ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA INGENIERÍA INDUSTRIAL

IMPLEMENTACIÓN DE UN SISTEMA SCADA PARA LA INTEGRACIÓN DE PROCESOS INDUSTRIALES BASADO EN UN AMBIENTE DE PROGRAMACIÓN *OPEN SOURCE*

Trabajo de Integración Curricular

Tipo: Propuesta Tecnológica

Presentado para optar al grado académico de:

INGENIERO INDUSTRIAL

AUTOR:

CRISTIAN XAVIER ROMERO CHICA

Riobamba - Ecuador

2023

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA INGENIERÍA INDUSTRIAL

IMPLEMENTACIÓN DE UN SISTEMA SCADA PARA LA INTEGRACIÓN DE PROCESOS INDUSTRIALES BASADO EN UN AMBIENTE DE PROGRAMACIÓN *OPEN SOURCE*

Trabajo de Integración Curricular

Tipo: Propuesta Tecnológica

Presentado para optar al grado académico de:

INGENIERO INDUSTRIAL

AUTOR: CRISTIAN XAVIER ROMERO CHICA

DIRECTOR: Ing. CALISPA AGUILAR MARCELO ESTEBAN Mgs.

Riobamba - Ecuador

2023

Yo, CRISTIAN XAVIER ROMERO CHICA declaro que el presente Trabajo de Integración Curricular es de mi autoría y los resultados de este son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular; El patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 22 de agosto de 2023

Cristian Xavier Romero Chica
172413322-6

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA INGENIERÍA INDUSTRIAL

El Tribunal de Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular: Tipo: Propuesta Tecnológica, **IMPLEMENTACIÓN DE UN SISTEMA SCADA PARA LA INTEGRACIÓN DE PROCESOS INDUSTRIALES BASADO EN UN AMBIENTE DE PROGRAMACIÓN *OPEN SOURCE***, realizado por el señor: **CRISTIAN XAVIER ROMERO CHICA**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud que el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. José Francisco Pérez Fiallos **PRESIDENTE DEL TRIBUNAL**		2023-08-22
Ing. Marcelo Esteban Calispa Aguilar **DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-08-22
Ing. Eduardo Francisco García Cabezas **ASESOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-08-22

DEDICATORIA

A mi familia, en especial a mis hermanas y mamá que me brindaron un apoyo incondicional y que estuvieron presentes en todas las etapas para concluir mi carrera profesional. A mi abuelo que ya no está presente pero siempre confió en todo lo que puedo lograr. A mi querida Laika que desde hace más de 5 años se convirtió en una constante compañía.

A mí, por todo el esfuerzo y dedicación puesto a lo largo de la etapa más importante de mi vida, por la perseverancia y fortaleza demostradas que me permitieron ser un mejor hombre día a día y conseguir los objetivos que me plantee.

Cristian

AGRADECIMIENTO

En primer lugar, agradezco a mi familia por su constante apoyo, sacrificio y ayuda brindada en todos los momentos vividos a lo largo de mi vida, sin ellos no sería el hombre que soy ahora.

A la Escuela Superior Politécnica de Chimborazo y a los profesores que conforma la Carrera de Ingeniería Industrial por la formación académica y ética de calidad que me brindaron a lo largo de la etapa universitaria.

A los compañeros y amigos de aula que fueron parte importante en mi formación académica, compañeros con los que compartí buenos y malos momentos, victorias y derrotas, sin ellos los logros alcanzados no serían posibles. A la comunidad latinoamericana Python por sus aportes y ayudas en el presente trabajo de titulación.

Cristian

ÍNDICE DE CONTENIDO

CAPÍTULO III

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

CAD:	Dibujo asistido por computadora
VCD:	Voltaje de corriente directa
VCA:	Voltaje de corriente alterna
RTU:	Unidad terminal remota
MTU:	Unidad terminal maestra
HMI:	Interface humano-máquina
SCADA:	Supervisión, control y adquisición de datos
GUI:	Interfaz de usuario gráfica
IDE:	Entorno de desarrollo integrado
PMMA:	Polimetilmetacrilato
IoT:	Internet de las cosas
VA:	Visión artificial
I2C:	Puerto y protocolo de comunicación
TCP:	Protocolo de control de transmisión
IP:	Protocolo de internet
Wi-Fi:	Red de comunicación inalámbrica

RESUMEN

En el presente trabajo de titulación se describe el desarrollo de un sistema de control, supervisión y adquisición de datos (SCADA), integrando recursos tecnológicos de *hardware* y *software* de código abierto con la finalidad de generar una propuesta tecnológica que se adapte a los sistemas productivos PYMES. Se inició con una revisión bibliográfica de los elementos que componen un SCADA, determinando los requerimientos para el desarrollo del sistema, en paralelo se realizó el levantamiento de información de los dos procesos industriales, el primero de selección y clasificación de botellas y el segundo de envasado lineal. A continuación, con el fin de lograr integrar los dos procesos industriales experimentales se realizó la selección del *hardware* principal, se empleó la tarjeta microcontrolador Arduino MEGA 2560 que permite el control de los procesos y gestión de la información obtenida, logrando la automatización de los mismos, posteriormente se generó una arquitectura de comunicación maestro-esclavo con el uso del Bus I2C, que permite el envío y recepción de datos desde los dos microcontroladores esclavos en cada proceso hacia el microcontrolador maestro que actúa como MTU. Además, dentro del proceso de construcción de los módulos de actuadores y controladores, primero se inició con el diseño de los modelos CAD en el *software* Inventor para posteriormente realizar la construcción y ensamblaje de los mismos. Posteriormente se tomaron los tiempos de latencia en la comunicación entre la interface hombre-máquina (HMI) y los procesos, obteniendo un tiempo promedio de 4,75 segundos en la activación y desactivación, la recepción de datos y almacenamiento de los mismos toma un tiempo promedio de 1,72 segundos. En conclusión, es factible desarrollar sistemas SCADA en base a herramientas de *software* y *hardware open source*, lo que permite el desarrollo tecnológico en las PYMES, disminuyendo costos de implementación de nuevas tecnologías que mejoren los procesos productivos.

Palabras clave: <SISTEMA DE CONTROL, SUPERVISIÓN Y ADQUISICIÓN DE DATOS (SCADA)> <AUTOMATIZACIÓN> <INTEGRACIÓN DE PROCESOS> <MICROCONTROLADORES> <SOFTWARE LIBRE>.

SUMMARY

This degree project describes the development of a Supervisory Control and Data Acquisition system (SCADA), integrating technological resources of hardware and open source software with the aim of generating a technological proposal that adapts to PYMES (acronym in Spanish) production systems. A bibliographic review of the elements that make up a SCADA was carried out, determining the requirements for the development of the system. Information was also gathered on the two industrial processes, one on the selection and classification of bottles and the other on linear packaging. The main hardware was selected to integrate the two experimental industrial processes. On the one hand, the Arduino MEGA 2560 microcontroller board was used to control the processes and manage the information received, thereby achieving the automation of the processes. On the other hand, a master-slave communication architecture was then created using the I2C bus, which allows sending and receiving data from the two slave microcontrollers in each process to the master microcontroller, which acts as a maximum transmission unit (MTU). In addition, within the construction process of the actuator and controller modules, the first step was the design of the Computer-Aided Design (CAD) models in Inventor software, followed by their construction and assembly. Subsequently, the latency times in the communication between the human-machine interface (HMI) and the processes were taken, obtaining an average time of 4.75 seconds in the activation and deactivation, and the reception and storage of data took an average time of 1.72 seconds. In conclusion, it is possible to develop SCADA systems based on open-source software and hardware tools, which enables technological development in PYMES and reduces the implementation costs of new technologies that improve production processes.

Keywords: <SUPERVISORY CONTROL AND DATA ACQUISITION (SCADA)>, <AUTOMATION>, <PROCESS INTEGRATION>, <MICROCONTROLLERS>, <FREE SOFTWARE>.

Angela Cecibel Moreno Novillo

0602603938

INTRODUCCIÓN

Las organizaciones están en constante búsqueda para optimizar sus procesos productivos y una forma de lograrlo es la automatización. En los últimos años, con el desarrollo de la tecnología han existido grandes avances en el área del automatización y control de plantas industriales. La capacidad de procesamiento de los sistemas informáticos ha ido en aumento y con ello también el procesamiento y adquisición de datos en los sistemas de control industrial. Los sistemas SCADA (por sus siglas en ingles *Supervisory Control And Data Acquisition*) supervisión, control y adquisición de datos facilitan la gestión en tiempo real de los procesos productivos en una planta, generando información de vital importancia para la toma de decisiones. (Pérez-López 2015, p. 4)

Los sistemas SCADA permiten supervisar, controlar y adquirir datos de las diferentes variables presentes en los procesos productivos. Para el correcto funcionamiento de un sistema SCADA es importante considerar necesario el desarrollo de un *software*, sistemas de comunicación entre los elementos de campo, una base de datos para almacenar los datos adquiridos y el desarrollo de una interface humano máquina o HMI (por sus siglas en ingles *Human-Machine Interface*). Gracias a todos estos elementos el sistema nos permite la correcta visualización de variables y posibles desviaciones de un proceso productivo en tiempo real y a distancia, facilitando realizar informes y toma de decisiones. (Rodríguez 2013, p. 18)

Para la creación de los sistemas SCADA existen aplicaciones y equipos desarrollado por empresas multinacionales. Aplicaciones que necesitan licencias para su correcto funcionamiento además que su código fuente no puede ser modificado y los módulos de campo tienen precios elevados. Esto no permite que estos sistemas se adapten a cambios que se den en el proceso, sin tener en cuenta que sus altos costos de instalación no permiten que puedan ser aplicados a procesos pequeños. Por este motivo es necesario el desarrollo de un sistema de automatización de bajo costo usando herramientas *open source* o código abierto que sean versátiles y aplicables a pequeños procesos en las micro, pequeñas y medianas industrias.

Los ambientes de programación *open source* se presentan como una alternativa más económica y versátil a los sistemas de programación SCADA tradicionales. Los sistemas SCADA desarrollados en ambientes de programación *open source* cuentan con una gran variedad de *software* y *hardware* de bajo costo.

El presente trabajo contiene el desarrollo de un sistema SCADA para el control, supervisión y adquisición de datos de dos procesos industriales, aplicados en dos procesos industriales didácticos desarrollados para la carrera de Ingeniería Industrial de la Escuela Superior Politécnica de Chimborazo, presentando soluciones asequibles y versátiles de automatización, supervisión y toma de datos en procesos productivos para las micro, pequeñas y medianas empresas.

CAPÍTULO I

1 DIAGNÓSTICO DEL PROBLEMA

1.1 Antecedentes

Los primeros pasos en la creación de sistemas y procesos automatizados para la industria se dan con el desarrollo de sistemas automáticos de lógica cableada, usando elementos o contactos de tipo físico, en este caso, cables, relés, temporizadores, contactores, contadores y demás. Contando también con sistemas de protección como son los guardamotores, termomagnéticos, variadores de frecuencia. Estos sistemas cumplían las funciones básicas para el control, mando, señalización y potencia. Al ser de carácter físico, su modificación se debe realizar cambiando el cableado y los elementos físicos del sistema, de modo que el mismo pueda cumplir la nueva función requerida. Los sistemas de lógica cableada aún son utilizados en máquinas y procesos productivos pequeños que no requieran cambios ni una alta inversión.

A partir de la década de 1960 en los países industrializados comienza el desarrollo de la lógica programable para la industria, dando versatilidad a los procesos industriales, ya que se facilitan los cambios de los procesos industriales. Los cambios ya no se realizan de manera física, solo es necesario que se realicen cambios en la programación del sistema para que este pueda realizar la nueva función solicitada, a pesar de las mejoras estos procesos son aislados. A mediados de la década de 1970, aparece el termino SCADA para definir los sistemas de supervisión, control y adquisición de datos. Los mismos permiten la integración de los distintos procesos productivos que pueden existir en una planta industrial. Generando un flujo de información que va desde la planta productiva hasta la gerencia, entregando información relevante para la toma de decisiones en la empresa. A partir del año 2 000 con la revolución de conectividad gracias al internet, los sistemas SCADA pueden entregar información sobre los procesos productivos en cualquier parte del mundo, facilitando aún más la gestión de los procesos industriales.(Antón et al. 2017, p. 98)

Los ambientes de programación *open source* o código abierto han generado grandes avances en lo que respecta al desarrollo tecnológico, gracias a la gran comunidad de programadores con la que cuenta. Los cambios en los códigos y librerías creadas permiten desarrollar sistemas de control, supervisión y adquisición de datos versátiles, que se pueden aplicar en pequeños procesos. En el 2016 en la Universidad de Cuenca se desarrolló un estudio comparativo entre los sistemas SCADA tradicionales y los sistemas creados en base a ambientes de programación *open source*, el objetivo de este estudio es crear un sistema SCADA económico y de fácil acceso para la pequeña y mediana empresa, ahorrando en costos de licencias y programadores lógicos en campo.(Minchala et al. 2016, p. 4657)

1.2 Planteamiento del problema

Las mipymes (micro, pequeñas y medianas empresas) existentes en el Ecuador son de vital importancia para la economía del país. Según (Instituto Nacional de Estadisticas y Censos 2021) en nuestro país las mipymes representan el 99,5% de empresas establecidas, generando el 27,6% de las ventas totales en todo el país. Uno de los principales problemas de las mipymes nacionales es que a pesar de su importancia en la economía nacional no están teniendo el impacto que deberían. La falta de innovación y desarrollo tecnológico están limitando su competitividad y supervivencia en el mercado.

Actualmente solo las empresas grandes pueden permitirse tener sistemas que les permita el control, supervisión y adquisición de datos, esto se da debido a que desarrollar un sistema SCADA conlleva un alto costo siendo muy poco asequible. Antes de adquirir estos sistemas es necesario tomar en cuenta parámetros como: el volumen de producción, la demanda, el tiempo de vida del producto. Estos sistemas normalmente desarrollados para las grandes industrias no se suelen ajustar a las necesidades de las empresas de menor escala generándose así una gran brecha en innovación tecnológica.

Las mipymes se caracterizar porque la mayoría de sus procesos, si no es que todos, son manuales y requieren de un operario, esto conlleva a que se generen errores, afectando seriamente el proceso productivo. El desarrollo de sistemas SCADA de bajo costo tiene como fin disminuir las pérdidas de dinero, tiempo, procesos ineficientes, desperdicios de materia prima, aumentando el orden y mejorando la estandarización y control de los procesos productivos.

Por tal motivo, es necesario el desarrollo de una opción de automatización, supervisión, control y de adquisición de datos asequible para las mipymes. Generando innovación tecnológica para ese gran porcentaje de empresas. De esta manera se podrá lograr aumentar la competitividad de las misma, brindándoles herramientas acordes a la nueva revolución industrial.

Las herramientas *open source* permiten generar soluciones para el desarrollo de sistemas SCADA, estas herramientas se ajustan a los requerimientos de las pequeñas y medianas empresas. El costo generado por el desarrollo de un sistema de supervisión, control y adquisición de datos en tiempo real en ambientes de programación de código abierto es relativamente bajo, además el *hardware* necesario para el funcionamiento del sistema integrado en las plataformas de programación *open source* es asequible.

1.3 Justificación

El uso de los sistemas de automatización industrial permite la gestión de los procesos productivos de una manera más ágil y mucho más fácil. Se mejoran la velocidad de respuesta ante posibles fallas o desviaciones del proceso. La información del proceso productivo se recopila de manera más rápida para su procesamiento y toma de decisiones. El desarrollo de un sistema SCADA en un ambiente de programación *open source* permite la automatización de procesos de pequeña escala, de manera económica, proporcionando los mismos beneficios de los sistemas SCADA de las grandes empresas.

La presente propuesta tecnológica es de gran importancia, porque aporta al desarrollo tecnológico industrial y permite optimizar los procesos productivos de las mipymes. Mejorando la competitividad de estas empresas, que por muchos años se han quedado atrás en lo que respecta a desarrollo tecnológico. En la actualidad los sistemas SCADA han aportado muchos beneficios a la gran industria, en gran medida gracias a sus amplias prestaciones, las mismas que pueden generar beneficios a la pequeña industria.

Por este motivo en el presente trabajo de titulación se propone desarrollar un sistema SCADA que integre dos procesos industriales didácticos (Envasadora de líquidos, clasificadora de botellas) de la carrera de Ingeniería Industrial, los que se integrarán permitiendo la visualización de los procesos mediante el uso de una interface hombre-máquina HMI. El sistema también permitirá el control, supervisión y almacenamiento en una base de datos de variables inmiscuidas en el proceso productivo. El desarrollo del proyecto sentará las bases para otros proyectos de automatización de menor escala, fomentando el desarrollo tecnológico en las mipymes.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema SCADA para la integración de procesos industriales basado en un ambiente de programación *open source*.

1.4.2 Objetivos Específicos

Realizar una investigación mediante un análisis bibliográfico sobre las herramientas de *software* y *hardware* en ambientes de programación *open source* para el desarrollo de un sistema SCADA que integre dos procesos experimentales didácticos considerados para este trabajo.

Seleccionar los elementos de *hardware* y *software* necesarios para la automatización de los procesos experimentales didácticos y su integración a un sistema orientado al control, supervisión y adquisición de datos.

Desarrollar una interfaz gráfica para la interacción del usuario con los procesos experimentales en una plataforma software de desarrollo.

Implementar el sistema SCADA vinculando la parte *hardware* de los procesos a la interfaz gráfica desarrollada.

Realizar pruebas de funcionalidad en el sistema SCADA para la determinación de tiempos de respuesta en sus diferentes etapas.

CAPÍTULO II

2 MARCO TEÓRICO

2.1 Antecedentes de investigación

A partir de la revisión bibliográfica de investigaciones anteriores realizadas por diferentes autores se puede determinar lo siguiente:

Aghenta e Iqbal presentan dentro de su artículo, el desarrollo y configuración de un sistema IoT (Internet de las cosas por sus siglas en ingles) de bajo costo basado en sistemas SCADA. El sistema es desarrollado para el control del voltaje en la generación de energía eléctrica, mediante el uso de paneles solares. Para la construcción del sistema se usó herramientas de *hardware* y *software open source*. El proyecto consta de elementos de campo (sensores y actuadores) representados por dos sensores de voltaje y un sensor de corriente, un RTU (Unidad terminal remota por sus siglas en inglés) en este caso un micro controlador ESP32 *Thing micro-controller* programado por medio del *software* Arduino, el mismo que sustituye al PLC tradicional y como MTU (Unidad terminal maestra por sus siglas en inglés) un Raspberry Pi 2 modelo B. La comunicación entre los dispositivos se da mediante un protocolo de comunicación Wi-Fi TCP/IP, usando un módulo Wi-Fi para el que la terminal RTU transmita los datos a un *router* el mismo que envía la información recolectada en campo a la MTU y a la HMI mediante el uso de protocolos de comunicación Modbus TCP/IP. Los datos recolectados son presentados en el *software* Thinger.IO una plataforma *open sorce* del Internet de las cosas. (Aghenta y Iqbal 2019, pp. 1-4)

En la conferencia de jornadas automáticas se presentó un estudio comparativo de entornos *open source* para sistemas SCADA, Portalo Calero indica que el principal objetivo es generar información útil y fiable para la elección de un sistema SCADA. Las características más importantes tomadas en cuenta para el estudio comparativo son: los protocolos de comunicación que dispone cada *software*, el lenguaje de programación en el que están desarrolladas las aplicaciones, el tipo de licencia que ofrecen, la compatibilidad con sistemas operativos, actualizaciones y nuevas versiones de *software*, conectividad a internet y repositorios en línea que permitan almacenar y revisar código elaborado por distintos desarrolladores. Generando así una tabla comparativa que permita seleccionar el *software* SCADA en función de las necesidades del desarrollador. (Portalo Calero et al. 2020, pp. 429-432)

Ashraf e Iqbal presentan en una conferencia los resultados de su proyecto titulado "*An open sorce SCADA for a solar water pumping system designed for Pakistani Conditions*". En el proyecto se desarrolló un sistema SCADA que permite la gestión de un sistema de captación de agua para riego usando energía solar para su funcionamiento. El sistema está pensado para el mejoramiento

de la producción agrícola en zonas aisladas de Pakistán, donde el suministro de energía eléctrica es nulo. El proyecto consta de una MTU usando una placa microordenador Raspberry Pi modelo 4B la misma que almacenará y ejecutará el sistema SCADA usando la plataforma *open source* Emoncms, un Arduino Mega 2560 que cumple la función de RTU recopilando información de los sensores del sistema para enviarla a la MTU además de activar el actuador, sensores de voltaje, nivel, temperatura, humedad, intensidad luminosa, de caudal, de nivel y como actuador la bomba para la recolección de agua. La recepción de datos se realizó mediante un protocolo de comunicación serial entre la RTU y la MTU. Los datos generados por el sistema se pudieron visualizar en la aplicación de código abierto en línea Emoncms, la misma que tiene como fin la medición de consumo de energía. Creando así una opción de monitoreo de sistemas de bajo costo y de fácil manejo. (Ashraf y Iqbal 2021, pp. 1403-1406)

Suarez en su tesis titulada "Diseño e implementación de un sistema SCADA para automatismos, basado *en hardware* y *software* libre" desarrolla diferentes simulaciones de sistemas de control y supervisión usando tecnología libre. La primera simulación se realiza el control de nivel de líquidos de dos tanques interconectados, se usa el microcontrolador Arduino como RTU, la transmisión de datos a la MTU o computadora/laptop se realiza mediante protocolos de comunicación Modbus, el proceso se representa en una HMI desarrollada en el programa Acimut Monitoriza, mediante un circuito con potenciómetros simulando el nivel y leds simulando las válvulas se puede visualizar la simulación físicamente. La segunda simulación es de una red SCADA multipuestos del nivel de dos tanques interconectado, en esta simulación se usan dos RTUs o dos placas Arduinos conectadas a la MTU y HMI mediante protocolos de comunicación Modbus, la interfaz HMI se desarrolla con el programa AdvancedHMI. La tercera simulación es sobre un sistema SCADA para el control de dos semáforos, se usan dos placas Arduino, simulando un sistema multipuestos, el *software* usado para la HMI es el Snap4Arduino, el mismo que permite la transmisión de información mediante comunicación serial. La última simulación es un sistema de control de temperatura en un solo puesto, el sensor de temperatura usado es el LM35 que se conecta al pin analógico de la placa Arduino, la misma que mediante comunicación serial se conecta a la MTU y mediante el uso del programa MyOpenLab se visualizan los datos obtenidos. (Suarez 2015, pp. 57-78)

2.2 Procesos industriales

Los procesos industriales se pueden definir como una serie de tareas secuenciales, las mismas que permiten la transformación de materias primas en bienes o productos finalizados. Los procesos industriales fueron desarrollados para permitir la fabricación de productos en grandes cantidades, de manera rápida y con el menor precio posible.

