Sgiro_NE",
        "name": "Driver Giro N.E.",
```

```
    "x": 700,
    "y": 80,
    "wires": []
},
{
    "id": "49c36fedbafbfef4",
    "type": "s7 out",
    "z": "535099e28c53be04",
    "endpoint": "36bc033410933cd0",
    "variable": "lectura_rpm_NE",
    "name": "Driver RPM N.E.",
    "x": 700,
    "y": 120,
    "wires": []
},
{
    "id": "af2658d8849b2d2d",
    "type": "s7 out",
    "z": "535099e28c53be04",
    "endpoint": "36bc033410933cd0",
    "variable": "pos_compuerta_NE",
    "name": "Driver Ángulo S.E.",
    "x": 710,
    "y": 160,
    "wires": []
},
{
    "id": "0d1d815ac9c759b4",
    "type": "serial in",
    "z": "535099e28c53be04",
    "name": "",
    "serial": "926949069681ee0f",
    "x": 130,
    "y": 240,
    "wires": [
        [
            "343cca3f27b5ed17",
            "9ef9ef344d1afcf4"
        ]
    ]
},
{
    "id": "343cca3f27b5ed17",
    "type": "change",
    "z": "535099e28c53be04",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "driver_no",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 310,
    "y": 240,
    "wires": [
        [
            "52b6505a9569a18b",
```

```
                "d0a000f8821eff4d",
                "d1133354c0ec0e64"
            ]
        ]
    },
    {
        "id": "52b6505a9569a18b",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var num_rpm_no = global.get(\"driver_no[5]\");\nglobal.set(\"driver_rpm_no\",
num_rpm_no.charCodeAt(0));\n\nmsg.payload= global.get(\"driver_rpm_no\");\n\n\nreturn
msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 240,
        "wires": [
            [
                "7429ae69300764a9"
            ]
        ]
    },
    {
        "id": "d0a000f8821eff4d",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var dgyro_no\nif (global.get(\"driver_no[1]\") === 'A')\n{\n    dgyro_no =
true;\n}\nelse \n{\n    dgyro_no = false;\n}\nglobal.set(\"driver_giro_no\",
dgyro_no);\n\nmsg.payload= global.get(\"driver_giro_no\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 200,
        "wires": [
            [
                "57e4ce76420d14d8"
            ]
        ]
    },
    {
        "id": "d1133354c0ec0e64",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var angulo_no = global.get(\"driver_no[2]\") + global.get(\"driver_no[3]\") +
global.get(\"driver_no[4]\");\nvar num_ang_no = parseInt(\"0x\"+ angulo_no) *
0.0296;\nglobal.set(\"driver_ang_no\", num_ang_no);\n\nmsg.payload=
global.get(\"driver_ang_no\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 280,
        "wires": [
            [
```

```
            "0834bfeef4f8c124"
        ]
    ]
},
{
    "id": "57e4ce76420d14d8",
    "type": "s7 out",
    "z": "535099e28c53be04",
    "endpoint": "36bc033410933cd0",
    "variable": "Sgiro_NO",
    "name": "Driver Giro N.O.",
    "x": 700,
    "y": 200,
    "wires": []
},
{
    "id": "7429ae69300764a9",
    "type": "s7 out",
    "z": "535099e28c53be04",
    "endpoint": "36bc033410933cd0",
    "variable": "lectura_rpm_NO",
    "name": "Driver RPM N.O.",
    "x": 710,
    "y": 240,
    "wires": []
},
{
    "id": "0834bfeef4f8c124",
    "type": "s7 out",
    "z": "535099e28c53be04",
    "endpoint": "36bc033410933cd0",
    "variable": "pos_compuerta_NO",
    "name": "Driver Ángulo N.O.",
    "x": 710,
    "y": 280,
    "wires": []
},
{
    "id": "be65103c9486df94",
    "type": "serial in",
    "z": "535099e28c53be04",
    "name": "",
    "serial": "350edfd6111f8379",
    "x": 130,
    "y": 360,
    "wires": [
        [
            "5cbc70d3c20161be"
        ]
    ]
},
{
    "id": "5cbc70d3c20161be",
    "type": "change",
    "z": "535099e28c53be04",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "driver_se",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
```

```
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 310,
        "y": 360,
        "wires": [
            [
                "b8cf74682947dfb4",
                "bdfa3830626c6a4c",
                "ae2252cf81b877cf"
            ]
        ]
    },
    {
        "id": "b8cf74682947dfb4",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var num_rpm_se = global.get(\"driver_se[5]\");\nglobal.set(\"driver_rpm_se\",
num_rpm_se.charCodeAt(0));\n\nmsg.payload= global.get(\"driver_rpm_se\");\n\n\nreturn
msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 360,
        "wires": [
            [
                "5f720c02eed1a352"
            ]
        ]
    },
    {
        "id": "bdfa3830626c6a4c",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var dgyro_se\nif (global.get(\"driver_se[1]\") === 'A')\n{\n    dgyro_se =
true;\n}\nelse \n{\n    dgyro_se = false;\n}\nglobal.set(\"driver_giro_se\",
dgyro_se);\n\nmsg.payload= global.get(\"driver_giro_se\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 320,
        "wires": [
            [
                "55b793b9f9bcbf4e"
            ]
        ]
    },
    {
        "id": "ae2252cf81b877cf",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var angulo_se = global.get(\"driver_se[2]\") + global.get(\"driver_se[3]\") +
global.get(\"driver_se[4]\");\nvar num_ang_se = parseInt(\"0x\"+ angulo_se) *
```

```
0.0296;\nglobal.set(\"driver_ang_se\", num_ang_se);\n\nmsg.payload=
global.get(\"driver_ang_se\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 400,
        "wires": [
            [
                "09203dd97d064340"
            ]
        ]
    },
    {
        "id": "55b793b9f9bcbf4e",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "Sgiro_SE",
        "name": "Driver Giro S.E.",
        "x": 700,
        "y": 320,
        "wires": []
    },
    {
        "id": "5f720c02eed1a352",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "lectura_rpm_SE",
        "name": "Driver RPM S.E.",
        "x": 700,
        "y": 360,
        "wires": []
    },
    {
        "id": "09203dd97d064340",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "pos_compuerta_SE",
        "name": "Driver Ángulo S.E.",
        "x": 710,
        "y": 400,
        "wires": []
    },
    {
        "id": "d7de5c55e0145c65",
        "type": "serial in",
        "z": "535099e28c53be04",
        "name": "",
        "serial": "43cb10220eaaabec",
        "x": 130,
        "y": 480,
        "wires": [
            [
                "87098de13d026915"
            ]
        ]
    },
    {
        "id": "87098de13d026915",
        "type": "change",
```

```
        "z": "535099e28c53be04",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "driver_so",
                "pt": "global",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 310,
        "y": 480,
        "wires": [
            [
                "d498a45a7cf247be",
                "9266c965585d521a",
                "aa0b0e781e11071c"
            ]
        ]
    },
    {
        "id": "d498a45a7cf247be",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var num_rpm_so = global.get(\"driver_so[5]\");\nglobal.set(\"driver_rpm_so\",
num_rpm_so.charCodeAt(0));\n\nmsg.payload= global.get(\"driver_rpm_so\");\n\n\nreturn
msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 480,
        "wires": [
            [
                "b91f545f24f06bcb"
            ]
        ]
    },
    {
        "id": "9266c965585d521a",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var dgyro_so\nif (global.get(\"driver_so[1]\") === 'A')\n{\n    dgyro_so =
true;\n}\nelse \n{\n    dgyro_so = false;\n}\nglobal.set(\"driver_giro_so\",
dgyro_so);\n\nmsg.payload= global.get(\"driver_giro_so\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 440,
        "wires": [
            [
                "31d625513618809b"
```

```
            ]
        ]
    },
    {
        "id": "aa0b0e781e11071c",
        "type": "function",
        "z": "535099e28c53be04",
        "name": "",
        "func": "var angulo_so = global.get(\"driver_so[2]\") + global.get(\"driver_so[3]\") +
global.get(\"driver_so[4]\");\nvar num_ang_so = parseInt(\"0x\"+ angulo_so) *
0.0296;\nglobal.set(\"driver_ang_so\", num_ang_so);\n\nmsg.payload=
global.get(\"driver_ang_so\");\n\n\nreturn msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 520,
        "wires": [
            [
                "43bd14333a17c263"
            ]
        ]
    },
    {
        "id": "31d625513618809b",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "Sgiro_SO",
        "name": "Driver Giro S.O.",
        "x": 700,
        "y": 440,
        "wires": []
    },
    {
        "id": "b91f545f24f06bcb",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "lectura_rpm_SO",
        "name": "Driver RPM S.O.",
        "x": 710,
        "y": 480,
        "wires": []
    },
    {
        "id": "43bd14333a17c263",
        "type": "s7 out",
        "z": "535099e28c53be04",
        "endpoint": "36bc033410933cd0",
        "variable": "pos_compuerta_SO",
        "name": "Driver Ángulo S.O.",
        "x": 710,
        "y": 520,
        "wires": []
    },
    {
        "id": "9ef9ef344d1afcf4",
        "type": "debug",
        "z": "535099e28c53be04",
        "d": true,
        "name": "",
        "active": true,
```

```
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 310,
    "y": 180,
    "wires": []
},
{
    "id": "45022f7300e5a55a",
    "type": "debug",
    "z": "535099e28c53be04",
    "d": true,
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 320,
    "y": 60,
    "wires": []
},
{
    "id": "d8fdd77c1d813634",
    "type": "debug",
    "z": "535099e28c53be04",
    "d": true,
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 700,
    "y": 40,
    "wires": []
},
{
    "id": "4de86dae80d1f566",
    "type": "debug",
    "z": "535099e28c53be04",
    "d": true,
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 880,
    "y": 100,
    "wires": []
},
```

```
{
    "id": "055c2d06ed5227a1",
    "type": "debug",
    "z": "535099e28c53be04",
    "d": true,
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 900,
    "y": 180,
    "wires": []
},
{
    "id": "92520c6ed17408f7",
    "type": "serial-port",
    "serialport": "COM3",
    "serialbaud": "115200",
    "databits": "8",
    "parity": "none",
    "stopbits": "1",
    "waitfor": "$",
    "dtr": "none",
    "rts": "none",
    "cts": "none",
    "dsr": "none",
    "newline": "@",
    "bin": "false",
    "out": "char",
    "addchar": "",
    "responsetimeout": "10000"
},
{
    "id": "36bc033410933cd0",
    "type": "s7 endpoint",
    "transport": "iso-on-tcp",
    "address": "192.168.0.241",
    "port": "102",
    "rack": "0",
    "slot": "1",
    "localtsaphi": "01",
    "localtsaplo": "00",
    "remotetsaphi": "01",
    "remotetsaplo": "00",
    "connmode": "rack-slot",
    "adapter": "",
    "busaddr": "2",
    "cycletime": "2",
    "timeout": "2000",
    "name": "1200",
    "vartable": [
        {
            "addr": "DB2,WORD0",
            "name": "lectura_rpm_NE"
        },
        {
            "addr": "DB2,WORD2",
            "name": "lectura_rpm_NO"
        },
        {
```

```
            "addr": "DB2,WORD4",
            "name": "lectura_rpm_SE"
        },
        {
            "addr": "DB2,WORD6",
            "name": "lectura_rpm_SO"
        },
        {
            "addr": "DB2,REAL8",
            "name": "pos_compuerta_NE"
        },
        {
            "addr": "DB2,REAL12",
            "name": "pos_compuerta_NO"
        },
        {
            "addr": "DB2,REAL16",
            "name": "pos_compuerta_SE"
        },
        {
            "addr": "DB2,REAL20",
            "name": "pos_compuerta_SO"
        },
        {
            "addr": "DB2,X24.0",
            "name": "Sgiro_NE"
        },
        {
            "addr": "DB2,X24.1",
            "name": "Sgiro_NO"
        },
        {
            "addr": "DB2,X24.2",
            "name": "Sgiro_SE"
        },
        {
            "addr": "DB2,X24.3",
            "name": "Sgiro_SO"
        }
    ]
},
{
    "id": "926949069681ee0f",
    "type": "serial-port",
    "serialport": "COM5",
    "serialbaud": "115200",
    "databits": "8",
    "parity": "none",
    "stopbits": "1",
    "waitfor": "$",
    "dtr": "none",
    "rts": "none",
    "cts": "none",
    "dsr": "none",
    "newline": "@",
    "bin": "false",
    "out": "char",
    "addchar": "",
    "responsetimeout": "10000"
},
{
    "id": "350edfd6111f8379",
    "type": "serial-port",
    "serialport": "COM7",
    "serialbaud": "115200",
```

```
        "databits": "8",
        "parity": "none",
        "stopbits": "1",
        "waitfor": "$",
        "dtr": "none",
        "rts": "none",
        "cts": "none",
        "dsr": "none",
        "newline": "@",
        "bin": "false",
        "out": "char",
        "addchar": "",
        "responsetimeout": "10000"
    },
    {
        "id": "43cb10220eaaabec",
        "type": "serial-port",
        "serialport": "COM9",
        "serialbaud": "115200",
        "databits": "8",
        "parity": "none",
        "stopbits": "1",
        "waitfor": "$",
        "dtr": "none",
        "rts": "none",
        "cts": "none",
        "dsr": "none",
        "newline": "@",
        "bin": "false",
        "out": "char",
        "addchar": "",
        "responsetimeout": "10000"
    }
]
```

APÉNDICE N. NODERED: Rx TSC_LAB y Tx PLC


```
[
    {
        "id": "012c8378beebad9b",
        "type": "tab",
        "label": "RxSERIAL_TxPLC",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "04d486016d0e5918",
        "type": "serial out",
        "z": "012c8378beebad9b",
        "name": "",
        "serial": "92520c6ed17408f7",
        "x": 730,
        "y": 120,
        "wires": []
    },
    {
        "id": "a9d20cc150d37921",
        "type": "debug",
        "z": "012c8378beebad9b",
        "d": true,
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 850,
        "y": 60,
        "wires": []
    },
    {
        "id": "d6dfb41d39ac2920",
        "type": "function",
        "z": "012c8378beebad9b",
        "name": "Compuerta N.E.",
        "func": "var inicio = '$';\nvar dutycycle_ne = String(global.get(\"DC_NE\"));\nconst
messages = new Array(3)\n\nif (dutycycle_ne.length === 1)\n{\n    messages[2] =
dutycycle_ne[0];\n    messages[1] = '0';\n    messages[0] = '0';\n}\nif (dutycycle_ne.length
=== 2)\n{\n    messages[2] = dutycycle_ne[1];\n    messages[1] = dutycycle_ne[0];\n
messages[0] = '0';\n}\nif (dutycycle_ne.length === 3)\n{\n    messages[2] = dutycycle_ne[2];\n
messages[1] = dutycycle_ne[1];\n    messages[0] = dutycycle_ne[0];\n}\nvar fin_dutycycle_ne =
String(messages[0] + messages[1] + messages[2]);\n\nvar gyro_ne;\nif (global.get(\"G_NE\") ===
true)\n{\n    gyro_ne = 'A';\n}\nelse \n{\n    gyro_ne = 'B';\n}\nvar fin =
'@'\n\nmsg.payload= inicio + gyro_ne + fin_dutycycle_ne + fin;\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 570,
        "y": 100,
        "wires": [
            [
                "04d486016d0e5918",
                "a9d20cc150d37921"
```

