UNIVERSIDAD TÉCNICA DEL NORTE

INSTITUTO DE POSTGRADO

MAESTRÍA EN MECATRÓNICA

"CONTROLADOR LÓGICO PROGRAMABLE BAJO SOFTWARE Y HARDWARE LIBRE"

Tesis de Maestría presentada en cumplimiento parcial de los requisitos de la Maestría en Mecatrónica: Mención Procesos Industriales

AUTOR:
Ing. Santiago Manuel Burbano Robles.

TUTOR:
PhD. Carlos Xavier Rosero Chandi

IBARRA – ECUADOR

2022

CERTIFICACIÓN DEL DIRECTOR DE TESIS

Como director del trabajo de investigación con el tema: "CONTROLADOR LÓGICO PROGRAMABLE BAJO SOFTWARE Y HARDWARE LIBRE", trabajo que fue realizado por Burbano Robles Santiago Manuel, previo a la obtención del título de Magister en mecatrónica mención procesos industriales, doy fe de que la obra mencionada reúne los requisitos y méritos suficientes para ser públicamente sustentada en juicio para ser oportunamente aprobada.

Ibarra, 18 de Mayo del 2022

Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI

PhD. Carlos Xavier Rosero Chandi

TUTOR

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL CONTACTO			
CÉDULA DE IDENTIDAD:	1003073358		
APELLIDOS Y NOMBRES:	Burbano Robles Santiago Manuel		
DIRECCIÓN:	Jorge Dávila Meza 5-34 y Luis Felipe Borja		
CORREO ELECTRÓNICO:	smburbanor@utn.edu.ec		
TELÉFONO FIJO:	NO	TELÉFONO MÓVIL	0989589285

DATOS DE LA OBRA	
TÍTULO:	CONTROLADOR LÓGICO PROGRAMABLE BAJO SOFTWARE Y HARDWARE LIBRE
AUTOR:	Santiago Manuel Burbano Robles
FECHA:	2022-05-18
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA	PREGRADO ☐ POSTGRADO ■
TÍTULO POR EL QUE OPTA:	Magíster en Mecatrónica Mención Procesos Industriales
TUTOR:	PhD. Carlos Xavier Rosero Chandi

2. CONSTANCIAS

El autor Santiago Manuel Burbano Robles, manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 7 días del mes de Julio del año 2022.

Ing. Santiago Manuel Burbano Robles

1003073358

REGISTRO BIBLIOGRÁFICO

Guía: POSGRADO – UTN

Fecha: Ibarra, 18 de mayo de 2022

Santiago Manuel Burbano Robles: "CONTROLADOR LÓGICO PROGRAMABLE BAJO SOFTWARE Y HARDWARE LIBRE" / GRADO MAGISTER EN MECATRÓNICA MENCIÓN PROCESOS INDUSTRIALES, Universidad Técnica del Norte, Ibarra.

DIRECTOR: PhD. Rosero Chandi Carlos Xavier

El objetivo general de esta tesis fue: Desarrollar un controlador lógico programable bajo software y hardware libre

Entre los objetivos específicos estaban: Especificar los componentes básicos que comprende un controlador lógico programable.

Diseñar el hardware necesario del PLC para ser usado en aplicaciones industriales.

Establecer el cargador de arranque y el entorno de desarrollo integrado para las aplicaciones finales.

Implementar el sistema para someterlo a pruebas bajo condiciones controladas de funcionamiento

Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI

PhD. Carlos Xavier Rosero Chandi

Firmado electrónicamente por:
SANTIAGO MANUEL BURBANO ROBLES

Ing. Santiago Manuel Burbano Robles

DEDICATORIA

A Gabriela y Emmanuel, la razón de mi vida.

AGRADECIMIENTO

A mi familia por todo el apoyo incondicional brindado durante este periodo académico.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Los controladores lógicos programables usados para la automatización de procesos industriales se presentan en el mercado con una gran variedad de marcas y modelos. Su costo es elevado dependiendo de las distintas funcionalidades que se necesite; de igual manera la licencia de software para realizar las aplicaciones tiene un costo extra y depende del fabricante del controlador. Además, por el extenso catálogo que ofrecen los fabricantes es imposible ser estudiados en la academia, tomando en cuenta que el propósito general de los controladores es el mismo. El presente trabajo propone el diseño y construcción de un controlador lógico programable que con su hardware de interfaz eléctrico y su software de procesamiento estén basados en estándares libres para su uso, distribución y modificación dependiendo de los requerimientos del desarrollador sin presentar ataduras por diseño electrónico o por limitaciones en las licencias de los equipos. Mediante investigación teórica y de campo se detalla las diferentes interfaces eléctricas de entrada y salida usadas en la industria, se diseña esquemáticos eléctricos; se describe una unidad de procesamiento y se fabrica el controlador lógico programable. Se presenta una opción de software que permite programar aplicaciones específicas de control y se realiza pruebas controladas que determinan el correcto funcionamiento y fiabilidad del mismo.

Palabras clave: Automatización, estándares libres, interfaces eléctricas.

ABSTRACT

Programmable logic controllers used for industrial process automation are presented in the market with a wide variety of brands and models. Their cost it's high, depending on the different functionalities that are needed; in the same way the software license to make the applications has an extra cost and depends on the controller manufacturer. In addition, due to the extensive catalog offered by manufacturers, it's impossible to be studied by the academy, taking into consideration that the general purpose of the controllers is the same. The present work proposes the design and construction of a programmable logic controller in which its electrical interface hardware and software processing are based on free standards for their use, distribution and modification; depending on the developer's requirements, without presenting ties with the electronic design or due to limitations in the licenses of the equipment. Through theoretical and field research, the different electrical input and output interfaces used in the industry are detailed, electrical schematics are designed; a processing unit is described and the programmable logic controller is manufactured. A software option is presented which allows to program specific control applications, and controlled tests are performed to determine its correct operation and reliability.

Key Words: Automation, free standards, electrical interfaces.

MARCO LEGAL

El objetivo del presente trabajo comprende la fabricación de componentes tecnológicos para aplicaciones industriales, no existe ninguna normativa gubernamental que norma la fabricación de componentes electrónicos de forma privada.

Como se establece en el libro III del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación; al tratarse de un diseño privado este debe ser registrado en el Servicio Nacional de Derechos Intelectuales (SENADI) en el área de propiedad industrial, organismo que se encarga de registrar y proteger los derechos intelectuales de toda persona natural o jurídica sobre sus invenciones, diseños industriales, circuitos integrados, marcas, signos distintivos, lemas comerciales y otros elementos relacionados con el mercado, la industria y el comercio.

El estado ecuatoriano en el servicio de contratación publica dispone del Reglamento para la adquisición de Software por parte de las entidades contratantes del sector público; este reglamento considera el valor agregado ecuatoriano para la adquisición de software por parte del sector público, tomando como prioridad productos o servicios que contemplen mayoritariamente valor agregado ecuatoriano.

Como objetivos del presente trabajo hará uso de software y hardware libre, los organismos internacionales que avalan este tipo de licenciamiento a nivel de software es la Open Source Iniciative que se encarga de certificar los diferentes tipos de licencias basadas en estándares libres; y para el hardware se encuentra la Open Source Hardware Association, organización encargada de la certificación de hardware abierto.

CAPÍTULO I

EL PROBLEMA

1.1 Planteamiento del problema

La automatización de procesos industriales a partir de la década de los 60 Controladores Industriales Inteligentes tuvo un avance exponencial con el desarrollo de los Controladores Lógicos Programables [PLC], el desarrollo de estos controladores facilitó la implementación de procesos en las cadenas de producción, no obstante por ser una tecnología nueva para ese entonces se veía limitado a las fábricas que más recursos administraban, con el pasar del tiempo varias empresas de diversos países se consolidaron como pioneros en el desarrollo y fabricación de los controladores.

A la actualidad, en el Ecuador desde las medianas empresas han optimizado sus procesos usando controladores lógicos programables con soluciones ofertadas por empresas locales o extranjeras, no obstante, para las pequeñas empresas y microempresas estos procesos no se han podido realizar por razones económicas Diagnostico del nivel de automatizacion en las pequeñas y medianas industrias de la ciudad de Cuenca, ya que este tipo de inversiones son elevadas en relación a la inversión total del negocio.

En el medio local la industrialización de procesos para cadenas productivas en cualquier área se ha estancado por el precio de los equipos necesarios para lograr la automatización, este tipo de empresas desarrollan sus productos con la adquisición de maquinaria individual por cada proceso y no se desarrolla una cadena de producción que optimice los recursos y genere rentabilidad Desarrollo HIstórico de la industria Manufacturera Ecuatorian y su matriz de exportación.. Las soluciones actuales del mercado cuentan con un extenso catálogo de productos, las mismas que son desarrolladas con

productos importados de fabricantes específicos que ponen a disposición hardware y software a medida.

La implementación de una solución basada en controladores lógicos programables Automatización con PLC: Soluciones industriales depende del número de dispositivos a controlar para determinar entradas y salidas necesarias, y a su vez las herramientas de software necesarias para cumplir este objetivo, cada fabricante pone a disposición su hardware y software que debe ser licenciado para su uso y al ser valores elevados es un limitante para su implementación.

De acuerdo con lo anterior, el presente trabajo de maestría propone el desarrollo y construcción de un Controlador Lógico Programable basado en hardware y software libre que pueda ser utilizado en la automatización de ciertos procesos industriales.

1.2 Antecedentes

La publicación realizada en el IECON 2018 Programmable Logic Controller: Open Source Hardware and Software for Massive Training presenta un proyecto en el cual desarrollan un controlador lógico programable de bajo costo usando software y hardware libre como una alternativa para enseñar a futuros profesionales de países en vías de desarrollo a usar controladores lógicos programables, plantean 2 alternativas, la primera usando microcontroladores para entender la lógica de programación y el uso de entradas y salidas, y una segunda opción usando un Raspberry Pi para aumentar las prestaciones y usar la lógica de programación por escalera basándose en el proyecto OpenPLC. Este trabajo se enfoca en facilitar el aprendizaje para los estudiantes sobre el uso de PLC en la industria, pero al estar diseñados electrónicamente para el uso académico no pueden ser usados en la industria.

La Federación Internacional de Control Automático IFAC International Fereration of Automatic en su publicación anual del año 2019 publica un trabajo de varios autores The Most Important Open Technologies for Design of Cost Efficient Automation Systems que presentan una revisión sobre varios componentes de software y hardware libre para la construcción de controladores lógicos programables, tomando como premisa que los PLC son el mecanismo más utilizado en la industria para la automatización de procesos, pero dependen de software y hardware licenciado que muchas veces es demasiado costoso, y para empresas con poco presupuesto no es factible su implementación. Se analizan varios componentes actualmente disponibles en el mundo colaborativo del software y hardware libre, se contempla la estructura de un sistema de control, estándares y protocolos de comunicación, estándares de programación basados en la norma IEC 61131-3 y el hardware abierto disponible. Se concluye que todas estas herramientas disponibles más el diseño específico a medida del desarrollador se puede construir un PLC en su totalidad que puede ser utilizado en la industria contando con certificaciones y estándares internacionales.

En la conferencia anual del año 2011 de ingeniería industrial avalada por IEEE se presentó un artículo An embedded PLC development for teaching in mechatronics education en el que se analiza que el mayor problema en el aprendizaje de controladores lógicos programables es la cantidad de marcas y modelos disponibles y que al ser costosos no se puede adquirir todos para su estudio y en el caso de la educación no se podrían albergar en un solo laboratorio. Es por esto que proponen la construcción de un PLC basado en microcontroladores ARM y el software LabVIEW que contemple el concepto básico de un controlador lógico programable, su estructura interna y sus interfaces eléctricas y electrónicas para el acople de señales otorgadas por sensores comerciales y la activación de actuadores.

Con este estudio se concluye que la fabricación de PLC de forma embebida es viable y a la vez se demuestra que se puede enseñar de una manera sencilla.

A la par de estas publicaciones en la actualidad hay varias empresas que ya han comenzado a desarrollar y comercializar controladores lógicos programables de calidad industrial siguiendo normativas industriales y certificaciones internacionales, 2 ejemplos de ellos son la empresa española IndustrialShields Soluciones industriales basadas en Open Source Hardware y la alemana Revolution Pi The Only Real industrial Raspberry Pi, que cuentan con un catálogo variado para soluciones de todo tipo relacionado a la industria. Estas empresas aportan una visión más clara de que un proyecto de esta magnitud si es viable y puede ser aplicado en cualquier parte del mundo.

1.3 Objetivos de la investigación

1.3.1 Objetivo general

Desarrollar un controlador lógico programable usando software y hardware libre para ser usado en aplicaciones industriales.

1.3.2 Objetivos específicos

- Especificar los componentes básicos que comprende un controlador lógico programable.
- Diseñar el hardware necesario del PLC para ser usado en aplicaciones industriales.
- Establecer el cargador de arranque y el entorno de desarrollo integrado para las aplicaciones finales.
- Implementar el sistema para someterlo a pruebas bajo condiciones controladas de funcionamiento.

1.4 Justificación

El desarrollo tecnológico globalizado permite alcanzar conocimientos que solo estaban al alcance de universidades o centros de investigación en lugares puntuales en el mundo, con este conocimiento adquirido ahora se desarrolla soluciones a partir de todo el conocimiento colaborativo y que la academia local lo promueve mediante programas de maestría.

Desarrollar un controlador lógico programable de bajo costo para aplicaciones específicas que solucionen problemáticas locales, incentiva a que la empresa local sea más productiva y pueda generar más plazas de empleo para desarrolladores de tecnología y técnicos de mantenimiento.

El desarrollo de tecnologías libres en cuanto a software Proyecto GNU y hardware Hardware Libre para una sociedad libre en los últimos años ha puesto a los desarrolladores tecnológicos independientes a la par de empresas pioneras en el mundo, alcanzando niveles de desarrollo colaborativo para que la única limitante sea el ingenio humano. Al desarrollar productos tecnológicos libres se contribuye como sociedad a masificar su uso y brindar soluciones independientes de licencias propietarias o cerradas para un hardware específico.

Al término del presente desarrollo de trabajo de maestría se obtendrá una visión clara de la posibilidad de producir controladores lógicos programables para el mercado local, ofertando soluciones a medida con garantía y servicio técnico local. Uno de los principios del hardware y software libre Proyecto GNU es la libertad de usar estos recursos con cualquier propósito, es por eso que este trabajo de investigación servirá para que futuros desarrolladores del área industrial focalicen sus investigaciones a partir de este trabajo y mejoren los resultados para el beneficio de la industria local y nacional.

CAPÍTULO II

MARCO TEÓRICO

2.1 Controladores lógicos programables

Un controlador lógico programable, más conocido como PLC por su abreviación en el idioma inglés puede definirse como un dispositivo electrónico capaz de almacenar y ejecutar instrucciones, operaciones lógicas y cálculos para el control de módulos de entrada y salida tanto analógicos como digitales sobre diferentes tipos de máquinas o procesosControladores Industriales Inteligentes.