2.2.1 Procesos de envasado

Los procesos de envasado o llenado de botellas son ampliamente usados en las industrias, este proceso consiste en la transferencia de un fluido desde un tanque reservorio hasta botellas o envases que permitan su distribución. Este proceso se puede ejecutar en función de variables como: volumen, peso, nivel o tiempo. (Pazmiño y Yépez 2015, p. 4)

2.2.2 Proceso de clasificación

Los procesos de clasificación son aplicados para un sin número de objetos en la industria, ya sea en la clasificación de frutas y verduras en la industria agrícola hasta en la clasificación de desperdicios para el reciclaje. Los procesos de clasificación permiten separar y clasificar objetos en función de ciertos parámetros requeridos. (Tierra y Galarza 2017, p. 7)

2.3 Herramientas de *software open source*

El termino *software open source* (en español código abierto) se refiere a los programas informáticos que cumplen con las cuatro libertades principales. Estas libertades se refieren a la libertad para la ejecución del programa en cualquier sitio o propósito en un tiempo indefinido, la libertad para adaptarlo en función de las necesidades del usuario, libertad para la redistribución del *software* y así colaborar con otras personas y libertad para mejorar el programa. (González, Seoane y Robles 2003, p. 16)

2.3.1 Lenguajes de programación

En la actualidad existen un sin número de lenguajes de programación para generar algoritmos que permiten controlar distintos aparatos, desde grandes máquinas industriales, sistemas informáticos, redes de comunicación hasta los elementos de la vida cotidiana como televisores cajeros y demás. Los algoritmos informáticos no son más que una secuencia de instrucciones introducidas en los procesadores de las máquinas, para que estas ejecuten una acción. (Hinojosa 2016, p. 15)
Existen dos tipos de lenguajes de programación, los lenguajes de bajo nivel que son más cercanos a la manera de trabajar de las máquinas y los lenguajes de alto nivel que es más entendible por los humanos. Los lenguajes de alto nivel permiten a las personas realizar algoritmos de una manera más fácil y entendible pero este lenguaje tiene que pasar por una serie de traducciones

para que sea entendido por la máquina. Dentro de esta traducción tenemos los lenguajes de tipo interpretado y compilado. (Hinojosa 2016, pp. 16-17)

Los lenguajes compilados constan de dos archivos, el código fuente y el binario. El código fuente es escrito por el programador en lenguaje de alto nivel dentro del programa llamado compilador, una vez compilado el programa se crea el archivo binario, mismo que es entendido por la máquina. Los lenguajes interpretados solo constan de un archivo llamado *script*, solo necesitan de un intérprete que lee y traduce las instrucciones del algoritmo a lenguaje máquina.(Hinojosa 2016, pp. 17-18)

2.3.2 Características de los lenguajes de programación

Según Moya (2009, p. 68) las principales características de los lenguajes de programación *open source* son:

- Los lenguajes de programación *open source* se usan y distribuyen libremente esto incluye su distribución comercial. La licencia es de uso libre.
- Puede ser ejecutado en distintos sistemas operativos.
- Se pueden desarrollar o utilizar módulos desarrollados por otros usuarios, reutilizando otros algoritmos.
- Disponen de una gran cantidad de librerías y módulos desarrollados para una gran cantidad de tareas.
- Soporta varios estilos de programación: orientada a objetos, estructurada o funcional.

2.3.3 Elementos de los lenguajes de programación

Los lenguajes de programación constan de distintos elementos de vital importancia para su funcionamiento, Downey (2017, pp. 3-29) destaca los siguientes:

- **Tipos de datos:** El lenguaje puede trabajar con distintos tipos de datos ya sean enteros, decimales, caracteres o cadena de caracteres.
- **Variables:** Son datos que se almacenan en la memoria del procesador, deben contar con nombre, indicar el tipo de dato y el dato.
- **Entradas y salidas:** Permiten el ingreso de valores desde periféricos de entrada y las salidas muestran datos en los periféricos de salida.
- **Operadores aritméticos:** Permiten realizar operaciones aritméticas sobre las variables del algoritmo.

- **Comentarios:** Permite escribir texto en lenguaje humano para la documentación del código, este texto no es ejecutado por el intérprete y solo es de utilidad para el programador.
- **Datos complejos:** Permiten almacenar una serie de datos en una sola variable, estos pueden ser matrices, vectores, listas, diccionarios o tuplas.
- **Estructuras de control de flujo:** Agrupan instrucciones de manera controlada, dentro de las estructuras de control tenemos:
    - **Estructuras de control condicionales:** Permiten la ejecución de una serie de instrucciones en función de una condición o pregunta.
    - **Estructuras de control iterativo:** Permiten la ejecución y repetición de una serie de instrucciones en función de una condición.
- **Funciones:** Son instrucciones contenidas en una cápsula, mismas que pueden ser ejecutadas cuando el programa llame a la función, se puede llamar a la función las veces que se necesite.
    - **Funciones incorporadas:** Son funciones que forman parte del lenguaje de programación, ya están preestablecidas.
    - **Funciones definidas por el usuario:** Son funciones creadas por el usuario en función de sus requerimientos y necesidades.
- **Módulos:** Son archivos con líneas de código y funciones que pueden ser usados en otros programas.

2.4 Herramientas de *hardware open source*

En la actualidad existen diferentes sistemas embebidos de código abierto que permiten la automatización de procesos, los elementos de *hardware* que podemos destacar son los PLC de código abierto, placas microcontroladores y miniordenadores. Estos elementos pueden ser adaptados a sistemas SCADA tomando en cuenta ciertas características como el número y tipo de entradas y salidas, tipo de protocolos de comunicación, memoria y capacidad de procesamiento. (Suarez 2015, p. 29)

2.4.1 PLC open source

Los PLC ilustración 2-1, son elementos de control de procesos, pueden trabajar como RTU o MTU, constan de 5 componentes básicos según Rodrigues (2014, p. 585-586):

- **Rack:** Permite la comunicación entre los distintos módulos que conforman el PLC.

- **Fuente de poder:** Provee de energía regulada de corriente directa a los diferentes módulos del PLC.
- **CPU:** Es el cerebro del PLC, es responsable por el procesamiento de la información recibida desde las entradas y la ejecución de órdenes a las salidas. Está formado por dos tipos de memorias, la memoria permanente que permite almacenar programas y la memora volátil que es usada para el procesamiento de las operaciones.
- **Entradas:** Leen las señales de los elementos de campo o sensores, estos pueden ser digitales o analógicos. Las entradas digitales son usadas para recibir información de encendido y apagado mientras que las entradas analógicas son usadas para medir magnitudes físicas.
- **Salidas:** Permiten enviar señales de activación a los actuadores dentro de un proceso.

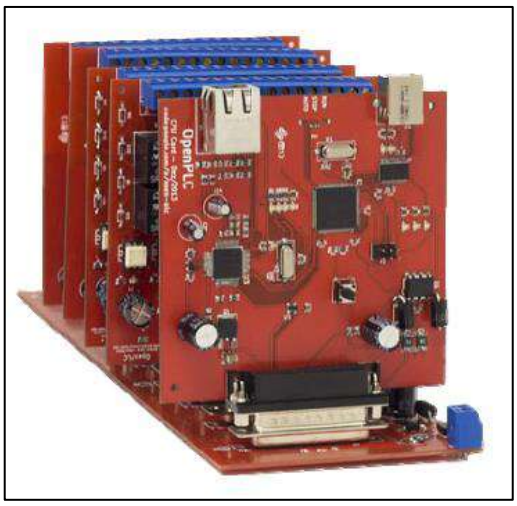


Ilustración 2-1: Prototipo de OpenPLC

Fuente: (Rodrigues Alves et al. 2014, p. 585)

2.4.2 Microcontroladores

Los microcontroladores son elementos de *hardware* usados en proyectos de electrónica, permiten recolectar información del exterior, procesarla y enviar señales de salida para interactuar con el exterior. Su mayor diferencia con las computadores y microprocesadores es la incorporación de los distintos elementos que los forman. (Ramírez Gorostizaga 2016, p. 13)

Los microcontroladores son circuitos integrados programables que cuentan con memoria (RAM y ROM) un microcontrolador, además de periféricos de entrada y salida. Entre sus principales ventajas tenemos el reducido espacio que ocupan, compatibilidad con diferentes elementos periféricos, además de sus bajos costos y su diversidad en el mercado. En la ilustración 2-2 se muestra una placa microcontrolador Arduino. (Ramírez Gorostizaga 2016, p. 14)

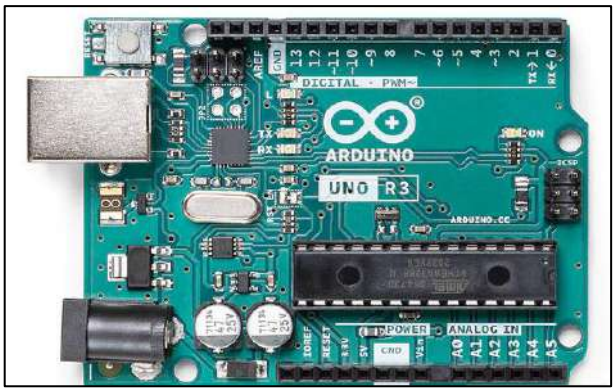


Ilustración 2-2: Placa microcontrolador Arduino

Fuente: (Arduino 2021)

2.4.3 Miniordenadores

Las placas miniordenadores elementos relativamente nuevos en el mundo de la tecnología, son usados en proyectos electrónicos de mayores exigencias, en memoria, procesamiento y almacenamiento de datos. Los miniordenadores cuentan con los mismos elementos que los ordenadores, pero el espacio que ocupan es muy reducido. (Delgado 2021, p. 33)

Los miniordenadores cuentan con módulos de memoria (RAM y ROM), más potentes que los microcontroladores, una unidad central de procesamiento (CPU), módulo wifi, ethernet y periféricos de entrada y salida. Su mayor ventaja radica en su potencia y velocidad de procesamiento. En la ilustración 2-3 se muestra un miniordenador Raspberry. (Delgado 2021, p. 33)

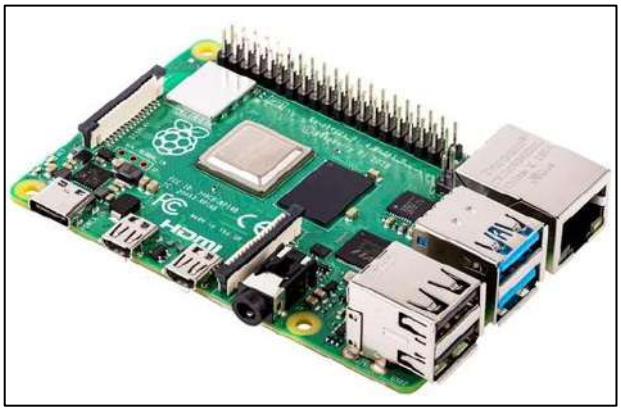


Ilustración 2-3: Placa miniordenador Raspberry Pi

Fuente: (Raspberry 2021)

2.5 Visión Artificial

La visión artificial o visión por computadora es un campo que forma parte de inteligencia artificial, mediante el uso de un algoritmo informático se recrea el proceso de visión y procesamiento de imágenes que es desarrollado por el cerebro humano. Para que un algoritmo de visión artificial pueda reconocer objetos en función de sus características principales es necesario entrenarlo, ya que el programa no tiene un conocimiento previo del objeto. (Gavrishev Mesheryakov 2022, p. 21)

2.5.1 Redes Neuronales

Son sistemas adaptativos de procesamiento de información inspirados en el funcionamiento del cerebro humano. Su estructura está basada en una gran cantidad de elementos llamados nodos o neuronas las cuales se comunican entre sí mediante el uso de enlaces. Cada neurona cuenta con un peso el mismo que indica la información requerida para la solución de un determinado problema. En la ilustración 2-4 se puede apreciar gráficamente una red neuronal. (Herrera, Barrios y Pérez 2014, p. 204)

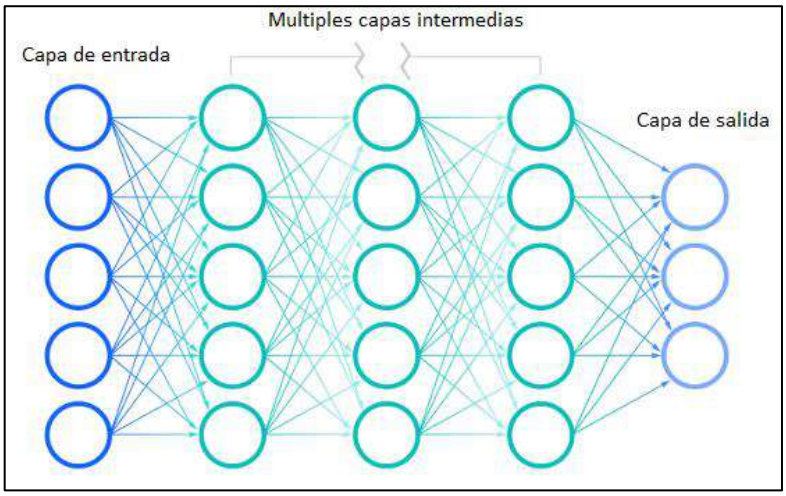


Ilustración 2-4: Representación gráfica de una red neuronal

Fuente: (Gavrishev Mesheryakov 2022, p. 14)

El principal objetivo de las redes neuronales es resolver problemas complejos que sería difíciles de resolver con programación convencional, para resolver estos problemas se sigue un proceso de entrenamiento y aprendizaje. La resolución de problemas se desarrolla en base a la experiencia creando su propio sistema de resolución. (Herrera, Barrios y Pérez 2014, p. 204)

2.5.1.1 Red neuronal convolucional

La red neuronal convolucional o (CNN por sus siglas en inglés), es una red multicapa lo que significa que cada nodo está enlazado con todos los nodos de la capa a continuación. Es principalmente usada en el desarrollo de algoritmos de reconocimiento, clasificación y visión artificial. (Gavrishev Mesheryakov 2022, p. 17)

Una de sus principales características se encuentra en el funcionamiento, la red comienza generando patrones a partir de formas simples, lo que implica que las primeras capas se centran en características como colores y siluetas, mientas el proceso avanza las capas a continuación se centran en características más complejas hasta realizar la detección. (Gavrishev Mesheryakov 2022, p. 17)

Según Gavrishev (2022, p. 18) la CNN está formada por tres tipos de capas:

- **La capa convolucional**, representada en la ilustración 2-5, es la primera capa dentro de la red neuronal, los elementos que la componen son los datos de entrada, filtro y un mapa de características. Procesa las imágenes mediante la creación de una matriz de pixeles, para facilitar el procesamiento de la imagen se la transforma de RGB a escala de grises, el tamaño de la imagen en pixeles determina el número de neuronas que se ocupan en la capa.

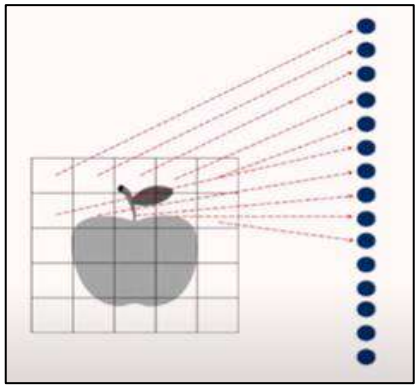


Ilustración 2-5: Capa convolucional

Realizado por: Romero Cristian, 2023

- **Capa de agrupación**, el objetivo de esta capa es reducir los datos de entrada pasándolos por un filtro. Al igual que la capa convolucional define características específicas de la imagen procesada, agrupándolas en agrupación máxima seleccionando los pixeles de mayor valor y agrupación media.
- **Capa completamente conectada**, también conocidas como capas ocultas son las capas finales de la red, en ellas se realizan la clasificación de los datos en dependencia de los rasgos extraídos en las capas anteriores, los resultados van de 0 a 1 en función de la confianza que tenga el modelo. Para que los resultados de salida se mantengan dentro de

un rango establecido se usan funciones de activación, la más usada es la función ReLU mostrada en la imagen 2-6.

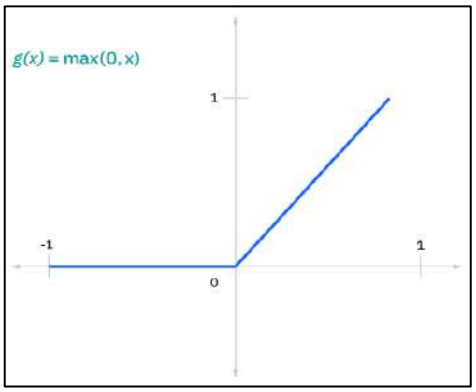


Ilustración 2-6: Función de activación ReLU

Fuente: (Gavrishev Mesheryakov 2022, p. 19)

2.5.2 Marcos de aprendizaje y arquitecturas de detección de objetos

Dentro del campo de la visión artificial, existen un sin número de algoritmos que permiten realizar tareas como detección, clasificación y localización de distintos objetos, ya sea en imágenes, videos o en tiempo real. Para facilitar el proceso de creación de estos algoritmos se han desarrollado marcos de aprendizaje o arquitecturas que permiten el entrenamiento de los modelos de redes neuronales. En la tabla 2-1 se indican los principales marcos de aprendizaje y arquitecturas de detección de objetos.(Gavrishev Mesheryakov 2022, p. 23)

Tabla 2-1: Marcos de aprendizaje y arquitecturas de detección de objetos *open source*

Herramienta	Tipo	Descripción
Tensor Flow	Marco de aprendizaje	Biblioteca de código abierto desarrollada por Google para el entrenamiento de modelos de aprendizaje profundo para inteligencia artificial.
PyTorch	Marco de aprendizaje	Biblioteca de código abierto desarrollada por Facebook para el entrenamiento de modelos de aprendizaje profundo para inteligencia artificial, su característica es la facilidad de uso.
Scikit - learn	Marco de aprendizaje	Biblioteca de código abierto desarrollada en Python para tareas de minería de datos, análisis estadísticos y demás.
Caffe2	Marco de aprendizaje	Biblioteca de código abierto desarrollada por Facebook para el entrenamiento de modelos de aprendizaje profundo para inteligencia artificial, su característica es la eficiencia de procesamiento.
Keras	Librerías	Biblioteca de código abierto para trabajar con Tensor Flow.

Yolo	Arquitectura	Familia de arquitecturas y modelos de detección previamente entrenados con el conjunto de datos COCO.

Fuente: (Gavrishev Mesheryakov 2022, p. 24)

Los modelos son archivos que almacenan el entrenamiento previamente realizado de una red neuronal, por este modelo son procesadas los cuadros de imágenes para la detección de objetos. La información es entregada en una tabla, en la misma se indica posición inicial, posición final, etiqueta del objeto y el nivel de confianza de la predicción. (Gavrishev Mesheryakov 2022, p. 23)

2.6 Sistema SCADA

El termino SCADA proviene del inglés *Supervisory Control And Data Acquisition* (Supervisión, control y adquisición de datos). Se puede definir como la tecnología que permite la recolección de datos, determinar el estatus de los equipos, el control y la supervisión de un proceso a distancia. Estos sistemas cumplen dos funciones principales: la adquisición de los datos y el control supervisado del proceso y los equipos. La información obtenida se puede recolectar de una gran cantidad de fuentes en procesos productivos complejos o dispersos geográficamente. Toda la información obtenida por el sistema SCADA debe ser automáticamente archivada. (Thomas y McDonald 2015, p. 5)

Los SCADA son sistemas de control de lazo cerrado o retroalimentados como se observa en la ilustración 2-7, se vigila constantemente las variables del proceso y el cumplimiento de las órdenes. Si es necesario, se efectúan acciones correctivas para mantener el proceso dentro de los límites permisibles. (Rodríguez 2013, p. 10)

Ilustración 2-7: Lazo de control cerrado

Realizado por: Romero. Cristian, 2023

2.6.1 Objetivos de los sistemas SCADA

Según Rodríguez (2013, p. 17-18) se busca desarrollar los sistemas SCADA con el fin de obtener los siguientes objetivos:

- **Economizar:** al realizar revisiones en el proceso de manera remota, evitando el uso de un operario.
- **Accesibilidad:** al obtener los datos de las variables del proceso, además de permitir modificar los parámetros de funcionamiento del mismo.
- **Mantenimiento:** En el sistema se pueden programar las fechas para realizar los mantenimientos preventivos, también se pueden almacenar los datos históricos de fallas.
- **Ergonomía:** con el fin que la relación operario-máquina sea lo más beneficiosa para el operario.
- **Gestión:** permite tratar los datos obtenidos mediante el uso de herramientas estadísticas, gráficos y tabuladores con el fin de analizar posibles mejoras del rendimiento.
- **Flexibilidad:** permite realizar modificaciones a los parámetros del proceso sin necesidad de realizar cambios físicos.
- **Conectividad:** permite la comunicación entre equipos de diferentes proveedores, evitando así la incompatibilidad y la falta de información.

2.6.2 Funciones de los sistemas SCADA

Rodríguez (2013, p. 18-19) menciona que las principales funciones y utilidades generadas por los sistemas SCADA son las siguientes:

- **Monitorización:** se puede obtener información de las variables del proceso en tiempo real y a distancia permitiendo una constante vigilancia por parte del operador.
- **Supervisión:** esta herramienta permite la gestión de información que facilita la toma de decisiones, además con el desarrollo de programas que supervisen y modifiquen ciertos parámetros se puede evitar la supervisión humana en algunos casos.
- **Adquisición de datos:** permite almacenar la información obtenida mediante el uso de base de datos, con el fin de tratarlos con el uso de herramientas estadísticas ¿.
- **Visualización de señales (alarmas y eventos):** presenta de manera visual eventos en tiempo real que se hayan generado en el proceso permitiendo poner en marcha acciones correctivas.
- **Mando:** permite a los operadores cambiar parámetros del proceso desde un computador.

- **Grabación de acciones:** permite almacenar varias configuraciones y parámetros de un proceso para cambiarlos de manera rápida y sencilla.
- **Garantiza la seguridad de los datos:** debe existir protección de los datos de influencias indeseadas sean intencionales o no.
- **Garantiza la seguridad en los accesos:** restringe el acceso a usuarios no autorizados, también registra los accesos y los cambios realizados.
- **Posibilidad de programación numérica:** permite realizar cálculos aritméticos complejos.

2.6.3 Criterios de diseño del sistema SCADA

Los sistemas automatizados SCADA deben cumplir con ciertos requerimientos, Ochoa y Velecela (2016, p. 55) mencionan las siguientes:

- Determinar la cantidad de variables que forman parte del proceso, para su almacenamiento en bases de datos, lo que permite su futuro procesamiento y gestión.
- Los componentes que forman parte del proceso, los sensores que toman la información en el proceso y los actuadores que ejecutan cada una de las funciones que generan el proceso.
- Número de variables requeridas en tiempo real, en el momento que se realice el proceso es necesario el control de ciertas variables importantes, estas variables se deben visualiza en tiempo real mediante el uso de la HMI.