```
        ]
    ]
},
{
    "id": "01ead35d9f9e4075",
    "type": "change",
    "z": "012c8378beebad9b",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "DC_NE",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 330,
    "y": 80,
    "wires": [
        [
            "d6dfb41d39ac2920"
        ]
    ]
},
{
    "id": "6f284a177b12b724",
    "type": "s7 in",
    "z": "012c8378beebad9b",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "velocidad_calc_NE",
    "diff": true,
    "name": "DutyCycle N.E.",
    "x": 120,
    "y": 80,
    "wires": [
        [
            "01ead35d9f9e4075"
        ]
    ]
},
{
    "id": "ed8ddeb98103569c",
    "type": "s7 in",
    "z": "012c8378beebad9b",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "giro_NE",
    "diff": true,
    "name": "Giro N.E.",
    "x": 140,
    "y": 120,
    "wires": [
        [
            "648882dd2cbfb545"
        ]
    ]
},
{
```

```
        "id": "648882dd2cbfb545",
        "type": "change",
        "z": "012c8378beebad9b",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "G_NE",
                "pt": "global",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 320,
        "y": 120,
        "wires": [
            [
                "d6dfb41d39ac2920"
            ]
        ]
    },
    {
        "id": "c17dbc716b183694",
        "type": "serial out",
        "z": "012c8378beebad9b",
        "name": "",
        "serial": "6ab7935b2274b5ec",
        "x": 730,
        "y": 240,
        "wires": []
    },
    {
        "id": "770f3a0ea5fb18a9",
        "type": "debug",
        "z": "012c8378beebad9b",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 750,
        "y": 180,
        "wires": []
    },
    {
        "id": "6ce99c87c38841e8",
        "type": "function",
        "z": "012c8378beebad9b",
        "name": "Compuerta N.E.",
        "func": "var inicio = '$';\nvar dutycycle_no = String(global.get(\"DC_NO\"));\nconst
messages = new Array(3)\n\nif (dutycycle_no.length === 1)\n{\n    messages[2] =
dutycycle_no[0];\n    messages[1] = '0';\n    messages[0] = '0';\n}\nif (dutycycle_no.length
=== 2)\n{\n    messages[2] = dutycycle_no[1];\n    messages[1] = dutycycle_no[0];\n
messages[0] = '0';\n}\nif (dutycycle_no.length === 3)\n{\n    messages[2] = dutycycle_no[2];\n
messages[1] = dutycycle_no[1];\n    messages[0] = dutycycle_no[0];\n}\nvar fin_dutycycle_no =
String(messages[0] + messages[1] + messages[2]);\n\nvar gyro_no;\nif (global.get(\"G_NO\") ===
```

```
true)\n{\n    gyro_no = 'A';\n}\nelse \n{\n    gyro_no = 'B';\n}\nvar fin =
'@'\n\nmsg.payload= inicio + gyro_no + fin_dutycycle_no + fin;\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 570,
        "y": 220,
        "wires": [
            [
                "c17dbc716b183694",
                "770f3a0ea5fb18a9"
            ]
        ]
    },
    {
        "id": "925f036fdf960869",
        "type": "change",
        "z": "012c8378beebad9b",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "DC_NO",
                "pt": "global",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 330,
        "y": 200,
        "wires": [
            [
                "6ce99c87c38841e8"
            ]
        ]
    },
    {
        "id": "35eb0fafb9e21bd8",
        "type": "s7 in",
        "z": "012c8378beebad9b",
        "endpoint": "8eb988afa28ab4f1",
        "mode": "single",
        "variable": "velocidad_calc_NO",
        "diff": true,
        "name": "DutyCycle N.O.",
        "x": 120,
        "y": 200,
        "wires": [
            [
                "925f036fdf960869"
            ]
        ]
    },
    {
        "id": "dece22202c699786",
        "type": "s7 in",
        "z": "012c8378beebad9b",
        "endpoint": "8eb988afa28ab4f1",
```

```
    "mode": "single",
    "variable": "giro_NO",
    "diff": true,
    "name": "Giro N.O.",
    "x": 140,
    "y": 240,
    "wires": [
        [
            "b92db9613c706ecd"
        ]
    ]
},
{
    "id": "b92db9613c706ecd",
    "type": "change",
    "z": "012c8378beebad9b",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "G_NO",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 320,
    "y": 240,
    "wires": [
        [
            "6ce99c87c38841e8"
        ]
    ]
},
{
    "id": "7b625cc949d97309",
    "type": "serial out",
    "z": "012c8378beebad9b",
    "name": "",
    "serial": "bcbfe41ea8ecc4c5",
    "x": 730,
    "y": 360,
    "wires": []
},
{
    "id": "b98b024b1de2f8ed",
    "type": "debug",
    "z": "012c8378beebad9b",
    "d": true,
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 750,
    "y": 300,
```

```
        "wires": []
    },
    {
        "id": "79f0267656bbb618",
        "type": "function",
        "z": "012c8378beebad9b",
        "name": "Compuerta N.E.",
        "func": "var inicio = '$';\nvar dutycycle_se = String(global.get(\"DC_SE\"));\nconst
messages = new Array(3)\n\nif (dutycycle_se.length === 1)\n{\n    messages[2] =
dutycycle_se[0];\n    messages[1] = '0';\n    messages[0] = '0';\n}\nif (dutycycle_se.length
=== 2)\n{\n    messages[2] = dutycycle_se[1];\n    messages[1] = dutycycle_se[0];\n
messages[0] = '0';\n}\nif (dutycycle_se.length === 3)\n{\n    messages[2] = dutycycle_se[2];\n
messages[1] = dutycycle_se[1];\n    messages[0] = dutycycle_se[0];\n}\nvar fin_dutycycle_se =
String(messages[0] + messages[1] + messages[2]);\n\nvar gyro_se;\nif (global.get(\"G_SE\") ===
true)\n{\n    gyro_se = 'A';\n}\nelse \n{\n    gyro_se = 'B';\n}\nvar fin =
'@'\n\nmsg.payload= inicio + gyro_se + fin_dutycycle_se + fin;\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 570,
        "y": 340,
        "wires": [
            [
                "7b625cc949d97309",
                "b98b024b1de2f8ed"
            ]
        ]
    },
    {
        "id": "a65bcd5b432e5a04",
        "type": "change",
        "z": "012c8378beebad9b",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "DC_SE",
                "pt": "global",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 330,
        "y": 320,
        "wires": [
            [
                "79f0267656bbb618"
            ]
        ]
    },
    {
        "id": "5a2b228da474ba14",
        "type": "s7 in",
        "z": "012c8378beebad9b",
        "endpoint": "8eb988afa28ab4f1",
        "mode": "single",
        "variable": "velocidad_calc_SE",
        "diff": true,
```

```
    "name": "DutyCycle S.E.",
    "x": 120,
    "y": 320,
    "wires": [
        [
            "a65bcd5b432e5a04"
        ]
    ]
},
{
    "id": "1486216a462dec84",
    "type": "s7 in",
    "z": "012c8378beebad9b",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "giro_SE",
    "diff": true,
    "name": "Giro S.E.",
    "x": 140,
    "y": 360,
    "wires": [
        [
            "5614d978ffd874e5"
        ]
    ]
},
{
    "id": "5614d978ffd874e5",
    "type": "change",
    "z": "012c8378beebad9b",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "G_SE",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 320,
    "y": 360,
    "wires": [
        [
            "79f0267656bbb618"
        ]
    ]
},
{
    "id": "c0e9db420d2e44c9",
    "type": "serial out",
    "z": "012c8378beebad9b",
    "name": "",
    "serial": "43cb10220eaaabec",
    "x": 730,
    "y": 480,
    "wires": []
},
{
    "id": "615f16c63eb8fdee",
```

```
        "type": "debug",
        "z": "012c8378beebad9b",
        "d": true,
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 750,
        "y": 420,
        "wires": []
    },
    {
        "id": "e180f57f41cf510c",
        "type": "function",
        "z": "012c8378beebad9b",
        "name": "Compuerta N.E.",
        "func": "var inicio = '$';\nvar dutycycle_so = String(global.get(\"DC_SO\"));\nconst
messages = new Array(3)\n\nif (dutycycle_so.length === 1)\n{\n    messages[2] =
dutycycle_so[0];\n    messages[1] = '0';\n    messages[0] = '0';\n}\nif (dutycycle_so.length
=== 2)\n{\n    messages[2] = dutycycle_so[1];\n    messages[1] = dutycycle_so[0];\n
messages[0] = '0';\n}\nif (dutycycle_so.length === 3)\n{\n    messages[2] = dutycycle_so[2];\n
messages[1] = dutycycle_so[1];\n    messages[0] = dutycycle_so[0];\n}\nvar fin_dutycycle_so =
String(messages[0] + messages[1] + messages[2]);\n\nvar gyro_so;\nif (global.get(\"G_SO\") ===
true)\n{\n    gyro_so = 'A';\n}\nelse \n{\n    gyro_so = 'B';\n}\nvar fin =
'@'\n\nmsg.payload= inicio + gyro_so + fin_dutycycle_so + fin;\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 570,
        "y": 460,
        "wires": [
            [
                "c0e9db420d2e44c9",
                "615f16c63eb8fdee"
            ]
        ]
    },
    {
        "id": "3892bb5c48853b79",
        "type": "change",
        "z": "012c8378beebad9b",
        "name": "",
        "rules": [
            {
                "t": "set",
                "p": "DC_SO",
                "pt": "global",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 330,
        "y": 440,
```

```
    "wires": [
        [
            "e180f57f41cf510c"
        ]
    ]
},
{
    "id": "c035fe063a6dc863",
    "type": "s7 in",
    "z": "012c8378beebad9b",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "velocidad_calc_SO",
    "diff": true,
    "name": "DutyCycle S.O.",
    "x": 120,
    "y": 440,
    "wires": [
        [
            "3892bb5c48853b79"
        ]
    ]
},
{
    "id": "7b10af267f380158",
    "type": "s7 in",
    "z": "012c8378beebad9b",
    "endpoint": "8eb988afa28ab4f1",
    "mode": "single",
    "variable": "giro_SO",
    "diff": true,
    "name": "Giro S.O.",
    "x": 140,
    "y": 480,
    "wires": [
        [
            "b0750d77b8065304"
        ]
    ]
},
{
    "id": "b0750d77b8065304",
    "type": "change",
    "z": "012c8378beebad9b",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "G_SO",
            "pt": "global",
            "to": "payload",
            "tot": "msg"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 320,
    "y": 480,
    "wires": [
        [
            "e180f57f41cf510c"
        ]
```

```
    ]
},
{
    "id": "cc0427bd74a71ba2",
    "type": "inject",
    "z": "012c8378beebad9b",
    "name": "",
    "props": [
        {
            "p": "payload"
        },
        {
            "p": "topic",
            "vt": "str"
        }
    ],
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "255",
    "payloadType": "num",
    "x": 140,
    "y": 580,
    "wires": [
        [
            "b68e942110bcb41a"
        ]
    ]
},
{
    "id": "b68e942110bcb41a",
    "type": "debug",
    "z": "012c8378beebad9b",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 420,
    "y": 600,
    "wires": []
},
{
    "id": "92520c6ed17408f7",
    "type": "serial-port",
    "serialport": "COM3",
    "serialbaud": "115200",
    "databits": "8",
    "parity": "none",
    "stopbits": "1",
    "waitfor": "$",
    "dtr": "none",
    "rts": "none",
    "cts": "none",
    "dsr": "none",
    "newline": "@",
    "bin": "false",
    "out": "char",
    "addchar": "",
    "responsetimeout": "10000"
```

```
},
{
    "id": "8eb988afa28ab4f1",
    "type": "s7 endpoint",
    "transport": "iso-on-tcp",
    "address": "192.168.0.241",
    "port": "102",
    "rack": "0",
    "slot": "1",
    "localtsaphi": "01",
    "localtsaplo": "00",
    "remotetsaphi": "01",
    "remotetsaplo": "00",
    "connmode": "rack-slot",
    "adapter": "",
    "busaddr": "2",
    "cycletime": "2",
    "timeout": "1500",
    "name": "enp_s1200",
    "vartable": [
        {
            "addr": "DB1,WORD0",
            "name": "velocidad_calc_NE"
        },
        {
            "addr": "DB1,X16.0",
            "name": "enable_NE"
        },
        {
            "addr": "DB1,WORD2",
            "name": "velocidad_calc_NO"
        },
        {
            "addr": "DB1,WORD4",
            "name": "velocidad_calc_SE"
        },
        {
            "addr": "DB1,WORD6",
            "name": "velocidad_calc_SO"
        },
        {
            "addr": "DB1,X16.1",
            "name": "enable_NO"
        },
        {
            "addr": "DB1,X16.2",
            "name": "enable_SE"
        },
        {
            "addr": "DB1,X16.3",
            "name": "enable_SO"
        },
        {
            "addr": "DB2,WORD0",
            "name": "rpm_driver_ne"
        },
        {
            "addr": "DB2,WORD2",
            "name": "rpm_driver_no"
        },
        {
            "addr": "DB2,WORD4",
            "name": "rpm_driver_se"
        },
        {
```

```
            "addr": "DB2,WORD6",
            "name": "rpm_driver_so"
        },
        {
            "addr": "DB2,WORD8",
            "name": "ang_driver_ne"
        },
        {
            "addr": "DB2,WORD10",
            "name": "ang_driver_no"
        },
        {
            "addr": "DB2,WORD12",
            "name": "ang_driver_se"
        },
        {
            "addr": "DB2,WORD14",
            "name": "ang_driver_so"
        },
        {
            "addr": "DB2,X16.0",
            "name": "giro_driver_ne"
        },
        {
            "addr": "DB2,X16.1",
            "name": "giro_driver_no"
        },
        {
            "addr": "DB2,X16.2",
            "name": "giro_driver_se"
        },
        {
            "addr": "DB2,X16.3",
            "name": "giro_driver_so"
        }
    ]
},
{
    "id": "6ab7935b2274b5ec",
    "type": "serial-port",
    "serialport": "COM5",
    "serialbaud": "115200",
    "databits": "8",
    "parity": "none",
    "stopbits": "1",
    "waitfor": "$",
    "dtr": "none",
    "rts": "none",
    "cts": "none",
    "dsr": "none",
    "newline": "@",
    "bin": "false",
    "out": "char",
    "addchar": "",
    "responsetimeout": "10000"
},
{
    "id": "bcbfe41ea8ecc4c5",
    "type": "serial-port",
    "serialport": "COM7",
    "serialbaud": "115200",
    "databits": "8",
    "parity": "none",
    "stopbits": "1",
    "waitfor": "$",
```