Un PLC a diferencia de un dispositivo electrónico tradicional es su capacidad de operar sus interfaces de entrada y salida bajo condiciones especiales, en las que se incluye, rangos y tipos de voltaje, rangos de temperatura, inmunidad al ruido eléctrico, resistencia a la vibración y al impacto.

Las característica más importantes es su operatividad en tiempo real, haciendo posible ejecutar acciones en el menor tiempo posible dependiendo de las condiciones de entrada, además de su flexibilidad para reprogramar sus tareas y ser adaptado para diferentes tipos de procesos.

2.2 Sistemas embebidos basados en hardware libre

El término hardware libre nació con el inicio de la informática por la década de los setenta Hardware Libre, en el que los aficionados a los ordenadores comenzaron a construir sus propias versiones con piezas de diferentes fabricantes, los nuevos esquemáticos eran compartidos y mejorados dependiendo del diseñador.

A la actualidad con el acceso más simplificado de componentes electrónicos, existen miles de proyectos bajo la denominación de hardware libre, cada proyecto ha presentado avances y novedades a distintos campos relacionados con la informática, electrónica, domótica, etc. Todos los proyectos denominados de hardware libre han tenido algún tipo de campaña de financiamiento para llegar al término del proyecto y luego salir a producción tomando en cuenta que sus esquemáticos siempre han sido liberados para que la comunidad pueda fabricarlos o rediseñarlos.

A continuación, se presentan varios ejemplos de proyectos de sistemas embebidos que en los últimos años han tenido mayor relevancia y sus características favorecen al diseño central de procesamiento para un PLC.

2.2.1 Arduino

El proyecto Arduino nace en la academia italiana por el año 2005, un grupo de profesores del Instituto de Diseño Interactivo de Ivrea diseñó un prototipo para facilitar la enseñanza en las aulas y que fuera de bajo costo Arduino Home Page, con el cierre del instituto el proyecto fue liberado para que la comunidad participe de la evolución, mejoras y sugerencias del proyecto.

El proyecto con el pasar del tiempo ha ganado gran aceptación en todo el mundo y su aporte a la educación a sido fundamental, su aporte a logrado desarrollar miles de proyectos y su iniciativa a generado que nuevos proyectos basados en hardware libre salgan a la luz.

El proyecto Arduino tiene varias placas de desarrollo, las mismas que han sido diseñadas para varios propósitos y funcionalidades, entre las más destacadas se puede nombrar los modelos UNO, Nano, MEGA, y sus modelos con comunicación inalámbrica enfocados al IoT[1], todas las placas tienen como componente principal un microcontrolador AVR marca Atmel y sus versiones dependen del modelo de la placa.

El proyecto Arduino aparte de liberar los esquemáticos eléctricos de sus placas, libero el entorno de programación para el desarrollo de aplicaciones que está basado en el lenguaje C++, este lenguaje derivado del lenguaje C que es muy robusto y popular a nivel mundial, está basado en una programación estructurada y también se basa en objetos: es por eso que se lo categoriza como lenguaje de programación híbrido.

Al tratarse de un proyecto libre se han desarrollado diferentes tipos de placas basadas en el proyecto Arduino con modificaciones tanto para producción, educación y pasatiempos caseros, y los entornos de programación también se han desarrollado de forma gráfica pudiendo programar por bloques.

Dependiendo de las placas, hay varios rangos de voltaje para trabajar, por eso necesario especificar la placa a ser usada para saber en que niveles de voltaje van a operar las entradas y salidas además de la alimentación de la placa.

2.2.2 Raspberry Pi

La fundación Rasperry Pi nace en el 2012 en el Reino Unido con el objetivo de enseñar informática y electrónica a colegios y universidades Raspberry Pi Foundation, tomando como objetivo desarrollar una computadora de tamaño reducido y de bajo costo. Su éxito esta basado en su alto rendimiento y bajo consumo de energía.

---

1 IoT Siglas en inglés de Internet de las cosas, que define la interconexión de elementos físicos sobre internet

A diferencia del proyecto Arduino un Raspberry Pi dispone de memoria RAM y puertos de entrada y salida de audio y video que con los periféricos necesarios podría convertirse en un computador de escritorio; y su mejora presenta un socket de puertos digitales y analógicos para periféricos electrónicos.

Los niveles de voltaje manejados por Raspberry Pi están en el orden de los 3.3V, para manejar dispositivos externos o adquirir señales es necesario realizar interfaces de acople ya que estos pines si sufren variaciones fuera de su rango nominal podrían dañar toda la placa.

Raspberry Pi cuenta por su arquitectura con varios sistemas operativos sobre los cuales puede operar, el proyecto más estable se basa en una distribución GNU/Linux denominada Raspberry Pi OS basado en Debian[2], bajo este sistema operativo se han desarrollado varias aplicaciones que van desde centros de entretenimiento hasta aplicaciones de visión artificial y minería de criptomonedas.

En el campo de la electrónica se han desarrollado una variedad de proyectos; gracias a su reducido tamaño y bajo consumo al poder funcionar solo con baterías. Al tratarse de un computador completo, para el manejo de entradas y salidas de propósito general se puede crear programas que se ejecuten automáticamente en lenguaje Python, C, Java o Scratch; siendo Python el más utilizado por su integración con el análisis de datos, visión por computadora o inteligencia artificial entre otros.

Con la evolución de los componentes electrónicos los Raspberry Pi han ido evolucionando, disponiendo ahora de versiones con más memoria RAM y diferentes tipos de conectores y opciones de configuración, además que se han desarrollado versiones miniatura o especificas para diferentes proyectos, la fundación Raspberry Pi también pensando en el

2 Distribución del sistema operativo desarrollado en el proyecto GNU/Linux

avance tecnológico ha desarrollado su línea industrial en la que presenta placas certificadas y potenciadas para ser usadas bajo condiciones industriales.

2.2.3 DevKits ESPRESSIF

Espressif es una empresa multinacional Espressif Company de fabricación de semiconductores que se ha especializado en desarrollar chips para aplicaciones inalámbricas, de bajo consumo y basadas en IoT y AIoT[3]; son los responsables de la creación de los famosos chips ESP en todas sus versiones los mismos que en la actualidad son los más usados para aplicaciones IoT, esta empresa esta comprometida con ofrecer soluciones robustas y energéticamente eficientes.

A su vez el libre acceso a su código tanto de tecnología como de sus soluciones han permitido que desarrolladores de todo el mundo creen sus propios dispositivos inteligentes basados en soluciones de Espressif y que estos se puedan interconectar.

Aparte de ofrecer los chips para crear soluciones personalizadas, Espressif dispone de kits de desarrollo para realizar prototipos e interactuar fácilmente con el chip elegido, estas placas de desarrollo cuentan con toda la circuitería extra para conectar sensores y actuadores, además de interfaces de comunicación para cargar los programas y su depuración.

Para su programación después de disponer de su cargador de arranque, los lenguajes más conocidos para ser programado son Python y C++ que usa el entorno de programación del proyecto Arduino.

Todos los chips de Espressif funcionan con niveles de voltaje de 3.3V, que deben ser tomados en cuenta para los diseños electrónicos en la toma de lecturas de sensores o en la ejecución de estados para sus pines de salida.

---

3 Siglas para la tecnología basada en internet de las cosas con inteligencia artificial.

2.3 Lenguajes de programación en la automatización industrial

Los lenguajes de programación son el nexo entre el sistema operativo de un controlador y el usuario que se encarga de ingresar las instrucciones, consisten en una secuencia de instrucciones que el CPU de un PLC las interpreta para analizar entradas o energizar salidas las cuales controlaran cualquier tipo de proceso o maquinaria Lenguajes de programación para PLC.

Para la programación de un controlador se pueden diferenciar dos tipos:

2.3.1 Lenguajes de bajo nivel

Esta clasificación se caracteriza por representarse con textos y se subdivide en:

2.3.1.1 Lista de instrucciones

Este lenguaje es el más complejo de interpretar por el usuario ya que usa instrucciones más directas hacia el procesador, en las cuales se indica directamente los registros que van a operar y los espacios de memoria en donde se ejecutara un proceso, en esta clasificación podemos tomar como ejemplo el lenguaje ensamblador.

2.3.1.2 Texto estructurado

Es un lenguaje con una sintaxis más entendible con el usuario, en la que se puede sistematizar expresiones matemáticas para las entradas analógicas y digitales, se puede estructurar bucles, funciones y condicionales, y su interprete puede distinguir entra mayúsculas y minúsculas; ejemplos para esta clasificación pueden ser Pascal, C++, Python, Java, etc.

2.3.2 Lenguajes de alto nivel

Estos tipos de lenguaje se caracterizan por ser muy amigables con el usuario, en el que de forma gráfica se puede programar las rutinas que realizará el controlador; se puede clasificar a la vez en:

2.3.2.1 Diagrama escalera

Es el lenguaje más utilizado en el campo de los PLC, ya que se asemeja a una escalera en donde se agregar funcionalidades como peldaños respetando la secuencia de izquierda a derecha y de arriba hacia abajo; se puede describir como la activación de entradas en la parte izquierda desde la línea de voltaje hacia la línea de la derecha o masa que representa las salidas.

2.3.2.2 Diagrama de bloques

Este lenguaje consiste en la utilización de bloques de procesos, cada bloque ya dispone de funcionalidades definidas a las cuales se les entrega variables de entrada que una vez después de cumplir su función entregaran una variable con la salida del proceso.

2.3.2.3 Diagrama de funciones secuenciales

Está compuesto por diagramas y gráficos que se unifican para crear secuencias de control. Este lenguaje proviene del estándar francés GRAFCET que también utiliza etapas, transiciones y acciones para su funcionamiento.

2.3.3 Interfaces para sensores y actuadores

Disponer de información sobre el estado de un proceso es vital para iniciar, continuar o detener las acciones secuenciales que conforman una automatización; este tipo de señales no siempre son eléctricas, por lo que se utilizan transductores para convertir a señales eléctricas; las señales provienen de magnitudes físicas que son recolectadas mediante sensores, se realizan un tratamiento de la señal para convertirle a algún parámetro eléctrico (tensión, corriente, frecuencia) y posteriormente es amplificada para ser enviada al controlador principal.

Cualquier tipo de fenómeno físico puede ser cuantificado; en la automatización industrial dependiendo del proceso y sus condiciones de operación se debe determinar las características estáticas y dinámicas del sensor, para que estén dentro de los parámetros de operatividad.

Hay una gran variedad de empresas que se han encargado de desarrollar sensores para la industria, por temas de seguridad y comercialización, aunque no es una regla general, la mayoría de sensores en el mercado trabajan con niveles de voltaje de 10 voltios para señales analógicas y 24 voltios para señales digitales, también acotar que existen sensores que

entregan su medición en una relación de corriente siendo los más conocidos los valores de 4 a 20 miliamperios.

Por otra parte, para que un proceso industrial cierre el ciclo de procesamiento un PLC debe ser capaz de accionar o controlar elementos eléctricos, neumáticos o hidráulicos, y para ello es necesario el manejo de actuadores, los mismos que se encargan de manejar distintos tipos de voltajes, corrientes o electro mecanismos para activar dichos elementos. Al igual que las entradas, por temas comerciales y de seguridad estos elementos trabajan en el orden de los 24 voltios no siempre siendo una regla general.

Como se analizó en la sección de sistemas embebidos, los procesadores o microcontroladores funcionan con niveles de voltaje de 5 o 3.3 voltios, a partir de esto es necesario el diseño de circuitería extra para el acondicionamiento de señales de las entradas y salidas para que puedan manejar estos niveles, además que por temas de seguridad de nuestros controlador es sumamente necesario desacoplar las señales de salida ya que se puede manejar corrientes alternas en la activación de elementos y el retorno de las mismas puede provocar daños en los componentes que funcionan con corriente continua.

2.4 Software para el diseño de sistemas embebidos

En consecuencia al planteamiento de la presente tesis que se basa en desarrollar un controlador lógico programable basado en hardware y software libre, se hace necesario también el detallar las herramientas de software libre utilizadas para el diseño tanto de los esquemáticos electrónicos, diseño de placas de circuito impreso, diseño asistido por computador CAD, software de programación.

2.4.1 KiCad

El software KiCad es un proyecto de código abierto KiCad EDA el cual permite la automatización de diseño electrónico, permite desarrollar proyectos desde la realización de esquemáticos y posteriormente el diseño del circuito impreso PCB, esta herramienta genera todos los archivos necesarios para la fabricación y producción en serie de placas ya que cumple con todas las normas y parámetros utilizados por los fabricantes.

El objetivo de KiCad es ofrecer un software de diseño electrónico multiplataforma[4] para que cualquier entusiasta, estudiante o profesional de la electrónica desarrolle sus proyectos sin limitación alguna.

2.4.2 FlatCAM

Los archivos gerber proporcionados por el software KiCad son utilizados por las empresas de fabricación industrial, para realizar prototipos de laboratorio se puede disponer de máquinas ruteadoras de control numérico CNC, las cuales permiten realizar placas de circuito impreso con precisión milimétrica, para esto es necesario que a partir de los archivos gerber y con las especificaciones de nuestra máquina se generen archivos G-code que son proporcionados por el software FlatCam.

Este software de código abierto multiplataforma esta escrito en lenguaje python y permite generar archivos G-code tanto de las pistas como de las perforaciones de las placas de circuito impreso FlatCAM: Free and Open-source PCB CAM, además de configurar velocidad de funcionamiento, cambios de herramientas y cortes finales.

4 Puede ser instalado en sistemas operativos Windows, Linux y Mac

2.4.3 FreeCad

FreeCAD es un software de código abierto que permite diseñar cualquier objeto de la vida real, a partir de planos en 2 dimensiones es capaz de proyectar objetos en 3 dimensiones que pueden ser modificados o variar sus parámetros iniciales.

Es un software multiplataforma (Windows, Mac y Linux) FreeCAD Your own 3D parametric modeler, altamente personalizable y extensible lee y escribe en muchos formatos de archivo abiertos como STEP, IGES, STL, SVG, DXF, OBJ, IFC, DAE y muchos otros, haciendo posible integrarlo sin problemas en cualquier proyecto.

Tiene un sin numero de aplicaciones, y varios campos de aplicación, en el se pueden desarrollar planos arquitectónicos, diseños de piezas mecánicas en 3D, simulación de elementos finitos, y cualquier proyecto que venga a nuestra mente, esta herramienta es sumamente potente ya que puede ser utilizada por profesionales o simplemente por entusiastas creativos.