2.7 Entorno SCADA

La producción no está aislada de los demás departamentos en una empresa, las organizaciones empresariales engloban todos sus departamentos para mejorar la gestión, calidad y productividad. Los sistemas SCADA permiten generar flujos de información en la empresa, los mismo que se integran en una pirámide de automatización que se conoce como CIM (*Computer Integrated Manufacturing* por sus siglas en inglés) manufactura integrada por computadora. En la ilustración 2-8 se puede observar gráficamente la pirámide de automatización. (Rodríguez 2013, p. 21)

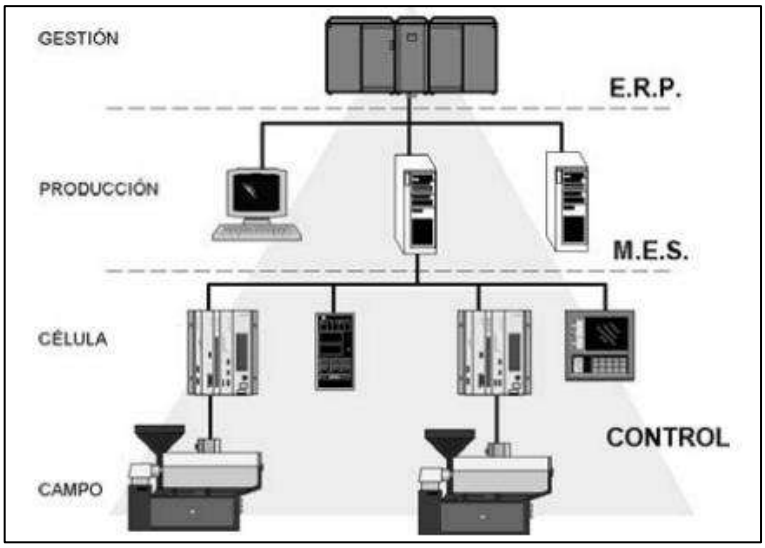


Ilustración 2-8: Pirámide de la automatización

Fuente: (Rodríguez 2013, p. 21)

2.8 *Hardware* SCADA

Los sistemas SCADA constan de ciertos elementos físicos para su correcto funcionamiento, los mismos que cumplen funciones importantes, la ilustración 2-9 indican gráficamente los elementos de un sistema SCADA. El sistema está formado por unidades terminales remotas o RTU que recolectan la información desde el campo por medio del uso de sensores, además permiten dar órdenes a los actuadores para realizar una función específica. Una vez recolectado los datos se los envía a una unidad terminal maestra que puede mostrar los datos y permite el control remoto del proceso. (Gordon y Deon 2004, p. 3)

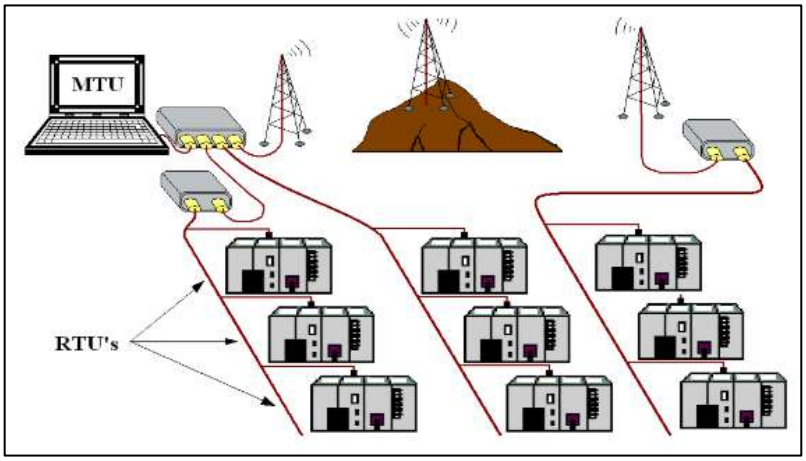


Ilustración 2-9: Componentes del sistema SCADA

Fuente: (Aguirre 2013, p. 8)

2.8.1 Unidad terminal remota (RTU)

Las RTU (*Remote Terminal Units* por sus siglas en inglés) son elementos físicos que se encuentran distribuidos en sectores estratégicos de los procesos a automatizar. Se podría decir que son los oídos, ojos y manos de los sistemas SCADA. El trabajo de los RTUs es recolectar la información desde los diferentes dispositivos de campo (sensores), procesar esa información y transmitir solo los datos necesarios a la estación maestra. De igual manera, reciben señales de control desde la unidad maestra y la distribuyen a los dispositivos de acción en el campo (actuadores). (Thomas y McDonald 2015, p. 5)

2.8.2 Unidad terminal maestra (MTU)

La MTU (*Master Terminal Unit* por sus siglas en inglés) son principalmente computadoras que se comunican con las RTUs en campo, su función principal es la de monitorizar y controlar el proceso o sistema mediante la recepción de información. Reciben datos y envían órdenes a las RTUs, además envían los datos para ser almacenados en una base de datos. (Thomas y McDonald 2015, p. 6)

2.8.3 Interface humano-máquina (HMI)

(*Human-Machine Interface* por sus siglas en inglés) como su nombre lo indica, se refiere al elemento físico, en este caso pantalla que permite la comunicación entre la MTU y el operario del proceso. La HMI nos muestra el proceso y alarmas de eventos que sucedan en el proceso mediante el uso de elementos gráficos. (Thomas y McDonald 2015, p. 6)

2.8.4 Elementos de campo

Son los distintos elementos que se encuentran dentro del proceso, los cuales permiten determinar el estado del mismo. Estos dispositivos permiten recolectar la información necesaria para que el sistema pueda realizar el control y supervisión. En resumen, los elementos de campo son los sensores que entregan los datos de las condiciones del sistema y actuadores que realizan acciones sobre el proceso. Ejemplos de sensores tenemos, los sensores fotovoltaicos para determinar el paso de un objeto, sensores capacitivos de proximidad que al igual que los anteriores determinan el paso de un objeto. Entre los ejemplos de los actuadores tenemos los cilindros neumáticos de

simple y de doble efecto, usados para realizar acciones de movimiento en los procesos industriales. (Moya 2009, p. 12)

2.9 *Software* SCADA

2.9.1 Comunicación

Existen diferentes métodos de comunicación entre equipos electrónicos, estos métodos se denominan protocolos de comunicación. La velocidad de comunicación entre dispositivos depende del propósito y la importancia, en función de estos parámetros se selecciona el protocolo de comunicación adecuado. En los sistemas SCADA el método de comunicación usado se denomina "maestro-esclavo".

El proceso de comunicación se da mediante un requerimiento de la MTU a la RTU, se dan instrucciones, se pide información del estado del proceso y se ordena a la RTU responder. La MTU siempre está a la espera información, una vez la RTU envía la respuesta la MTU finaliza la comunicación. La comunicación se da de la misma manera en cada una de las distintas RTUs que se encuentren en el sistema SCADA. La RTU no inicia la comunicación, esto es tarea de la MTU. (Boyer 2004, p. 53)

2.9.1.1 Topología de comunicación

Para el correcto funcionamiento de un sistema SCADA los dispositivos que lo componen deben estar conectados y comunicados entres si, el tipo de conexión dependerá de las necesidades del proceso. La topología usada es física por el tipo de conexión de los elementos y lógica por la manera de transmitir la información en la red de comunicación. (Thomas y McDonald 2015, p. 79)

Según Thomas y McDonald (2015, p. 79-81) los principales tipos de topologías de comunicación SCADA son:

- **Punto a punto o multipunto:** Los dispositivos de un sistema pueden estar conectados de distintas maneras, si existen dos dispositivos estos pueden estar conectados juntos mediante el uso de una línea de comunicación esta conexión se denomina punto a punto. La conexión multipunto se desarrolla con más de dos dispositivos conectados a una misma línea, los dispositivos usan la línea de comunicación simultáneamente. En la ilustración 2-10 se detalla la topología.

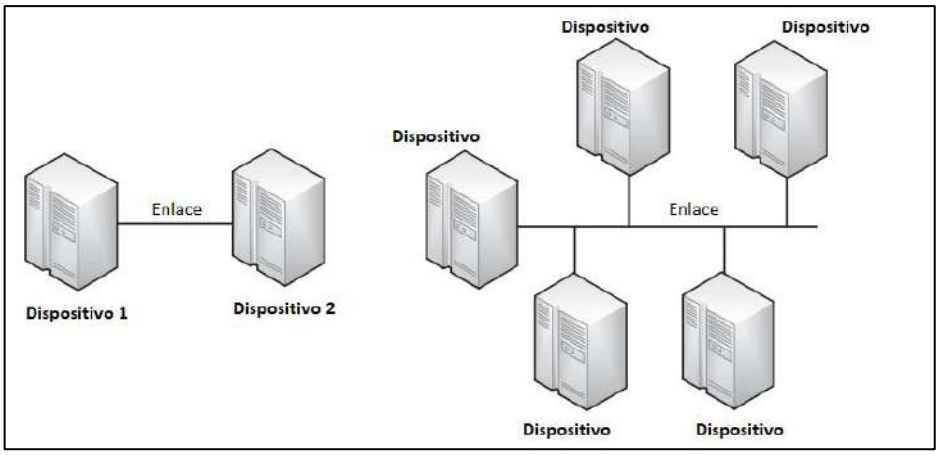


Ilustración 2-10: Topología de comunicación punto a punto y multipunto

Fuente: (Thomas y McDonald 2015, p. 80)

- **Bus:** Esta topología de comunicación está conformado por nodos, es flexible y se puede conectar de manera simple y redundante. Se puede adaptar a cualquier tipo de comunicación y es robusta ya que cualquier fallo en un nodo no afecta al bus. Aunque flexible y muy usada tiene ciertas desventajas, ya que los errores en el bus son difíciles de identificar, los mensajes se pueden perder si no son recibidos por un nodo y un nodo puede no enviar un mensaje a tiempo en momentos de exceso de tráfico en el bus.
- **Anillo:** Forma un ciclo cerrado, todos los nodos de la red forman un anillo incluido el maestro, la comunicación se da en una sola dirección pasando de nodo en nodo hasta llegar al maestro. Los principales problemas en esta topología son el incremento o reducción de nodos y si la comunicación se detiene es difícil su detección.
- **Estrella:** Como su nombre lo indica, la red forma una estrella y en el centro se encuentra conectado el terminal maestro, su configuración, mantenimiento, aumento y disminución de nodos es sencilla. Su mayor desventaja es si existe una falla en la termina maestra, ya que esto genera una falla en toda la red.
- **Malla:** Se considera una mejora de la topología anillo ya que su sistema es redundante, su uso depende del requerimiento de la red. Al ser un sistema redundante se pueden evitar los problemas de comunicación existentes.
- **Flujo de datos:** Esta topología se puede dar de manera simple o doble. La comunicación en la topología de flujo de datos simple se da en una sola vía, un dispositivo envía y otro solo recibe la información, una vez liberada la red pueden invertirse los papeles. Pero en el flujo de datos doble los dos dispositivos pueden transmitir y recibir información al mismo tiempo.

2.9.1.2 Técnicas de comunicación de los sistemas SCADA

Para la jerarquización y comunicación entre dispositivos en un sistema SCADA es importante tener en cuenta las técnicas de comunicación en los sistemas SCADA, Thomas y McDonald (2015, p. 81-82) indican que las principales técnicas de comunicación son:

- **Maestro – Esclavo:** Esta técnica de comunicación consta de dos clases de dispositivos. El dispositivo maestro que controla los tiempos y gestiona la comunicación haciendo requerimientos y recibiendo la información por parte de los dispositivos esclavos. El dispositivo esclavo que controla los procesos en el campo y recibe las órdenes del dispositivo maestro. El uso de recursos es mínimo y el dispositivo maestro tiene control total sobre la inicialización de la comunicación en los dispositivos esclavos.
- **Colega a colega:** Permite que cualquier dispositivo de la red inicie la comunicación, aunque la estación maestra recibe la mayoría de la información. Al aumentar el número de nodos el rendimiento disminuye.
- **Multi – Colega:** A diferencia de la técnica anterior la comunicación se da por dos vías y los dispositivos pueden hacer requerimientos a varios nodos en la red.

2.9.2 Datos

Dentro de los procesos industriales existen distintas clases de variables que entran en juego, algunas de esas variables son: presión, caudal, cantidad de producto, peso, volumen. Para la adquisición de los datos de estas variables se usan los sensores, mismos que transmiten las señales generadas a las RTUs.

La señal transmitida puede ser analógica o digital, el caso que la señal sea digital no es necesaria la conversión. Las señales analógicas por el contrario deben ser tratadas y convertidas a señales digitales, para su transmisión mediante el uso del sistema de comunicación del SCADA.(Thomas y McDonald 2015, p. 82)

2.9.2.1 Componentes de la comunicación de datos

Según Thomas y McDonald (2015, p. 83) los componentes presentes en la comunicación de datos en un sistema SCADA son:

- **Mensaje:** Son los datos o información a comunicar.
- **Remitente:** Es el dispositivo que envía el mensaje, principalmente es una RTU.
- **Receptor:** Es el dispositivo destino del mensaje, principalmente es una MTU.

- **Medio:** Elemento físico por donde se transmite el mensaje desde el remitente al receptor.
- **Protocolo:** Es una serie de convenciones que controlan los datos, define que, como y cuando se comunica la información.

2.9.2.2 Transmisión de datos digitales

Para la transmisión de datos digitales entre dispositivos mediante el uso de elementos físicos, primero se debe transformar los datos digitales en señales digitales. El proceso de tratamiento de datos se llama codificación, este proceso es realizado por el microcontrolador que trabaja como RTU. Los datos digitales trabajan con el voltaje de referencia del microcontrolador o dispositivo RTU. En la ilustración 2-11 se detalla el funcionamiento de los datos digitales en una válvula. (Thomas y McDonald 2015, p. 83)

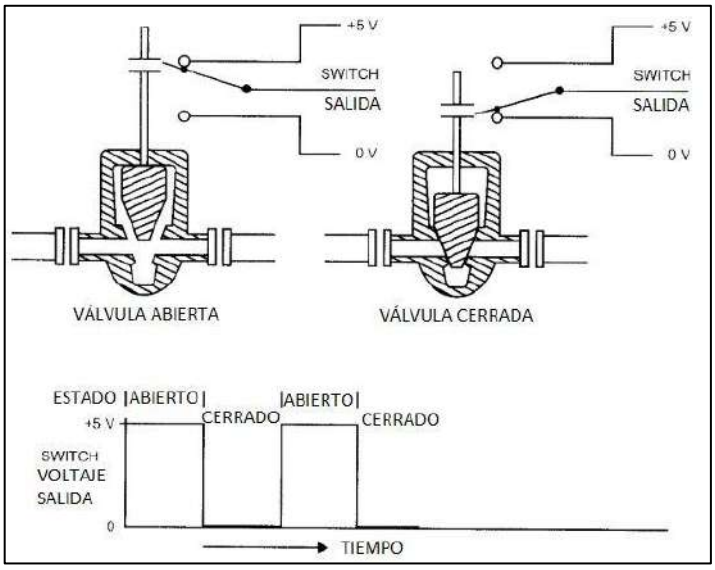


Ilustración 2-11: Datos digitales

Fuente: (Boyer 2004, p. 54)

2.9.2.3 Transmisión de datos analógicos

Los datos analógicos son valores infinitos que varían en función del tiempo. Estos valores representan magnitudes físicas presentes en los procesos productivos (volumen, caudal, peso, tiempo, distancia y demás), para su lectura y posterior tratamiento se las representa en magnitudes eléctricas, principalmente voltaje. (Thomas y McDonald 2015, p. 14)

Los sensores transmiten los datos a la RTU que se encarga de la recepción, lectura y posterior tratamiento de los datos analógicos a datos digitales. En la ilustración 2-12 se representa la posición de una válvula en función de una señal analógica.

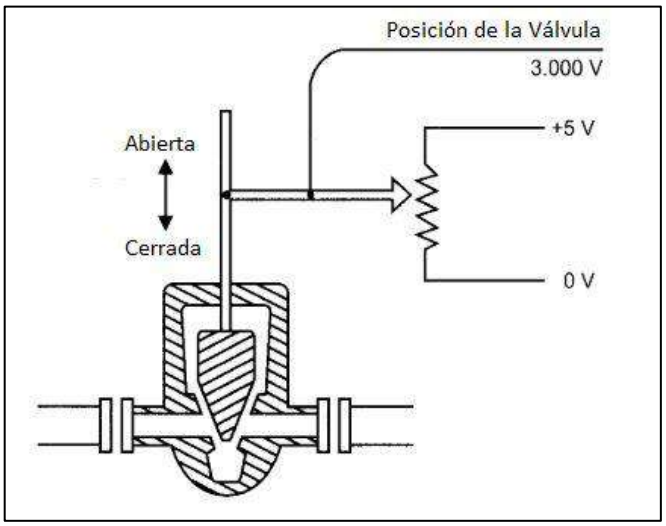


Ilustración 2-12: Datos analógicos

Fuente: (Boyer 2004, p. 56)

2.9.3 Interface gráfica

Se define como los elementos gráficos para la visualización del proceso industrial, mostrando el estado del mismo con el uso de imágenes y texto. Permite el control y supervisión de los procesos de una manera fácil e intuitiva. (Rodríguez 2013, p. 49)

Los programas para la creación de la interfaz gráfica contienen librerías que facilitan su desarrollo permitiendo al desarrollador generar distintas interfaces para distintos procesos. La ilustración 2-13 muestra la interfaz gráfica de un proceso de monitoreo y control de líquidos.

Ilustración 2-13: Interfaz gráfica para el control de líquidos

Fuente: (Suarez 2015, p. 66)

2.9.3.1 Alarmas y eventos

Los sistemas SCADA están en un proceso de constante vigilancia de los procesos productivos. Esta vigilancia consta de la comparación de los parámetros que influyen en el proceso y pueden generar sucesos no deseados, generando problemas de funcionamiento. Estos sucesos requieren de la intervención de un operario para evitar el paro del proceso. (Rodríguez 2013, p. 51)

Existen otros sucesos o eventos denominados normales, son eventos que indican el estado de funcionamiento del proceso, no se requiere intervención del operario y se registran de forma automática en una base de datos. La ilustración 2-14 indica un sistema de muestra y almacenamiento de alarmas y eventos desarrollado en un sistema SCADA. (Rodríguez 2013, p. 51)

Ilustración 2-14: Pantalla de alarmas

Fuente: (Rodríguez 2013, p. 51)

2.9.4 Base de datos

La información obtenida dentro de un proceso productivo debe ser tomada, tratada y almacena de alguna forma, en este punto son necesarias las bases de datos, misma que permiten guardar información de distinto tipo y origen de manera organizada y jerarquizada para su futura consulta y procesamiento. (Rodríguez 2013, p. 46)

Entre los elementos que conforman una base de datos tenemos:

- **Dato:** Es el elemento más pequeño, formado por bits o bytes, por sí solo no tiene ningún significado.
- **Campo:** Es el espacio de almacenamiento de un tipo específico de datos.
- **Registro:** Colección de datos iguales o diferentes que se encuentran relacionados.
- **Archivo:** Es una colección de registros relacionados que cuentan con una estructura homogénea.

CAPÍTULO III

3 MARCO METODOLÓGICO

Para el desarrollo e implementación del sistema SCADA es necesario determinar ciertas especificaciones y procedimientos. Este capítulo está enfocado en describir la metodología que se siguió para el desarrollo de la integración de los dos procesos industriales en un sistema SCADA ilustración 3-1, indicando los cambios realizados, características de los elementos usados, planos de conexión y la programación del sistema. El desarrollo del sistema está enfocado en el uso de herramientas de *software* y *hardware open source* o de acceso libre.

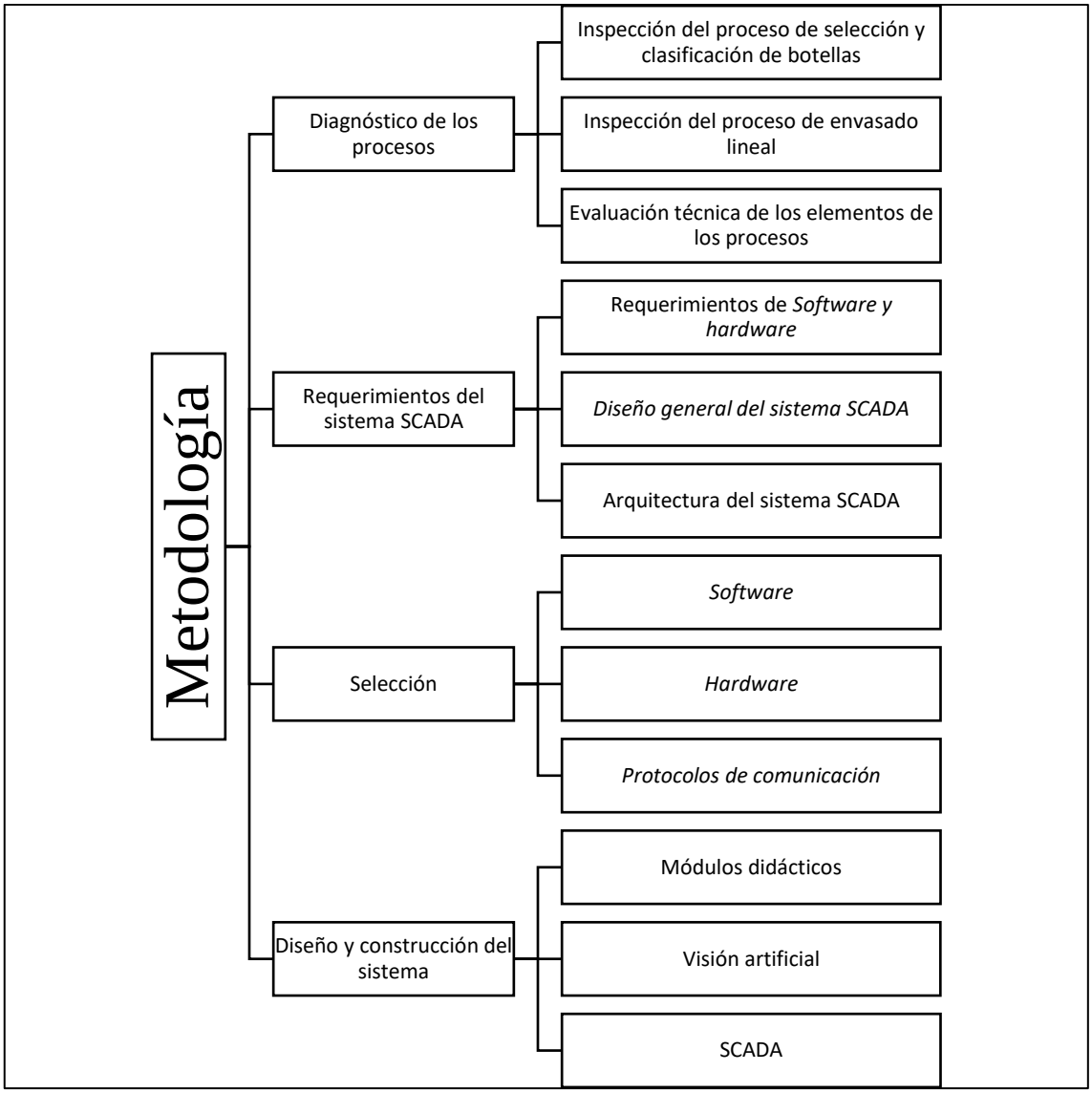


Ilustración 3-1: Diagrama de metodología para el diseño del sistema SCADA

Realizado por: Romero, Cristian, 2023

3.1 Diagnóstico de los procesos industriales

Como primer punto, se requiere realizar el levantamiento de información referente a los dos procesos industriales a integrar en el sistema. Revisar el estado de cada uno de los componentes de los procesos, así como los diagramas de conexión y su funcionamiento.

3.1.1 Inspección del proceso de selección y clasificación de botellas

Es un sistema experimental de procesamiento de materiales a pequeña escala, construido en función de parámetros técnicos que se encuentra en la Escuela de Ingeniería Industrial. A lo largo, desde su construcción el proceso ha sufrido cambios en la manera en que procesa y selecciona los materiales.

En su desarrollo ilustración 3-2 el proceso fue diseñada en base a dos sistemas, mando y potencia. El sistema de potencia encargado de ejecutar las acciones físicas mediante el uso de actuadores eléctricos y neumáticos. El sistema de mando y control fue desarrollado en torno a un PLC o Controlador Lógico Programable, equipo que recibe información desde los sensores en el proceso para poder generar acciones mediante el control de los actuadores.

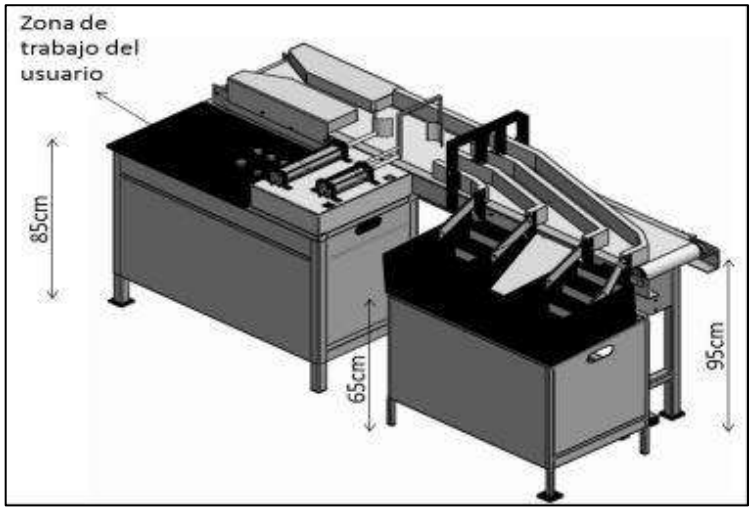


Ilustración 3-2: Proceso de selección y clasificación de botellas

Fuente: (Montes y Plasencia 2015, p. 30)

El siguiente cambio fue la implementación de un sistema de visión artificial. Para poder desarrollar el nuevo sistema de control fueron necesarios cambios importantes, se reemplazó el PLC de tipo industrial por una placa programable Arduino. Los sensores fueron removidos y en su lugar se usa una cámara para la toma de imágenes de los elementos que ingresan. El procesamiento de las imágenes ilustración 3-3 es realizado en una computadora mediante el uso del *software* Matlab, enviando una señal a la placa programable Arduino para que active los

actuadores neumáticos que no han sido modificados. El sistema de potencia no ha sido cambiado ya que se puede acoplar al nuevo diseño. Los planos de los sistemas que forman parte del proceso se pueden ver en el anexo A.