```
        "dtr": "none",
        "rts": "none",
        "cts": "none",
        "dsr": "none",
        "newline": "@",
        "bin": "false",
        "out": "char",
        "addchar": "",
        "responsetimeout": "10000"
    },
    {
        "id": "43cb10220eaaabec",
        "type": "serial-port",
        "serialport": "COM9",
        "serialbaud": "115200",
        "databits": "8",
        "parity": "none",
        "stopbits": "1",
        "waitfor": "$",
        "dtr": "none",
        "rts": "none",
        "cts": "none",
        "dsr": "none",
        "newline": "@",
        "bin": "false",
        "out": "char",
        "addchar": "",
        "responsetimeout": "10000"
    }
]
```

APÉNDICE O. TSC_LAB HTTP

```
/*
   ***************************** TSC-Lab ******************************
   ******************** Proyecto de titulación ************************
   By: Carlos Zúñiga Reyes

*/

#include <OneWire.h>
#include <WiFi.h>
#include <HTTPClient.h>
//separador library
#include <Separador.h>
Separador s;

// WiFi
const char *ssid = "your_ssid"; // Enter your WiFi name
const char *password = "your_password";  // Enter WiFi password

//Your Domain name with URL path or IP address with path
const char* serverName = "http://192.168.0.8:1880/update-sensor";
//repleace  192.168.0.101 for your addres
const char* serverNameGet = "http://192.168.0.8:1880/get-sensor";
//repleace  192.168.0.101 for your addres

String rpm_http = "";

//motor
int motor1Pin1 = 33;
int motor1Pin2 = 25;
int enable1Pin = 32;


// Setting PWM properties
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 0;

//move
String move_motor = "counterclockwise";


int encoder = 27;

// banderas para enviar una sola vez el dato de conexión exitosa
int band1= 0, band2=0;

void motor( void *pvParameters );
void comands( void *pvParameters );
//void enviar( void *pvParameters );
void RPM( void *pvParameters );

//volatile int counter = 0;
int counter = 0;

// Gethttp
String sensorReadings;
float sensorReadingsArr[3];
```

```
float ang_compuerta = 0; // variable donde se guardará la posición actual de
la compuerta con cada interrupción activada
int quotient=0; // variable que guardara el resto
int x=0,y,temp = 0; // variables para el calculo del resto
char hexadecimalNumber[4]="000"; // arreglo donde se guardara la codificacion

// variables de recepción y transmisión para los mensajes hacia y desde el
NODERED
char MensajeRx[7] = "PPPPPP";
char MensajeTx[8] = "$AAAAA@";
char Duty[4] = "PPP";

void interruption()    // Function that runs during each interrupt
{
  counter++;
  if (MensajeRx[1] == 'A')  // a favor de las manecillas del reloj
  {
    ang_compuerta = ang_compuerta + 0.0296;
  }
  if (MensajeRx[1] == 'B')  // en contra de las manecillas del reloj
  {
    ang_compuerta = ang_compuerta - 0.0296;
  }
  quotient = ang_compuerta/0.0296;
  while(quotient!=0){

            temp = quotient % 16;

            //To convert integer into character
            if( temp < 10)
            {
                   temp =temp + 48;
            }

            else
            {
                temp = temp + 55;
            }

            hexadecimalNumber[x++]= temp;
            quotient = quotient / 16;
        }
    MensajeTx[2] = hexadecimalNumber[2];
    MensajeTx[3] = hexadecimalNumber[1];
    MensajeTx[4] = hexadecimalNumber[0];

    hexadecimalNumber[2] = '0';
    hexadecimalNumber[1] = '0';
    hexadecimalNumber[0] = '0';

    x = 0;

    Serial.print (MensajeTx);

}

void setup() {

    //wifi
```

```
  WiFi.mode(WIFI_STA);

  Serial.begin(115200);
  // sets the pins as outputs:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);



  // configure LED PWM functionalitites
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(enable1Pin, pwmChannel);


  attachInterrupt(encoder, interruption, RISING);


  xTaskCreatePinnedToCore(
    motor
    ,  "MotorDC"          // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 1


  xTaskCreatePinnedToCore(
    comands
    ,  "Comands"          // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 //core 1


  xTaskCreatePinnedToCore(
    RPM
    ,  "RPM"               // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 0


  //wifi
  WiFi.mode(WIFI_STA);
}
```

```
void loop() {
  connect_wifi();
  gethost(ServerName);
  publicData();
  vTaskDelay(999);
}

void motor( void *pvParameters ) {
  while (1) {

    if (move_motor == "clockwise") {
      // Move the DC motor forward at maximum speed
      digitalWrite(motor1Pin1, LOW);
      digitalWrite(motor1Pin2, HIGH);
      ledcWrite(pwmChannel, dutyCycle);
    } else if (move_motor == "counterclockwise") {
      // Stop the DC motor
      digitalWrite(motor1Pin1, HIGH);
      digitalWrite(motor1Pin2, LOW);
      ledcWrite(pwmChannel, dutyCycle);
    }

  }

}



void comands(void *pvParameters) {
  while (1) {
    if (Serial.available()) {
      String string = Serial.readStringUntil('@');
      string.toCharArray(MensajeRx, 7);
      Duty[0] = MensajeRx[2];
      Duty[1] = MensajeRx[3];
      Duty[2] = MensajeRx[4];
      dutyCycle = atol(Duty);
      MensajeTx[1] = MensajeRx[2];
      if (MensajeRx[2] == 'A') {
       //Serial.println("Motor move on clockwise");
        move_motor = "clockwise";
       }
       if (MensajeRx[2] == 'B') {
         //Serial.println("Motor move on counterclockwise");
          move_motor = "counterclockwise";
       }
    }
  }
}




void RPM( void *pvParameters ) {


  while (1) {

    vTaskDelay(999);
```

```
    MensajeTx[5] = counter;
    //Serial.print(counter*60); // eliminado porque quiere obtener las rps
    //Serial.print(counter); // agregado porque quiere obtener las rps
    //Serial.println(" RPM");     // quitado por actualización de protocólo
    rpm_http = String(counter);
    counter = 0;

  }



}


void connect_wifi(){
  // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, password);  // Connect to WPA/WPA2 network. Change
this line if using open or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
}


void publicData(){
  WiFiClient client;
      HTTPClient http;

      // Your Domain name with URL path or IP address with path
      http.begin(client, serverName);

      // Specify content-type header
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      // Data to send with HTTP POST
      String httpRequestData =
"api_key=tPmAT5Ab3j7F9&sensor=DS18B20&value1="+MensajeTx+"&value2="+rpm_http;

      // Send HTTP POST request
      int httpResponseCode = http.POST(httpRequestData);


    if(band1=0)
      {
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      band1++;
      }

      // Free resources
      http.end();
}
```

```
String httpGETRequest(const char* serverName) {
  WiFiClient client;
  HTTPClient http;

  // Your Domain name with URL path or IP address with path
  http.begin(client, serverName);

  // Send HTTP POST request
  int httpResponseCode = http.GET();

  String payload = "{}";

  if (httpResponseCode>0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    payload = http.getString();
    Serial.println(payload);
    dutyCycle = payload.toInt();
  }
  else {
      if(band2=0)
      {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
        band2++;
      }
  }
  // Free resources
  http.end();

  return payload;
}
```

APÉNDICE P. TSC_LAB Planta

```
/*
   ***************************** TSC-Lab ******************************
   ******************** Proyecto de titulación ************************
   By: Carlos Zúñiga Reyes

*/

//separador library
#include <Separador.h>
Separador s;

//motor
int motor1Pin1 = 33;
int motor1Pin2 = 25;
int enable1Pin = 32;


// Setting PWM properties
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 0;

//move
String move_motor = "counterclockwise";


int encoder = 27;


void motor( void *pvParameters );
void comands( void *pvParameters );
//void enviar( void *pvParameters );
void RPM( void *pvParameters );

//volatile int counter = 0;
int counter = 0;

float ang_compuerta = 0; // variable donde se guardará la posición actual de
la compuerta con cada interrupción activada
int quotient=0; // variable que guardara el resto
int x=0,y,temp = 0; // variables para el calculo del resto
char hexadecimalNumber[4]="000"; // arreglo donde se guardara la codificacion

// variables de recepción y transmisión para los mensajes hacia y desde el
NODERED
char MensajeRx[7] = "PPPPPP";
char MensajeTx[8] = "$AAAAA@";
char Duty[4] = "PPP";

void interruption()    // Function that runs during each interrupt
{
  counter++;
  if (MensajeRx[1] == 'A')  // a favor de las manecillas del reloj
  {
    ang_compuerta = ang_compuerta + 0.0296;
  }
  if (MensajeRx[1] == 'B')  // en contra de las manecillas del reloj
```

```
  {
    ang_compuerta = ang_compuerta - 0.0296;
  }
  quotient = ang_compuerta/0.0296;
  while(quotient!=0){

            temp = quotient % 16;

            //To convert integer into character
            if( temp < 10)
            {
                   temp =temp + 48;
            }

            else
            {
                temp = temp + 55;
            }

            hexadecimalNumber[x++]= temp;
            quotient = quotient / 16;
        }
    MensajeTx[2] = hexadecimalNumber[2];
    MensajeTx[3] = hexadecimalNumber[1];
    MensajeTx[4] = hexadecimalNumber[0];

    hexadecimalNumber[2] = '0';
    hexadecimalNumber[1] = '0';
    hexadecimalNumber[0] = '0';

    x = 0;

    Serial.print (MensajeTx);

}

void setup() {
  Serial.begin(115200);
  // sets the pins as outputs:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);



  // configure LED PWM functionalitites
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(enable1Pin, pwmChannel);


  attachInterrupt(encoder, interruption, RISING);


  xTaskCreatePinnedToCore(
    motor
    ,  "MotorDC"         // Descriptive name of the function (MAX 8
characters)
    ,  2048              // Size required in STACK memory
```

```
    ,  NULL              // INITIAL parameter to receive (void *)
    ,  1                 // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL              // Variable that points to the task (optional)
    , 1);                // core 1


  xTaskCreatePinnedToCore(
    comands
    ,  "Comands"         // Descriptive name of the function (MAX 8
characters)
    ,  2048              // Size required in STACK memory
    ,  NULL              // INITIAL parameter to receive (void *)
    ,  1                 // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL              // Variable that points to the task (optional)
    , 1);                //core 1


  xTaskCreatePinnedToCore(
    RPM
    ,  "RPM"              // Descriptive name of the function (MAX 8
characters)
    ,  2048              // Size required in STACK memory
    ,  NULL              // INITIAL parameter to receive (void *)
    ,  1                 // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL              // Variable that points to the task (optional)
    , 1);                // core 0



}

void loop() {
}

void motor( void *pvParameters ) {
  while (1) {

    if (move_motor == "clockwise") {
      // Move the DC motor forward at maximum speed
      digitalWrite(motor1Pin1, LOW);
      digitalWrite(motor1Pin2, HIGH);
      ledcWrite(pwmChannel, dutyCycle);
    } else if (move_motor == "counterclockwise") {
      // Stop the DC motor
      digitalWrite(motor1Pin1, HIGH);
      digitalWrite(motor1Pin2, LOW);
      ledcWrite(pwmChannel, dutyCycle);
    }

  }

}



void comands(void *pvParameters) {
  while (1) {
```

```
    if (Serial.available()) {
      String string = Serial.readStringUntil('@');
      string.toCharArray(MensajeRx, 7);
      Duty[0] = MensajeRx[2];
      Duty[1] = MensajeRx[3];
      Duty[2] = MensajeRx[4];
      dutyCycle = atol(Duty);
      MensajeTx[1] = MensajeRx[2];
      if (MensajeRx[2] == 'A') {
       //Serial.println("Motor move on clockwise");
        move_motor = "clockwise";
       }
       if (MensajeRx[2] == 'B') {
         //Serial.println("Motor move on counterclockwise");
          move_motor = "counterclockwise";
       }
    }
  }
}




void RPM( void *pvParameters ) {


  while (1) {

    vTaskDelay(999);
    MensajeTx[5] = counter;
    //Serial.print(counter*60); // eliminado porque quiere obtener las rps
    //Serial.print(counter); // agregado porque quiere obtener las rps
    //Serial.println(" RPM");     // quitado por actualización de protocólo
    counter = 0;

  }



}
```

APÉNDICE Q. TSC_LAB LAZO ABIERTO

```
/*
   ****************************** TSC-Lab ******************************
  This practice is aboutNode-Red (with wi-fi)
*/
//initial setting for data acquisition
int dutyCycleInitial = 255;
int dutyCycleFinish = 0;
int period = 13000;
int cycles = 10;
int dutyCycle = 0;

//separador library
#include <Separador.h>
Separador s;

//motor
int motor1Pin1 = 33;
int motor1Pin2 = 25;
int enable1Pin = 32;
int motor_status = 1;

// Setting PWM properties
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;

//move
String move_motor = "counterclockwise";

int encoder = 27;

void motor( void *pvParameters );
//void enviar( void *pvParameters );
void RPM( void *pvParameters );
void pwm( void *pvParameters );

volatile int counter = 0;

void interruption()    // Function that runs during each interrupt
{
  counter++;
}

void setup() {
  Serial.begin(115200);
  // sets the pins as outputs:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  // configure LED PWM functionalitites
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(enable1Pin, pwmChannel);
  attachInterrupt(encoder, interruption, RISING);
```

```
  xTaskCreatePinnedToCore(
    motor
    ,  "MotorDC"          // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 1

  xTaskCreatePinnedToCore(
    RPM
    ,  "RPM"               // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 0

  xTaskCreatePinnedToCore(
    pwm
    ,  "PWM"               // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 0
}

void loop() {
}

void motor( void *pvParameters ) {
  while (1) {
      digitalWrite(motor1Pin1, HIGH);
      digitalWrite(motor1Pin2, LOW);
      ledcWrite(pwmChannel, dutyCycle*255);
      //vTaskDelay(period);
  }
}