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Descripción del área de estudio

El presente estudio de investigación se realizó en las oficinas de la empresa Inprise Soluciones Tecnológicas Empresariales, que se encuentra domiciliada en la ciudad de Ibarra en la calle José Miguel Leóro y Sánchez y Cifuentes. Inprise está dedicada al desarrollo de soluciones integrales para empresas enfocándose en software y hardware a la medida. La empresa lleva 10 años en el mercado teniendo como sus principales clientes al sector del transporte público y la empresa privada. En la figura 1 se muestra la ubicación de la empresa donde se realizó el estudio.

Figura 1:

Ubicación de Inprise

3.2 Enfoque y tipo de investigación

La propuesta metodológica y tecnológica avanzada tuvo como objetivo solucionar un problema en particular por lo que fue necesario elegir un enfoque de investigación ingenieril, el mismo que está basado en todos los conocimientos, habilidades y destrezas de ingeniería que se deben llevar a cabo para solucionar dicho problema puntual.

Para el desarrollo del enfoque metodológico se propuso las siguientes etapas: identificación y formulación del problema, análisis del problema, búsqueda de soluciones posibles, evaluación de diferentes alternativas y selección de la solución óptima y la especificación de la solución escogida.

Parte del desarrollo fue documental, ya que se tomó en cuenta una base de datos técnica de diferentes sensores y actuadores usados en la industria, de tal manera que el dispositivo a desarrollar trabaje con los mismos.

Se realizó un trabajo explicativo donde se detalló todos los componentes necesarios para desarrollar un controlador lógico programable como un producto final, y también fue un trabajo aplicado, ya que se obtuvo un dispositivo tecnológico para ser usado en la industria.

3.3 Procedimientos

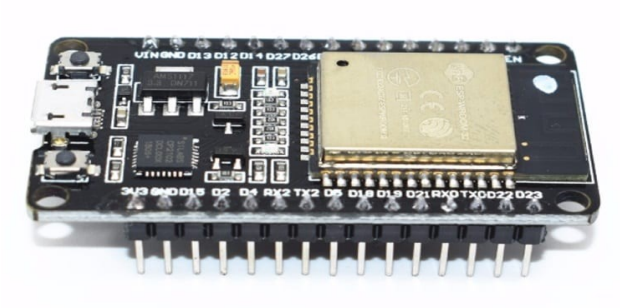
3.3.1 Diseño electrónico

3.3.1.1 Sistema embebido principal

Después de analizar varias tipos de sistemas embebidos expuestos en la sección 2, y analizando sus características para la construcción del controlador lógico programable se eligió el kit de desarrollo de propósito general de la empresa Espressif en su modelo ESP32-S2-WROOM, en su versión comercial ESP32 DEVKIT V1 como se muestra en la figura 2.

Figura 2:

ESP32 DEVKIT V1, versión de desarrollo



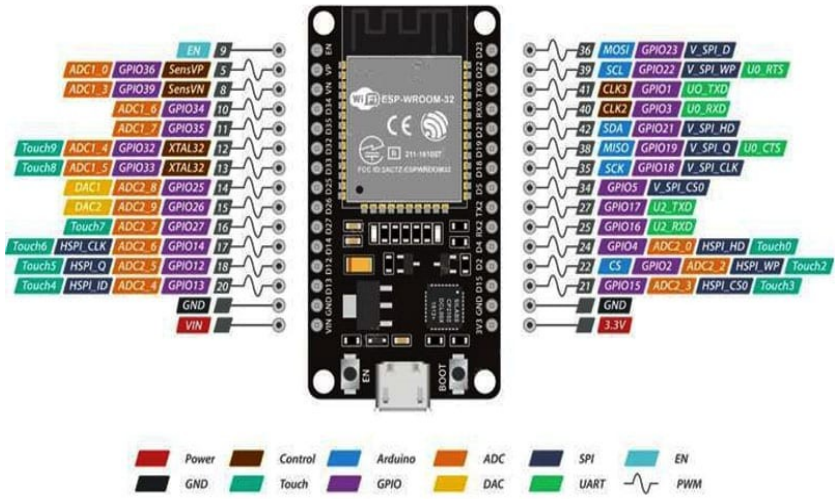
Nota. Adaptado de Solectro, tienda en linea,2022, (https://solectroshop.com/8165-large_default/esp32-wroom-32-devkit-v1-placa-con-wifi-y-bluetooth.jpg)

3.3.1.1.1 Diagrama de pines

Esta versión de desarrollo cuenta con 30 pines que se detallan a continuación, estos pueden ser usados como alimentación, entradas y salidas analógicas, entradas y salidas digitales y puertos de comunicación específicos.

Figura 3:

Diagrama de pines ESP32 DEVKIT V1



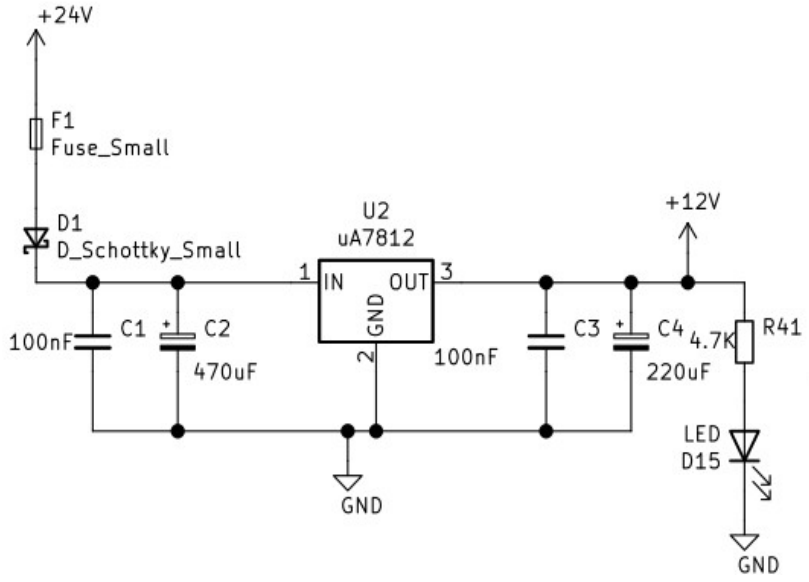
Nota. Tomado de UNIT Electronics, 2021, (https://uelectronics.com/wp-content/uploads/2018/06/AR0453-ESP32-V2.jpg

3.3.1.2 Fuentes de alimentación

La entrada principal de alimentación es de 24V que esta protegida mediante un fusible de 2A y un diodo para evitar retornos de corriente.

Para la alimentación de los componentes analógicos de comparación se utiliza un regulador de 12 voltios LM7812 que proporciona 1 amperio de corriente que se encuentra configurado como lo especifica su hoja de datos con capacitores cerámicos y electrolíticos, además de un led indicador de funcionamiento.

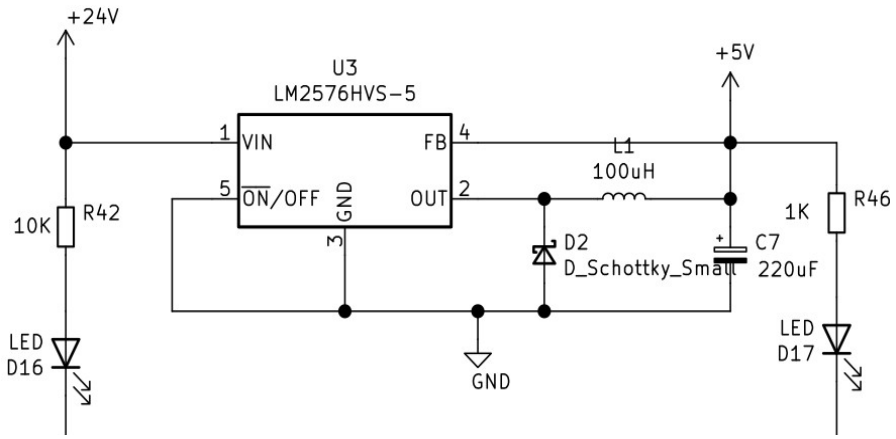
Figura 4:



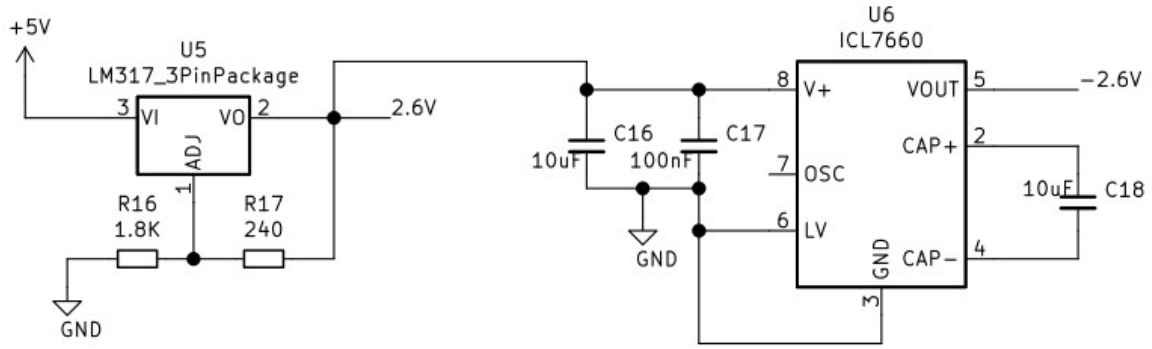
Para la alimentación de 5 voltios se utilizó el circuito integrado LM2576 que provee una salida de hasta 3 amperios, su configuración esta tomada de su hoja de especificaciones

en la que utiliza una bobina de 100 uH y un condensador electrolítico de 220uF, se integró un led monitor para verificar su funcionamiento.

Figura 5:



Para la interfaz de entrada analógica de 0-20mA es necesario retro alimentar los amplificadores operacionales con voltajes de 2.6 voltios y -2.6 voltios que se logra con el uso del circuito integrado LM317 para reducir el voltaje de 5 voltios a 2.6 y el circuito ICL7660 para convertir negativamente el voltaje de 2.6 voltios.



3.3.1.3 Interfaces de entrada

3.3.1.3.1 Entradas digitales

Tomando en cuenta los siguientes parámetros y valores otorgados por las hojas técnicas de los componentes se procede a diseñar las entradas digitales.

Tabla 1:

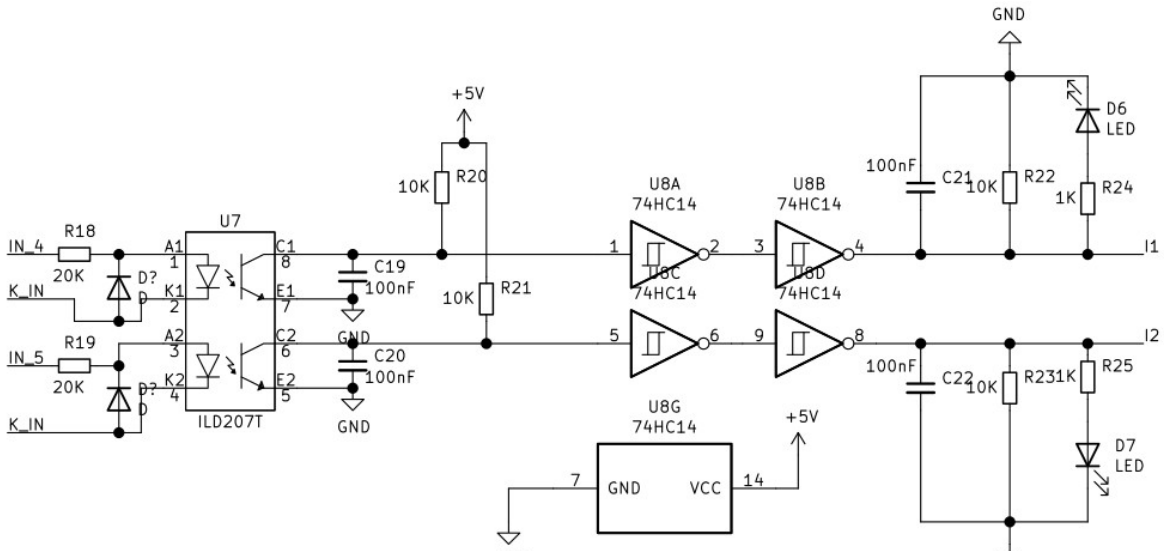
Especificaciones técnicas entradas digitales

DATOS TÉCNICOS	PLC v1
Número de entradas	2
Tensión nominal	24 V DC a 10 mA
Tensión continua admisible	30V DC
Grupo de aislamiento	1
Señal 1 lógica (mín.)	18 V DC a 5 mA
Señal 0 lógica (max.)	5 V DC a 1 mA

Mediante el circuito integrado optoaclopador ILD207 disponemos de 2 canales independientes y desacoplados para la lectura de valores de 24 voltios, la resistencia de 20 kilo ohmios y el diodo en paralelo a la entrada de optoacoplador sirve para limitar el voltaje de funcionamiento y una posible inversión de polaridad de entrada.

Una vez polarizado el optoacoplador la señal de salida es rectificada mediante el circuito integrado 74HC14 que consiste en una compuerta Schmitt Trigger que entrega directamente al pin del ESP32 una señal TTL en el pin digital. Para disponer de un monitoreo visible de activación de las entradas se acoplo un led en paralelo al pin digital del ESP32. En la figura 7 se muestra el esquemático de 2 entradas digitales.

Figura 7:



3.3.1.3.2 Entrada analógica 0-10V

De igual manera para las entradas analógicas se detalla en la tabla 2 los parámetros básicos con los que se realizó el diseño.

Tabla 2:

Especificaciones técnicas de entradas analógicas 0-10 voltios

DATOS TÉCNICOS	PLC V1
Número de entradas	2
Rango total	0-10 V DC
Resolución	12 bits
Impedancia	≥100 KΩ
Corriente de suministro operativa mínima	30 uA
Tensión soportada máxima	32 V DC

Para realizar lecturas de 0 a 10 voltios se configura la entrada mediante un amplificador operacional LM358 que está alimentado a 12 voltios en su pin positivo y 0 voltios en su pin negativo, configurado como un amplificador no inversor con una ganancia de 0.3 que ha su salida es controlado por un diodo Zener a 3.3 voltios para garantizar los niveles aceptados por el pin analógico del ESP32.