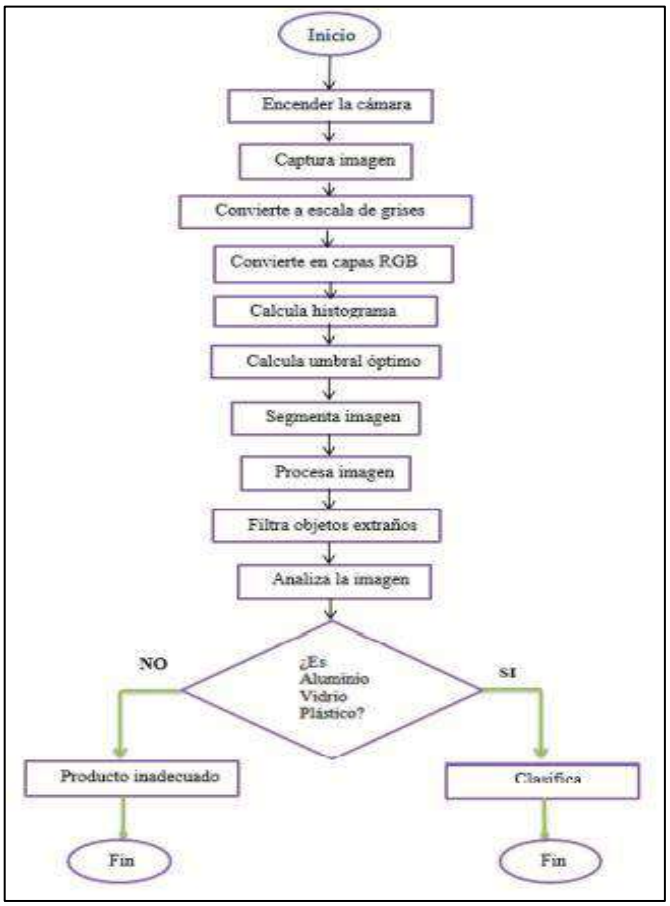


Ilustración 3-3: Diagrama de flujo del procesamiento de imágenes

Fuente: (Tierra y Galarza 2017, p. 57)

3.1.1.1 Revisión de los sistemas del proceso de selección y clasificación de botellas

Para poder determinar el estado actual del proceso es necesario hacer una evaluación de los sistemas por separado, inspeccionar los sistemas y cada uno de sus componentes. El proceso de selección y clasificación de botellas ilustración 3-4 consta de tres diferentes tipos de sistemas: mando, potencia, neumáticos, ver los planos en el anexo A.

Ilustración 3-4: Proceso de selección y clasificación de botellas

Realizado por: Romero, Cristian, 2023

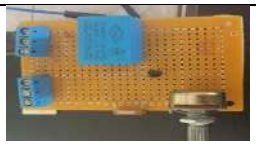
El sistema de mando que se encuentra tablero de control ilustración 3-5, permite controlar el proceso de selección de botellas mediante un sensor de ingreso de objetos y la toma de imágenes por parte de una cámara USB, el procesamiento de las imágenes se realiza en una computadora mediante un programa de visión artificial creado en el *software* MATLAB, una vez realizado el procesamiento se envía un comando a la placa microcontrolador ARDUINO para la activación de los actuadores que permiten la clasificación de las botellas en los diferentes contenedores.

Ilustración 3-5: Tablero de mando y control

Realizado por: Romero, Cristian, 2023

En la tabla 3-1 se detallan cada uno de los elementos que son parte del sistema de mando y control del proceso. El diagrama de conexión del sistema y del circuito fotovoltaico se encuentra en el anexo A.

Tabla 3-1: Elementos de control del proceso de clasificación de botellas

Cantidad	Nombre	Características	Imagen
1	Pulsador de marcha	-CAMSCO -PB-XAM	
1	Pulsador de paro	-CAMSCO -PB-XAM	
1	Luz piloto	-STECK -S-LDS110 -Alimentación: 110VCA	
1	Arduino	-Mega 2560	
1	Modulo relé	-Módulo de 4 relés	
1	Fuente	-TPS-2405 -Alimentación: 110VCA -Salida = 24VCD	
1	Relé electromagnético	-FATO -MK2P-I -Bobina: 24VCD -8 pines	
1	Circuito fotovoltaico	-Foto resistencia -Relé -Potenciómetro 10K -Bornera -Diodo 2N3904	
1	Cámara	-Logitech -C170 -30 FPS -5 megapíxeles	

Realizado por: Romero, Cristian, 2023

El sistema de potencia permite la activación del motor de la banda, de esta manera se pueden transportar las botellas desde el ingreso, al área de toma de imágenes y hasta la sección de clasificación de botellas, además se incluye el sistema de iluminación. El sistema de potencia es alimentado con corriente alterna y controlado por el sistema de mando y control. En la ilustración 3-6 se muestra el tablero y el motor que forman parte del sistema de potencia.

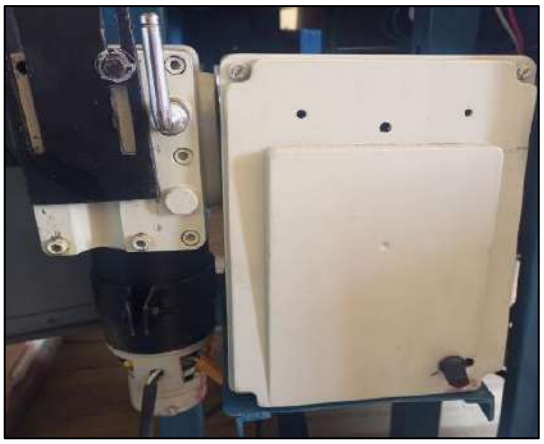
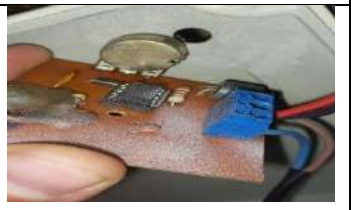


Ilustración 3-6: Tablero del sistema de potencia y motor reductor

Realizado por: Romero, Cristian, 2023

En la tabla 3-2 se detallan cada uno de los elementos que intervienen en el sistema de potencia del proceso de selección y clasificación de botellas. El diagrama de conexión del sistema y del circuito variador de velocidad del motor se encuentra en el anexo A

Tabla 3-2: Elementos de potencia del proceso de clasificación de botellas

Cantidad	Nombre	Características	Imagen
1	Motor reductor	-Dongchang -Reductor sin fin corona -Alimentación: 24VCD	
1	Transformador	-Laytor -TRI-50-C -Alimentación: 110VCA -Salida: 24VCD	
1	Disyuntor	-SASSIN -3SB1-63N	
1	Disyuntor	-CHNT -NB1-63H	
1	Variador de velocidad del motor-reductor	-Potenciómetro 100K -Diodo 6 A 8 MIC -Circuito integrado 555 -Transistor IRFZ44N -Capacitores cerámicos -Resistencias -Borneras	

3	Luces	-Alimentación: 110VCA	

Realizado por: Romero, Cristian, 2023

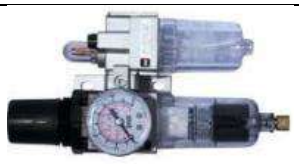
El sistema neumático está conformado por los actuadores neumáticos, controlados por las electroválvulas. Desde el sistema de mando se envía una señal para activar las electroválvulas las mismas que cambian de estado y permiten el paso del aire comprimido a uno de los dos compartimentos del cilindro de doble efecto ilustración 3-7 extendiendo el vástago para realizar la clasificación de botellas, una vez finalizado el proceso de clasificación la electroválvula regresa a su estado inicial y permite el paso de aire para que el vástago del cilindro regrese a la posición inicial.

Ilustración 3-7: Sistema neumático, cilindros de doble efecto

Realizado por: Romero, Cristian, 2023

En la tabla 3-3 se detallan cada uno de los elementos que conforman el sistema neumático. El diagrama de conexión del sistema se encuentra en el anexo A.

Tabla 3-3: Elementos neumáticos del proceso de clasificación de botellas

Cantidad	Nombre	Características	Imagen
1	FRL (Filtro-Regulador-Lubricado)	-NAN -AW2000-02	
2	Electroválvula	-Checkman -4V210-08 -Monoestable -Alimentación: 24VCD	

1	Cilindro	-Carrera: 140mm -Doble efecto	
1	Cilindro	-Carrera: 300mm -Doble efecto	

Realizado por: Romero, Cristian, 2023

3.1.2 Inspección del proceso de envasado lineal

Es un proceso diseñado específicamente para ser utilizada de manera didáctica en el laboratorio de automatización de la Escuela de Ingeniería Industrial, ilustración 3-8. El objetivo principal es mejorar el aprendizaje técnico de los estudiantes en el área de automatización y control. El diseño y construcción del proceso de envasado se realizó en función de parámetros técnicos, usando elementos utilizados en la industria.

Una vez encendido el proceso se deben ingresar tres envases plásticos los mismo que ingresan en el área de envasado, ya que el envasado se realiza en tres válvulas dosificadoras, en el siguiente proceso los envases llenos ingresan al área de tapado, mediante un sistema simple a base de gravedad se colocan las tapas en los envases, a continuación, un cilindro neumático presiona las tapas, tapando los envases y finalizando el proceso. El proceso consta de cuatro diferentes sistemas: mando y control, potencia, neumático e hidráulico.

Ilustración 3-8: Proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

3.1.2.1 Revisión de los sistemas del proceso de envasado lineal

El sistema de mando y control que se encuentra ubicado en el tablero eléctrico ilustración 3-9 está encargado de la recepción y procesamiento de información desde los sensores, activando los actuadores que permiten realizar las acciones correspondientes. En este caso específico el proceso consta de dos sensores capacitivos que permiten detectar los envases en el área de envasado y el área de tapado. Una vez detectados los envases en cualquiera de las dos áreas el sistema procesa esa información y envía señales a los actuadores activándose en función a una secuencia establecida. El elemento que controla este proceso es un PLC o Control Lógico Programable, mismo elemento que ha sido cambiado por versiones más recientes, el proceso fue diseñado y construido con un PLC S7-200 y posteriormente se cambió a la versión S7-1200.

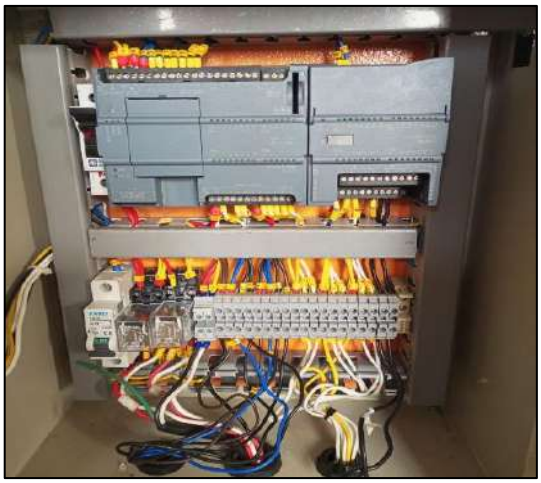


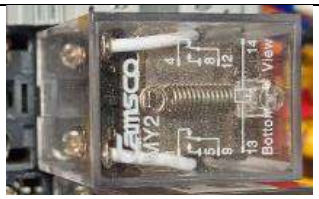
Ilustración 3-9: Tablero de mando y control

Realizado por: Romero, Cristian, 2023

En la tabla 3-4 se detallan los elementos que forman parte del sistema de mando y control del proceso de envasado línea. Los planos de conexión se encuentran en el anexo B.

Tabla 3-4: Elementos de mando y control del proceso de envasado lineal

Cantidad	Nombre	Características	Imagen
1	PLC (Programador Lógico Programable)	-Siemens -S7-1200 -CPU 1214C -AC/DC/RLY -Alimentación: 24 VCD	

1	Módulo de ampliación	-Siemens -SIMATIC S7-1200 -E/S Digitales -SM1223 -Alimentación: 24VCD	
1	Pulsador de paro	-CAMSCO -PB-XAM	
2	Luz piloto	-STECK -S-LDS110 -Alimentación: 110VCA	
1	Pulsador de marcha	-CAMSCO -PB-XAM	
2	Relé electromagnético	-CAMSCO -MY2 -Bobina: 110VCA -14 pines	
2	Sensores capacitivos	-TOKY -TC-18P8C -Alimentación: 10-30 VCD	

Realizado por: Romero, Cristian, 2023

El sistema de potencia se encarga de activar el motor ilustración 3-10 que permite el movimiento de la banda transportadora y la bomba hidráulica para el llenado de los envases. Son elementos vitales para el proceso, primero el motor que impulsa la banda permite el transporte de los envases por las distintas áreas del proceso, la bomba hidráulica permite el llenado de los envases, cumpliendo con el propósito, el envasado de líquidos.

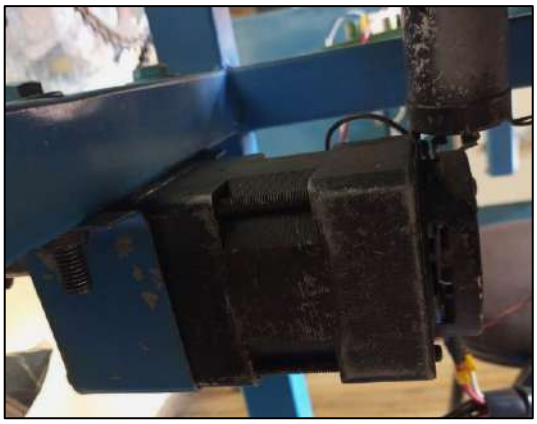


Ilustración 3-10: Motor reductor del sistema de potencia

Realizado por: Romero, Cristian, 2023

En la tabla 3-5 se detallan los elementos que forman parte del sistema de potencia, incluyendo la bomba hidráulica. En el anexo B se encuentran los planos de conexión del sistema.

Tabla 3-5: Elementos de potencia del proceso de envasado lineal

Cantidad	Nombre	Características	Imagen
1	Motor reductor	-Alimentación: 110VCA -0.87 A -3400RPM	
1	Bomba hidráulica	-HYUNDAI -HYQ860 -Alimentación: 110VCA -3450RPM	
1	Disyuntor	-FATO -C65N -C10	
1	Porta fusibles	-32A -380VCA	

Realizado por: Romero, Cristian, 2023

El sistema neumático es parte principal de los actuadores que ejecutan las distintas acciones en el proceso. Compuesto por electroválvulas ilustración 3-11 y cilindros de doble efecto, reciben la señal desde el sistema de mando ejecutando acciones físicas. Para su funcionamiento es necesario el uso de aire comprimido y electricidad para la activación de las electroválvulas.

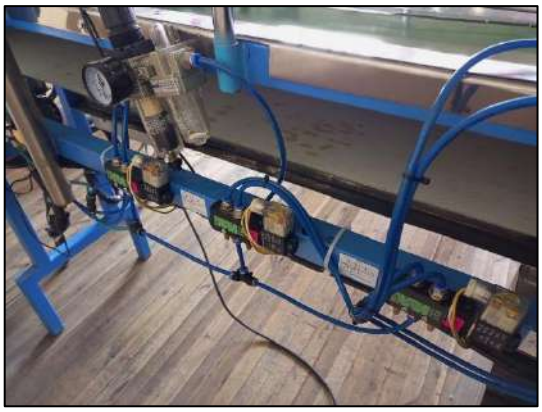
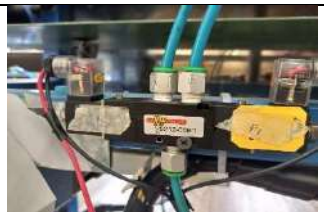


Ilustración 3-11: Electroválvulas del sistema neumático

Realizado por: Romero, Cristian, 2023

En la tabla 3-6 se detallan todos los elementos que forman parte del sistema neumático del proceso de envasado lineal. En el anexo B se muestra el diagrama de conexión del sistema.

Tabla 3-6: Elementos neumáticos del proceso de envasado lineal

Cantidad	Nombre	Características	Imagen
1	FRL (Filtro-Regulador-Lubricado)	-NAM -AL2000-02 -Pmax 10MPa	
3	Electroválvula	-ASCO -QTV52-08 -Monoestable -Alimentación: 110VCA	
1	Electroválvula	-NTP -V5212-06E1 -Biestable -Alimentación: 110VCA	
2	Cilindro de detención	-Doble efecto -Carrera: 50mm -D vástago: 5mm -D émbolo: 16mm	
1	Cilindro de llenado	-Doble efecto -Carrera: 100mm -D vástago: 5mm -D émbolo: 16mm	

1	Cilindro de tapado	-Doble efecto -Carrera: 60mm -D vástago: 12mm -D émbolo: 40mm	

Realizado por: Romero, Cristian, 2023

El sistema hidráulico permite el envasado de las botellas. El líquido a envasar se traslada desde el depósito en la parte inferior hasta las válvulas dosificadoras, las cuales permiten el paso del fluido hasta las botellas. El sistema hidráulico es controlado por el sistema de mando y para su correcto funcionamiento es necesario el uso de electricidad para la bomba hidráulica y las válvulas dosificadoras ilustración 3-12.

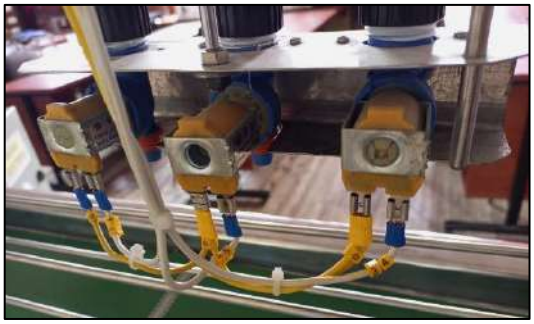


Ilustración 13-2: Válvulas dosificadoras del sistema hidráulico

Realizado por: Romero, Cristian, 2023

En la tabla 3-7 se detallan los elementos que forman parte del sistema hidráulico del proceso de envasado lineal. En el anexo B se muestra el diagrama de conexión del sistema.

Tabla 3-7: Elementos hidráulicos del proceso de envasado lineal

Cantidad	Nombre	Características	Imagen
3	Válvulas dosificadoras	*-Water Inlet Valve* -Alimetación: 110VCA	
1	Bomba hidráulica	-HYUNDAI -HYQ860 -Alimentación: 110VCA -3450RPM	

Realizado por: Romero, Cristian, 2023

3.1.3 Evaluación técnica de los elementos de los procesos industriales

Mediante la revisión visual y de funcionamiento de cada uno de los elementos que forman parte de los sistemas de los procesos industriales experimentales se logró hacer un análisis cualitativo, clasificando a los componentes en dos categorías aceptable y no aceptable. Los resultados son presentados en la tabla 3-8 para el proceso de selección y clasificación de botellas y en la tabla 3-9 para el proceso de envasado lineal.

Tabla 3-8: Estado de los elementos del proceso de clasificación de botellas

Elemento	Evaluación	Estado	Estado total
Pulsador de marcha	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Pulsador de paro	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Luz piloto	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Arduino	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Relé	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Fuente	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Relé electromagnético	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Circuito fotovoltaico	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Cámara	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Motor reductor	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Transformador	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Disyuntor Q1	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Disyuntor Q2	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Variador de velocidad	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Luces	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
FRL (Filtro-Regulador-Lubricado)	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	

Electroválvula	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Cilindro	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Cilindro	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	

Realizado por: Romero, Cristian, 2023

Tabla 3-9: Estado de los elementos del proceso de envasado lineal

Elemento	Evaluación	Estado	Estado total
PLC (Programador Lógico Programable)	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Módulo de ampliación	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Pulsador de paro	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Luz piloto	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Pulsador de marcha	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Relé electromagnético	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Sensores capacitivos	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Motor reductor	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Bomba hidráulica	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Disyuntor	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Porta fusibles	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
FRL (Filtro-Regulador-Lubricado)	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Electroválvula monoestable	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Electroválvula biestable	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Cilindro de detención	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Cilindro de llenado	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	

Cilindro de tapado	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	
Válvulas dosificadoras	Estado físico	Aceptable	Aceptable
	Estado funcional	Aceptable	

Realizado por: Romero, Cristian, 2023

3.2 Requerimientos del sistema SCADA

Para el desarrollo del tema propuesto en el trabajo de integración curricular, es necesario cumplir con determinadas exigencias. Requerimientos fundamentados teóricamente y que son necesarios para el correcto funcionamiento del sistema SCADA.

3.2.1 Requerimientos de software y hardware

3.2.1.1 Requerimientos de software para el control visual

En el procesamiento de imágenes en la selección y clasificación de botellas es necesario que se cumplan requerimientos definidos en función de las necesidades del proyecto. Los requerimientos se indican en la tabla 3-10.

Tabla 3-10: Requerimientos de *software* para el procesamiento de imágenes

N°	Requerimientos
1	Tipo de licencia *open source*
2	Protocolos de comunicación serial
3	*Software* con librerías necesarias para procesamiento de imágenes
4	Librerías para la gestión de archivos y base de datos
5	Comunicación con bases de datos
6	Facilidad en el desarrollo de la programación

Realizado por: Romero, Cristian, 2023

3.2.1.2 Requerimientos de software del sistema SCADA

Los distintos requerimientos para los elementos *software* necesarios para el desarrollo del sistema SCADA en función de tecnologías *open source* se encuentran detallados en la tabla 3-11. Además, es importante tener en cuenta la facilidad para la escritura de código y la comunidad de soporte existente.

Tabla 3-11: Requerimientos de *software* para el sistema SCADA

N°	Requerimientos
1	Compatibilidad con el sistema operativo *Windows*

2	Tipo de licencia *open source*
3	Librerías para el desarrollo de GUI (Interfaz gráfica de usuario)
4	Protocolos de comunicación serial
5	Compatibilidad con plataforma de base de datos *open source*
6	Protocolo de enlace a base de datos
7	Facilidad para el desarrollo del sistema

Realizado por: Romero, Cristian, 2023

3.2.1.3 Requerimientos de hardware del sistema SCADA

Para el desarrollo del sistema SCADA es importante integrar los procesos usando elementos físicos que cumplan con las necesidades del proyecto, que sean de fácil acceso y que cuenten con una gran comunidad de soporte. Estos requerimientos se especifican en la tabla 3-12.

Tabla 3-12: Requerimientos de *hardware* para el sistema SCADA

N°	Requerimientos
1	Tipo de licencia *open source*
2	Protocolos de comunicación serial
4	Compatibilidad con lenguajes de programación *open source*
5	Tamaño acorde con los módulos didácticos de conexión
6	Bajo costo

Realizado por: Romero, Cristian, 2023

3.2.2 Diseño general del sistema SCADA

En la ilustración 3-13 se muestra el diseño general de todo el sistema, que va desde el funcionamiento de cada una de los procesos, su integración y el sistema SCADA que las controla.