// Calcula las RPM quye tiene el motor
void RPM( void *pvParameters ) {
  while (1) {
    vTaskDelay(999);
    //Serial.println(counter * 60);//255 -> 0x32 0x35 0x35
    Serial.write(counter);//0-255
    counter = 0;
  }
}

// Read del PWM que viene desde Matlab
void pwm( void *pvParameters ) {
  while (1) {
```

```
    //Serial.println("hola");
    if (Serial.available())
    {
      String string = Serial.readStringUntil('\n');
      dutyCycle = string.toInt();
    }
  }
}
```

APÉNDICE R. MATLAB Gráfica datos recopilados

```
%%
Datos=load('3.csv'); %Cargar los datos del excel y guardarlos en la variable
OUT=Datos(:,1); %Salida del Sistema
IN=Datos(:,2); %Entrada del Sistema (Verificar que este valor este entre 0 y
255, caso contrario *255)
Time=(0:1:length(Datos)-1)'; %Vector de tiempo
% Gráfica de los datos
figure(1)
plot(Time,IN); %Se grafica la entrada del sistema
title('Entrada: Altos y Bajos del Microcontrolador')
ylabel('Señal del microcontrolador')
xlabel('Tiempo [s]')
grid on
hold on
plot(Time,OUT);
```

APÉNDICE S. MATLAB Función de Transferencia

```
clear;clc;%clear all
%%
G=tf(d2c(bj20120));%<-----------------------
%%
num=cell2mat(G.Numerator);
den=cell2mat(G.Denominator);
FTmotor=tf(num,den)
```

APÉNDICE T. TSC_LAB LAZO CERRADO

```
/*
   ****************************** TSC-Lab ******************************
  This practice is aboutNode-Red (with wi-fi)
*/
//initial setting for data acquisition
int dutyCycleInitial = 255;
int dutyCycleFinish = 0;
int period = 13000;
int cycles = 10;
int dutyCycle = 0;

//separador library
#include <Separador.h>
Separador s;

//motor
int motor1Pin1 = 33;
int motor1Pin2 = 25;
int enable1Pin = 32;
int motor_status = 1;

// Setting PWM properties
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;

//move
String move_motor = "counterclockwise";

int encoder = 27;

void motor( void *pvParameters );
//void enviar( void *pvParameters );
void RPM( void *pvParameters );
void pwm( void *pvParameters );

volatile int counter = 0;

void interruption()     // Function that runs during each interrupt
{
  counter++;
}

void setup() {
  Serial.begin(115200);
  // sets the pins as outputs:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  // configure LED PWM functionalitites
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(enable1Pin, pwmChannel);
  attachInterrupt(encoder, interruption, RISING);

  xTaskCreatePinnedToCore(
```

```
    motor
    ,  "MotorDC"          // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 1

  xTaskCreatePinnedToCore(
    RPM
    ,  "RPM"               // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 0

  xTaskCreatePinnedToCore(
    pwm
    ,  "PWM"               // Descriptive name of the function (MAX 8
characters)
    ,  2048               // Size required in STACK memory
    ,  NULL               // INITIAL parameter to receive (void *)
    ,  1                  // Priority, priority = 3 (configMAX_PRIORITIES - 1)
is the highest, priority = 0 is the lowest.
    ,  NULL               // Variable that points to the task (optional)
    , 1);                 // core 0
}

void loop() {
}

void motor( void *pvParameters ) {
  while (1) {
      digitalWrite(motor1Pin1, HIGH);
      digitalWrite(motor1Pin2, LOW);
      ledcWrite(pwmChannel, dutyCycle);
      //vTaskDelay(period);
  }
}

// Calcula las RPM quye tiene el motor
void RPM( void *pvParameters ) {
  while (1) {
    vTaskDelay(999);
    //Serial.println(counter * 60);//255 -> 0x32 0x35 0x35
    Serial.write(counter);//0-255
    counter = 0;
  }
}

// Read del PWM que viene desde Matlab
void pwm( void *pvParameters ) {
  while (1) {
    //Serial.println("hola");
    if (Serial.available())
```