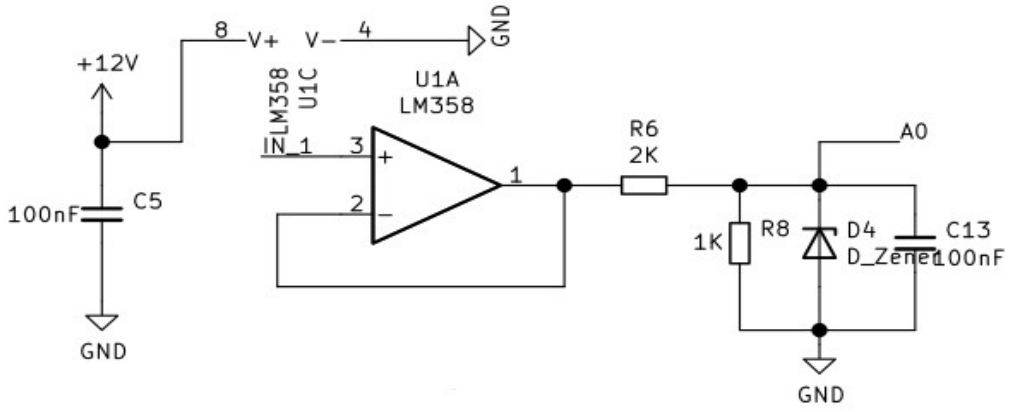


Figura 8:

Esquemático entrada analógica 0-10 voltios

3.3.1.3.3 Entrada analógica 4-20mA

A continuación se presenta en tabla 3 los parámetros para el diseño de la entrada analógica.

Tabla 3:

Especificaciones técnicas de entrada analógica 4-20mA

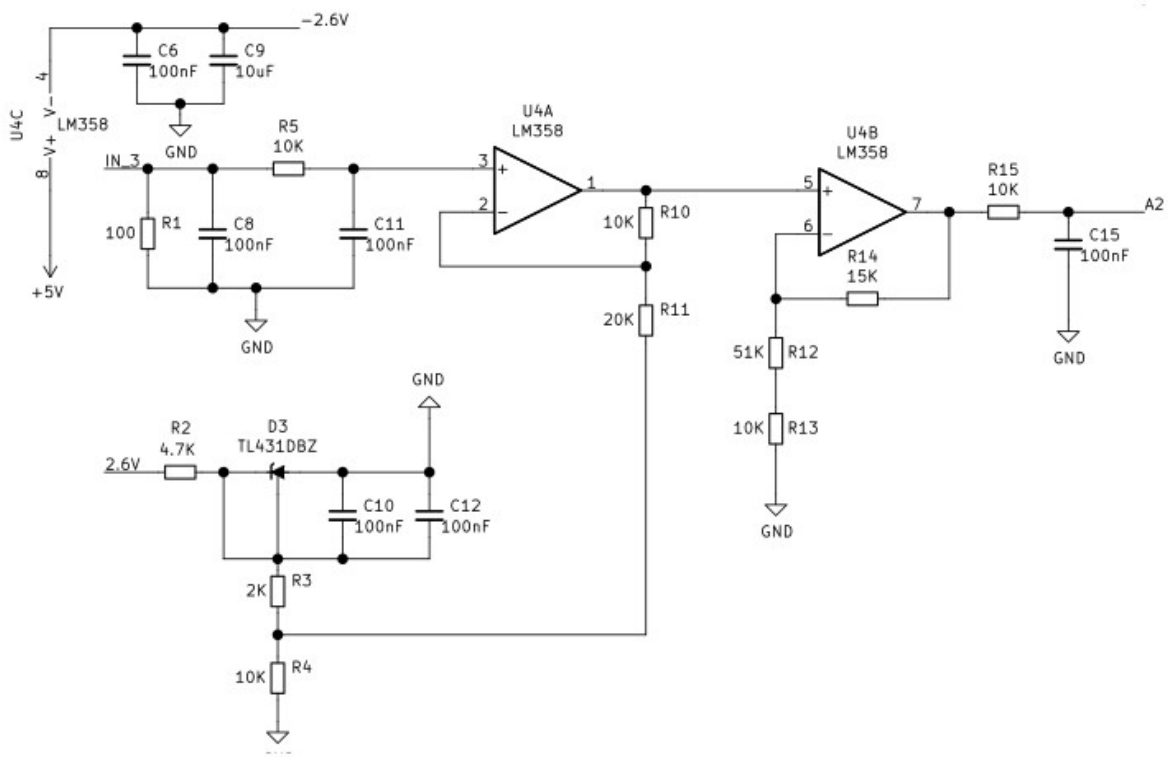
DATOS TÉCNICOS	PLC V1
2 Número de entradas	3 1
4 Rango total	5 0-4 mA
6 Resolución	7 12 bits
8 Impedancia	9 ≥100 KΩ
10 Corriente de suministro operativa mínima	11 30 uA
12 Tensión soportada máxima	13 32 V DC

Para la entrada de corriente analógica primero se realiza una conversión de corriente a voltaje mediante una resistencia en paralelo y se realiza una lectura de voltaje con el amplificador operacional LM358, configurado como en un seguidor de tensión no invertido que permite llevar los niveles de voltaje de 0.4 voltios

cuando la corriente es 4 miliamperios y 2 voltios para 20 miliamperios. Posteriormente la señal vuelve a ser amplificada por un LM358 para que el rango de voltaje este entre 0 y 3.3 voltios, esta señal es conectada directamente al pin analógico del ESP32 como se muestra en la figura 9.

Figura 9:

Esquemático de entrada analógica 4-20mA

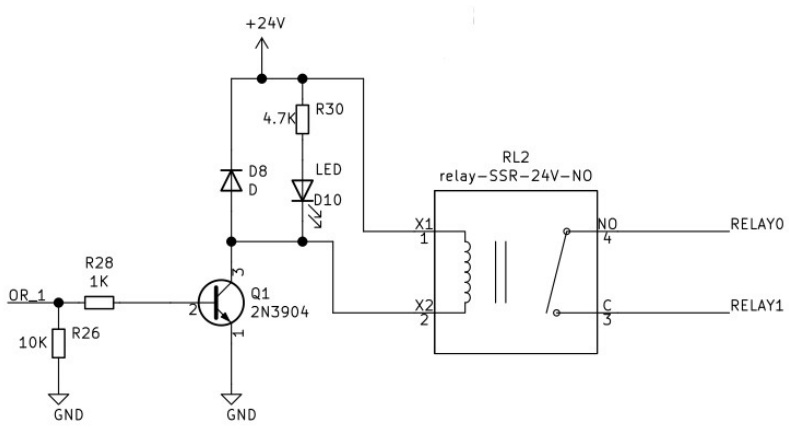


4.1.4 Interfaces de salida

4.1.4.1 Salida DC a relé

El PLC esta diseñado con 2 salidas digitales para manejar cargas de corriente continua de hasta 10 amperios que cumplen su propósito mediante relés de 24 voltios de estado sólido que son polarizados mediante un transistor 2N3904, configurado como inversor de señal para activar su bobina y un led adicional como luz indicadora de activación de la salida, la figura 10 muestra el esquemático realizado.

Figura 10:

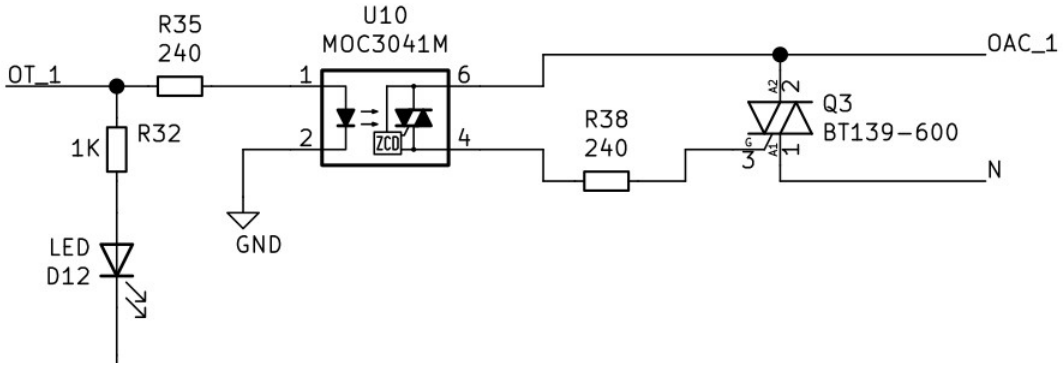


Esquemático de salida

13.1 4.1.4.2 Salidas de corriente alterna mediante TRIAC

Para manejar cargas de corriente alterna directamente desde el PLC se ha diseñado 3 salidas opto aisladas mediante el circuito integrado MOC3041 que ofrece un desacople óptico de la etapa de corriente alterna con la corriente continua suministrada desde el ESP32, para el manejo de cargas de corriente alterna se uso el TRIAC BTA16-600 este ofrece alto rendimiento en aplicaciones de corriente alterna de onda completa e inmunidad al ruido con cargas hasta de 16 amperios, además se integró un led para verificar el estado de la salida.

Figura 11:



14 **4.1.5 Interfaces de comunicación**

4.1.5.1 Comunicación RS-232

El ESP32 dispone de pines dedicados para comunicación serial RS232 con niveles de voltaje TTL, y solo se dispuso de terminales de conexión directa para acceder a dicha comunicación

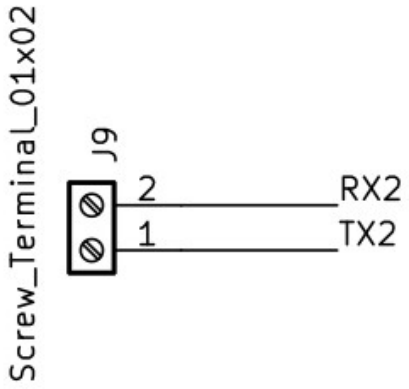


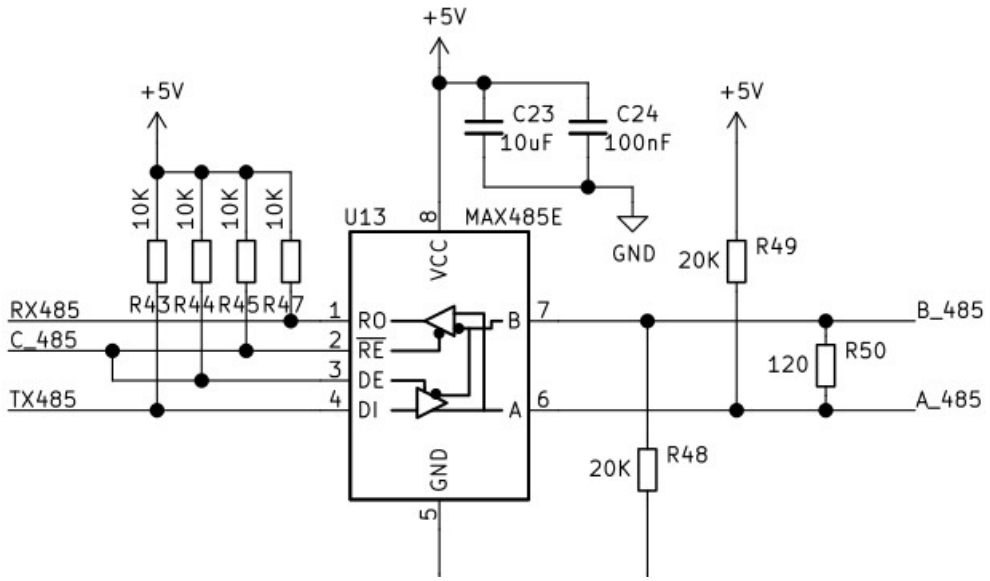
Figura 12:

Pines de comunicación RS232

4.1.5.2 Comunicación RS-485

Para la conexión de algunos periféricos industriales se ha contemplado disponer de una comunicación RS485, que se logra usando los pines seriales RS232 del ESP32 y la interfaz de comunicación otorgada por el circuito integrado MAX485 que mediante la configuración definida en su hoja de datos otorga una comunicación duplex controlada con un pin digital desde el ESP32 para enviar o recibir información directa con los periféricos RS485.

Figura 13:

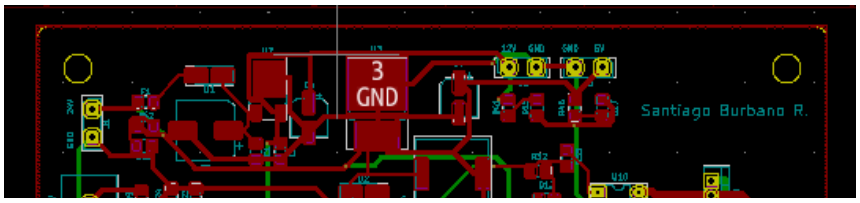


4.2 Diseño de placa de circuito impreso

Una vez realizado todo el diseño de componentes, estos están acoplados en un solo esquemático junto con borneras de conexión que facilitan la interconexión de periféricos. El controlador lógico programable cuenta con varios componentes por cada interfaz de entrada y salida y se consideró el diseño de la placa de circuito impreso en su mayoría con componentes SMD que reducen considerablemente el tamaño de la placa.

Además como es concebida como un producto de alto rendimiento con calidad industrial la placa se envió a un centro profesional de fabricado de PCB para cumpla con los estándares de calidad y además permita diseñar con más de una capa de cobre.

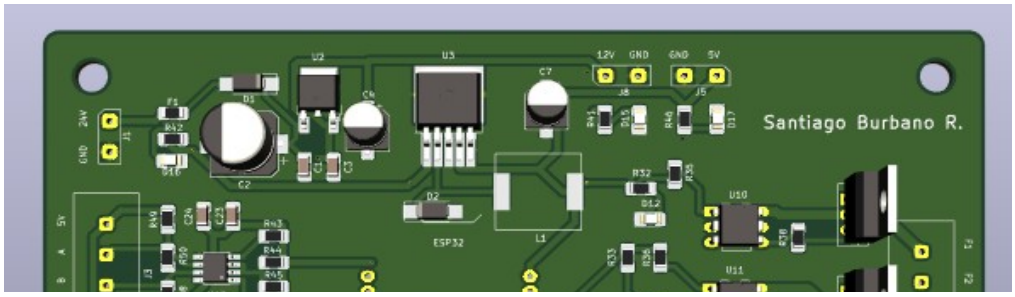
Figura 14:



El software KiCad a partir del esquemático electrónico permite diseñar el PCB siguiendo reglas de diseño predeterminadas y también definir criterios propios para el espacio entre vías, número de capas, grosor de pistas, corte y serigrafía como un producto final de ensamblaje.

Además que esta herramienta permite obtener una vista 3D de la placa, para constatar el acabado final y los detalles de presentación de la placa, una vez verificado todo se procede a exportar el diseño en archivos de trabajo GERBER que son los archivos que la empresa encargada de la fabricación de PCB necesita.