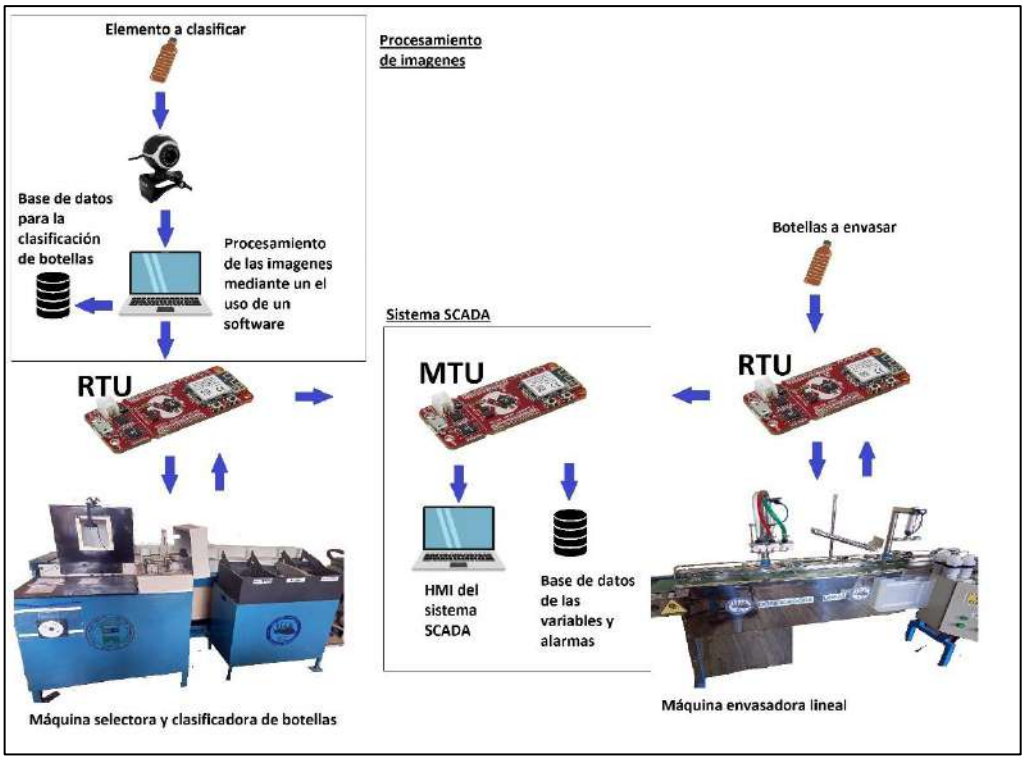


Ilustración 3-13: Diseño general del sistema SCADA

Realizado por: Romero, Cristian, 2023

De manera general se puede explicar el funcionamiento de todo el sistema, comenzando con el proceso de selección y clasificación de botellas, el proceso inicia con el ingreso de una botella a clasificar, la cámara toma las imágenes y mediante el uso de una base de datos y un *software* desarrollado para la clasificación se procesa las imágenes obtenidas desde la cámara. Desde la PC se envía una señal a la RTU para activar los actuadores necesarios para la clasificación. En este proceso es importante tener en cuenta la cantidad de botellas procesadas, así como el tipo de material que las componen.

El proceso de envasado lineal funciona usando sensores para el ingreso de las botellas, existen dos secciones dentro del proceso, la sección de llenado y de tapado. La RTU recibe la información del ingreso de botellas desde sensores y procede a activar los actuadores en función de una secuencia de funcionamiento. La variable de la que depende el proceso es la del tiempo de llenado ya que puede variar en dependencia de la viscosidad del fluido a envasar.

El sistema SCADA que integra los dos procesos industriales debe tener en cuenta el uso de una MTU o unidad terminal maestra que permite controlar las RTUs de cada uno de los procesos. El sistema debe contar con algunos requerimientos como:

- Permitir la monitorización, supervisión y adquisición de datos de los procesos productivos integrados en el sistema.
- Considerar la independencia entre el control automático y el control manual.
- Debe ser desarrollado con herramientas de *hardware* y *software open source*.

- El sistema debe operar sin la acción de un operario, pero de ser necesario el operario podrá realizar el control manual de los procesos.
- Las RTUs debe tener una secuencia básica de control: arranque, detención.
- Las variables involucradas en el proceso deben ser almacenadas en una base de datos para su futuro procesamiento.
- La HMI o interface humano-máquina del sistema debe ser intuitiva y de fácil manejo para el operario permitiendo la visualización de alarmas y eventos generados en el proceso.
- Se debe garantizar la seguridad de los datos y el control de los procesos restringiendo el acceso a personas no autorizadas.

3.2.3 Arquitectura del sistema SCADA

En la ilustración 3-14 se puede apreciar la arquitectura general del sistema SCADA, en el sistema se integran dos procesos industriales los mismos que pueden ser controlados por el usuario mediante el uso de herramientas de control y visualización.

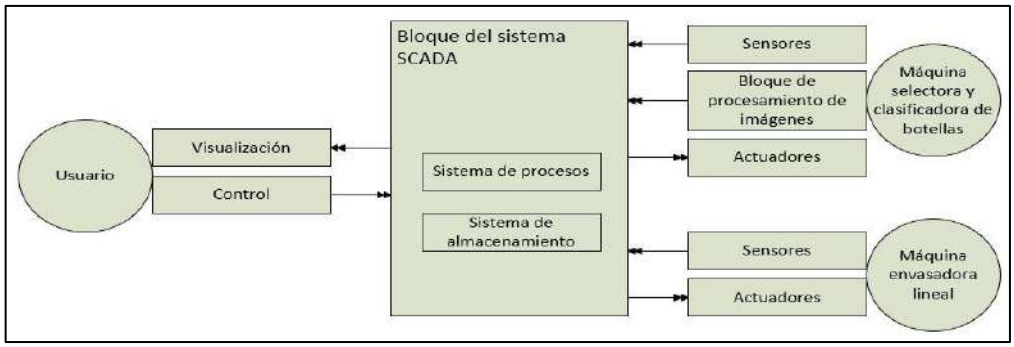


Ilustración 3-14: Arquitectura del sistema SCADA

Realizado por: Romero, Cristian, 2023

El sistema SCADA está conformado por distintos bloques y herramientas, sus funciones se detallan a continuación:

- Visualización: Es una herramienta que permite al usuario obtener información en tiempo real del proceso observando en una pantalla el estado del mismo.
- Control: Permite la comunicación directa para el control de los procesos industriales entre el operario y el sistema SCADA.
- Bloque del sistema SCADA: Conformado por el sistema SCADA y todos los elementos de *hardware* y *software* que lo componen. La información obtenida por las RTUs en cada uno de los procesos pasa a la MTU mediante una red de comunicación, de la misma forma la MTU se comunica con la terminal SCADA denominada sistema de procesos donde se interpreta, procesa y se muestra la información en una interface amigable con el usuario o HMI, desde esta interface se puede controlar así como observar el funcionamiento del

proceso en tiempo real, para finalizar los datos obtenidos se almacenan en una base de datos o sistema de almacenamiento para su análisis posterior.

- Sensores: Son elementos de campo que se encargan de obtener información del estado del proceso productivo.
- Actuadores: Son elementos que reciben señales eléctricas para poner en marcha el proceso mediante acciones físicas, son controlados desde el bloque de sistema SCADA.
- Bloque de procesamiento de imágenes: Se refiere al sistema de visión artificial con el que trabaja el proceso de selección y clasificación de botellas, se encarga de clasificar las botellas mediante el uso de un *software* de reconocimiento visual, dicho algoritmo procesa las imágenes y envía una señal a la RTU indicando el tipo de material de la botella procesada.

3.3 Selección

Identificados los requerimientos para la creación del sistema y desarrollado el diseño general para la integración de los procesos industriales con la respectiva arquitectura del sistema SCADA, se procede a seleccionar las herramientas de *hardware* y *software* necesarias para la puesta en marcha del proyecto.

3.3.1 Selección de software

3.3.1.1 Selección del software para el procesamiento de imágenes

En la selección del *software* para el procesamiento de imágenes en la selección y clasificación de botellas se evaluaron los lenguajes de programación más usados en la actualidad tabla 3-13.

Tabla 3-13: Selección de *software* para el procesamiento de imágenes

Lenguajes de programación	*Python*	*Java*	*C#*
Tipo de licencia *open source*	PSFL (Licencia de *software* libre permisiva)	GNU GLP (Licencia pública general)	GNU GLP (Licencia pública general)
Protocolos de comunicación serial	Serial Ethernet	Serial Ethernet	Serial Ethernet
Librerías para el procesamiento de imágenes	Sí	Sí	Sí
Librería para la gestión de archivos	Sí	Sí	Sí
Comunicación con base de datos	Sí	Sí	Sí
Facilidad en el desarrollo de la programación	Fácil	Medio	Difícil

Realizado por: Romero, Cristian, 2023

El lenguaje de programación *Python* fue seleccionado para el desarrollo del sistema del procesamiento de imágenes, este lenguaje de código abierto cumple con todos los requerimientos necesarios. El tipo de licencia del lenguaje de programación *Python* es PSFL o licencia de *software* libre permisiva es un tipo de licencia de código abierto u *open source*. Los protocolos de comunicación dentro del lenguaje de programación cumplen con lo requerido, *Python* cuenta con las librerías para el desarrollo de programas de procesamiento de imágenes además de comunicación con múltiples bases de datos *open source*, pero su principal ventaja frente a otros lenguajes es su facilidad en el desarrollo de código para la programación y su comunidad de soporte.

3.3.1.2 Selección del software del sistema SCADA

En la tabla 3-14, se comparan los distintos lenguajes de programación para el desarrollo del sistema SCADA. Los lenguajes de programación seleccionados cuentan con una gran popularidad, versatilidad y cuentan con grandes comunidades de soporte y foros de usuarios.

Tabla 3-14: Selección de *software* para el sistema SCADA

Lenguajes de programación	*Python*	*Java*	*C#*
Compatibilidad con el SO *Windows*	Sí	Sí	Sí
Tipo de licencia *open source*	PSFL	GNU GLP	GNU GLP
Librerías para el desarrollo de GUI	Sí	Sí	Sí
Protocolos de comunicación serial	Sí	Sí	Sí
Compatibilidad con plataformas de base de datos	Sí	Sí	Sí
Protocolo de enlace a base de datos	Sí	Sí	Sí
Facilidad para el desarrollo del sistema	Fácil	Medio	Difícil

Realizado por: Romero, Cristian, 2023

El lenguaje de programación *Python* ilustración 3-15 cumple con los requerimientos de *software* necesarios para el desarrollo del sistema SCADA. Cuenta con compatibilidad con el sistema operativo Windows lo que permite el uso extendido del *software*, el tipo de licencia es PSFL o licencia de *software* libre permisiva de código abierto (*open source*). Cuenta con distintas librerías para el desarrollo de interfaces gráficas de usuarios GUI, comunicación serial, compatibilidad y protocolos de comunicación con base de datos de tipo *open source*. A diferencia de los otros lenguajes de programación *Python* es más fácil para aprender y para escribir código, obteniendo una importante ventaja.

Ilustración 3-15: Logo de Python

Fuente: (Python 2021)

Una vez seleccionada el lenguaje de programación en el que se desarrollará el sistema SCADA, se elige la base de datos para el almacenamiento de información recolectada de los procesos. La base de datos SQLite cuenta con características acordes a los requerimientos de *software* y alcance del proyecto. Sus características se presentan en la tabla 3-15.

Tabla 3-15: Características de la base de datos SQLite

	Características
	Completa y se almacena en un solo archivo
	Rápida ya que puede trabajar solo con memoria
	Fácil de añadir en grandes proyectos
	Sin dependencias externas
	Puede ser usada en distintas plataformas: Android, BSD, iOS, Linux, Mac, Solaris, VxWorks, y Windows (Win32)

Realizado por: Romero, Cristian, 2023

3.3.2 Selección de hardware

3.3.2.1 Selección de hardware del sistema SCADA

Para la selección de los elementos de *hardware* que cumplirán las funciones de MTU y RTU dentro del sistema SCADA se evaluaron los elementos más usados, de fácil acceso y con una comunidad de soporte más numerosa tabla 3-16.

Tabla 3-16: Selección de *hardware* del sistema SCADA

Elemento de *hardware*	*Arduino*	*ESP32*	*Raspberry*
Tipo de licencia *open source*	OSHW (*Hardware* libre)	OSHW (*Hardware* libre)	OSHW (*Hardware* libre)
Protocolos de comunicación serial	Serial Ethernet Wifi	Serial Ethernet Wifi	Serial Ethernet Wifi
Compatibilidad con lenguajes de programación *open source*	Sí	Sí	Sí
Tamaño acorde con los módulos didácticos de conexión	Sí	Sí	Sí
Bajo costo	Sí	Sí	No

Realizado por: Romero, Cristian, 2023

Para cumplir las funciones de MTU y RTU dentro del proyecto se seleccionó el *hardware* Arduino ilustración 3-16, el mismo que cumple con todos los requerimientos para la selección. El tipo de licencia de los elementos desarrollados por Arduino es OSHW *hardware* libre, cumpliendo con el requerimiento de licencia *open source*. El protocolo de comunicación necesario está dentro de los protocolos de comunicación permisible los mismos que son: serial, ethernet y wifi. Arduino cuenta con compatibilidad con distintos lenguajes de programación, su tamaño permite integrarlo en el módulo didáctico de conexión y cuenta con un bajo costo que lo hace asequible. Es importante mencionar que los elementos de *hardware* Arduino son de fácil acceso y su comunidad de soporte cuenta con un gran número de usuarios.

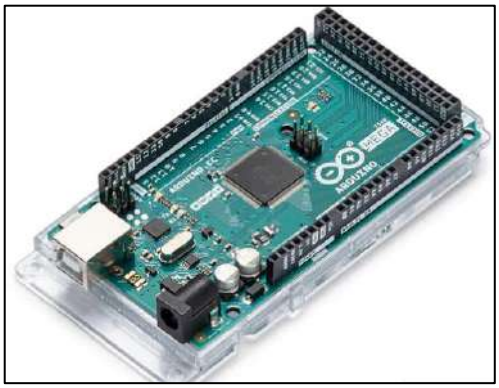


Ilustración 3-16: Arduino Mega 2560

Fuente: (Arduino 2021)

La marca Arduino cuenta con diferentes series de microcontroladores, los más usados y de mayor disponibilidad son: Mega, Uno, Nano. Gracias a su gran cantidad de salidas digitales y analógicas para el desarrollo del proyecto, se usará el microcontrolador de la serie Mega 2560, sus características se detallan en la tabla 3-17.

Tabla 3-17: Tabla de características del microcontrolador Arduino Mega

Microcontrolador	Mega 2560
Voltaje de operación	5 VCD
Voltaje en las entradas digitales (Recomendado)	7 – 12 VCD
Límite de voltaje en las entradas digitales (Máximo)	6 – 20 VCD
Pines digitales de entrada y salida	54 pines
Pines analógicos de entrada	16 pines
Memoria	256 KB
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz
Largo	101,52 mm
Ancho	53,3 mm
Peso	37 g

Fuente: (Arduino 2021)

Otro de los elementos de *hardware* a implementar son los relés para el control de los actuadores y una fuente de poder para la alimentación de los sensores. Arduino cuenta con módulos relé que permiten el control de los mismos de manera fácil. Las principales características de los módulos relé se detallan en la tabla 3-18.

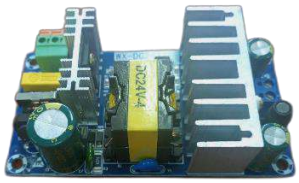
Tabla 3-18: Tabla de características del módulo relé de 4 canales

	Voltaje Operacional	5 VCD
	Señal de control	3,3 – 5 VCD
	Relé	SRD-05VDC-SL-C
	Capacidad máxima	10A / 250VCA
	Tiempo de acción	5 – 10 ms
	Indicadores led de activación	-

Fuente: (AV Electronics 2021)

La fuente de poder para la alimentación de los sensores debe tomar en consideración el voltaje referencial de los mismos, que va de los 10 a los 30 voltios de corriente directa. Las principales características de la fuente de poder se presentan en la tabla 3-19.

Tabla 3-19: Tabla de características de la fuente de poder

	Voltaje de entrada	110/220 VCA
	Señal de salida	21 VCD
	Número de salidas	1
	Amperaje	6 A
	Indicador led de activación	

Fuente: (AV Electronics 2021)

3.3.3 Protocolos de comunicación

Los microcontroladores Arduino cuentan con protocolos que permiten la comunicación entre distintos dispositivos, dentro de estos protocolos tenemos: I2C, SPI, UART. Para el proyecto se seleccionó el protocolo de comunicación BUS I2C, estándar desarrollado en función de una comunicación maestro esclavo que facilita y simplifica en gran medida la comunicación entre las RTUs y la MTU. Las principales características se detallan en la tabla 3-20.

Tabla 3-20: Tabla de características del estándar I2C

Líneas de comunicación	SCL (Línea de pulso de reloj)
	SDA (Línea de datos)
Tipo de comunicación	*Half – duplex* (Envía o recibe información no ambos)
Tipo de sistema	Maestro – Esclavo

Número de dispositivos maestros	Multi – maestro
Número de dispositivos esclavos	Multi – esclavo
Topología	Bus de transmisión
Velocidad de datos	Baja: hasta 100 kbps
	Media: hasta 400 kbps
	Alta: hasta 1 Mbps
	Máxima: 3,4 Mbps
Tipo de bus	Bidireccional
Reloj	Señal común del reloj maestro
Protocolo	Bit de inicio, bit de detención

Fuente: (García 2014, p. 25)

3.4 Diseño y construcción de los módulos didácticos

3.4.1 Diseño de los módulos didácticos

Los módulos didácticos de conexión formarán parte de cada una de los procesos a integrar en el sistema SCADA, de manera que faciliten la conexión entre elementos para que los estudiantes puedan realizar prácticas de automatización y control. Los módulos deben integrar todos los elementos de mando y potencia. Entre las principales características para el diseño y construcción de los módulos didácticos de conexión tenemos:

- Facilidad en la construcción
- Elementos de bajo costo
- Materiales de fácil acceso
- Facilidad para su operación
- Seguro
- Fácil mantenimiento
- Fácil montaje y desmontaje de conexiones
- Debe contener conexiones para todos los elementos de mando y potencia
- Elementos de *hardware open source*

3.4.1.1 Diseño CAD de los módulos didácticos del proceso de clasificación de botellas

Antes de realizar la construcción de los módulos didácticos de conexión se realizará el desarrollo de los diseños CAD, tomando en cuenta las características especificadas. Para crear el modelo se usó el *software* Autodesk Inventor Professional 2023, ya que cuenta con una gran cantidad de prestaciones, además que facilita licencias para estudiantes, simplificando su adquisición y uso.

Los diseños se crearon con chapas metálicas para las bases y plástico PMMA (Polimetilmetacrilato) para la superficie, en cada diseño se incluye los elementos que formarán parte de los sistemas de control. Se pensó en dos módulos por proceso, el primer módulo contará con las conexiones para los elementos de potencia, el segundo con las conexiones para los elementos de mando.

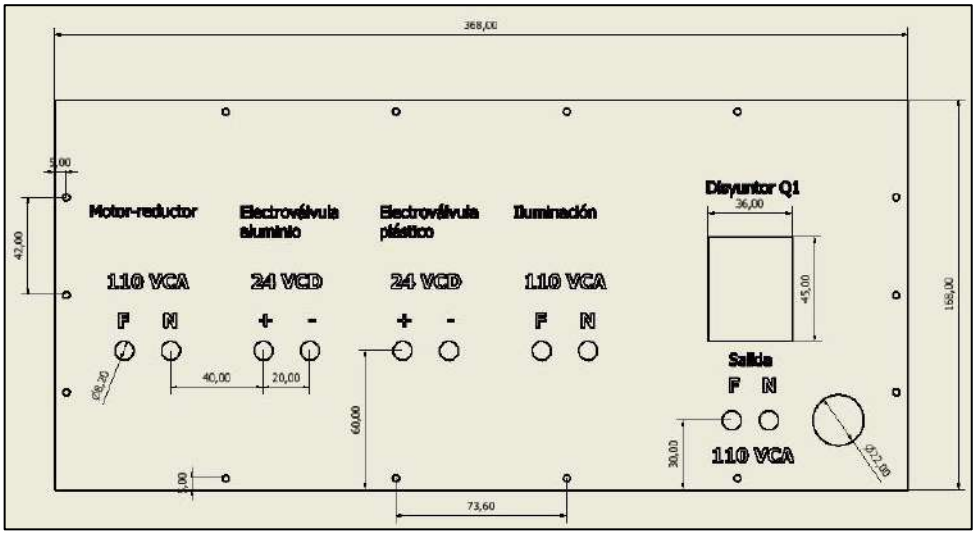


Ilustración 3-17: Placa de PMMA del módulo de actuadores del proceso de clasificación

Realizado por: Romero, Cristian, 2023

La ilustración 3-17 representa a la placa de PMMA del módulo de actuadores del proceso de selección y clasificación de botellas, la misma que cuenta como actuadores al motor-reductor de la banda transportadora, las dos electroválvulas para la clasificación de las botellas y el sistema de iluminación, además cuenta con el sistema de alimentación del sistema, planos en el anexo C.

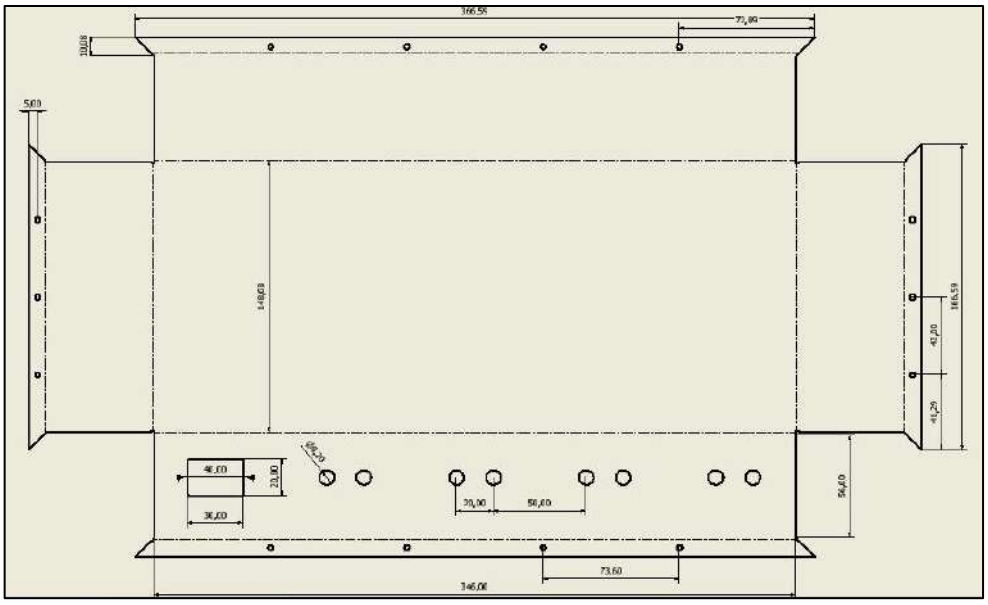


Ilustración 3-18: Base del módulo de actuadores del proceso de clasificación

Realizado por: Romero, Cristian, 2023

La ilustración 3-18 representa el plano desarrollado de la base del módulo de actuadores del proceso de selección y clasificación de botellas, esta base permitirá colocar distintos elementos de protección del sistema, planos en el anexo C.

Creados cada uno de los elementos necesarios en el módulo de actuadores, se puede realizar el ensamblaje y cableado del módulo, generando un modelo CAD muy realista. En la ilustración 3-19 se muestra el resultado del ensamblaje.

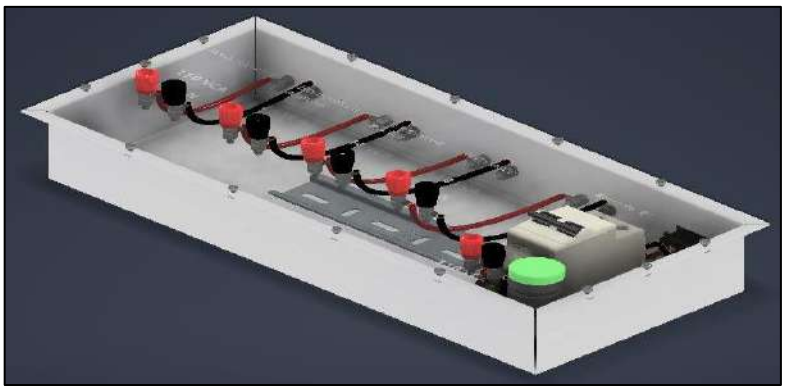


Ilustración 3-19: Ensamblaje del módulo de actuadores

Realizado por: Romero, Cristian, 2023

El segundo módulo del proceso de selección y clasificación de botellas es el módulo de control, cuenta con una base metálica de mayor tamaño que en el módulo de actuadores, ya que cuenta con un mayor número de elementos. La ilustración 3-20 muestra el plano desarrollado de la base del módulo realizada en chapa metálica, planos en el anexo C.