```
    {
      String string = Serial.readStringUntil('\n');
      dutyCycle = string.toInt();
    }
  }
}
```

APÉNDICE U. BLOQUE DE PROGRAMAS PLC

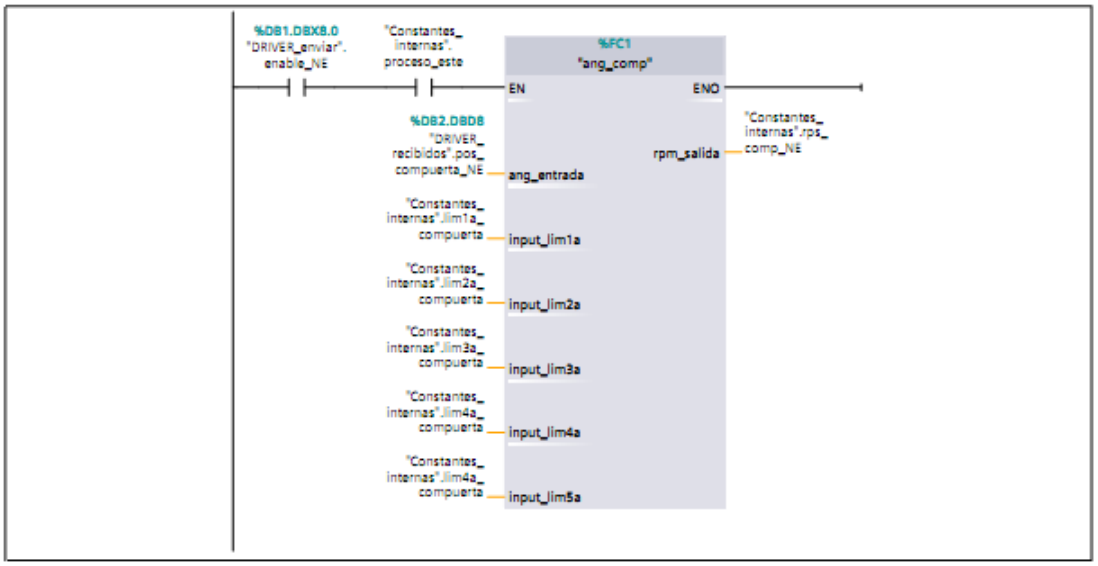
Bloques de programa

Main [OB1]

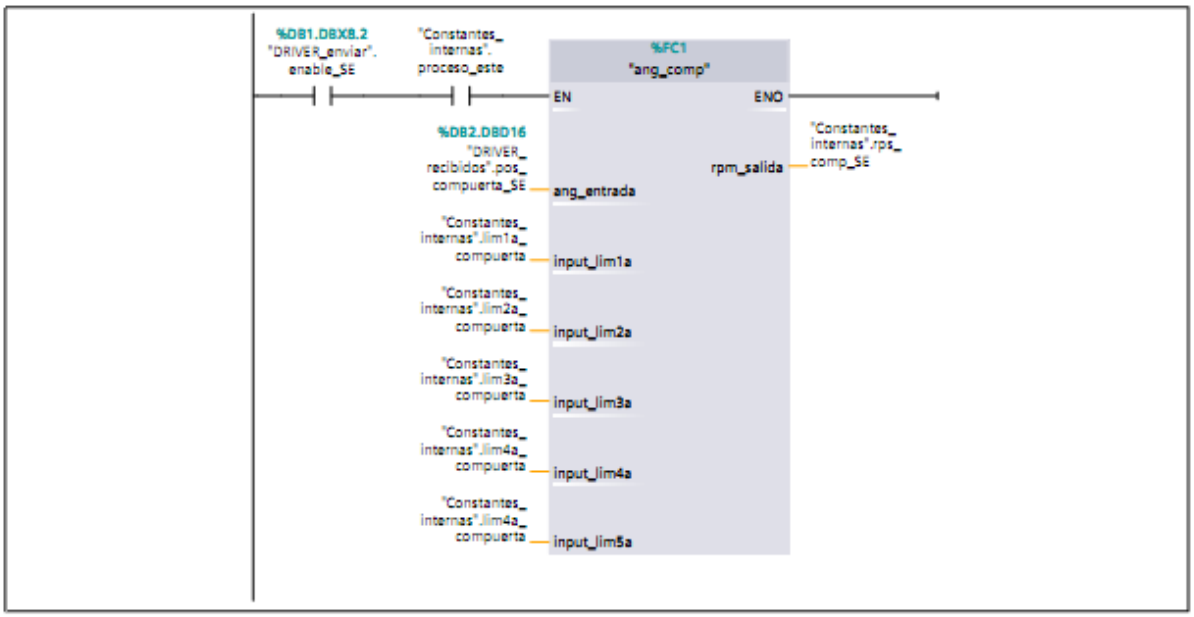
Main Propiedades					
General					
Nombre	Main	**Número**	1	**Tipo**	OB
Idioma	KOP	**Numeración**	Automático		
Información					
Título	"Main Program Sweep (Cycle)"	**Autor**		**Comentario**	
Familia		**Versión**	0.1	**ID personalizado**	

Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

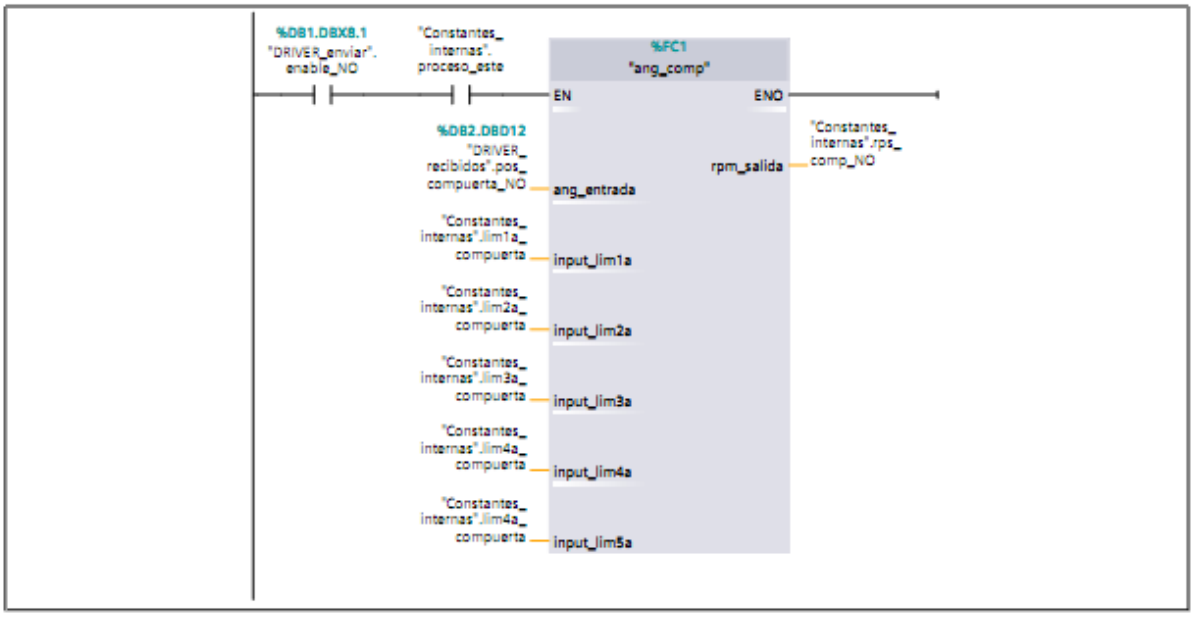
Segmento 1: Compuerta N.E. Apertura



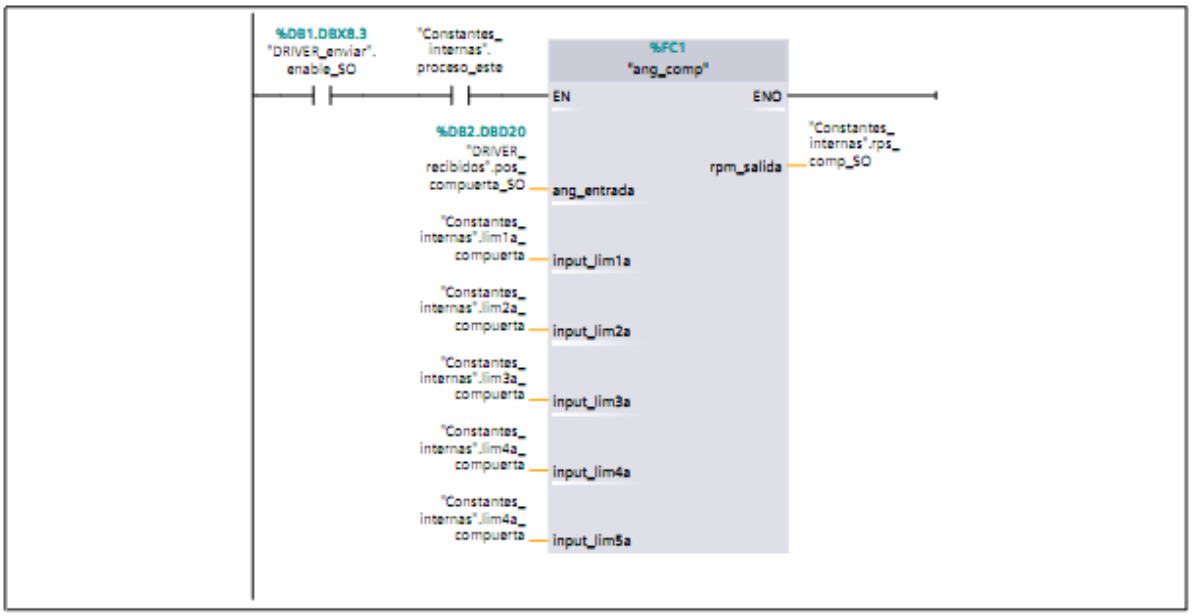
Segmento 2: Compuerta S.E. Apertura



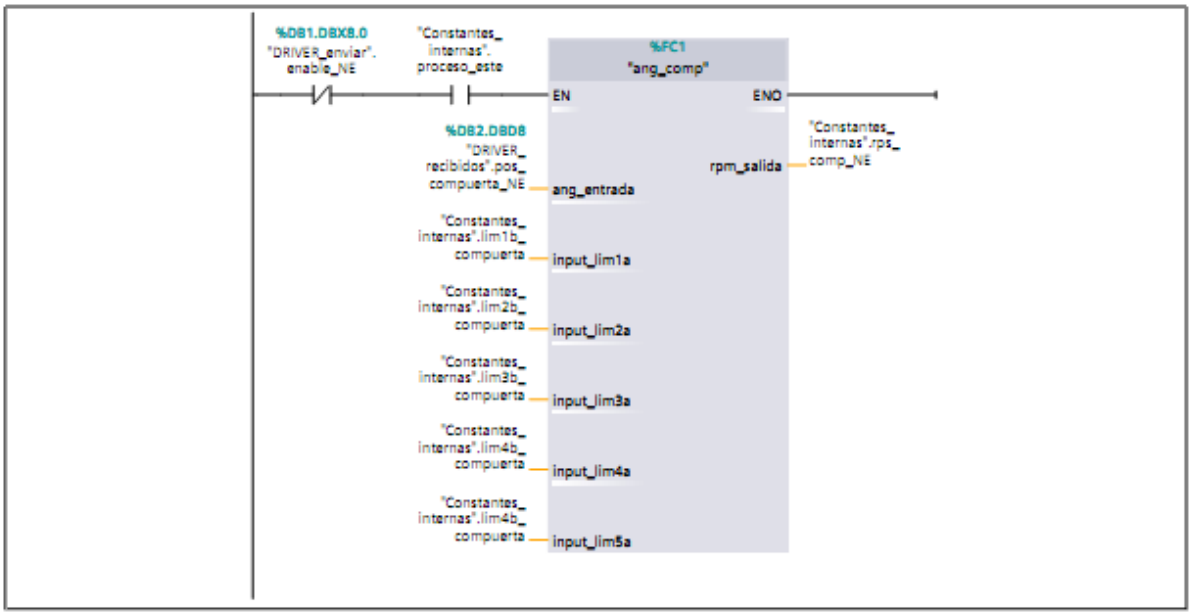
Segmento 3: Compuerta N.O. Apertura



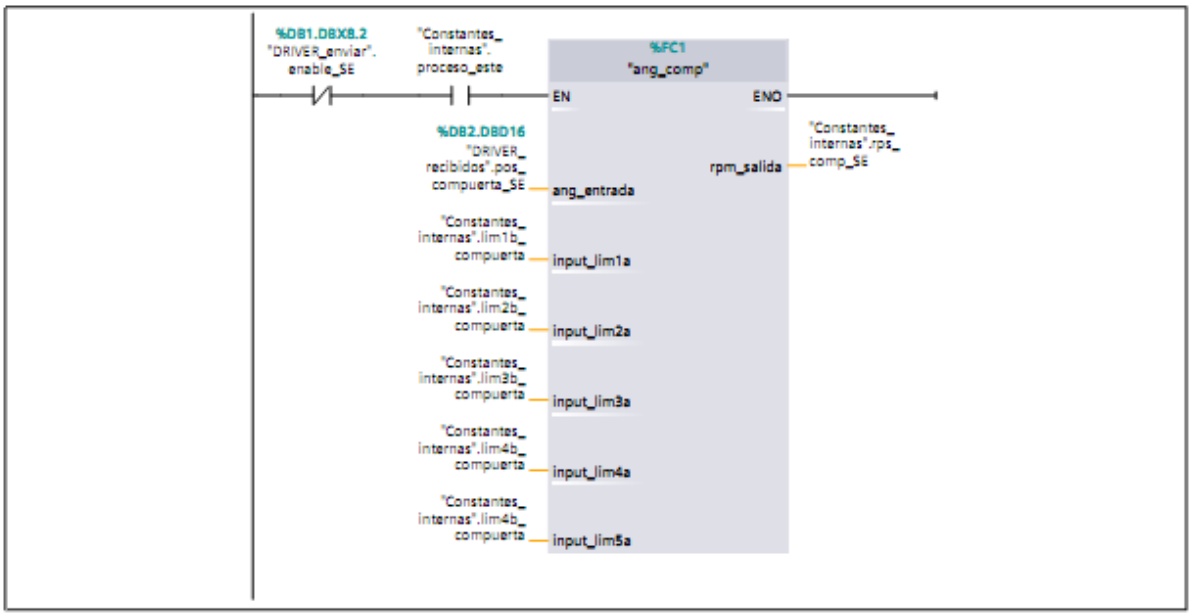
Segmento 4: Compuerta S.O. Apertura



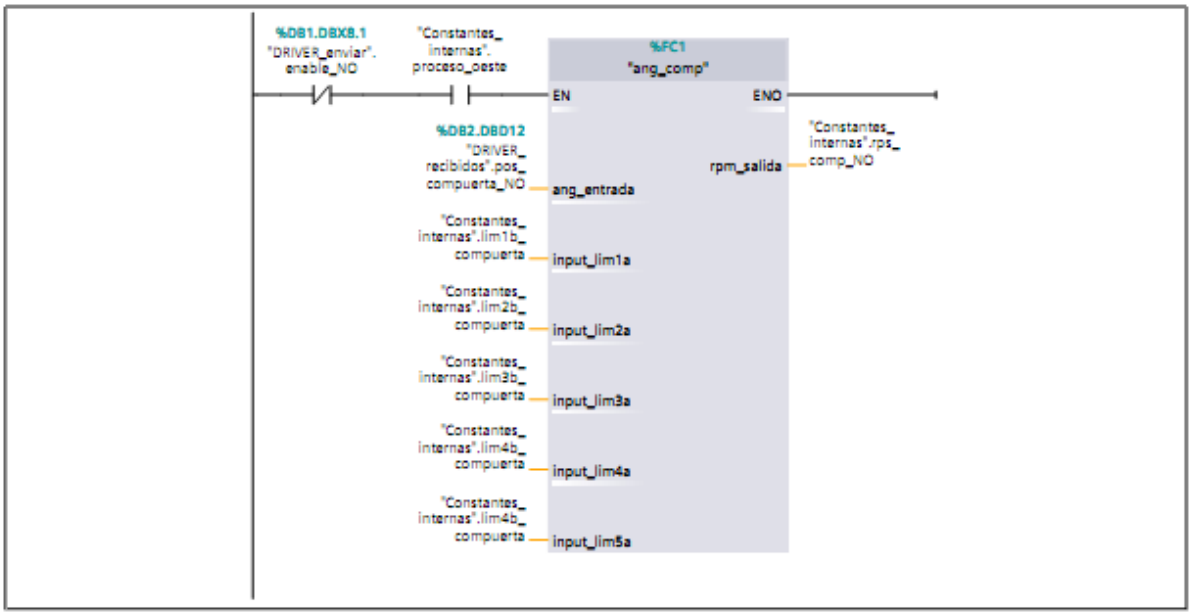
Segmento 5: Compuerta N.E. Cierre



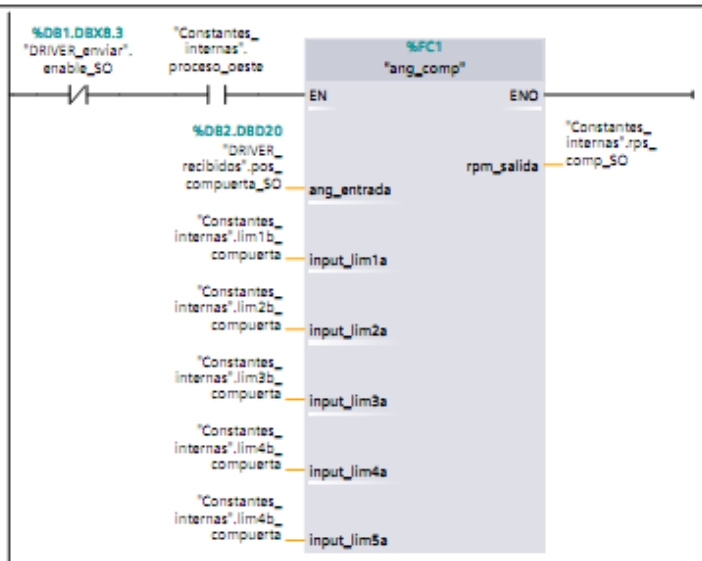
Segmento 6: Compuerta S.E. Cierre



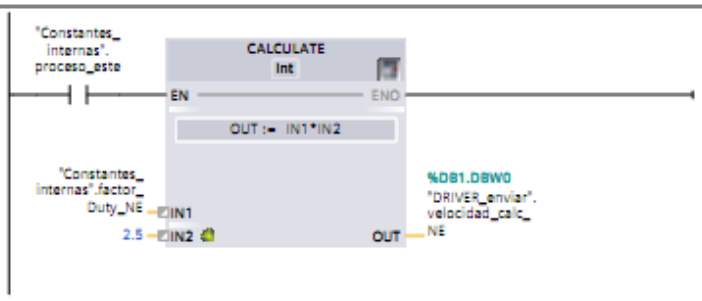
Segmento 7: Compuerta N.O. Cierre



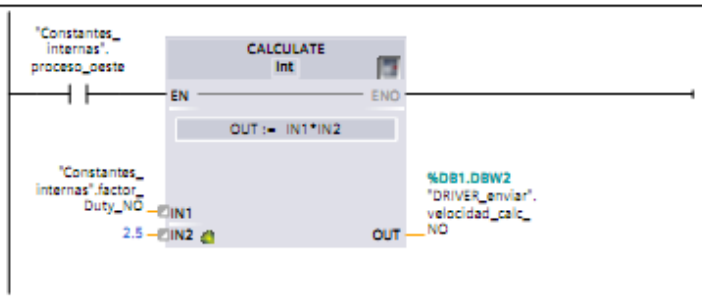
Segmento 8: Compuerta S.O. Cierre



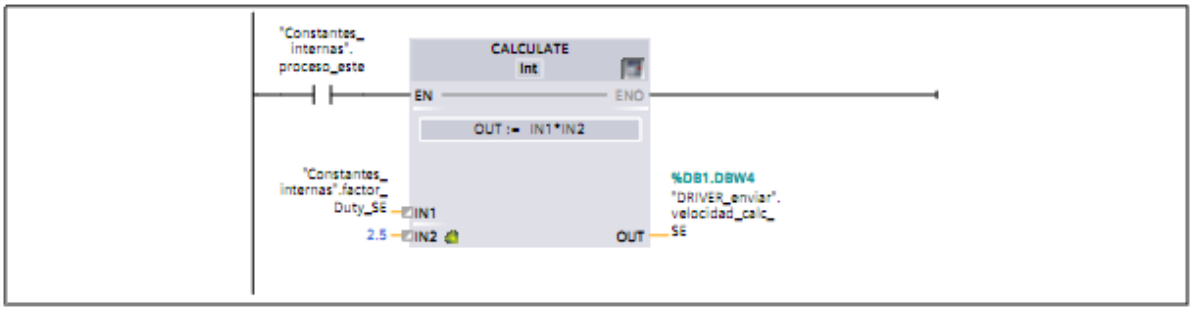
Segmento 9:



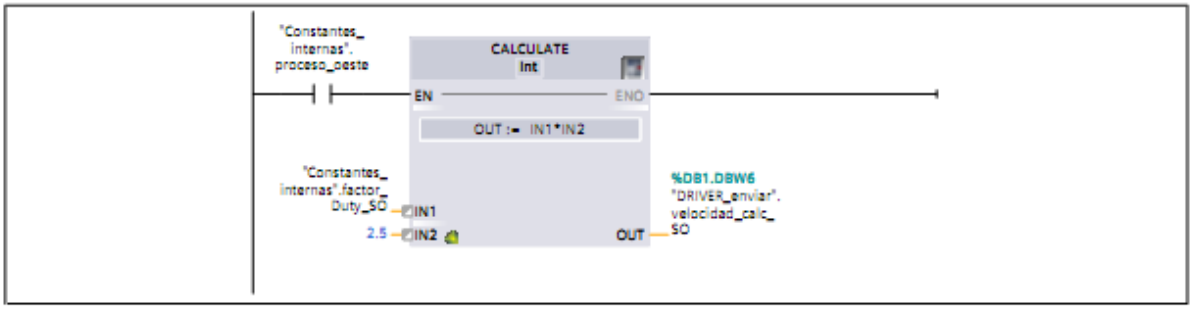
Segmento 10:



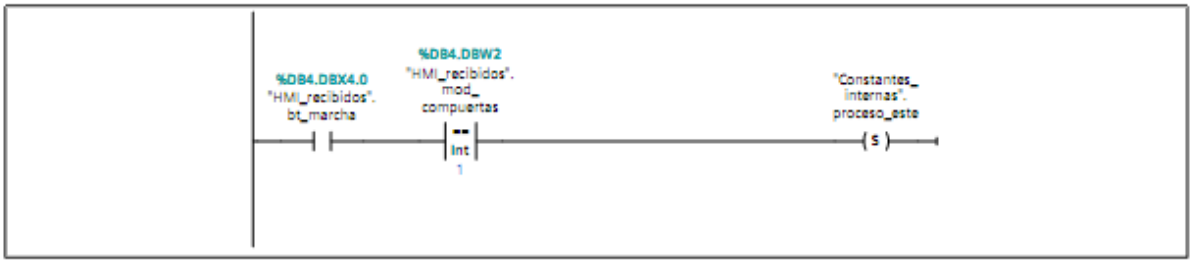
Segmento 11:



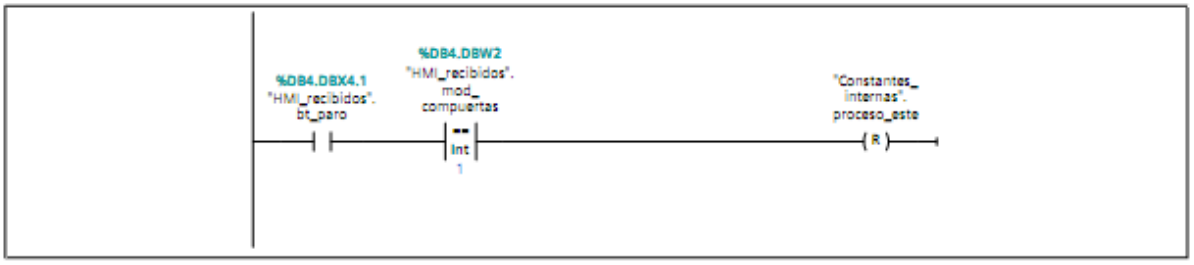
Segmento 12:



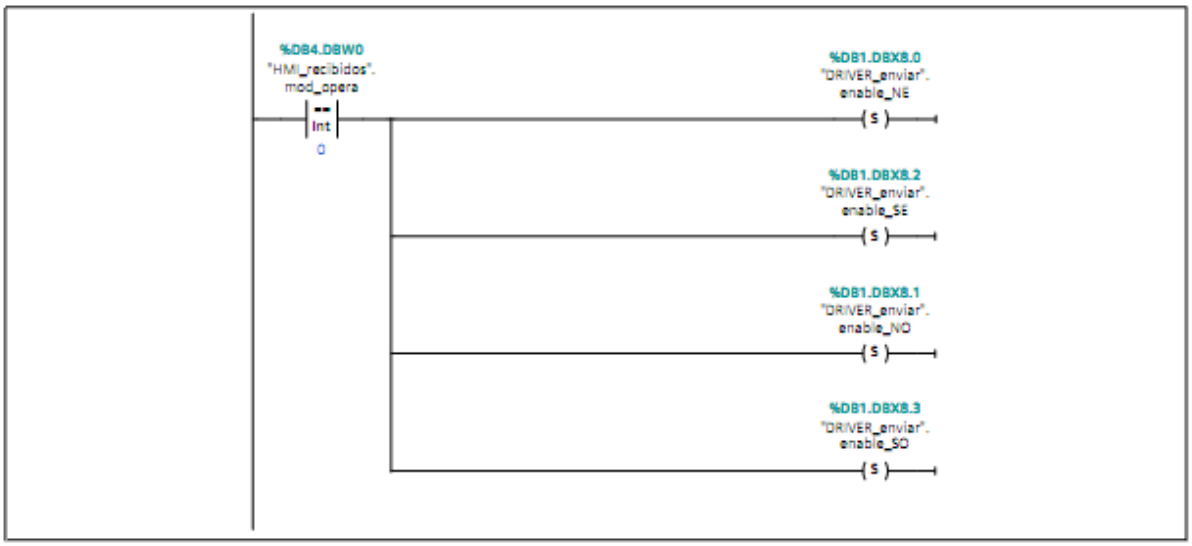
Segmento 13:



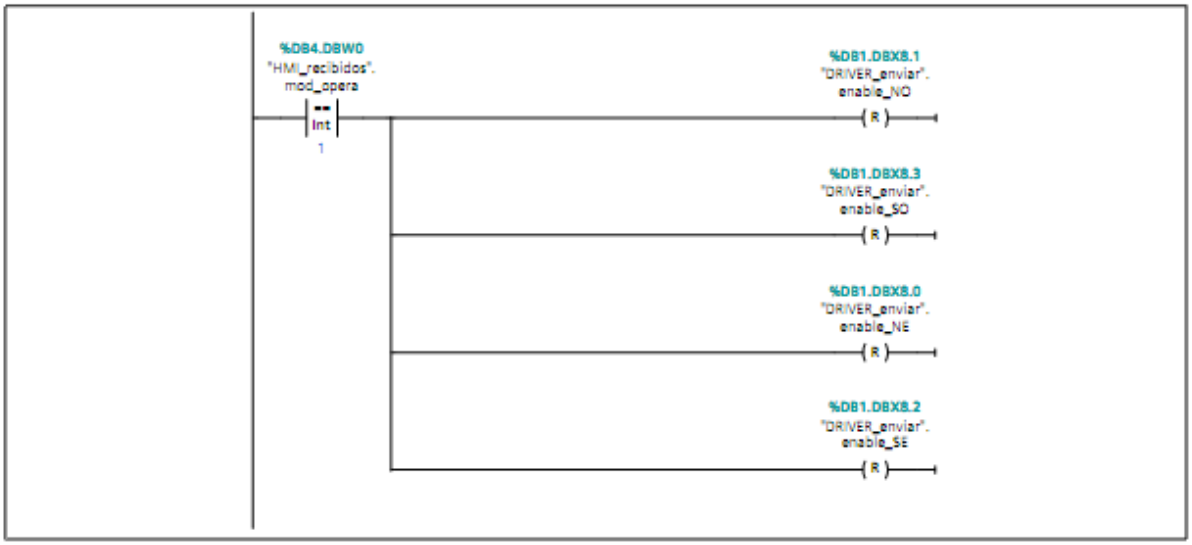
Segmento 14:



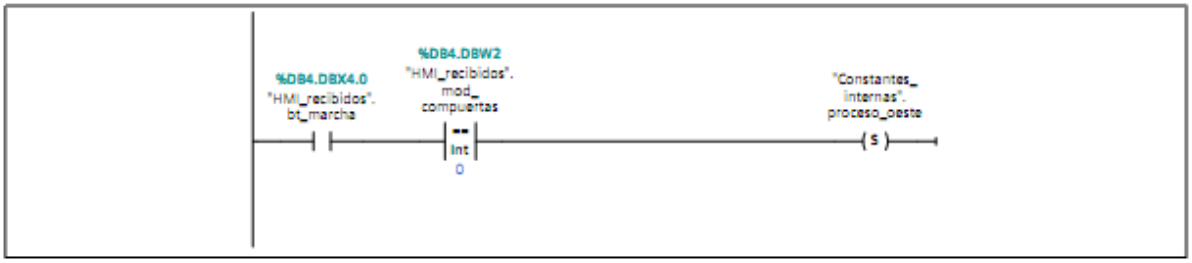
Segmento 15: Esta activación proviene del HMI enviado al PLC y este puede mantar esa bandera a el driver del motor en caso de ser necesario usarlo



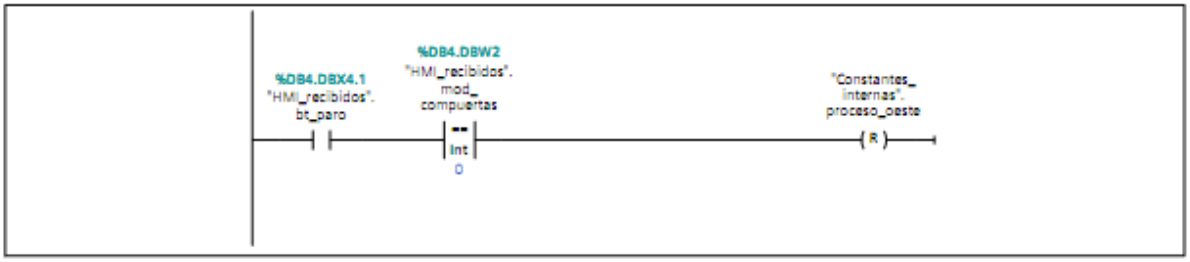
Segmento 16: Esta activación proviene del HMI enviado al PLC y este puede mantar esa bandera a el driver del motor en caso de ser necesario usarlo



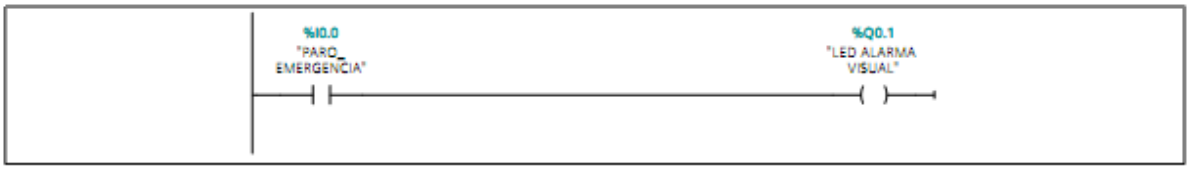
Segmento 17:



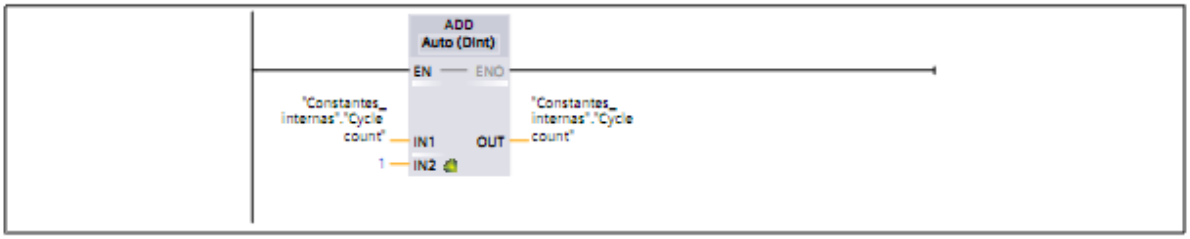
Segmento 18:



Segmento 19:



Segmento 20:



Bloques de programa

DRIVER_enviar [DB1]

DRIVER_enviar Propiedades					
General					
Nombre	DRIVER_enviar	Número	1	Tipo	DB
Idioma	DB	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizado	

Nombre	Tipo de datos	Offset	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ Static										
velocidad_calc_NE	Int	0.0	0	False	True	True	True	False		velocidad enviada al driver del motor NE, el cálculo de la velocidad se da usando la posición del encoder
velocidad_calc_NO	Int	2.0	0	False	True	True	True	False		velocidad enviada al driver del motor NE, el cálculo de la velocidad se da usando la posición del encoder
velocidad_calc_SE	Int	4.0	0	False	True	True	True	False		velocidad enviada al driver del motor NE, el cálculo de la velocidad se da usando la posición del encoder
velocidad_calc_SO	Int	6.0	0	False	True	True	True	False		velocidad enviada al driver del motor NE, el cálculo de la velocidad se da usando la posición del encoder
enable_NE	Bool	8.0	false	False	True	True	True	False		Esta activación de giro proviene del HMI enviado al PLC, si es 1 gira con las manecillas del reloj
enable_NO	Bool	8.1	false	False	True	True	True	False		Esta activación de giro proviene del HMI enviado al PLC, si es 1 gira con las manecillas del reloj
enable_SE	Bool	8.2	false	False	True	True	True	False		Esta activación de giro proviene del HMI enviado al PLC, si es 1 gira con las manecillas del reloj

Nombre	Tipo de datos	Offset	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
enable_SO	Bool	8.3	false	False	True	True	True	False		Esta activación de giro proviene del HMI enviado al PLC, si es 1 gira con las manecillas del reloj

Bloques de programa

DRIVER_recibidos [DB2]

DRIVER_recibidos Propiedades					
General					
Nombre	DRIVER_recibidos	Número	2	Tipo	DB
Idioma	DB	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizado	

Nombre	Tipo de datos	Offset	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ Static										
lectura_rpm_NE	Int	0.0	0	False	True	True	True	False		Lectura de las rpm del motor NE
lectura_rpm_NO	Int	2.0	0	False	True	True	True	False		Lectura de las rpm del motor NO
lectura_rpm_SE	Int	4.0	0	False	True	True	True	False		Lectura de las rpm del motor SE
lectura_rpm_SO	Int	6.0	0	False	True	True	True	False		Lectura de las rpm del motor SO
pos_compuerta_NE	Real	8.0	0.0	False	True	True	True	False		Lectura de la posición actual de la compuerta NE
pos_compuerta_NO	Real	12.0	0.0	False	True	True	True	False		Lectura de la posición actual de la compuerta NO
pos_compuerta_SE	Real	16.0	0.0	False	True	True	True	False		Lectura de la posición actual de la compuerta SE
pos_compuerta_SO	Real	20.0	0.0	False	True	True	True	False		Lectura de la posición actual de la compuerta SO
giro_NE	Bool	24.0	false	False	True	True	True	False		Lectura de sentido en el que gira la compuerta NE
giro_NO	Bool	24.1	false	False	True	True	True	False		Lectura de sentido en el que gira la compuerta NO
giro_SE	Bool	24.2	false	False	True	True	True	False		Lectura de sentido en el que gira la compuerta SE
giro_SO	Bool	24.3	false	False	True	True	True	False		Lectura de sentido en el que gira la compuerta SO

Bloques de programa

HMI_enviar [DB3]

HMI_enviar Propiedades					