Figura 15:



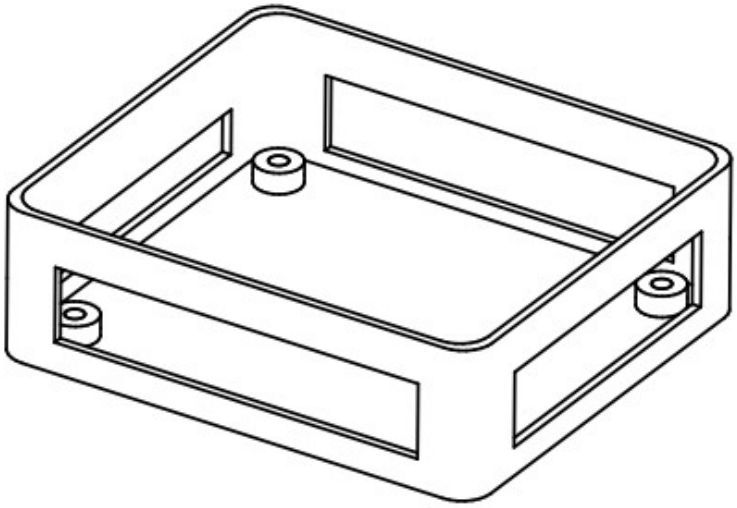
4.3 Diseño de carcasa y componentes externos

Para el diseño que va a servir de carcasa y protección para la placa del circuito impreso se usa el software FreeCAD que permite dibujar planos en 2 dimensiones y luego convertirlos en elementos de 3 dimensiones que pueden ser fabricados en una impresora 3D. Esta carcasa permite montar el controlador sobre un riel DIN utilizado en tableros electrónicos de automatización, y a su vez orificios de conexión para los periféricos tanto de entrada como de salida.

El diseño de la PCB esta diseñado con 4 orificios en sus esquinas, es por eso que la carcasa dispone de puntos de apoyo para sujetar la PCB con tornillos, que permiten una fácil desinstalación para soporte y mantenimiento del controlador. En la figura 16 se presenta la vista isométrica de la carcasa y en anexos el plano mecánico completo.

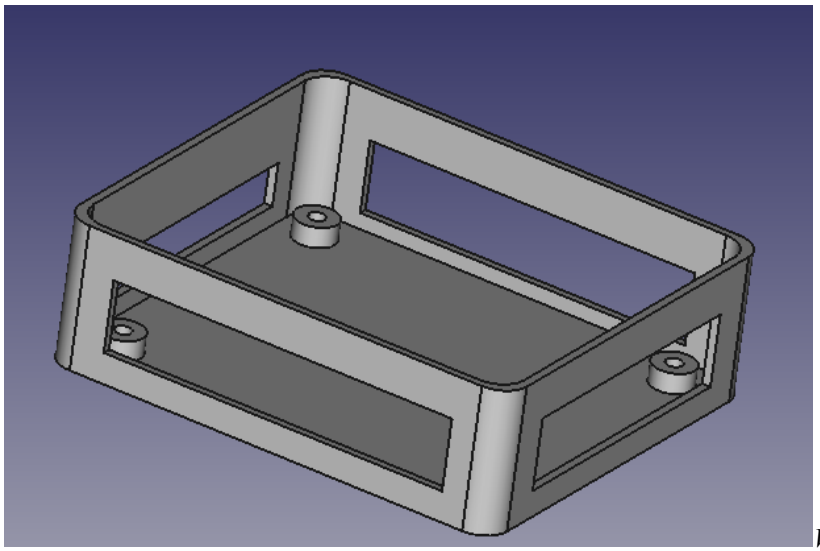
Figura 16:

Vista isométrica de la carcasa de protección



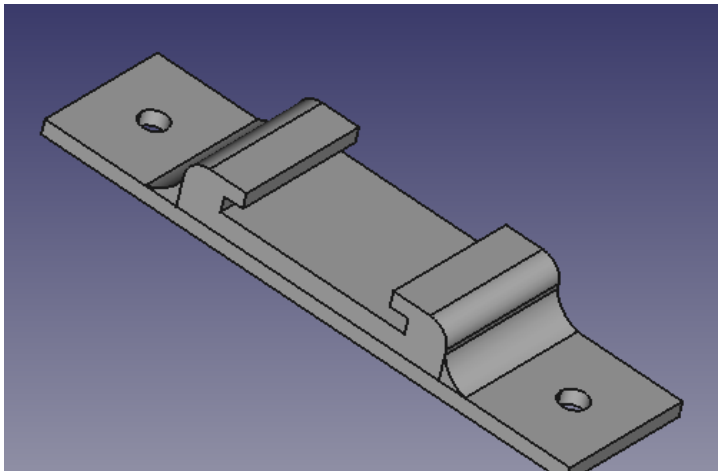
La carcasa esta construida con polímero termoplástico PLA que soporta temperaturas hasta 180 grados centígrados y garantiza protección a los componentes que están en su interior, se diseño las paredes de la carcasa con un espesor de 3mm y su base de 5mm en donde ésta se sujeta mediante tornillos la riel DIN. En las figuras 17 y 18 se muestra la vista en 3D de las carcasa de protección y el soporte para la riel DIN.

Figura 17:



Vista 3D de la carcasa de

Figura 18:



CAPITULO IV

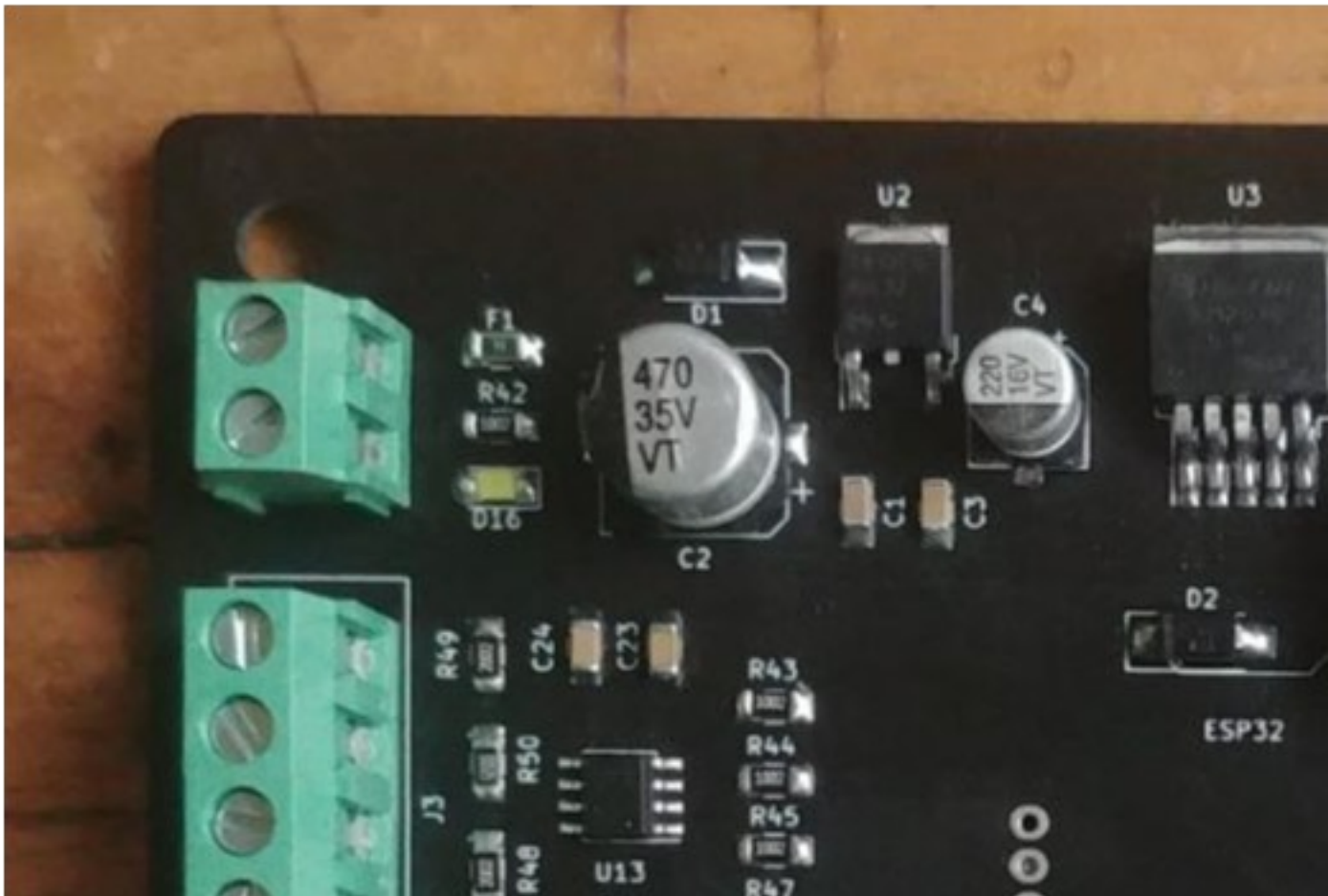
ENSAMBLAJE, PUESTA EN MARCHA Y PRUEBAS

4.1 Ensamblaje

Una vez lista la placa de circuito impreso enviado por la empresa fabricante, y con la lista de materiales se procede a soldar todos los componentes, tanto los de montaje superficial SMD como los de inserción, este proceso se realizó manualmente utilizando herramientas locales y componentes de soldadura que se encuentran en el mercado local.

Al disponer de una placa fabricada en máquinas profesionales, su aislamiento es el indicado y los espacios de soldadura son los exactos para cada componente, lo que facilita su montaje, una vez terminado el proceso de soldadura se procede a limpiar la placa con alcohol isopropílico que desvanece cualquier resto de pasta de soldadura para evitar contacto entre elementos.

Figura 19:



e

q

i

s

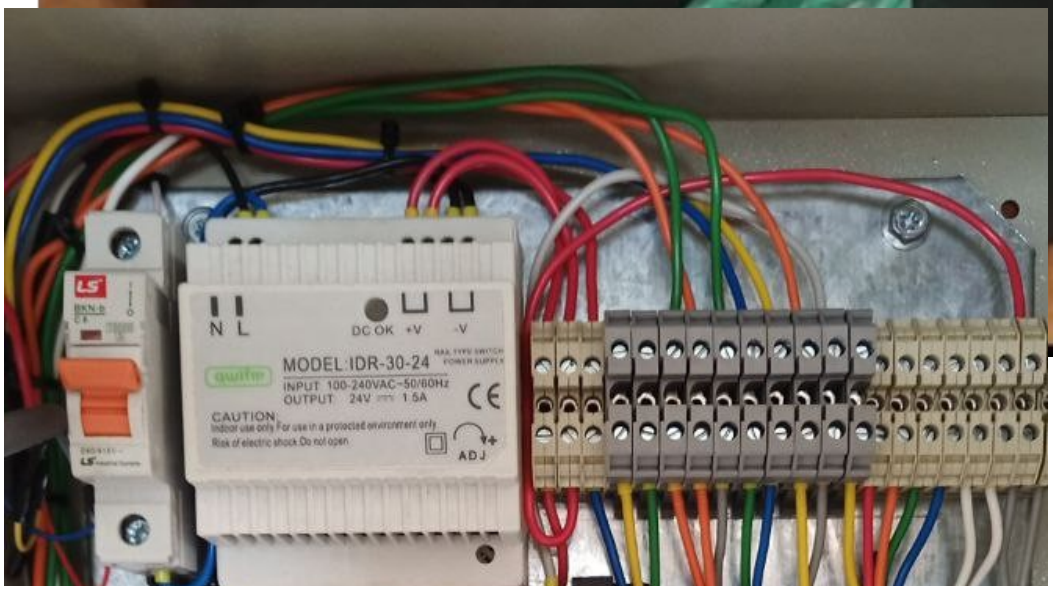
a

a

s

c

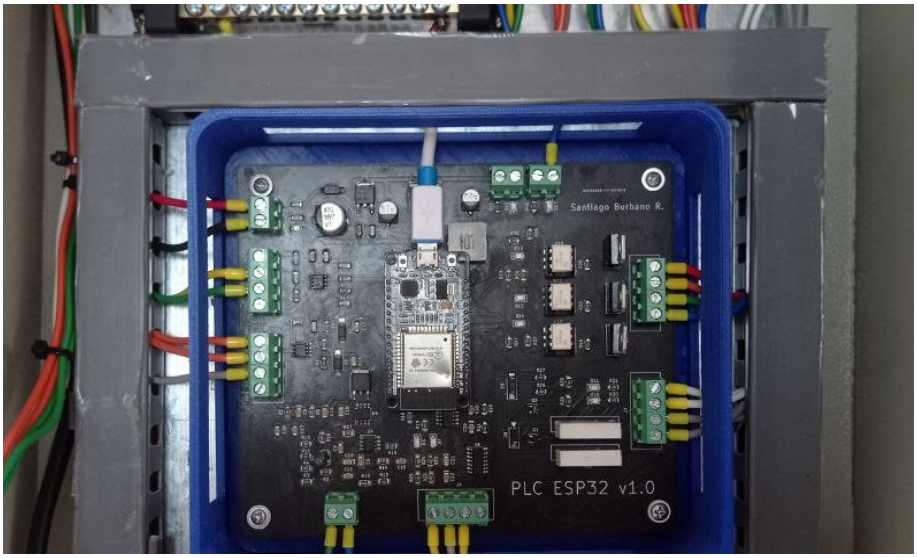
Figura 20:



Como se describió en el capítulo anterior se diseñó una caja plástica que sirva de base para el controlador lógico programable, esta base dispone de 2 soportes para ser insertados en la riel DIN de la caja metálica, por efectos de presentación de este trabajo de titulación no se contempló una tapa plástica para el controlador ya que es necesario que se observen los componentes de la tarjeta desarrollada.

En la parte inferior de la caja esta colocado el controlador lógico programable en su caja plástica y se ha dispuesto ubicar canaletas a su alrededor para realizar el cableado hacia las borneras y que estas queden cubiertas y selladas.

Figura 21:



Para las conexiones entre el controlador y los periféricos se dispuso de cable número 18 AWG que presenta flexibilidad y soporta voltajes hasta 600 voltios y cargas hasta 6 amperios suficientes para el proyecto, además que se dispone en el mercado de varios colores que facilitan la identificación de las conexiones dentro del tablero.

En la parte exterior de la caja metálica correspondiente a la tapa se dispuso a modo de pruebas una pantalla HMI y 2 botones físicos y un selector para realizar ejemplos de funcionalidad, los mismo que están sujetos por sus componentes físicos propios de los componentes.

Figura 22:

4.2 Operatividad

Una vez realizado todo el montaje físico de el controlador lógico programable tanto de sus componentes de hardware como cableado eléctrico, se procede a encender el sistema y a tomar las primeras mediciones de voltaje para comprobar que que los niveles de voltaje sean los correctos de acuerdo al diseño realizado, estos voltajes comprenden los 24 voltios, 12 voltios y 5 voltios como fuentes principales de alimentación de componentes y una fuente auxiliar de 2.6 voltios.

Después de comprobar visualmente y térmicamente que no se dispone de ningún elemento que presente fallas primarias se procede a conectar el controlador lógico programable al computador mediante un cable USB.