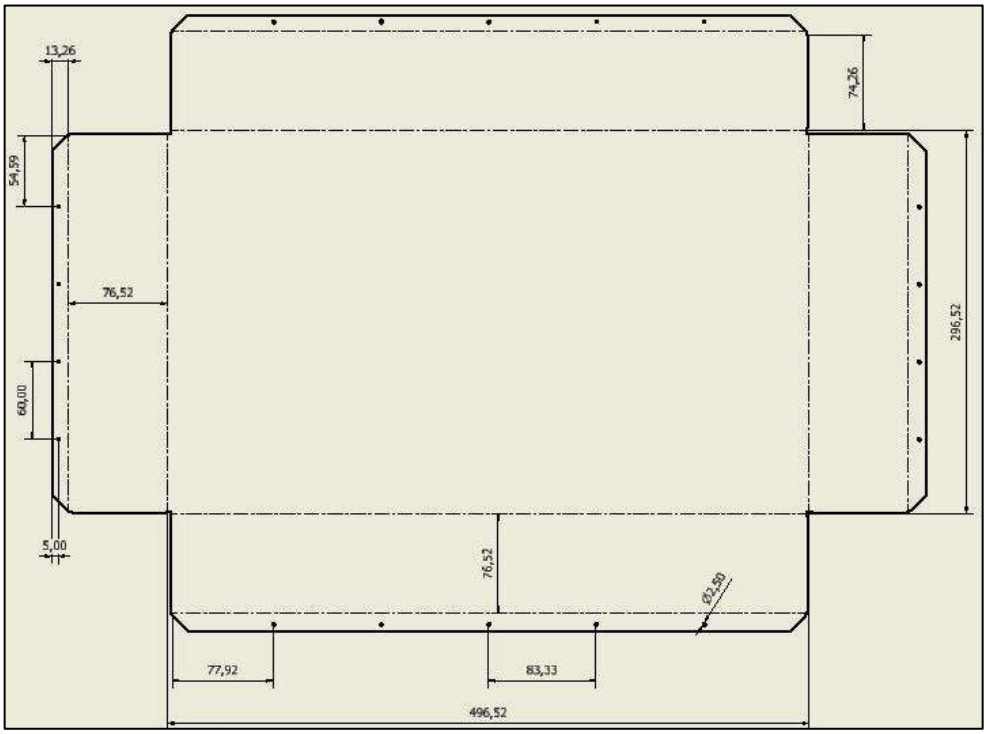


Ilustración 3-20: Base del módulo de control del proceso de clasificación

Realizado por: Romero, Cristian, 2023

La placa PMMA del módulo de control cuenta con una mayor cantidad de elementos, la principal función de esté modulo es dotar de elementos para el control del proceso de selección y clasificación de botellas. La ilustración 3-21 representa a la placa del módulo, planos en el anexo C.

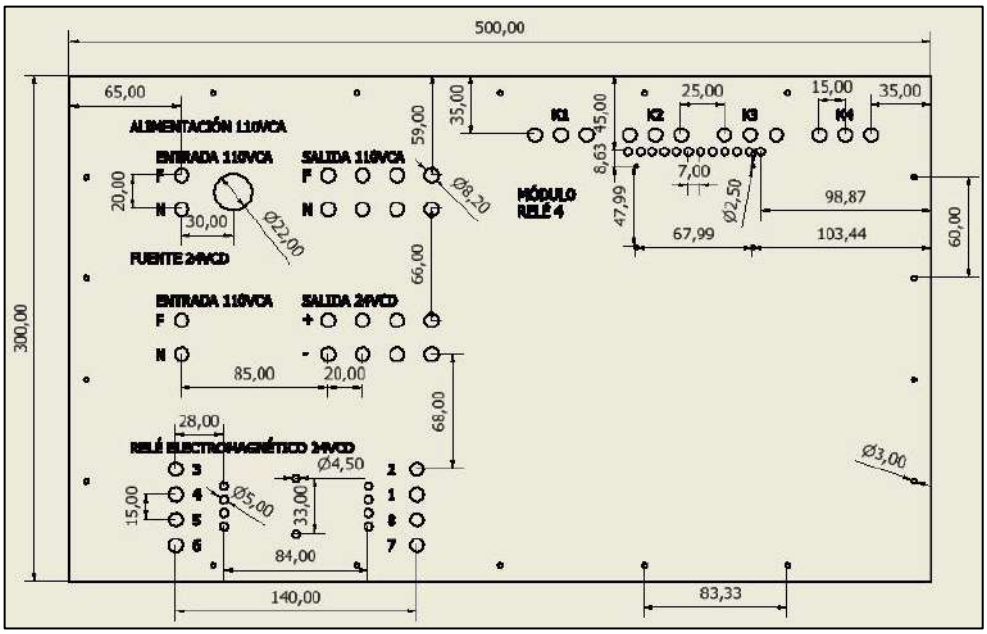


Ilustración 3-21: Placa de PMMA del módulo de control del proceso de clasificación

Realizado por: Romero, Cristian, 2023

Al final de la creación de cada uno de los elementos del módulo, se desarrolla el ensamblaje para saber el aspecto final del mismo. El módulo cuenta con todos los elementos necesarios para realizar el proceso de control del proceso. En la ilustración 3-22 se muestra el ensamblaje.

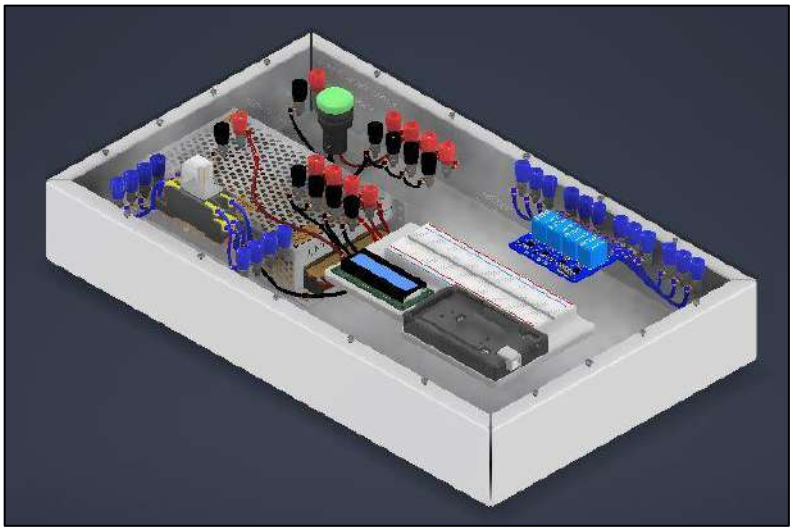


Ilustración 3-22: Ensamblaje del módulo de control

Realizado por: Romero, Cristian, 2023

3.4.1.2 Diseño CAD de los módulos didácticos del proceso de envasado lineal

Para el diseño de los módulos es necesario tener en cuenta la cantidad de sensores y actuadores con que se cuentan, al igual que los módulos del proceso de selección y clasificación de botellas, el diseño está pensado en dos módulos, uno de mando y otro de actuadores. Los módulos están desarrollados de manera que realizar las conexiones entre los elementos sea fácil.

La ilustración 3-23 representa a la placa de PMMA del módulo de actuadores o control del proceso de envasado lineal, la misma que cuenta con conexiones para dos sensores capacitivos, para el motor de la banda transportadora, la bomba hidráulica, para tres electroválvulas monoestables, una electroválvula biestable y para los dosificadores de llenado, para el control de la salida de alimentación se cuenta con espacio para un disyuntor y una luz piloto como indicador. Ver el plano en el anexo D.

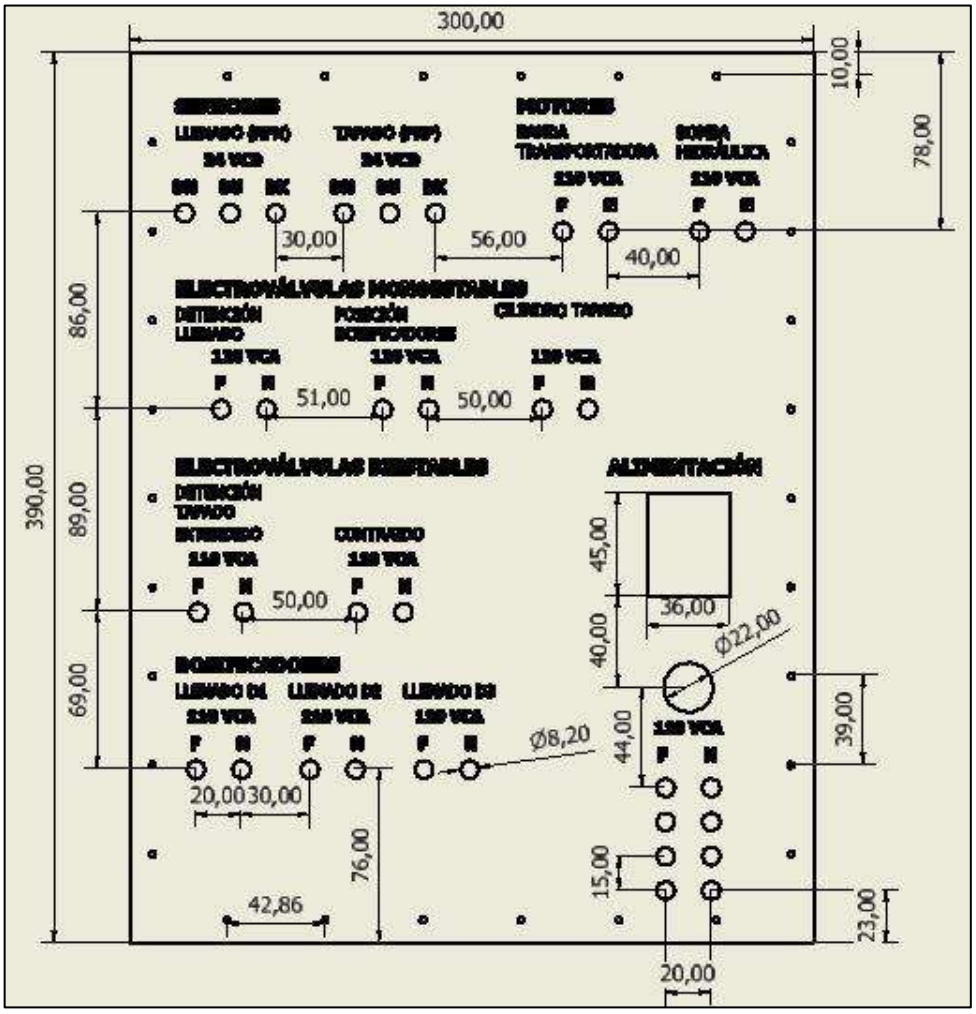


Ilustración 3-23: Placa PMMA del módulo de actuadores del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

La ilustración 3-24 es la representación del desarrollo de la base, diseñada en chapa metálica. El diseño de la base es muy similar a las anteriores, salvo unos por sus medidas y unos agujeros que permiten el paso de los cables para las conexiones. Ver el plano en el anexo D.

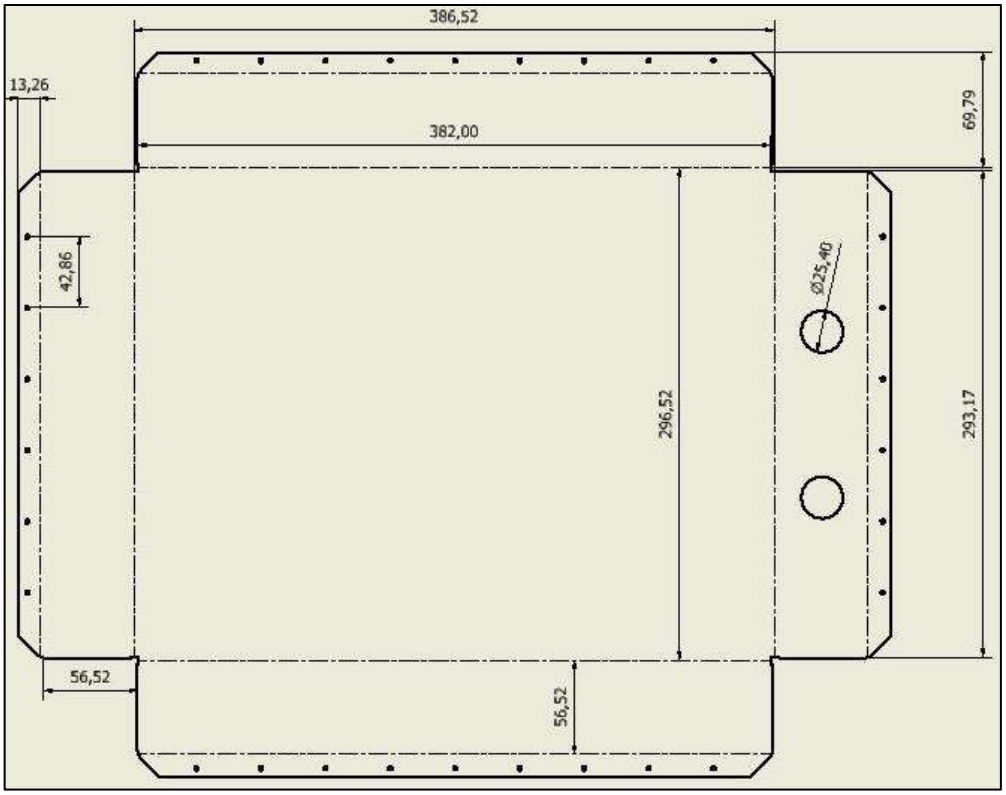


Ilustración 3-24: Desarrollo de la base de actuadores del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

Una vez realizado el desarrollo CAD de cada elemento incluido en el módulo de actuadores, se realiza el ensamblaje. En la Ilustración 3-25 se muestra el resultado final del ensamblaje.

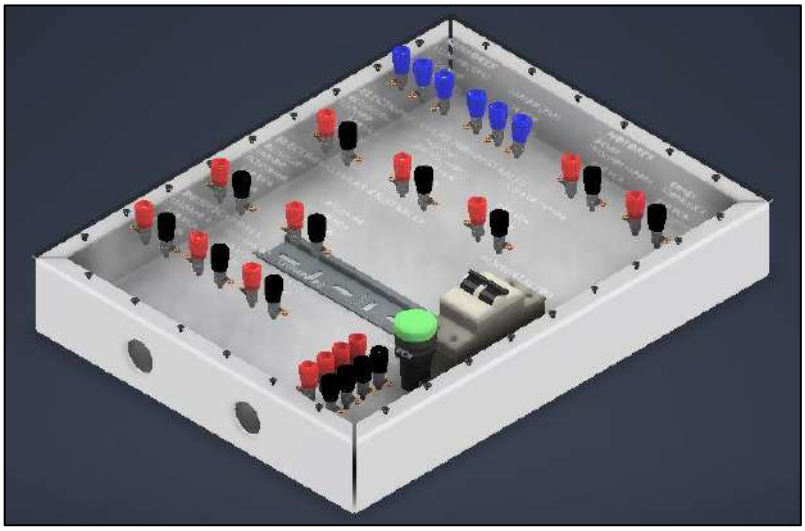


Ilustración 3-25: Ensamblaje del módulo de actuadores

Realizado por: Romero, Cristian, 2023

El segundo módulo o módulo de mando del proceso de envasado lineal cuenta con los elementos necesarios para el control de cada actuador. Los principales elementos del módulo son la placa PMMA ilustración 3-26 y la base metálica ilustración 3-27, el módulo cuenta con el espacio necesario para los relés, el microcontrolador Arduino con sus respectivos elementos y espacio para realizar la alimentación del sistema. Ver los planos en el anexo D.

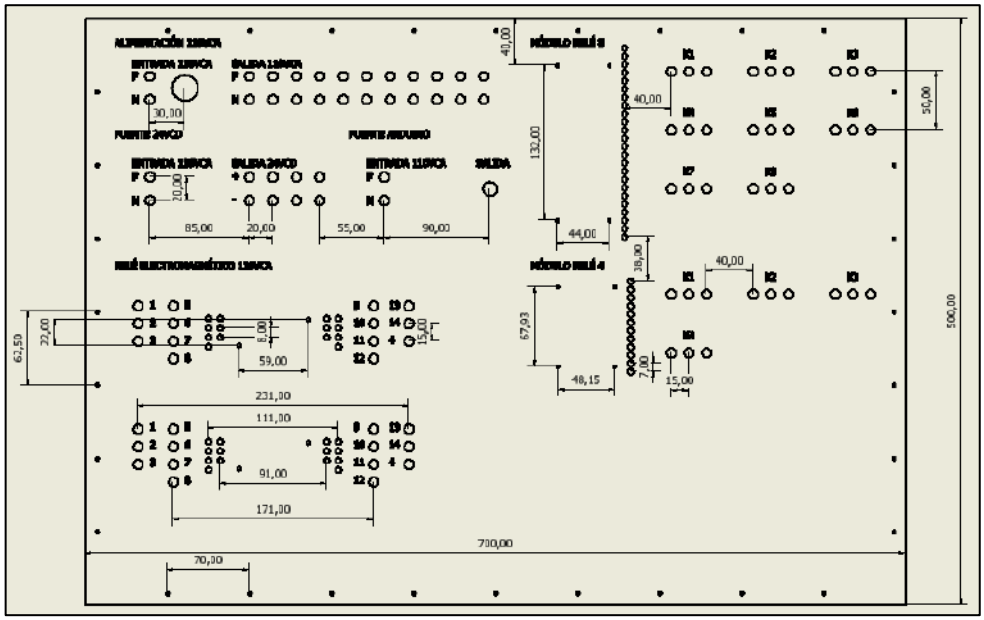


Ilustración 3-26: Placa de PMMA del módulo de control del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

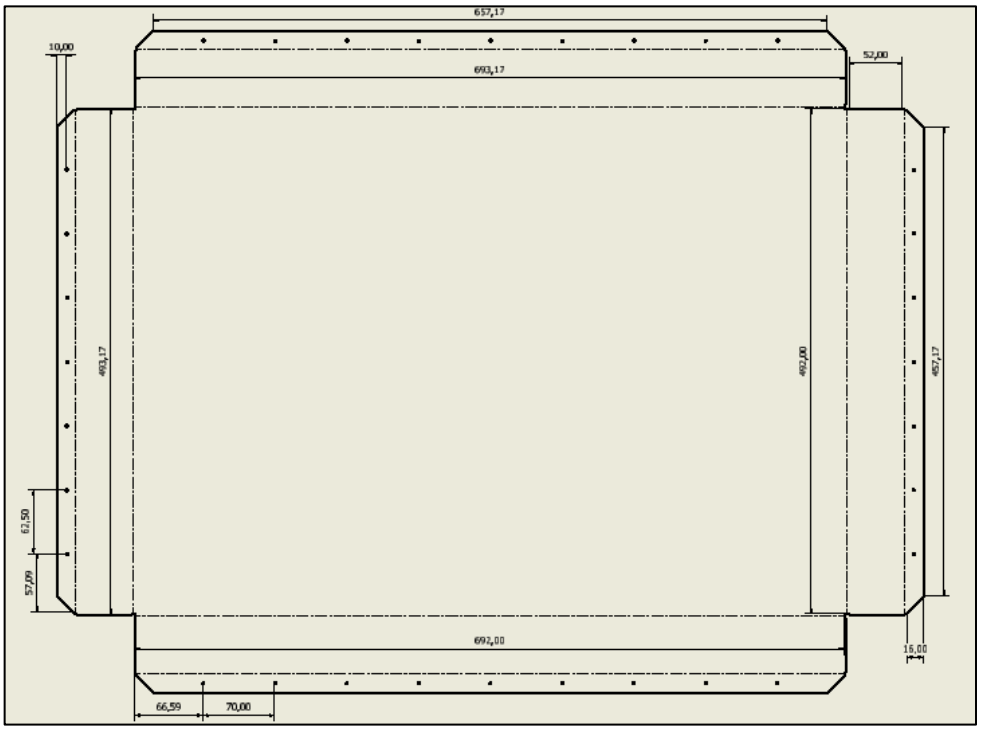


Ilustración 3-27: Desarrollo de la base de mando del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

Finalizado el desarrollo de todos los elementos del módulo de mando, se realiza el ensamblaje para poder visualizar el resultado final. Los diseños CAD dan una idea precisa del resultado final además que aportan con los planos para la construcción. En la ilustración 3-28 se puede observar el módulo de mando ensamblado.

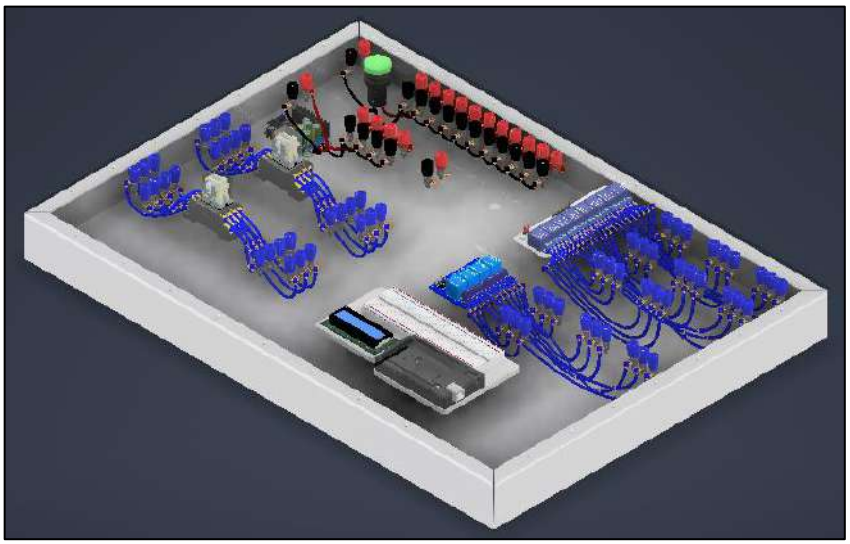


Ilustración 23-8: Ensamblaje del módulo de mando del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

3.4.2 Construcción de los módulos didácticos

Desarrollados los modelos CAD de los módulos didácticos y determinados los materiales usados en los mismos, además de los elementos necesarios para su ensamblaje, el siguiente paso es la construcción. Para la construcción de las bases metálicas se seleccionó chapas metálicas estructurales por su resistencia, durabilidad y elasticidad para el proceso de doblado. En la ilustración 3-29 se muestra las bases de los módulos didácticos.

Ilustración 3-29: Bases de los módulos didácticos

Realizado por: Romero, Cristian, 2023

Para la placa que soportan las conexiones y los elementos que forman parte del módulo se seleccionó el polímero PMMA por su alta resistencia a los golpes, aislante eléctrico, resistencia al envejecimiento y condiciones ambientales, además de su estética. Las placas ilustración 3-30 son de color negro con etiquetas y acabados realizadas por láser.

Ilustración 3-30: Placa PMMA

Realizado por: Romero, Cristian, 2023

Para el ensamblaje de los módulos son necesarios algunos componentes, entre los que tenemos elementos de control, elementos que permiten la integración del control junto con los elementos de potencia, fuentes de alimentación y luces piloto para indicar la energización de los módulos. En la tabla 3-21 (A) se listan los elementos necesarios para el ensamblaje de los módulos del proceso de selección y clasificación de botellas y en la tabla 3-21 (B) del proceso de envasado lineal.

Tabla 3-21 (A): Tabla de componentes para el proceso de clasificación

Cantidad	Componente
2	Bases de chapa metálica
2	Placas de acrílico PMMA
2	Luz Piloto
2	Disyuntor
1	Conector de entrada de fuente de poder
52	Jacks tipo banana (Rojo/Negro)
12	Jack tipo banana (Azules)
1	Módulo de 4 relés
1	Relé electromagnético 24V
1	Fuente de poder de 24V
1	Arduino Mega 2560

1	Pantalla LCD
1	Protoboard
1	Base para Arduino, LCD, Protoboard
1	Case para Arduino Mega 2560

Realizado por: Romero, Cristian, 2023

Tabla 3-21 (B): Tabla de componentes para el proceso de envasado

Cantidad	Componente
2	Bases de chapa metálica
2	Placas de acrílico PMMA
2	Luz Piloto
2	Disyuntor
68	Jacks tipo banana (Rojo/Negro)
66	Jack tipo banana (Azules)
1	Módulo de 4 relés
1	Módulo de 8 relés
2	Relé electromagnético 24V
1	Fuente de poder de 24V
1	Fuente para Arduino 12V
1	Arduino Mega 2560
1	Pantalla LCD
1	Protoboard
1	Base para Arduino, LCD, Protoboard
1	Case para Arduino Mega 2560

Realizado por: Romero, Cristian, 2023

Para realizar las conexiones de una manera didáctica, se usaron borneras *jack* tipo banana en todos los módulos a construir ilustración 3-31. Las mismas permiten conectar y desconectar los elementos de mando y de potencia de manera fácil y rápida. El objetivo del uso de las borneras es permitir realizar conexiones rápidas para que los estudiantes puedan realizar sus prácticas de control y automatización.