General					
Nombre	HMI_enviar	Número	3	Tipo	DB
Idioma	DB	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizado	

Nombre	Tipo de datos	Offset	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ Static										
grados_NE	Real	0.0	0.0	False	True	True	True	False		los grados sensados en tiempo real por el encoder NE y que serán enviados al HMI
grados_NO	Real	4.0	0.0	False	True	True	True	False		los grados sensados en tiempo real por el encoder NO y que serán enviados al HMI
grados_SE	Real	8.0	0.0	False	True	True	True	False		los grados sensados en tiempo real por el encoder SE y que serán enviados al HMI
grados_SO	Real	12.0	0.0	False	True	True	True	False		los grados sensados en tiempo real por el encoder SO y que serán enviados al HMI
rpm_NE	Real	16.0	0.0	False	True	True	True	False		Valor de rpm sensados para mostrar en HMI compuerta NE
rpm_NO	Real	20.0	0.0	False	True	True	True	False		Valor de rpm sensados para mostrar en HMI compuerta NO
rpm_SE	Real	24.0	0.0	False	True	True	True	False		Valor de rpm sensados para mostrar en HMI compuerta SE
rpm_SO	Real	28.0	0.0	False	True	True	True	False		Valor de rpm sensados para mostrar en HMI compuerta SO
bt_emergencia	Bool	32.0	false	False	True	True	True	False		Aviso de activación manual de paro de emergencia
fin_cerrado	Bool	32.1	false	False	True	True	True	False		Aviso de apertura

Nombre	Tipo de datos	Offset	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
fin_abierto	Bool	32.2	false	False	True	True	True	False		Aviso de cierre

Bloques de programa

HMI_recibidos [DB4]

HMI_recibidos Propiedades					
General					
Nombre	HMI_recibidos	Número	4	Tipo	DB
Idioma	DB	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizado	

Nombre	Tipo de datos	Offset	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ Static										
mod_opera	Int	0.0	0	False	True	True	True	False		Modo de operación, 0 para apertura y 1 para cierre
mod_compuertas	Int	2.0	0	False	True	True	True	False		Modo de compuertas, 0 para las compuertas OESTE y 1 para las compuertas ESTE
bt_marcha	Bool	4.0	false	False	True	True	True	False		Señal de inicio de proceso de apertura o cierre
bt_paro	Bool	4.1	false	False	True	True	True	False		Señal de paro de proceso de apertura o cierre

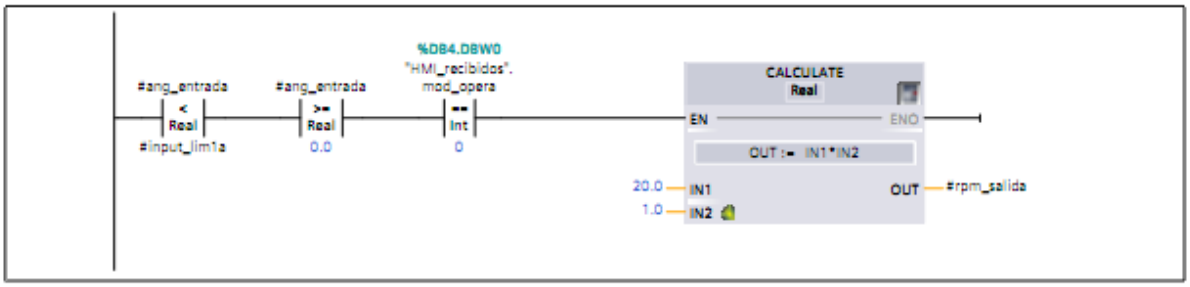
Bloques de programa

ang_comp [FC1]

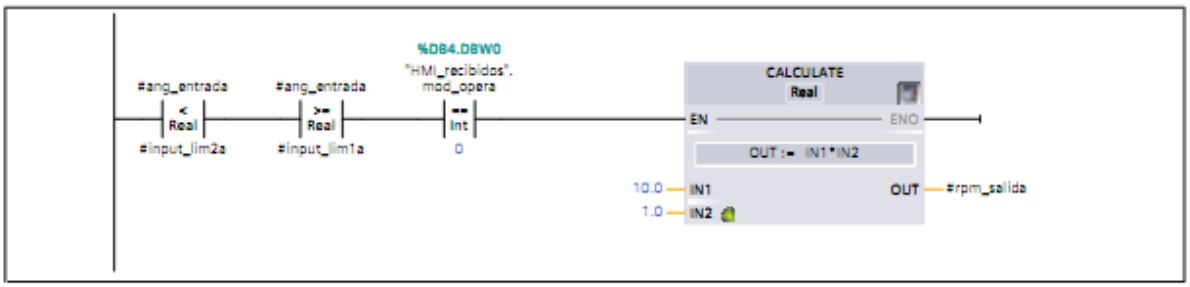
ang_comp Propiedades					
General					
Nombre	ang_comp	Número	1	Tipo	FC
Idioma	KOP	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personali-zado	

Nombre	Tipo de datos	Valor predet.	Comentario
▼ Input			
ang_entrada	Real		
input_lim1a	Real		
input_lim2a	Real		
input_lim3a	Real		
input_lim4a	Real		
input_lim5a	Real		
▼ Output			
rpm_salida	Real		
InOut			
Temp			
Constant			
▼ Return			
ang_comp	Void		

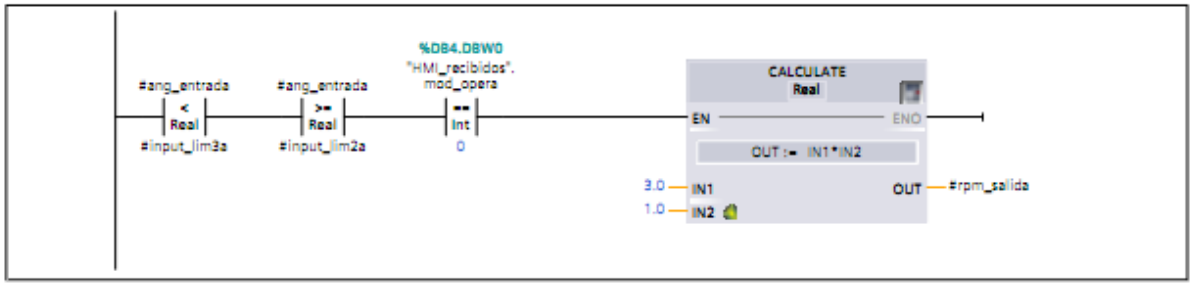
Segmento 1: APERTURA



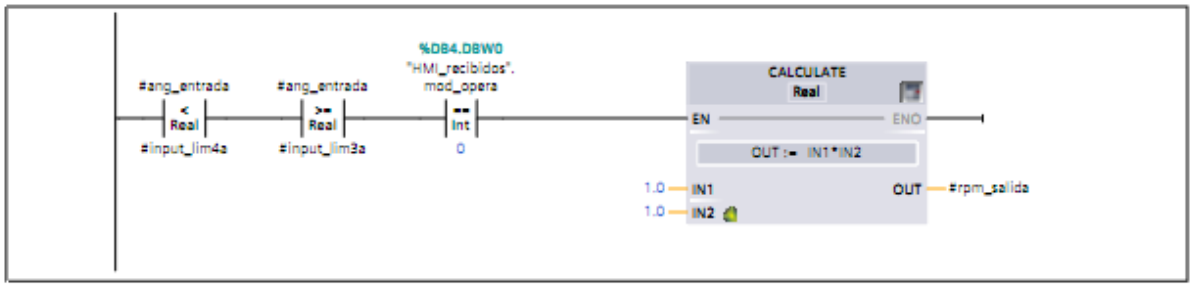
Segmento 2: APERTURA



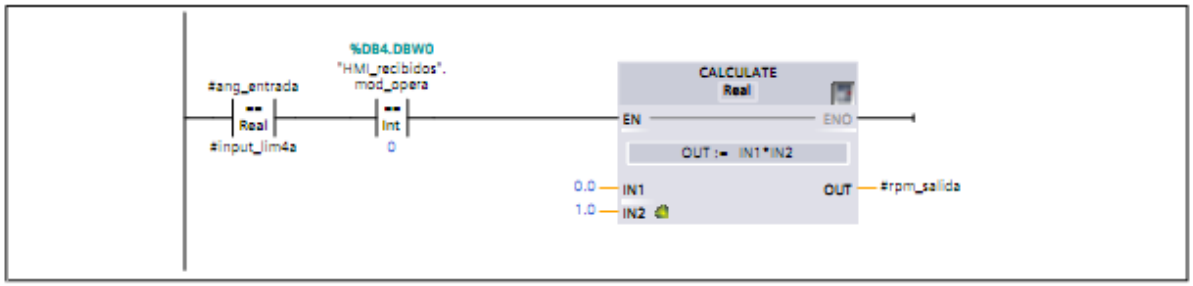
Segmento 3: APERTURA



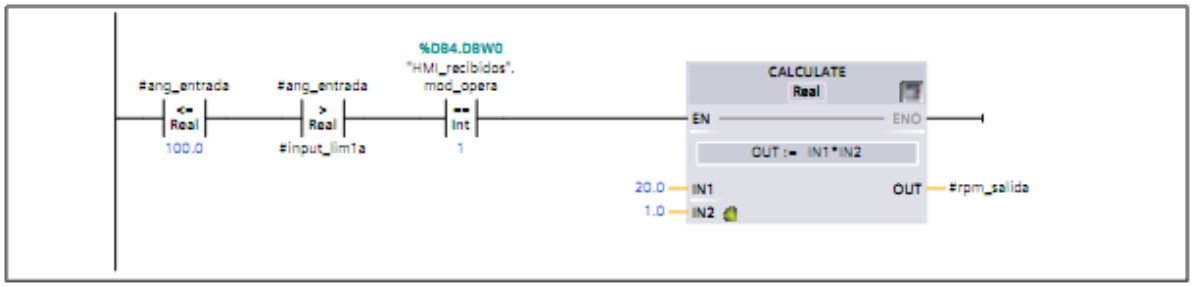
Segmento 4: APERTURA



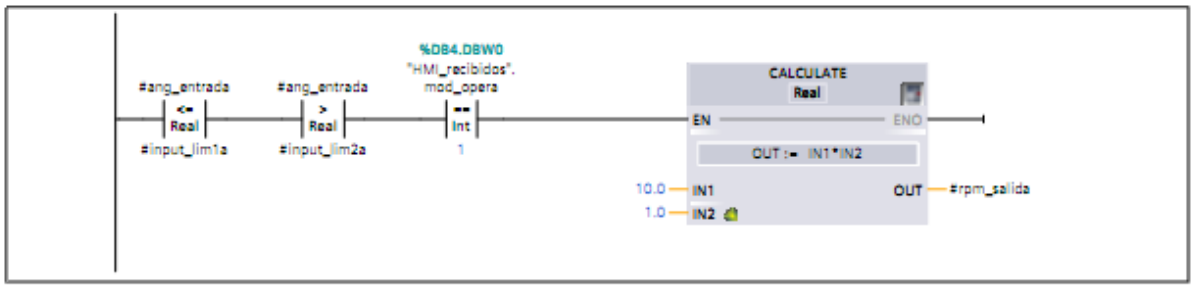
Segmento 5: APERTURA



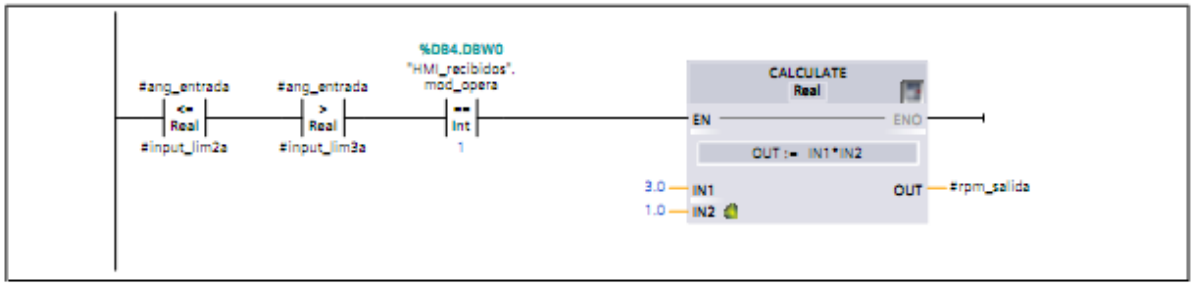
Segmento 6: CIERRE



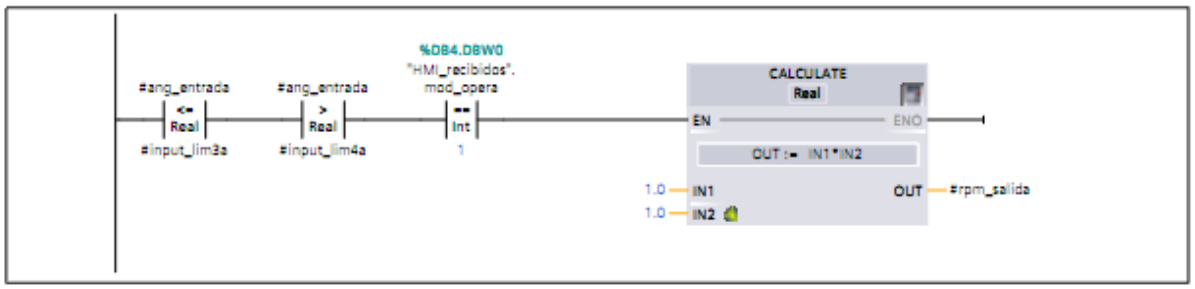
Segmento 7: Cierre



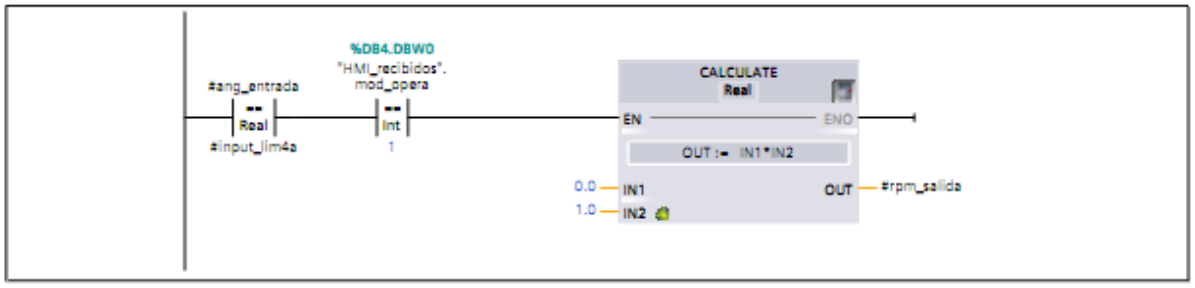
Segmento 8:



Segmento 9:



Segmento 10:



Bloques de programa

Constantes_internas [DB5]

Constantes_internas Propiedades					
General					
Nombre	Constantes_internas	Número	5	Tipo	DB
Idioma	DB	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizado	

Nombre	Tipo de datos	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ Static									
lim1a_compuerta	Real	75.0	False	True	True	True	False		límite de apertura
lim2a_compuerta	Real	80.0	False	True	True	True	False		límite de apertura
lim3a_compuerta	Real	85.0	False	True	True	True	False		límite de apertura
lim4a_compuerta	Real	89.0	False	True	True	True	False		límite de apertura
lim5a_compuerta	Real	84.0	False	True	True	True	False		límite de apertura
lim6a_compuerta	Real	85.0	False	True	True	True	False		límite de apertura
lim7a_compuerta	Real	86.0	False	True	True	True	False		límite de apertura
lim8a_compuerta	Real	87.0	False	True	True	True	False		límite de apertura
lim9a_compuerta	Real	88.0	False	True	True	True	False		límite de apertura
lim10a_compuerta	Real	89.0	False	True	True	True	False		límite de apertura
lim11a_compuerta	Real	90.0	False	True	True	True	False		límite de apertura
lim1b_compuerta	Real	16.0	False	True	True	True	False		límite de cierre
lim2b_compuerta	Real	11.0	False	True	True	True	False		límite de cierre
lim3b_compuerta	Real	6.0	False	True	True	True	False		límite de cierre
lim4b_compuerta	Real	2.0	False	True	True	True	False		límite de cierre
lim5b_compuerta	Real	3.0	False	True	True	True	False		límite de cierre
lim6b_compuerta	Real	5.0	False	True	True	True	False		límite de cierre

Nombre	Tipo de datos	Valor de arranque	Rema-nencia	Accesi-ble desde HMI/OPC UA/Web API	Es-cribible desde HMI/OPC UA/Web API	Visible en HMI Engi-neer-ing	Valor de ajuste	Super-visión	Comentario
lim7b_compuerta	Real	4.0	False	True	True	True	False		límite de cierre
lim8b_compuerta	Real	3.0	False	True	True	True	False		límite de cierre
lim9b_compuerta	Real	2.0	False	True	True	True	False		límite de cierre