4.2.1. Cargador de arranque

Como se explicó en capítulos anteriores el chip principal del sistema embebido es el circuito integrado ESP32 que su fabricante la empresa Espressif desarrolla un cargado de arranque que realiza la configuración mínima de módulos internos, inicializa las opciones de cifrado y seguridad si están configurados, selecciona la aplicación de la partición para iniciar y carga la imagen en la RAM. Cabe mencionar además que este gestor de arranque se encuentra en la dirección de memoria 0x1000 de la Flash.

El código del gestor de arranque se encuentra disponible en el repositorio de GitHub de Espressif y va en su versión 4.4 que puede ser descargado y modificado si se desea controlar de una forma particular algún módulo del ESP32.

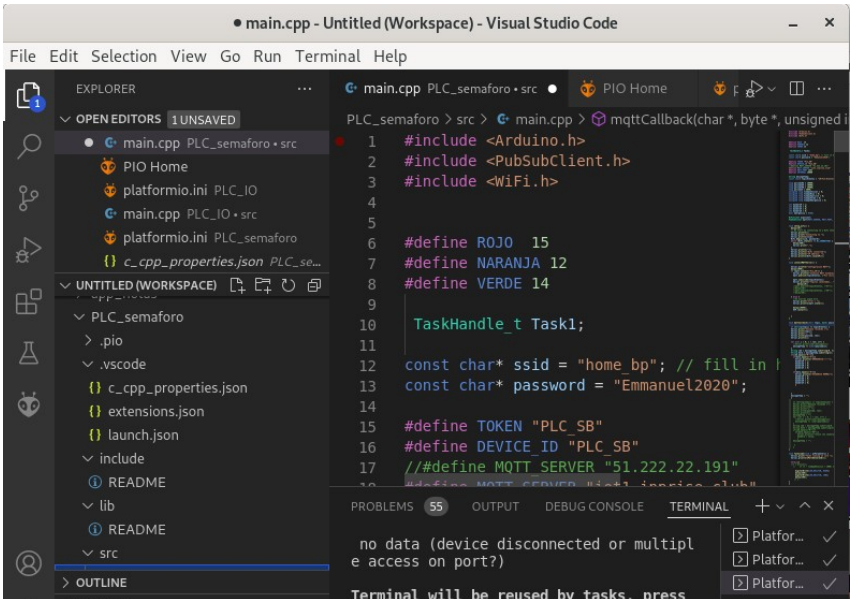
4.2.2 Entorno de desarrollo de aplicaciones

Para la programación de aplicaciones el sistema embebido va a ser programado mediante lenguaje de bajo nivel de texto estructurado, al momento existen dos lenguajes de programación disponibles para desarrollar aplicaciones en el ESP32, el primero es mediante lenguaje C++ y el segundo mediante Python que en su versión para sistemas embebidos ha sido denominada microPython.

Una vez definido que lenguaje se va utilizar se puede acceder a este entorno por varias interfaces de desarrollo, la más popular en el lenguaje C++ es el entorno de desarrollo del proyecto Arduino, que mediante librerías otorgadas por Espressif añade las distintas tarjetas electrónicas basadas en ESP32 y su programación y carga de programas se lo realiza como una tarjeta Arduino tradicional. Para el lenguaje de programación en microPython se puede desarrollar desde una terminal o utilizar un editor de código como por ejemplo uPyCraft.

Por defecto las placas de desarrollo de Espressif vienen precargadas con el firmware para ser programadas mediante lenguaje C++, es por eso que el entorno de desarrollo elegido es el software Visual Studio Code, que ofrece mejores prestaciones de edición, formato de código y acceso a librerías, y de igual manera que en el entorno Arduino se puede cargar directamente los programas a la placa. La interfaz es muy amigable y puede ser instalado en cualquier sistema operativo, cabe mencionar que este entorno de desarrollo también puede servir si se desea utilizar el ESP32 con microPython.

Figura 23:



4.3 Pruebas de funcionamiento

Una vez instalado y configurado todos los componentes necesarios para desarrollar aplicaciones se procede a cargar el programa básico de ejemplo que consiste en hacer parpadear el led integrado a la placa ESP32, que funciona con normalidad, con esta prueba se determina que todos los componentes de software a nivel entorno de desarrollo están listos y podemos proceder a realizar ejemplos prácticos de funcionamiento.

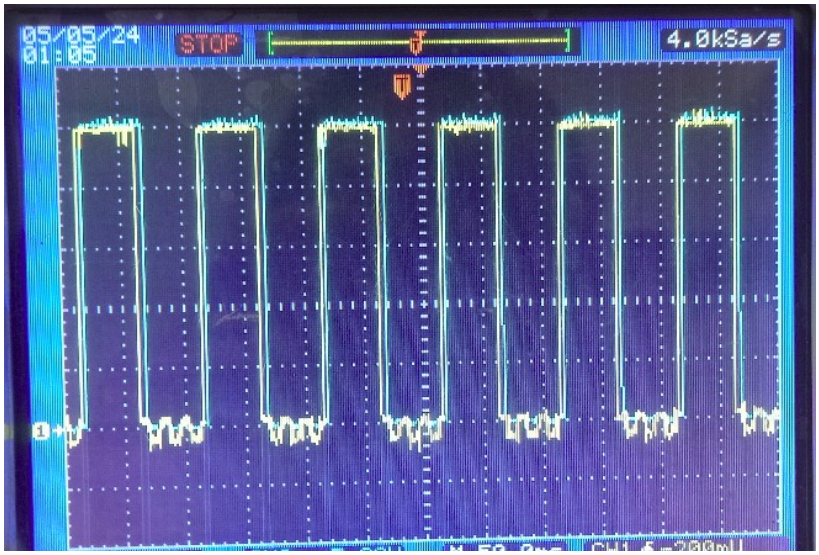
4.3.1 Evaluación de tiempo de respuesta para una entrada y salida digital

La prueba controlada a ser analizada como un caso de laboratorio consisten en determinar el tiempo de respuesta de un evento de entrada hacia una salida, tomando en cuenta que la placa esta diseña con 2 tipos de salidas digitales, una salida con relé de estado sólido y otra salida con TRIAC, las pruebas en el laboratorio se realizaron con una señal cuadrada con un ciclo de trabajo al 50% .

Para realizar esta prueba se programo el ESP32 con una rutina en la que se adquiere la señal en un entrada digital y el valor recibido es entregado a el pin de salida tanto a la salida de relé como a la de transistor.

La primera prueba se realizó a una frecuencia de 1Hz en la que mediante un osciloscopio se analizó la onda de salida con respecto a la de entrada y después se fue aumentando la frecuencia para determinar la variación en la salida.

Figura 24:



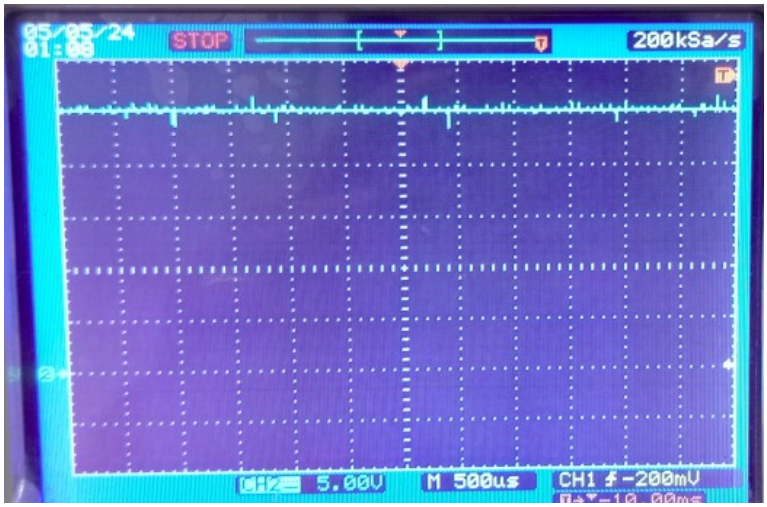
Al realizar las pruebas respectivas y con la ayuda de el osciloscopio se determinó que para la señal de salida en el transistor el tiempo de respuesta es 5 milisegundos, y para el transistor la salida esta en el orden los 10 picosengundos.

Figura 25:

Figura 26:

El sistema se ve afectado cuando sobrepasa en nivel de respuesta a la salida quedando en un estado lógico alto, este valor esta determinado por el nivel de adquisición de la señal a la entrada, este valor al sobrepasar los 50 Hz no puede ser procesado y se mantiene en un solo valor lógico.

Figura 27:



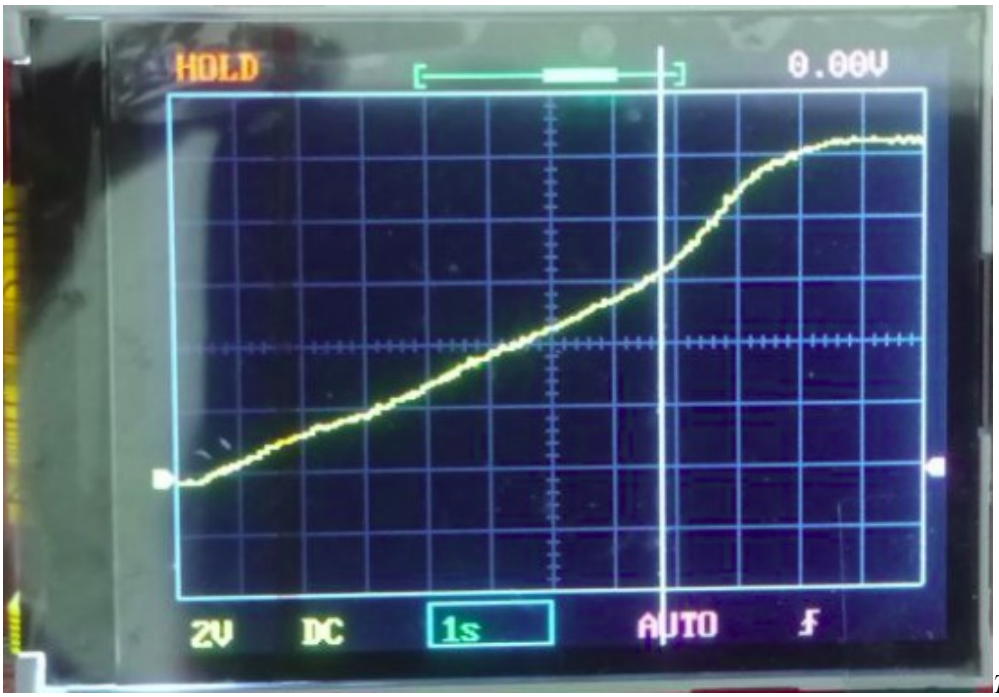
4.3.2 Evaluación lectura de señales analógicas

El ESP32 como unidad principal de procesamiento del PLC, cuenta con un módulo interno de conversión analógica a digital de 12 bits que se pueden representar 4096 valores en el rango de 0 a 3.3 voltios.

Tomando en cuenta el diseño original del controlador, para realizar pruebas de funcionamiento, primero la señal analógica debe pasar por el circuito de acoplamiento que reduce los niveles de voltaje a los niveles permitidos en el ESP32, esta circuitería esta diseñada con amplificadores operacionales LM358 que pueden trabajar hasta con señales de máximo 1 Mhz por lo cual su tiempo no representa un retardo para la lectura con el ESP32.

Idealmente se considera una lectura lineal de datos pero el fabricante y la prueba realizada muestran que para valores menores a 0.1 voltios la lectura siempre responde con el valor 0 y para lectura mayores a 3.1 voltios la respuesta es el valor digital de 4095. Este fenómeno se puede observar en la figura 28.

Figura 28:



Toma de

Para las lecturas analógicas, mediante código de programación estos valores se pueden validar de tal forma que no representen alteraciones en las mediciones y se entregue datos correctos sobre el mínimo y máximo valor capturado. Al contar el ESP32 con un reloj de procesamiento a 160Mhz se garantiza que se puede realizar alrededor de 1000 lectura en un periodo de 1 segundo tomando en cuenta los ciclos de reloj para las demás instrucciones.

4.3.3 Ejemplos básicos usando el PLC

A continuación se presentan 2 ejemplos básicos de funcionamiento del controlador lógico programable, con el fin de presentar una idea de las múltiples aplicaciones que se pueden dar en la industria tomando en cuenta que el único limitante es nuestro ingenio.

4.3.3.1 Semáforo vial y ejemplo IoT

Para este ejemplo de prueba se ha dispuesto la utilización de un semáforo real de instalación para control del transito en la ciudad, que funciona con corriente alterna de 110 voltios o 220 voltios, que va a ser controlado directamente desde el controlador lógico programable y pueda reportar su funcionamiento hacia internet mediante el protocolo MQTT como un dispositivo IoT.

El semáforo se conecta directamente a los pines de salida del TRIAC los mismos que son activados como una salida digital para cada color del semáforo y a su vez la información del estado esta siendo enviada a un servidor MQTT para su representación gráfica o para realizar acciones particulares sobre el semáforo.

Figura 29:

Imagen real del semáforo conectado

El sistema IoT se configuró mediante un servidor MQTT nativo bajo el proyecto de código abierto EMQX, además de node-red que es una herramienta de programación que permite interconectar hardware y software mediante diagramas de flujo simples. A partir de node-red se puede acceder a interfaces prediseñadas que muestran el funcionamiento del hardware que envía información MQTT y también enviar información al controlador.