Ilustración 3-31: Placa PMMA con borneras Jack tipo banana

Realizado por: Romero, Cristian, 2023

Las conexiones, las fuentes de energía se colocan en la parte interior de los módulos, para el cableado se usan terminales tipo anillo, con la finalidad que las conexiones sean resistentes. En la ilustración 3-32 se muestra el interior de un módulo de control del proceso de selección y clasificación de botellas.

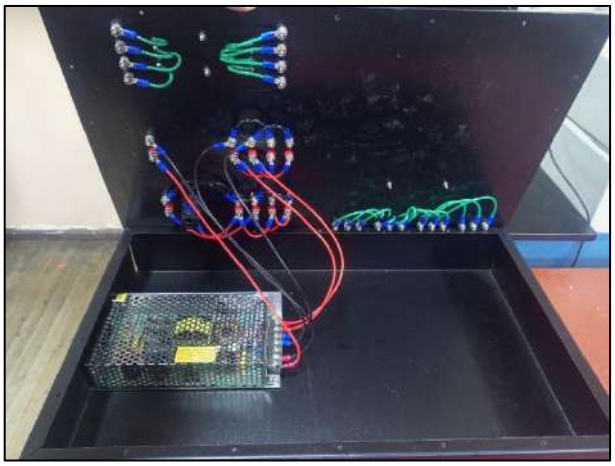


Ilustración 3-32: Conexiones internas de los módulos

Realizado por: Romero, Cristian, 2023

Los módulos ensamblados son cuatro en total, divididos en dos partes, dos por cada proceso, en la ilustración 3-33 (A) se muestran los módulos del proceso de selección y clasificación de botellas, y la ilustración 3-33 (B) representa a los módulos del proceso de envasado lineal.

Ilustración 3-33 (A): Módulos del proceso de selección y clasificación de botellas

Realizado por: Romero, Cristian, 2023

Ilustración 3-33 (B): Módulos del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

3.5 Visión Artificial

Antes de comenzar con el desarrollo del *software* SCADA los procesos a integrar deben funcionar de manera correcta integrando *software* que cumpla con los requerimientos antes mencionados. El proceso de selección y clasificación de botellas cuenta con un programa de visión artificial desarrollado con el *software* MATLAB, mismo que no cuenta con una licencia *open source*, por

tal motivo es necesario el desarrollo del programa en un *software* que cumpla con los requerimientos.

El lenguaje de programación seleccionado para el desarrollo del programa que cumpla con el reconocimiento visual es Python, usando una IDE (Entorno de desarrollo integrado por sus siglas en ingles) de libre acceso que permita desarrollar el algoritmo. *Pycharm Community* cuenta con una licencia *open source* que permite la descarga y el uso del *software* de manera libre y gratuita.

3.5.1 Entrenamiento del modelo

Para comenzar con el proceso de entrenamiento del modelo se deben seleccionar los objetos a detectar y clasificar, en este caso botellas de plástico, vidrio y latas de aluminio. En la tabla 3-22 se detallan las principales características de cada uno de los objetos seleccionados.

Tabla 3-22: Tabla de características de los objetos

Materia	Descripción	Imagen	Peso (Kg)	Dimensiones (mm)	
				Ancho	Altura
Vidrio	Envase color negro		0,25	60	200
Plástico	Envase color café		0,25	60	175
Aluminio	Envase color negro		0,25	55	200

Realizado por: Romero, Cristian, 2023

Para el entrenamiento de la red neuronal que permita el procesamiento y clasificación de los objetos es necesario seguir ciertos pasos:

- Crear un *dataset* (base de datos), tomar y almacenar imágenes de los objetos que se van a detectar y clasificar, la base de datos se detalla en la ilustración 3-34.

Ilustración 3-34: Base de datos para el reconocimiento visual

Realizado por: Romero, Cristian, 2023

- Etiquetar los objetos a detectar y clasificar en el *dataset*, uno para entrenamiento y otro para validación mediante el uso de la plataforma web *Make Sense* en la página www.makesense.ai, almacenando los archivos que soporta el marco de aprendizaje YOLO. En la ilustración 3-35 se muestra gráficamente el proceso de etiquetado.

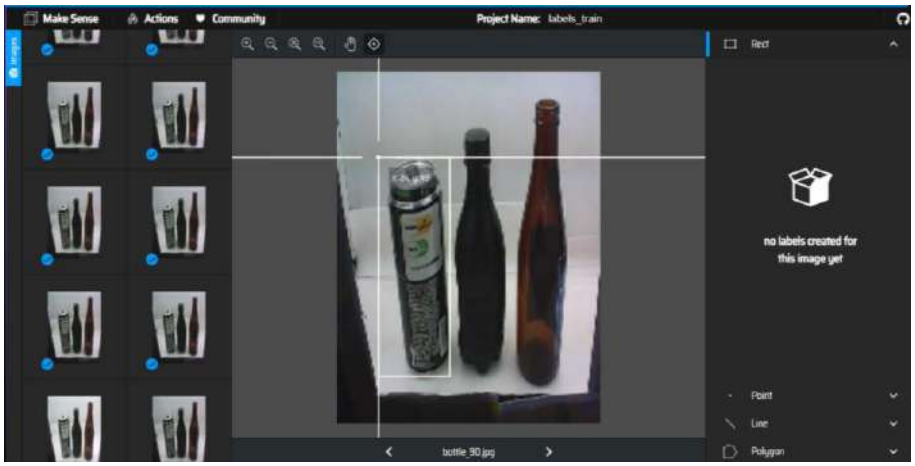


Ilustración 3-35: Etiquetado de los elementos a clasificar

Realizado por: Romero, Cristian, 2022

- En la plataforma *Google Colab* de la página www.colab.research.google.com, se realiza el entrenamiento del modelo, entre los criterios tomados en cuenta para el entrenamiento tenemos el número de ciclos de procesamiento (*epochs* = 150), la inferencia de imagen (*img* = 640) y el tamaño de la matriz de procesamiento de la imagen (*batch* = 4). En la ilustración 3-36 se muestra la interface para el entrenamiento del modelo. Ver el proceso completo de entrenamiento en el anexo E.

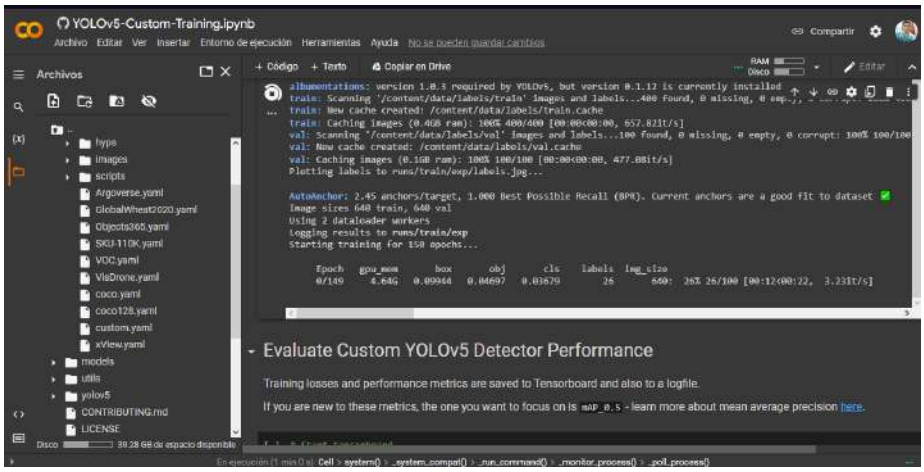


Ilustración 3-36: Plataforma de entrenamiento *Google Colab*

Realizado por: Romero, Cristian, 2023

- Finalizado el entrenamiento se descarga la red neuronal y se descargan los archivos de configuración YOLOV5 desde la plataforma Pytorch, los archivos se instalan en el entorno virtual del programa de reconocimiento visual.

3.5.2 Algoritmo de visión artificial

Finalizado el entrenamiento del modelo, es necesario desarrollar un algoritmo que permita la visualización de la detección y clasificación de botellas. La visualización se debe desarrollar en pantalla usando una interface fácil de usar, además una vez detectado el objeto se debe enviar la información al microcontrolador para que active la secuencia de clasificación.

En la ilustración 3-37 se muestra el diagrama de flujo con la secuencia para la detección de botellas, misma secuencia que se detalla a continuación:

- Para la correcta ejecución del algoritmo se deben usar distintas librerías, las mismas que ya fueron instaladas previamente. Para enviar información al microcontrolador Arduino se usan las librerías *serial* y *time*. Para la detección de objetos se usan las librerías *torch, cv2, numpy* y para acceder a la información obtenida en *dataframe* (tablas), pandas.
- Usando las librerías ya importadas se realiza la conexión para la comunicación con el microcontrolador Arduino.
- Agregando la dirección del modelo ya entrenado, se realiza la lectura del mismo para la detección de las botellas.
- Seleccionando una cámara se inicia la captura de video.
- Para que el algoritmo pueda realizar una detección continua se crea un bucle de repetición infinito, pero para finalizarlo cuando se requiera solo es necesario presionar la tecla ESC.
- Dentro del bucle de repetición se realiza la detección del objeto.

- Lo siguiente es acceder a la información que entrega el modelo, esta información se presenta en una tabla o *dataframe,* entre los datos que se presentan tenemos la posición del objeto, la confianza y la etiqueta del objeto. El dato que se requiere es la etiqueta la misma que nos indica el tipo de material del objeto.
- La información obtenida se compara, si es plástico, o aluminio, en el caso que no se cumpla ninguna de las condiciones anteriores el material el vidrio, en cada caso se envía la información al microcontrolador Arduino.
- Se muestra la imagen para poder visualizar la detección de los objetos, el código de reconocimiento visual se puede ver en el anexo F.

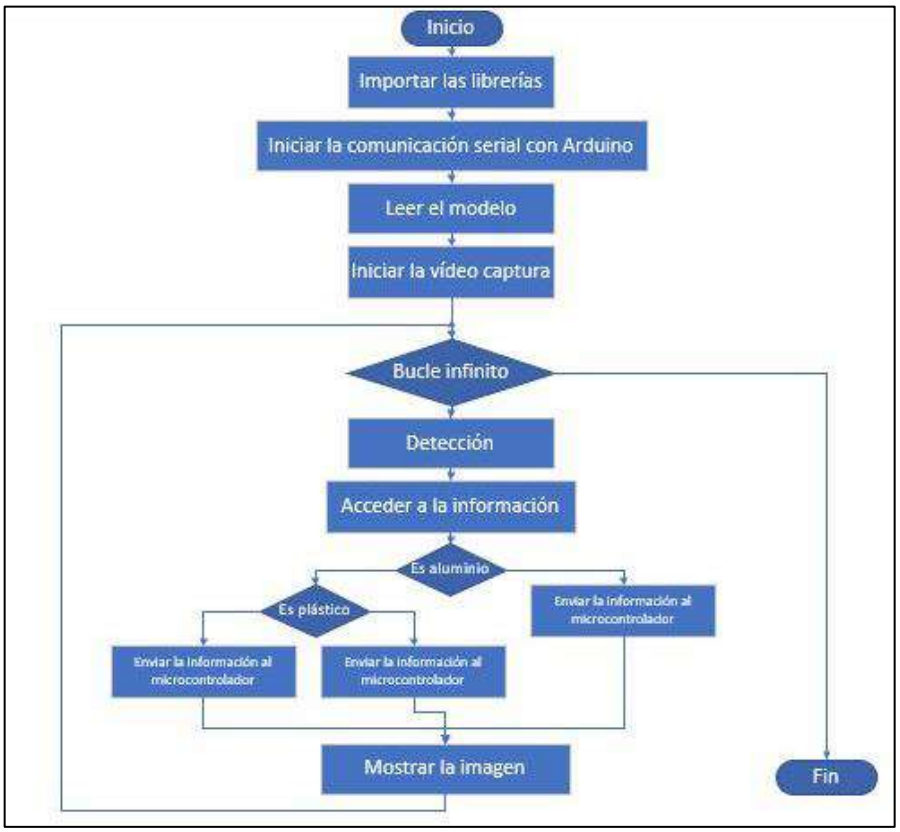


Ilustración 3-37: Diagrama de flujo del algoritmo de visión artificial

Realizado por: Romero, Cristian, 2023

3.5.3 Interfaz gráfica del proceso de visión artificial

Para la presentación de los fotogramas ilustración 3-38 y la detección de objetos en pantalla se necesita el desarrollo de una interfaz gráfica que sea simple, fácil de usar e intuitiva con el usuario. La interface debe presentar los fotogramas y el material del objeto detectado.

Ilustración 3-38: Proceso de selección y clasificación de botellas

Realizado por: Romero, Cristian, 2023

En el desarrollo de la interfaz gráfica intervienen las librerías *Tkinter* necesaria para crear aplicaciones de escritorio, además de las librerías *imutils* para llamar funciones de dibujo denominadas *OpenCV* y *PIL* para el manejo de imágenes. En la ilustración 3-39 se muestran los elementos incluidos en la interfaz gráfica, el código se puede observar en el anexo G.

Ilustración 3-39: Interface para el proceso de selección y clasificación de botellas

Realizado por: Romero, Cristian, 2023

3.6 Sistema SCADA

El proceso de desarrollo del sistema SCADA se organiza por proceso, al final se integran los dos procesos en el sistema y se desarrolla la HMI. Cada proceso cuenta con un microcontrolador que hace el trabajo de RTU, con una programación específica en función del proceso, las RTUs se integran a un microcontrolador que hace el trabajo de MTU y envía la información necesaria al HMI en un ordenador.

3.6.1 Proceso de selección y clasificación de botellas

3.6.1.1 Algoritmo de control del proceso de selección y clasificación de botellas

Finalizado el proceso de detección y clasificación de objetos se realiza el envío de datos desde el algoritmo en Python a la placa microcontrolador Arduino, la misma que hará las veces de RTU. El microcontrolador controlará la entrada de señales desde el sensor y el algoritmo de visión artificial para ejecutar acciones activando los actuadores ya sean eléctricos o neumáticos.

Para la creación del algoritmo del microcontrolador que permita realizar el control del proceso de selección y clasificación de botellas se usa la IDE de programación del fabricante. Arduino cuenta con una IDE diseñada para sus microcontroladores, permite la descarga y uso de librerías, facilidad para cargar y guardar el programa en su propia sintaxis de programación basada en el lenguaje C++.

En la ilustración 3-40 se muestra el diagrama de flujo base para el desarrollo del algoritmo de control del proceso de selección y clasificación, para su creación en la IDE Arduino se siguieron los pasos a continuación:

- Como primer paso para el desarrollo del algoritmo se importan las librerías *LiquidCrystal* para el uso de la pantalla LCD y *Wire* para usar el bus de comunicación I2C.
- A continuación, se realiza la declaración de variables, así como la declaración de pines a usar.
- Como siguiente paso se desarrolla la función *void setup*, dentro de esta función se inician los protocolos de comunicación necesarios para enviar y recibir información, entre estos protocolos de comunicación tenemos el *serial* y el *wire* incluyendo la inicialización de la pantalla LCD.
- Se define el funcionamiento de los pines del microcontrolador, en el caso de los sensores y botones son pines de entrada, para los actuadores son pines de salida.
- Para finalizar la función *void setup* se define el estado inicial de los pines de salida, el estado puede ser *HIGH* para apagado y *LOW* para encendido.

- Se continua con la función *void loop* que es donde se ejecuta la secuencia del proceso, en esta función del algoritmo se recibe los pulsos de los botones para el encendido y apagado, se recibe la información del sensor fotovoltaico para el inicio de la secuencia.
- La función *void loop* finaliza con una estructura condicional anidada que permite comparar el dato enviado desde el algoritmo de Python obtenido desde el puerto serial, así se puede realizar una secuencia determinada en función del material reconocido.
- Se finaliza el desarrollo del algoritmo con las funciones para los eventos de petición y recepción de datos para la comunicación I2C. Ver el algoritmo de control del proceso en el anexo H.

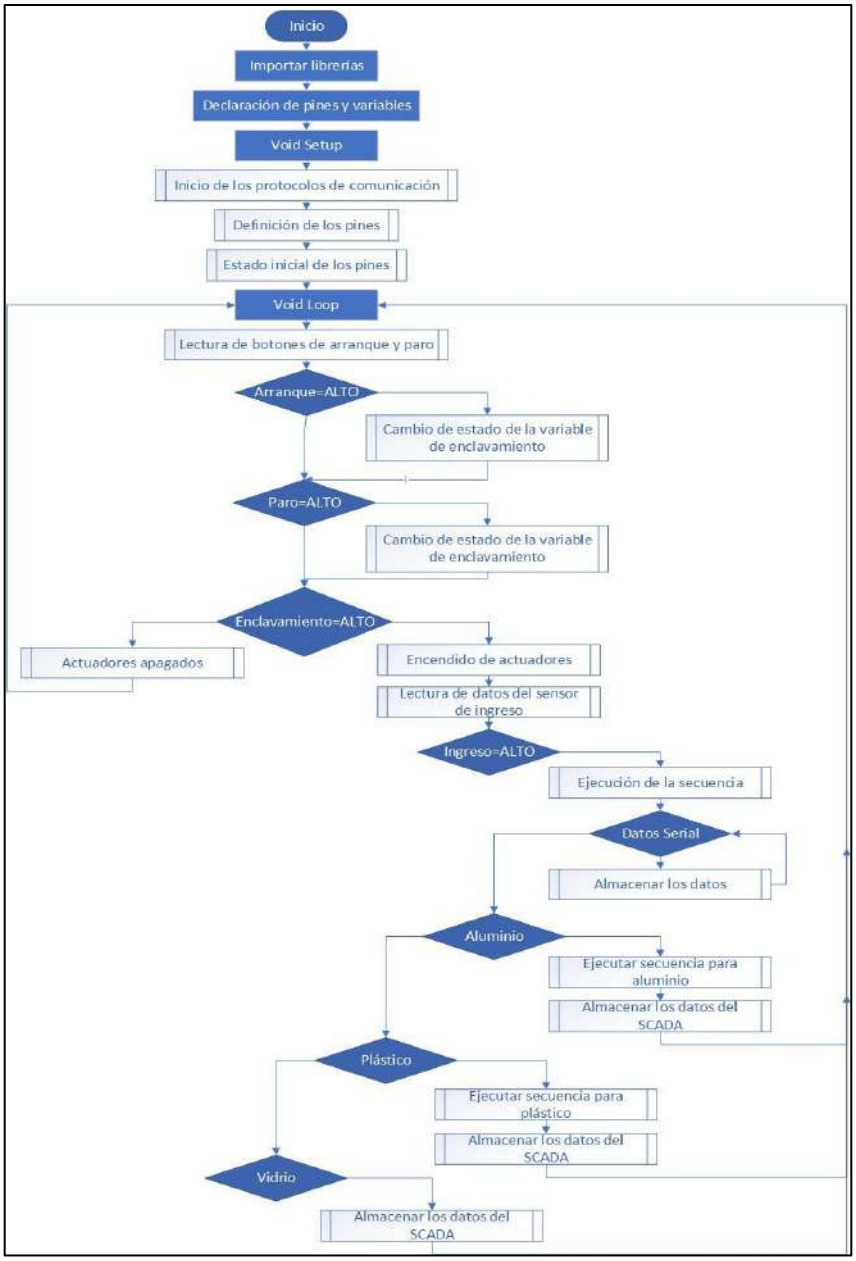


Ilustración 3-40: Diagrama de flujo de control del proceso de clasificación

Realizado por: Romero, Cristian, 2023

3.6.1.2 Conexiones de hardware del proceso de selección y clasificación de objetos

El proceso de clasificación de botellas cuenta con elementos de *hardware* que permiten realizar el control del mismo. En la ilustración 3-41 se aprecia el diagrama de conexiones de los elementos usados, para la creación del diagrama se usó el programa *Fritzing*.

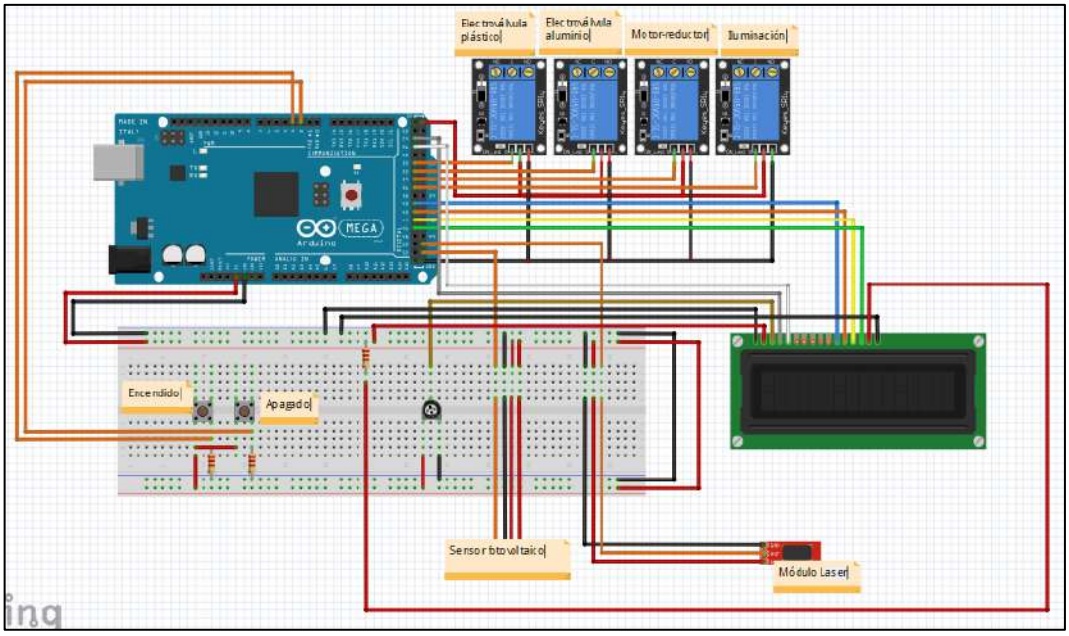


Ilustración 3-41: Diagrama de conexiones del proceso de clasificación de botellas

Realizado por: Romero, Cristian, 2023

Los elementos de *hardware* usados en el control del proceso son los siguientes:

- **Arduino Mega AT2560:** Recibe información desde los sensores, procesa e interpreta la información recibida y envía señales para ejecutar acciones mediante los actuadores. Cuenta con pines analógicos y pines digitales que pueden ser programados como pines de entrada o salida en función de la necesidad.
- **Módulo de 4 relés:** Es usado para activar o desactivar los actuadores del proceso, estos necesitan voltajes mayores a los entregados por las salidas del microcontrolador, el módulo relé permite el paso de una potencia mayor sin causar daños al microcontrolador.
- **Pantalla LCD 16x2:** Permite mostrar información del proceso, es activada por el microcontrolador.
- **Botones:** Son los encargados de enviar una señal de encendido y apagado del proceso.
- **Módulo laser:** Activado por el microcontrolador, envía un haz de luz laser a la foto resistencia del circuito del sensor fotovoltaico.

- **Sensor fotovoltaico:** Envía una señal al microcontrolador una vez se obstruya el haz de luz laser, esto indica el ingreso de un objeto al proceso. En la ilustración 3-42 se representa el circuito del sensor.