lim10b_compuer-ta	Real	1.0	False	True	True	True	False		límite de cierre
lim11b_compuer-ta	Real	0.0	False	True	True	True	False		límite de cierre
rps_comp_NE	Real	0.0	False	True	True	True	False		velocidad calculada del angulo de puerta
rps_comp_NO	Real	0.0	False	True	True	True	False		velocidad calculada del angulo de puerta
rps_comp_SE	Real	0.0	False	True	True	True	False		velocidad calculada del angulo de puerta
rps_comp_SO	Real	0.0	False	True	True	True	False		velocidad calculada del angulo de puerta
factor_Duty_NE	Real	0.0	False	True	True	True	False		
factor_Duty_NO	Real	0.0	False	True	True	True	False		
factor_Duty_SE	Real	0.0	False	True	True	True	False		
factor_Duty_SO	Real	0.0	False	True	True	True	False		
proceso_este	Bool	false	False	True	True	True	False		
proceso_oeste	Bool	false	False	True	True	True	False		
Cycle count	DInt	0	False	True	True	True	False		

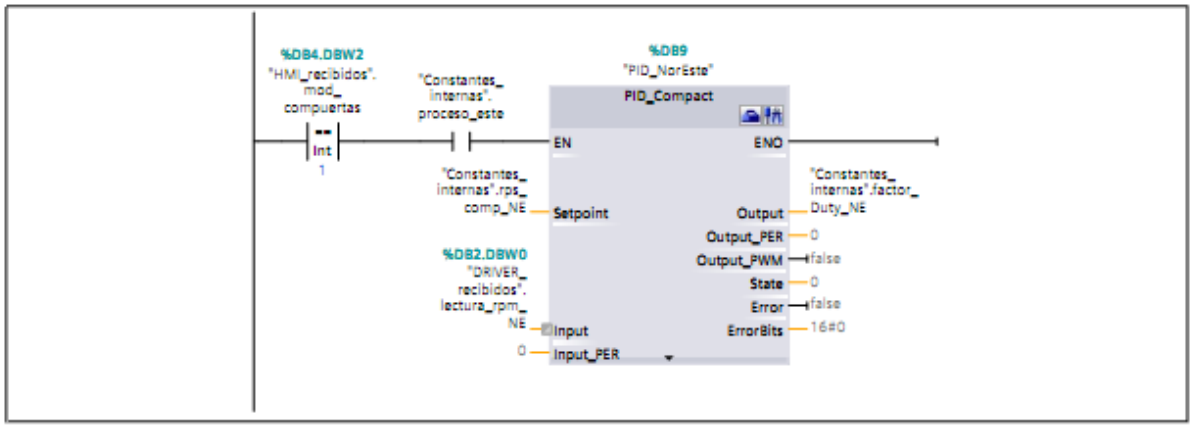
Bloques de programa

Cyclic interrupt [OB30]

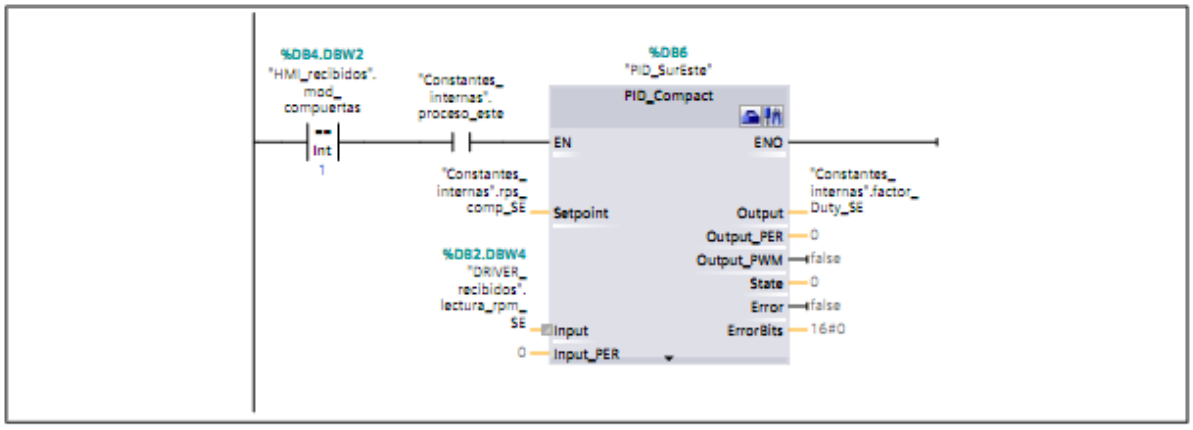
Cyclic interrupt Propiedades					
General					
Nombre	Cyclic interrupt	Número	30	Tipo	OB
Idioma	KOP	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizado	

Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Initial_Call	Bool		Initial call of this OB
Event_Count	Int		Events discarded
Temp			
Constant			

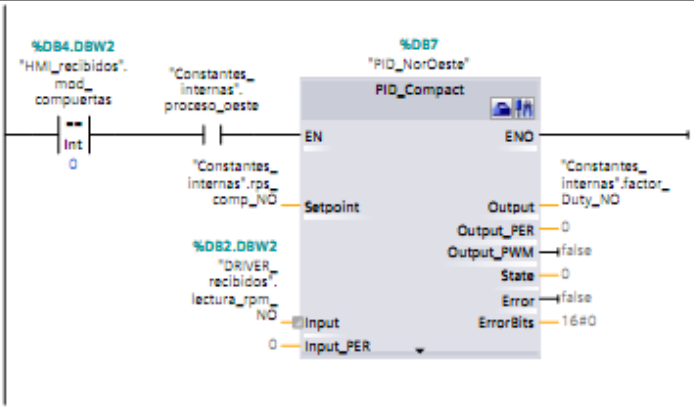
Segmento 1:



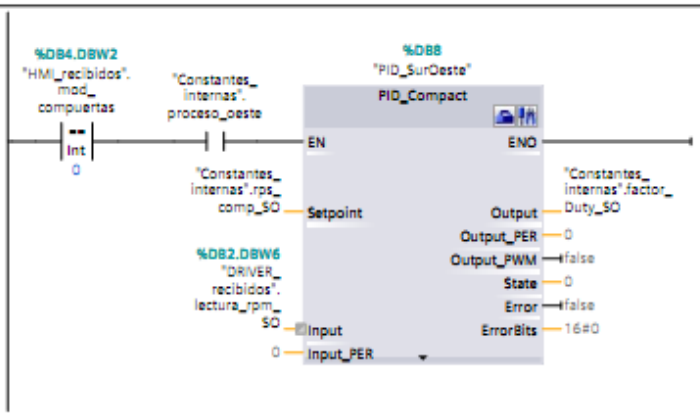
Segmento 2:



Segmento 3:



Segmento 4:



Bloques de programa / Bloques de sistema / Recursos de programa

PID_Compact [FB1130]

PID_Compact Propiedades					
General					
Nombre	PID_Compact	Número	1130	Tipo	FB
Idioma	SCL	Numeración	Automático		
Información					
Título	Compact PID_Controller with self-tuning	Autor	SIMATIC	Comentario	
Familia	COMPPID	Versión	2.3	ID personalizado	PID_Cmpt

Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ Input									
Setpoint	Real	0.0	No remanente	True	True	True	False		controller setpoint input
Input	Real	0.0	No remanente	True	True	True	False		current value from process in REAL format
Input_PER	Int	0	No remanente	True	True	True	False		current value from peripheral input
Disturbance	Real	0.0	No remanente	True	True	True	False		disturbance intrusion
ManualEnable	Bool	false	No remanente	True	True	True	False		activate manual value to overwrite output value
ManualValue	Real	0.0	No remanente	True	True	True	False		manual value
ErrorAck	Bool	false	No remanente	True	True	True	False		reset error message
Reset	Bool	false	No remanente	True	True	True	False		reset the controller
ModeActivate	Bool	false	No remanente	True	True	True	False		enable mode
▼ Output									
ScaledInput	Real	0.0	No remanente	True	False	True	False		current value after scaling
Output	Real	0.0	No remanente	True	False	True	False		output value in REAL format
Output_PER	Int	0	No remanente	True	False	True	False		analog output value
Output_PWM	Bool	false	No remanente	True	False	True	False		pulse width modulated output value
SetpointLimit_H	Bool	false	No remanente	True	False	True	False		setpoint reached upper limit
SetpointLimit_L	Bool	false	No remanente	True	False	True	False		setpoint reached lower limit

Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
InputWarning_H	Bool	false	No remanente	True	False	True	False		current value reached upper warning level
InputWarning_L	Bool	false	No remanente	True	False	True	False		current value reached lower warning level
State	Int	0	No remanente	True	False	True	False		current mode of operation (0 Inactive, 1 SUT, 2 TIR, 3 Automatic, 4 Manual, 5 Substitute output)
Error	Bool	false	No remanente	True	False	True	False		error flag
ErrorBits	DWord	16#0	Remanente	True	False	True	False		error message
▼ InOut									
Mode	Int	4	Remanente	True	True	True	False		mode selection
▼ Static									
InternalDiagnostic	DWord	0	No remanente	False	False	False	False		internal diagnostic and version handling
InternalVersion	DWord	DW#16#02030003	No remanente	True	False	True	False		version of controller
InternalRTVersion	DWord	0	No remanente	False	False	False	False		version of runtime
IntegralResetMode	Int	4	No remanente	True	True	True	True		0 smooth, 1 clear, 2 keep, 3 overwrite initial output, 4 like setpoint change
OverwriteInitialOutputValue	Real	0.0	No remanente	True	True	True	False		initialisation of output value for override control
RunModeByStartup	Bool	true	No remanente	True	True	True	True		activate Mode after CPU restart
LoadBackUp	Bool	false	No remanente	True	True	True	False		restore last parameter set
SetSubstituteOutput	Bool	true	No remanente	True	True	True	True		assignment of output value in State = 5 (FALSE = last valid value, TRUE = SubstituteOutput)
PhysicalUnit	Int	0	No remanente	True	False	True	True		unit of measurement of the process value and setpoint

Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
PhysicalQuantity	Int	0	No remanente	True	False	True	True		physical quantity of the process value and setpoint
ActivateRecoverMode	Bool	true	No remanente	True	True	True	True		FALSE = go to inactive by error, TRUE = activate error treatment
Warning	DWord	16#0	Remanente	True	False	True	False		warning message
WarningInternal	DWord	16#0	Remanente	True	False	True	False		warning message
Progress	Real	0.0	No remanente	True	False	True	False		progress of current phase in percent