Figura 30:

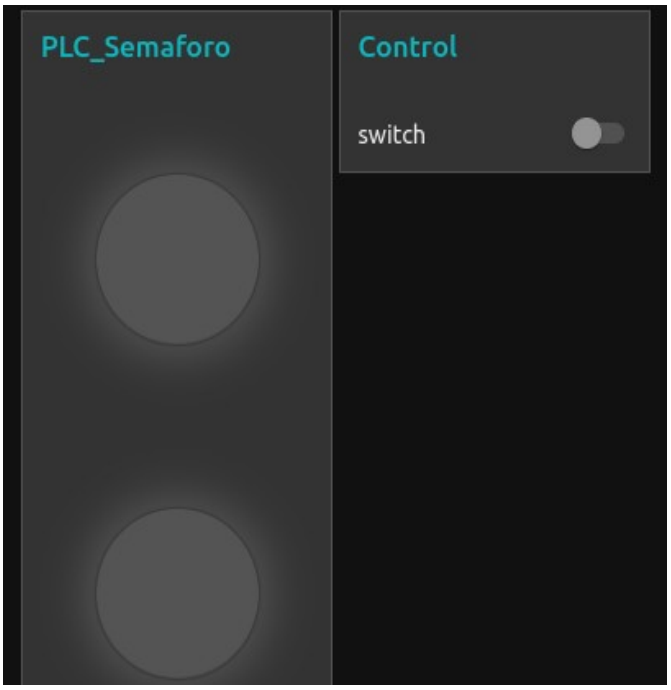
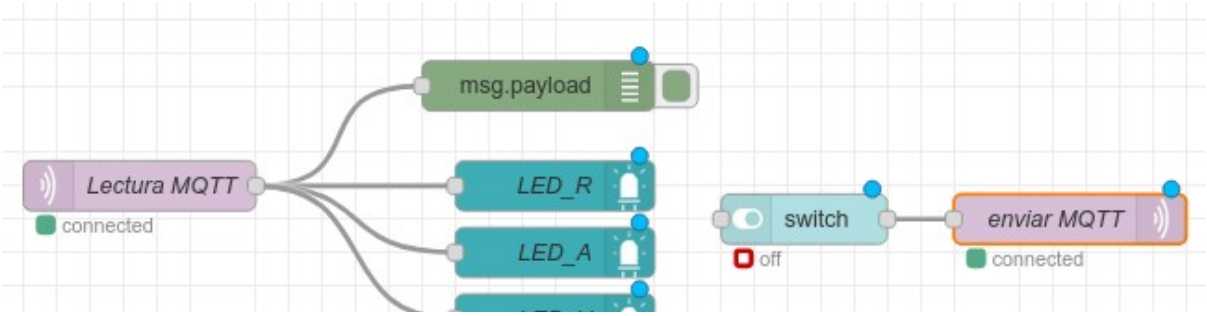


Figura 31:

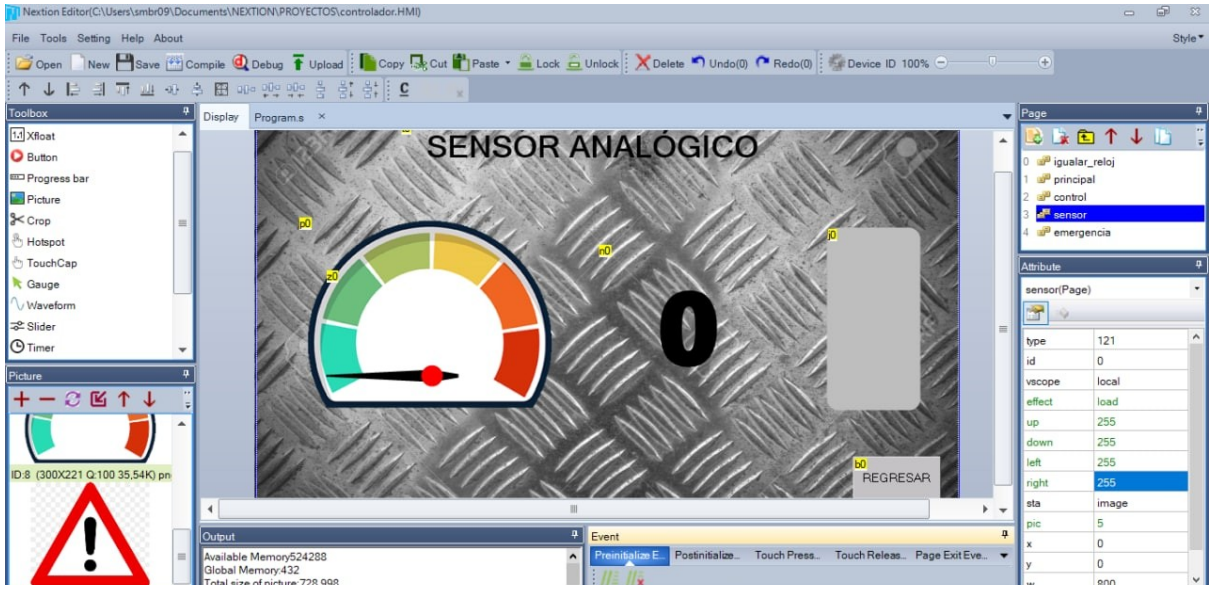


4.3.3.2 Lectura analógica y comunicación serial

Para este ejemplo se realiza la medición de un sensor analógico entre 0 y 10 voltios que otorga un valor variable que va a ser visualizado de 3 formas diferentes en la pantalla HMI mediante comunicación serial, además se configura un valor específico para que accione una salida a un relé que ejemplificara la activación de un actuador.

La pantalla HMI de la marca NEXTION ofrece un entorno de desarrollo amigable para desarrollar las diferentes interfaces en un proyecto, facilitando su uso y con una amplia documentación y librerías en C++. Para la representación de el valor analógico capturado se realiza una gráfica con una aguja que gira con respecto a un eje de diferentes tonalidades que representa el valor mínimo y máximo entregado, un número decimal que muestra el valor obtenido y una barra de estado vertical que muestra el valor del dato obtenido de forma proporcional hasta llegar al limite superior.

Figura 32:



Una vez cargado las interfaces en la pantalla física, el controlador lógico programable será el encargado de leer la entrada analógica y enviar mediante comunicación serial las etiquetas y el valor para que sean representadas en la pantalla.

Figura 33:

A continuación en la tabla 4 se presenta un muestreo para determinar el error de las lecturas analógicas en la entrada del controlador, en el procesamiento del ADC y en la presentación final del valor medido, tomando en cuenta que se realizó variaciones de voltaje entre 0 y 10 voltios.

Tabla 4:

Error porcentual para valores analógicos

Valor Analógico (V)	Entrada ESP32	Procesamiento ADC	Valor entregado HMI	% de error
0	0	0	0	0
1	0,321	408	0,9963	0,37
2	0,664	817	1,9951	0,24
3	0,982	1230	3,0037	0,12
4	1,33	1639	4,0024	0,06
5	1,651	2047	4,9988	0,02
6	1,98	2455	5,9951	0,08
7	2,33	2870	7,0085	0,12
8	2,652	3276	8	0
9	2,98	3689	9,0085	0,09
10	3,34	4095	10	0
			Promedio	**0,1 %**

Analizando la tabla numero 4 se determina que el valor de error no representa alteraciones considerables en las lecturas, ya que el convertidor analógico digital del ESP32 puede realizar mediciones en rangos de 0.8 mV.

CONCLUSIONES

- El controlador lógico programable desarrollado cumple con el objetivo planteado en el presente trabajo de titulación, y puede ser usado en un entorno industrial para realizar tareas especificas y de precisión.
- Un controlador lógico programable debe disponer de 5 componentes fundamentales, una fuente de alimentación, entrada y salida de señales, unidad de procesamiento, y una unidad de programación de aplicaciones.
- Al disponer de una unidad de procesamiento con características especificas de niveles de voltaje, es necesario realizar una circuitería adicional para el acoplamiento de señales tanto digitales como analógicas que son usadas en el entorno industrial.
- Para el cargador de arranque de la unidad de procesamiento se uso el otorgado el fabricante sin ninguna modificación, para futuras versiones se puede modificar este cargador para aplicaciones especificas.
- Para el desarrollo de aplicaciones finales se detallo las diferentes opciones de programación quedando abierta la posibilidad de que el usuario final escoja el entorno de programación que mas se sienta cómodo.
- Las pruebas realizadas en el controlador lógico programable determinaron que el diseño si responde a las condiciones industriales para tareas repetitivas y de activación de actuadores con niveles de voltaje superiores a la unidad de procesamiento.
- La versión desarrollada para el presente trabajo de titulación es una versión inicial y puede ser replicada o mejorada dependiendo de la aplicación especifica donde vaya a ser utilizada, sin ningún tipo de restricción tanto de hardware como de software.

RECOMENDACIONES

- Los sistemas embebidos que están operando actualmente en entornos industriales han sido valorados y certificados por organismos técnicos que avalan su funcionamiento y seguridad, para el presente controlador lógico programable y futuras versiones es necesario realizar una certificación para que pueda ser comercializado.
- Los esquemáticos presentados en este trabajo de titulación corresponden a diseños para la unidad de procesamiento elegida, para ser replicados en otra unidad deben ser evaluados tanto a nivel de voltaje como en consumo de corriente.
- El entorno de desarrollo de programación puede ser elegido acorde a las exigencias del desarrollador, y no es un limitante de uso siempre y cuando se desarrolle en el lenguaje de programación que esta cargada la unidad de procesamiento.
- Por propósitos educativos se realizó un diseño extendido de la placa, pero para propósitos de aplicación o comerciales debe ser reducido en su tamaño y disposición de conectores tanto de entrada como de salida.

TRABAJO FUTURO

En el presente trabajo se diseñó y construyó un controlador lógico programable para aplicaciones industriales, que a lo largo de su desarrollo se pudo plantear alguna lineas de investigación y de diseño para trabajos futuros, a continuación se presentan algunas ideas para trabajos futuros que derivan del presente trabajo de titulación y no estuvieron contempladas en los objetivos planteados.

- Desarrollar un cargador de arranque propio con documentación en español para futuras versiones y mejoras aplicadas en sistemas embebidos basados en ESP32.
- Realizar versiones especificas del controlador lógico programable diseñado para tareas o aplicaciones finales de consumo.
- Diseñar una estructura final de ensamblaje compacta para el controlador lógico programable.

Bibliografía

UNNED,2011: Departamento de Ingeniería Eléctrica, Electrónica y Control, Controladores Industriales Inteligentes, , http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE6_1_1.pdf

Sánchez, 2010, p. 47-49: INGENIUS, Diagnostico del nivel de automatización en las pequeñas y medianas industrias de la ciudad de Cuenca, 2010, doi:https://doi.org/10.17163/ings.n4.2010.05

Guerra Procel & Martin-Montaner, 2014, p.504-521: Guerra Procel, Martin-Montaner, J., Desarrollo Histórico de la industria Manufacturera Ecuatoriana y su matriz de exportación., 2014, https://revistapublicando.org/revista/index.php/crv/article/view/591

PLC Desing, 2017: PLC Desing, Automatización con PLC: Soluciones industriales, 2017, http://plcdesign.xyz/automatizacion-con-plc-soluciones-industriales/

Easwaran, E., Kushalkar, R., Tigadi, N., Moudgalya, K., Chipkar, A., Akshai, M., & Alves, T. (2018). Programmable Logic Controller: Open Source Hardware and Software for Massive Training. IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2422-2427. doi:10.1109/IECON.2018.8592772

IFAC, 2016: ifac-control.org, International Fereration of Automatic, 2016, https://www.ifac-control.org

Kazala, R., & Stracynski, P. (2019). The Most Important Open Technologies for Design of Cost Efficient Automation Systems. 19th IFAC Conference on Technology, Culture and International Stability TECIS 2019, 52(25), 391-396. doi:https://doi.org/10.1016/j.ifacol.2019.12.567

Pratumsuwan, P., & Pongaen, W. (2011). An embedded PLC development for teaching in mechatronics education. 6th IEEE Conference on Industrial Electronics and Applications, 1477 - 1481. doi:DOI:10.1109/ICIEA.2011.5975823

Boot & Work Corp. S.L. (2021). Soluciones industriales basadas en Open Source Hardware. Recuperado el 3 de Mayo de 2021, de Industrial Shields: https://www.industrialshields.com/es_ES/

Revolution Pi, 2021: Revolution Pi, The Only Real industrial Raspberry Pi, , https://revolution.kunbus.com

GNU.org. (2021). FreeSoftware Foundation. Recuperado el 11 de Junio de 2021, de Proyecto GNU: https://www.gnu.org/philosophy/lessig-fsfs-intro.es.html

Pillku, 2012: Pillku amantes de la libertad, Hardware Libre para una sociedad libre, , https://pillku.org/hardware-libre-para-una-sociedad-libre/

EcuRed, 2019: Enciclopedia EcuRed, Hardware Libre, , https://www.ecured.cu/Hardware_libre

Arduino, 2022: , Arduino Home Page, , https://www.arduino.cc/

Raspberry Pi: , Raspberry Pi Foundation, , https://www.raspberrypi.org/

Espressif, 2022: , Espressif Company, , https://www.espressif.com/en/company/about-espressif

Seika, 2021: , Lenguajes de programación para PLC, , https://www.seika.com.mx/5-lenguajes-de-programacion-para-plc/

KiCad, 2022: , KiCad EDA, , https://www.kicad.org/about/kicad/

FlatCAM, 2022: , FlatCAM: Free and Open-source PCB CAM, , http://flatcam.org/

FreeCAD, 2022: , FreeCAD Your own 3D parametric modeler, , https://www.freecadweb.org/

ANEXOS

COSTOS REFERENCIALES

Controlador lógico programable

Tabla 5:

Costos controlador lógico programable

DESCRIPCIÓN	PRECIO
Placa PCB + Gastos de envío	20
Componentes SMD + Gastos de envío	10
Componentes THT + Gastos de envío	15
Materiales de soldadura	5
Base plástica PLA	15
TOTAL	**65**

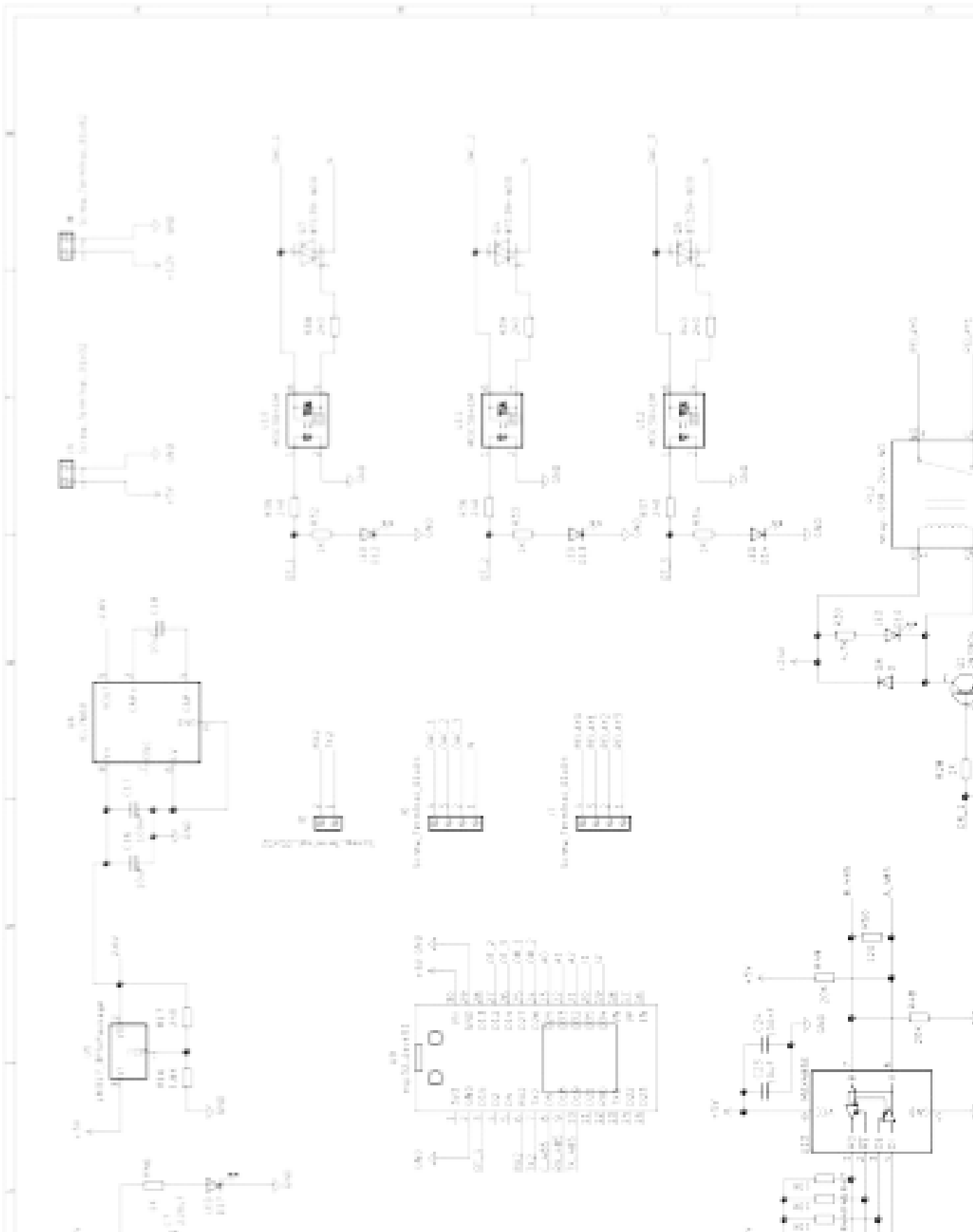
Tablero de control

Tabla 6:

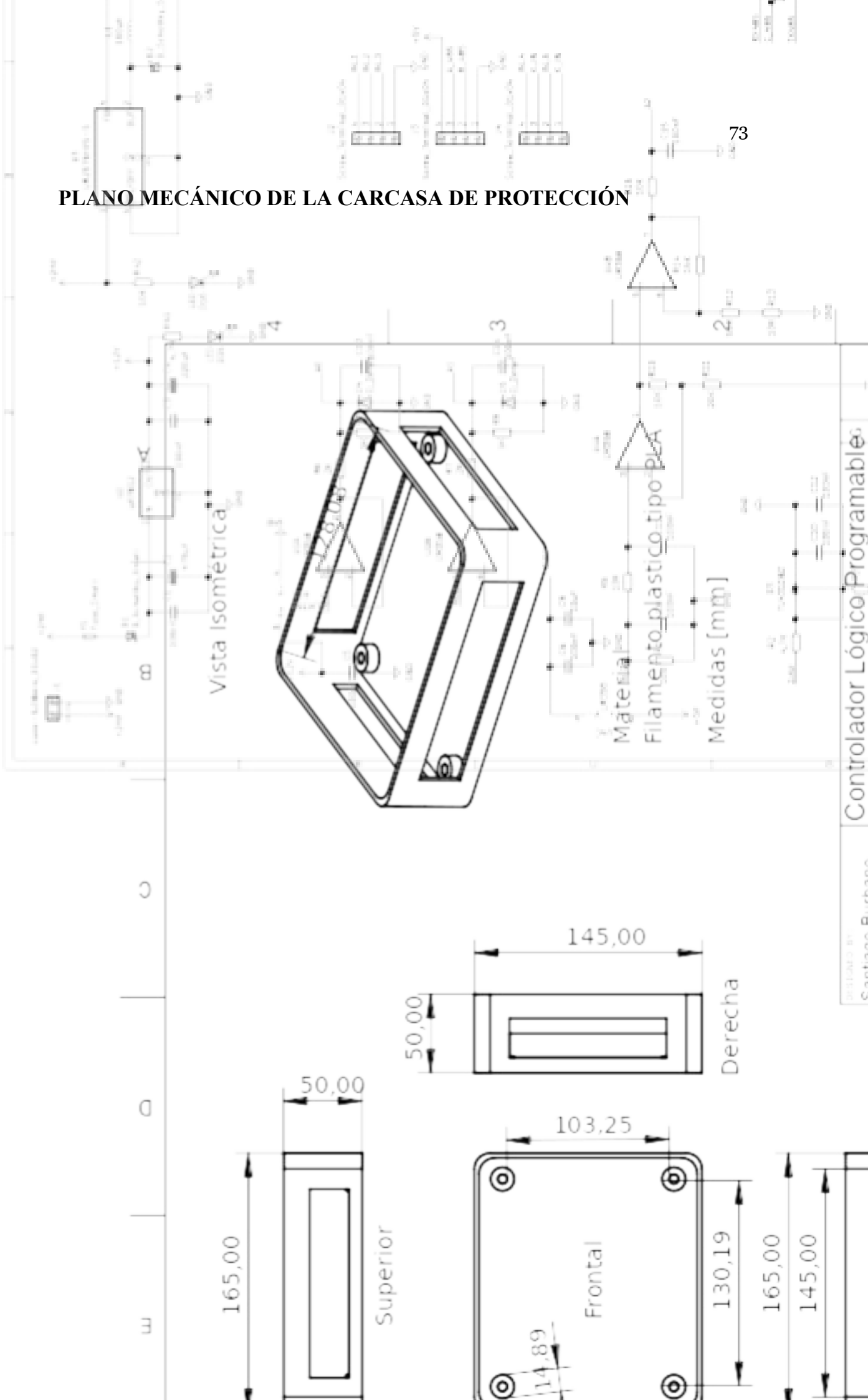
Costos tablero de control

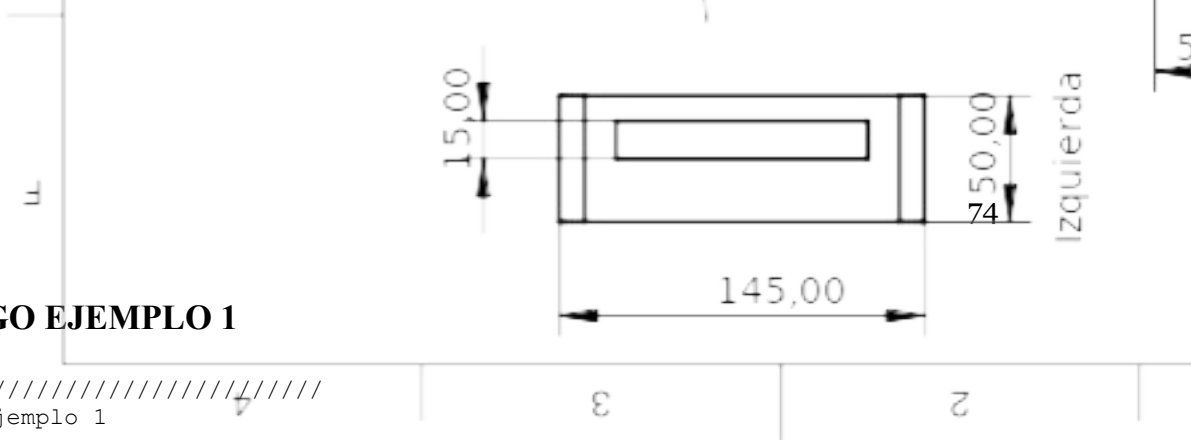
DESCRIPCIÓN	PRECIO
Caja metálica	60
Rieles DIN	3
Borneras	25
Canaletas	4
Cable AWG18	8
Botones	10
Pantalla HMI	100
Fuente 24VDC	35
Breaker 6A	8
TOTAL	**253**

ESQUEMÁTICO COMPLETO



PLANO MECÁNICO DE LA CARCASA DE PROTECCIÓN





CÓDIGO EJEMPLO 1

```
////////////////////////////
//    Ejemplo 1
//
//  secuencia de un semáforo de 3 estados
//  envío de información mediante conexión
//  inalámbrica al internet con protocolo MQTT
//  recepción de información para cambio de
//  estado de semáforo a modo precaución
//
////////////////////////////////
#include <Arduino.h>
#include <PubSubClient.h>
#include <WiFi.h>
#define ROJO  15
#define NARANJA 12
#define VERDE 14

 TaskHandle_t Task1;

const char* ssid = "home_bp";
const char* password = "Emmanuel2020";

#define TOKEN "PLC_SB"
#define DEVICE_ID "PLC_SB"
#define MQTT_SERVER "iot1.inprise.club"
#define MQTT_PORT 1883
#define INTERVAL 10000

String messageTemp;
const char* topicEventos = "IN/PLC/Eventos";

long periodoRA = 10000;
long periodoAV = 10000;
long periodoVR = 10000;
long periodoE = 2000;
unsigned long tiempoInicial = 0;
unsigned long tiempoRojo = 0;
unsigned long tiempoAmarillo = 0;
unsigned long tiempoVerde = 0;
unsigned long tiempoEmergencia = 0;

int banderaR = 1;
int banderaA = 0;
int banderaV = 0;
int banderaE = 0;
bool emergencia = HIGH;

WiFiClient espClient;
PubSubClient mqtt(MQTT_SERVER, MQTT_PORT, espClient);
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void connectMQTTServer() {
  Serial.println("Configuracion MQTT");
  delay(1000);
  if (mqtt.connect("PLC_SB")) {
    Serial.println("Conexion MQTT establecida...");
    mqtt.publish(topicEventos, ("PLC iniciando..." ));

    mqtt.subscribe(topicEventos);
    Serial.println("Topicos conectados...");
        delay(10);

  } else {
    Serial.print("error = ");
    Serial.println(mqtt.state());

    delay(10000);
    ESP.restart();

  }
}

void mqttCallback(char* topic, byte* payload, unsigned int len) {

  if (String(topic) == topicEventos) {
  Serial.print("mensaje recibido [");
  Serial.print(topic);
  Serial.print("]: ");
  Serial.write(payload, len);
  Serial.println();

  for (int i = 0; i < len; i++) {
    messageTemp += (char)payload[i];
  }
  String cmd = messageTemp.substring(0, 2);
  String data = messageTemp.substring(2);
  if(cmd.equals("M-")){
    if(data.equals("E")){
      Serial.println("EMERGENCIA!!!!!");
      banderaE = 1;
      banderaA = 0;
      banderaR = 0;
      banderaV = 0;
      }
    if(data.equals("N")){
      Serial.println("SECUENCIA NORMAL");
      banderaE = 0;
      banderaA = 0;
      banderaR = 1;
      banderaV = 0;
      }
  }
  messageTemp = "";
  }
}

void Task1code(void * pvParameters) {
  Serial.print("Task1 running on core ");
  Serial.println(xPortGetCoreID());
```

```
  for(;;){

      digitalWrite(LED_BUILTIN, HIGH);
      delay(300);
      digitalWrite(LED_BUILTIN, LOW);
      delay(300);
    }
}

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN,OUTPUT);
  xTaskCreatePinnedToCore(
                    Task1code,   /* Task function. */
                    "Task1",     /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                       &Task1,       /* Task handle to keep track of created
task */
                    0);          /* pin task to core 0 */
  delay(500);
  setup_wifi();
  delay (50);
  connectMQTTServer();
  delay(50);
  mqtt.setCallback(mqttCallback);

  pinMode(ROJO,OUTPUT);
  pinMode(NARANJA,OUTPUT);
  pinMode(VERDE,OUTPUT);
  delay(500);
  digitalWrite(ROJO,LOW);
  digitalWrite(NARANJA,LOW);
  digitalWrite(VERDE,LOW);
  delay(500);
  Serial.println("Iniciando semaforo...");

}

void loop() {

    if(!mqtt.connected())
  {
    connectMQTTServer();
  }
    mqtt.loop();

  if ((millis() > (tiempoInicial + periodoVR)) && (banderaR == 1)){
    tiempoRojo = millis();
    tiempoEmergencia = millis();
    digitalWrite(ROJO,HIGH);
    digitalWrite(NARANJA,LOW);
    digitalWrite(VERDE,LOW);

    Serial.println("rojo");
    banderaR = 0;
    banderaA = 1;
    banderaV = 0;

    mqtt.publish(topicEventos, ("R"));
  }
  if ((millis() > (tiempoRojo + periodoRA)) && (banderaA == 1)){
    tiempoAmarillo = millis();
```

```
    tiempoEmergencia = millis();
    digitalWrite(ROJO,LOW);
    digitalWrite(NARANJA,HIGH);
    digitalWrite(VERDE,LOW);
    Serial.println("amarillo");
    banderaR = 0;
    banderaA = 0;
    banderaV = 1;
    mqtt.publish(topicEventos, ("A"));
  }
  if ((millis() > (tiempoAmarillo + periodoAV)) && (banderaV == 1)){
    tiempoInicial = millis();
    tiempoEmergencia = millis();
    digitalWrite(ROJO,LOW);
    digitalWrite(NARANJA,LOW);
    digitalWrite(VERDE,HIGH);
    Serial.println("verde");
    banderaR = 1;
    banderaA = 0;
    banderaV = 0;
    mqtt.publish(topicEventos, ("V"));
  }

  if ((millis() > (tiempoEmergencia + periodoE)) && (banderaE == 1)){

    tiempoEmergencia = millis();
    digitalWrite(ROJO,emergencia);
    digitalWrite(NARANJA,!emergencia);
    digitalWrite(VERDE,LOW);

    if(emergencia){
      mqtt.publish(topicEventos, ("E"));
    } else {
      mqtt.publish(topicEventos, ("e"));
    }
    emergencia = !emergencia;
    }
}
```

CÓDIGO EJEMPLO 2

```
// ejemplo de uso de pantalla HMI mediante comunicacion serial
#include <Arduino.h>

#define salidaA 27

TaskHandle_t Task1;

int n = 0;
int m = 0;
double entrada = 0;

int valor_anterior ;
int umbral = 3;

void setup() {
  Serial.begin(115200);
  Serial2.begin(115200);
  pinMode(salidaA,OUTPUT);
  Serial.println("iniciando..");
  digitalWrite(salidaA,LOW);
```

```
}

void loop() {

    entrada = analogRead(25);
    Serial.println(entrada);

    m = entrada / 24.8;
    n = m * 1.8;
    Serial2.print("z0.val=");
    Serial2.print(  n  );
    Serial2.write(0xff);
    Serial2.write(0xff);
    Serial2.write(0xff);

    Serial2.print("va0.val=");
    Serial2.print(  m  );
    Serial2.write(0xff);
    Serial2.write(0xff);
    Serial2.write(0xff);

    Serial2.print("j0.val=");
    Serial2.print(  m  );
    Serial2.write(0xff);
    Serial2.write(0xff);
    Serial2.write(0xff);
    //n++;

    if(m > 80){
      digitalWrite(salidaA,HIGH);
    }else{
      digitalWrite(salidaA,LOW);
    }
    m++;
     if(m==101){

      m=0;
      }
  delay(1000);

  if (Serial1.available()) {
    int inByte = Serial1.read();
    Serial.write(inByte);
  }

}
```