CurrentSetpoint	Real	0.0	No remanente	True	False	True	False		current active setpoint value
CancelTuningLevel	Real	10.0	No remanente	True	True	True	True		cancel level for setpoint change during tuning
SubstituteOutput	Real	0.0	No remanente	True	True	True	True		substitute output value in case of error
▼ Config	PID_CompactConfig		No remanente	True	True	True	True		configuration data set
InputPerOn	Bool	true	No remanente	True	True	True	True		activate peripheral input
InvertControl	Bool	false	No remanente	True	True	True	True		invert control direction
InputUpperLimit	Real	120.0	No remanente	True	True	True	True		input (process value) upper limit
InputLowerLimit	Real	0.0	No remanente	True	True	True	True		input (process value) lower limit
InputUpperWarning	Real	3.402822e+38	No remanente	True	True	True	True		input (process value) upper level warning
InputLowerWarning	Real	-3.402822e+38	No remanente	True	True	True	True		input (process value) lower level warning
OutputUpperLimit	Real	100.0	No remanente	True	True	True	True		output value upper limit
OutputLowerLimit	Real	0.0	No remanente	True	True	True	True		output value lower limit
SetpointUpperLimit	Real	3.402822e+38	No remanente	True	True	True	True		setpoint upper limit value
SetpointLowerLimit	Real	-3.402822e+38	No remanente	True	True	True	True		setpoint lower limit value
MinimumOnTime	Real	0.0	No remanente	True	True	True	True		PWM minimum on time
MinimumOffTime	Real	0.0	No remanente	True	True	True	True		PWM minimum off time

Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
▼ InputScaling	PID_Scaling		No remanente	True	True	True	True		input scaling
UpperPointIn	Real	27648.0	No remanente	True	True	True	True		high value (input range of scaling)
LowerPointIn	Real	0.0	No remanente	True	True	True	True		low value (input range of scaling)
UpperPointOut	Real	100.0	No remanente	True	True	True	True		high value (output range of scaling)
LowerPointOut	Real	0.0	No remanente	True	True	True	True		low value (output range of scaling)
▼ CycleTime	PID_CycleTime		No remanente	True	True	True	True		data set for cycle time estimation
StartEstimation	Bool	true	No remanente	True	True	True	False		start automatic estimation of call cycle time
EnEstimation	Bool	true	No remanente	True	True	True	True		enable estimation of call cycle time
EnMonitoring	Bool	true	No remanente	True	True	True	True		enable monitoring of call cycle time
Value	Real	0.1	No remanente	True	True	True	True		call cycle time
▼ CtrlParamsBackUp	PID_CompactControlParams		No remanente	True	True	True	True		saved parameter set
Gain	Real	1.0	No remanente	True	True	True	True		proportional gain
Ti	Real	20.0	No remanente	True	True	True	True		reset time
Td	Real	0.0	No remanente	True	True	True	True		derivative time
TdFiltRatio	Real	0.2	No remanente	True	True	True	True		filter coefficient for derivative part
PWeighting	Real	1.0	No remanente	True	True	True	True		weigthing of proportional part in direct, feedback path
DWeighting	Real	1.0	No remanente	True	True	True	True		weigthing of derivative part in direct, feedback path
Cycle	Real	1.0	No remanente	True	True	True	True		PID Controller cycle time
▼ PIDSelfTune	PID_CompactSelfTune		No remanente	True	True	True	True		data set for self tuning
▼ SUT	PID_Compact_SUT		No remanente	True	True	True	True		data set for start up tuning
CalculateParams	Bool	false	No remanente	True	True	True	False		recalculate control parameters with parameters of startup tuning

Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible desde HMI/OPC UA/Web API	Escribible desde HMI/OPC UA/Web API	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
TuneRule	Int	0	No remanente	True	True	True	True		tuning rule for SUT (0-CHR PID,1-CHR PI)
State	Int	0	No remanente	True	False	True	False		current phase of start up tuning
▼ TIR	PID_Compact_TIR		No remanente	True	True	True	True		data set for tuning in run
RunIn	Bool	false	No remanente	True	True	True	False		activate run in set-point without controling
Calculate-Params	Bool	false	No remanente	True	True	True	False		recalculate control parameters with parameters of tuning in run
TuneRule	Int	0	No remanente	True	True	True	True		tuning rule for TIR (0-2-A PID auto,fast,slow;3-ZN PID;4-ZN PI;5-ZN P)
State	Int	0	No remanente	True	False	True	False		current phase of tuning in run
▼ PIDCtrl	PID_CompactControl		No remanente	True	True	True	True		data for controling part
PIDInit	Bool	false	No remanente	True	True	True	False		initialization of controller
IntegralSum	Real	0.0	No remanente	True	True	True	False		signal of integral part
▼ Retain	PID_CompactRetain		Remanente	True	True	True	True		retain data
▼ CtrlParams	PID_CompactControlParams		Remanente	True	True	True	True		actual parameter set
Gain	Real	1.0	Remanente	True	True	True	True		proportional gain
Ti	Real	20.0	Remanente	True	True	True	True		reset time
Td	Real	0.0	Remanente	True	True	True	True		derivative time
TdFiltRatio	Real	0.2	Remanente	True	True	True	True		filter coefficient for derivative part
PWeighting	Real	1.0	Remanente	True	True	True	True		weigthing of proportional part in direct, feedback path
DWeighting	Real	1.0	Remanente	True	True	True	True		weigthing of derivative part in direct, feedback path
Cycle	Real	1.0	Remanente	True	True	True	True		PID Controller cycle time