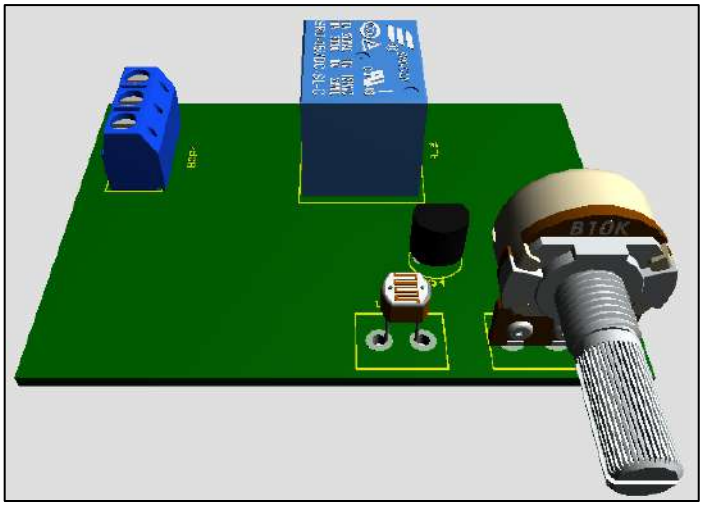


Ilustración 3-42: Diseño del circuito del sensor fotovoltaico

Realizado por: Romero, Cristian, 2023

En la tabla 3-23 se detallan los pines digitales usados para el control del proceso de selección y clasificación de botellas.

Tabla 3-23: Tabla de elementos de *hardware* del proceso de clasificación

Arduino	Tipo (entrada/salida)	Actuador/Sensor
Digital 2	Entrada	Botón de apagado
Digital 3	Entrada	Botón de encendido
Digital 24	Salida	Pin RS de la pantalla LCD
Digital 26	Salida	Pin RW de la pantalla LCD
Digital 30	Salida	Relé K1 Electroválvula plástico
Digital 32	Salida	Relé K2 Electroválvula aluminio
Digital 34	Salida	Relé K3 Motor reductor
Digital 36	Salida	Relé K4 Iluminación
Digital 40	Salida	D4 de la pantalla LCD
Digital 42	Salida	D5 de la pantalla LCD
Digital 44	Salida	D6 de la pantalla LCD
Digital 46	Salida	D7 de la pantalla LCD
Digital 51	Salida	Módulo laser
Digital 53	Entrada	Sensor fotovoltaico

Realizado por: Romero, Cristian, 2023

3.6.2 Proceso de envasado lineal

3.6.2.1 Algoritmo de control del proceso de envasado lineal

El algoritmo de control se desarrolló en la IDE del microcontrolador Arduino. En la ilustración 3-43 se muestra el diagrama de flujo guía para la construcción del algoritmo que controla el proceso. Ver el algoritmo en el anexo I.

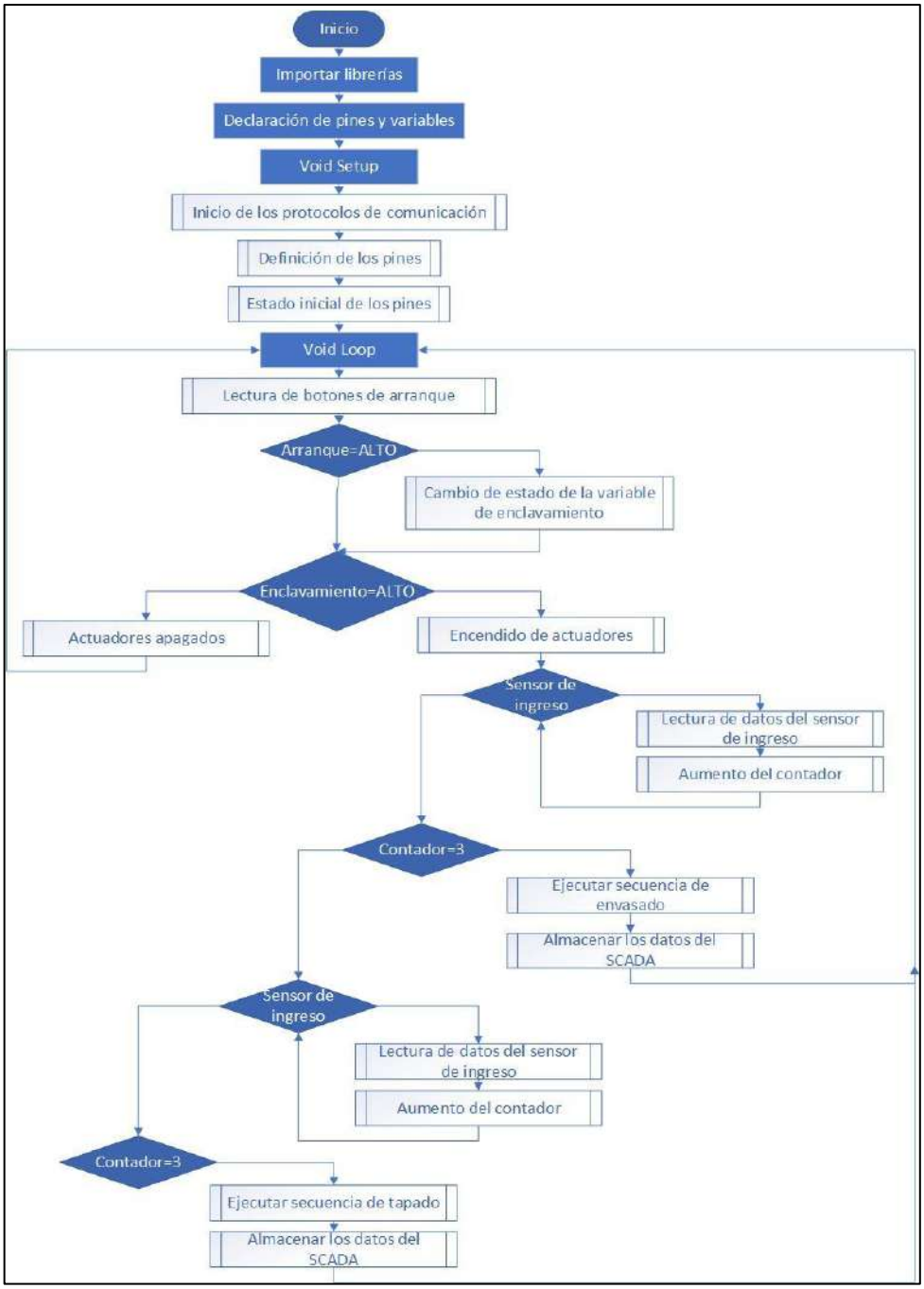


Ilustración 3-43: Diagrama de flujo de control del proceso de envasado lineal

Realizado por: Romero, Cristian, 2022

Para la creación del algoritmo dentro de la IDE Arduino que será la segunda RTU del sistema SCADA se siguieron los siguientes pasos:

- El primer paso a seguir es importar la librería necesaria para el uso del bus de comunicación I2C.
- Declaración de las variables y los pines que se usarán para la recepción de señales y envió de órdenes a los actuadores.
- A continuación, se pasa al desarrollo de la función *void setup,* se genera el código que permite iniciar los protocolos de comunicación serial, I2C junto con la declaración de funcionalidad de los pines, para botones y sensores los pines son de entrada, para los actuadores los pines son de salida.
- La repetición de la secuencia de la maquina se desarrolla dentro de la función *void loop*, la misma que inicia con la lectura del pin del botón de inicio, iniciando la secuencia, una vez activado tres veces el sensor para el área de llenado, se activa la siguiente secuencia, para finalizar con la secuencia del área de tapado, la misma que inicia por la activación del sensor de tapado.
- El algoritmo se finaliza con las funciones de comunicación, la función de evento de petición, envía los datos al microcontrolador maestro y la función de evento de recepción, recibe las órdenes dadas desde el maestro.

3.6.2.2 Conexiones de hardware del proceso de envasado lineal

El proceso de envasado lineal cuenta con una mayor cantidad de actuadores y sensores, por lo tanto, el número de elementos de control aumenta en comparación con el proceso anterior. En la ilustración 3-44 se representa el diagrama de conexión de los elementos de *hardware*.

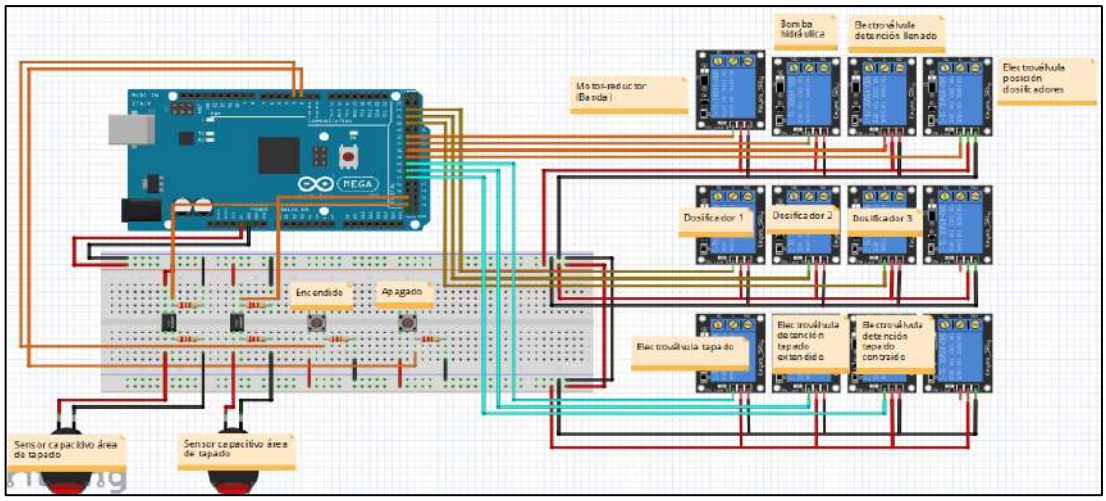


Ilustración 3-44: Diagrama de conexiones del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

Entre los elementos de *hardware* destacados en el proceso de control del envasado lineal tenemos:

- **Arduino MegaAT2560:** Microcontrolador que se encarga de procesar las señales obtenidas en el proceso, ya sean obtenidas por los sensores o enviar señales para accionar los actuadores.
- **Optoacopladores:** Elementos que permiten generar una conexión entre dos elementos de diferente potencia. Permiten obtener las señales de los sensores capacitivos sin dañar las entradas del microcontrolador.
- **Módulo de 8 relés:** Recibe la señal desde el microcontrolador para energizar los actuadores, mediante el uso de una interface de potencia activa los actuadores.
- **Sensores capacitivos:** Son los encargados de detectar el ingreso de un objeto en los distintos sectores del proceso, envían las señales al microcontrolador para su procesamiento.

En la tabla 3-24 se detallan los elementos usados en para las conexiones del sistema de control del proceso de envasado lineal.

Tabla 3-24: Tabla de elementos de *hardware* del proceso de envasado

Arduino	Tipo (entrada/salida)	Actuador/Sensor
Digital 2	Entrada	Botón de apagado
Digital 3	Entrada	Botón de encendido
Digital 24	Salida	Motor-reductor (Banda)
Digital 26	Salida	Bomba hidráulica
Digital 28	Salida	Electroválvula detención de llenado
Digital 32	Salida	Electroválvula posición dosificadores
Digital 34	Salida	Dosificador 1
Digital 36	Salida	Dosificador 2
Digital 38	Salida	Dosificador 3
Digital 40	Salida	Electroválvula tapada
Digital 42	Salida	Electroválvula detención tapado extendido
Digital 44	Salida	Electroválvula detención tapado contraído
Digital 50	Salida	Sensor capacitivo área de tapado
Digital 52	Salida	Sensor capacitivo área de llenado

Realizado por: Romero, Cristian, 2023

3.6.3 Integración de los procesos industriales

3.6.3.1 Algoritmo para el control de la MTU

Una vez que los procesos industriales se encuentran funcionando el siguiente paso es la integración de procesos industriales. La integración se da mediante el uso del protocolo de comunicación I2C con la técnica de comunicación maestro-esclavo, el dispositivo maestro o MTU integra las dos RTU que gobiernan cada uno de los procesos. En la ilustración 3-45 se muestra el diagrama de flujo seguido para el diseño del algoritmo y en el anexo J se presenta el algoritmo programado en la IDE Arduino.

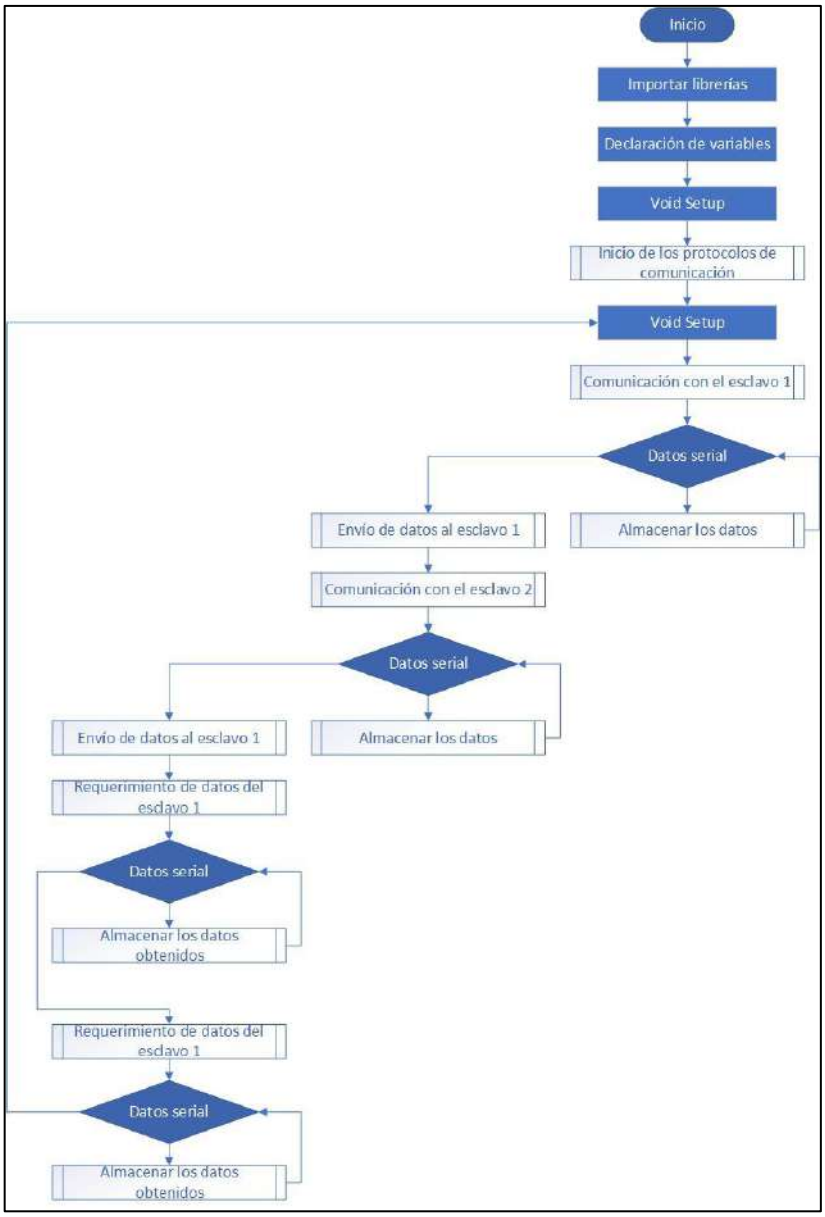


Ilustración 3-45: Diagrama de flujo de la integración de procesos

Realizado por: Romero, Cristian, 2023

Para la creación del algoritmo de integración de los dos procesos se siguieron los siguientes pasos:

- Como primer paso para el desarrollo del programa se llama a la librería *Wire* la misma que permite la comunicación mediante el uso del bus I2C.
- El siguiente paso es declarar las variables usadas en el programa, estas variables almacenarán los datos recibidos desde los dispositivos esclavos y servirán para almacenar los datos a enviar.
- En el *Void setup* donde se ejecuta el estado inicial del proceso, se inicial la comunicación *Serial* y la comunicación *Wire*.
- En el bucle *Void loop* se ejecuta el envío de datos a los esclavos uno y dos, mediante el uso de la estructura condicional *If* se pregunta si existen datos en el puerto serial, estos datos serán enviados a los dispositivos esclavos.
- En el mismo bucle se ejecuta los procesos de recepción de datos desde los dispositivos esclavos, mediante el uso de la estructura de repetición *While* se pregunta si existen datos en el puerto *Wire*, se almacenan los datos en una variable para poder leerlos en el puerto serial.

3.6.3.2 Conexiones de hardware para el control de la MTU

La conexión para la comunicación I2C es relativamente sencilla, en la ilustración 3-46 se muestra la estructura y la topología del bus de comunicación. Para el proceso de integración se utilizan los dos microcontroladores que gobiernan los proceso, estos actúan como los esclavos de la conexión o RTUs y un microcontrolador que actúa como maestro o MTU.

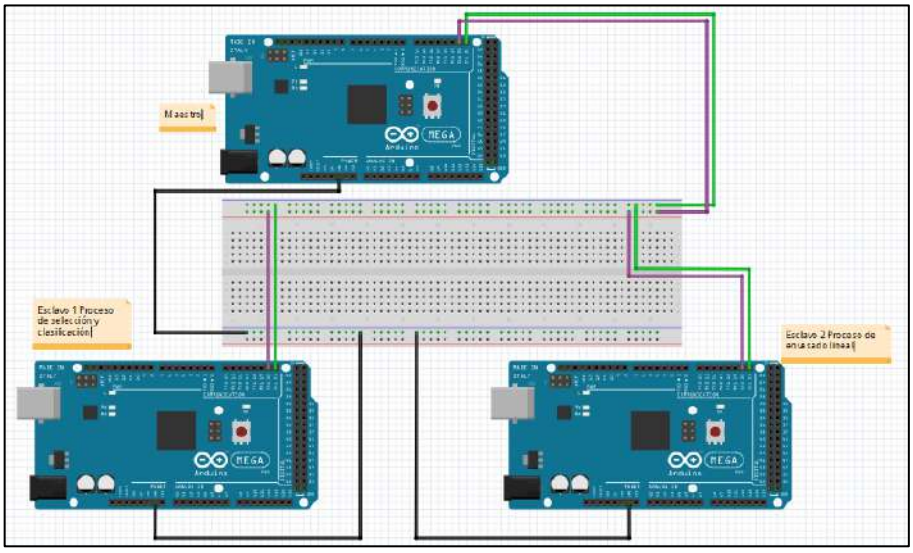


Ilustración 3-46: Diagrama de conexiones de la comunicación I2C

Realizado por: Romero, Cristian, 2023

3.6.4 Interfaz gráfica HMI

El desarrollo de la GUI para la HMI consta de dos fases fundamentales, la primera etapa es el diseño y desarrollo de los elementos gráficos de la HMI, y la segunda etapa es la integración de los elementos gráficos con los datos obtenidos desde la MTU. Para el desarrollo del algoritmo se usaron dos módulos creados en función de programación orientada a objetos.

La primera etapa de construcción de la HMI consta de las siguientes funciones para su construcción:

- **Módulo *HMI_modulo*:** Es el primer módulo desarrollado, cuenta con todos los elementos necesarios para el funcionamiento de la HMI. Ver *script* en el anexo K.
- **Librerías:** Para la creación de los elementos visuales es necesario importar las librerías que permiten su uso, la librería *tkinter* permite el uso de botones, ventanas, etiquetas, fotos y demás elementos gráficos. La librería *PIL* permite la gestión de imágenes dentro del programa.
- ***Class HMI (Frame):*** Define los atributos, argumentos y métodos usados para el desarrollo de la interface HMI.
- ***Def __init__(self):*** Es la función de construcción del algoritmo**,** las funciones llamadas dentro del constructor se ejecutan directamente iniciado el programa.
- ***Def visual (self):*** En esta función se construyen y ejecutan los elementos visuales de la HMI, entre los que tenemos:
    - **Botones:** Permiten la ejecución de comandos mediante la pulsación de un elemento en la HMI, su principal función es encender y apagar los procesos productivos.
    - **Imágenes:** Son usadas principalmente para que el usuario entienda el funcionamiento del proceso, con las imágenes se busca que la interface sea más intuitiva y fácil de usar.
    - **Etiquetas:** Usadas para los encabezados del programa, así como para etiquetar ciertos elementos que forman parte de la interface.
    - **Tablas:** Son usadas para mostrar los eventos de cada uno de los procesos en la HMI.
- **Módulo *main*:** Dentro de este módulo se ejecuta la interface gráfica HMI, mediante el llamado de la librería y el módulo antes desarrollado. Ver *script* en el anexo K.
- **Librerías:** Las librerías usadas son *tkinter* y se llama para su ejecución al módulo HMI anteriormente desarrollado.
- **Def main ():** Esta función contiene la creación de la ventana principal, así como los llamados para ejecutar todos los elementos incluidos en la misma ventana.

En la ilustración 3-47 está representada la interface gráfica HMI del proyecto, cuenta con todos los elementos mencionados anteriormente.

Ilustración 3-47: Interface gráfica HMI

Realizado por: Romero, Cristian, 2023

En la segunda etapa del desarrollo de la interface HMI está formada por las funciones que reciben y envían los datos al microcontrolador MTU, entre las que tenemos:

- **Módulo *HMI_modulo* (*Frame*):** Dentro del primer módulo se encuentran todas las funciones de lectura y envío de datos. Ver *script* en el anexo K.
- ***Def iniciar_com* (*self*):** En esta función se ejecutan las líneas de código para iniciar el proceso de comunicación serial entre la interface Python y el microcontrolador MTU. En una variable se almacenan los datos del objeto conexión. En la ilustración 3-48 se muestra el diagrama de flujo de la función.

Ilustración 3-48: Función de comunicación serial

Realizado por: Romero, Cristian, 2023

- ***Def cerrar_com (self):*** Esta función permite cerrar la comunicación serial con el microcontrolador. En la ilustración 3-49 se muestra el proceso de cierre de la comunicación serial.

Ilustración 3-49: Función de cierre de comunicación serial

Realizado por: Romero, Cristian, 2023

- ***Def encender_maq (self):*** Esta función está ligada al botón de inicio y permite el envío de un dato a la MTU, este dato permite encender los procesos y un indicador visual en la interface HMI. La función es igual para los dos procesos. En la ilustración 3-50 se muestra el diagrama de flujo del proceso dentro de la función.

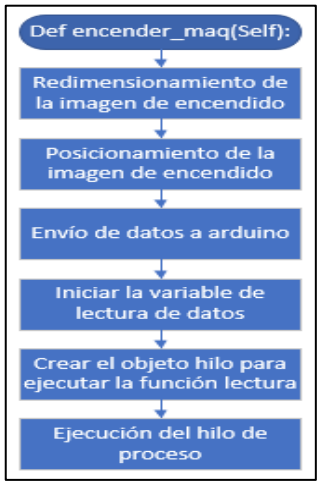


Ilustración 3-50: Función de encendido del proceso

Realizado por: Romero, Cristian, 2023

- ***Def apagar_maq (self):*** Esta función está ligada al botón finalizar y permite el envío de otro dato que permite el apagado de los procesos y cambia de estado el indicador visual de la HMI. En la ilustración 3-51 se representa el diagrama de flujo del proceso dentro de la función.

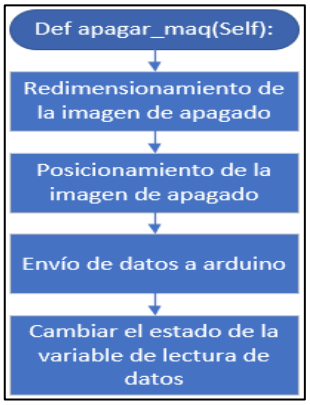


Ilustración 3-51: Función de apagado del proceso

Realizado por: Romero, Cristian, 2023

- ***Def lectura_com* (*self*):** El objetivo de esta función es la constante lectura de datos desde el microcontrolador a la interface, el fin de la lectura es adquirir datos del proceso, control y supervisión del proceso junto con el envío de datos a la tabla de eventos. En la ilustración 3-52 se muestra el diagrama de flujo del proceso que lleva a cabo la función.

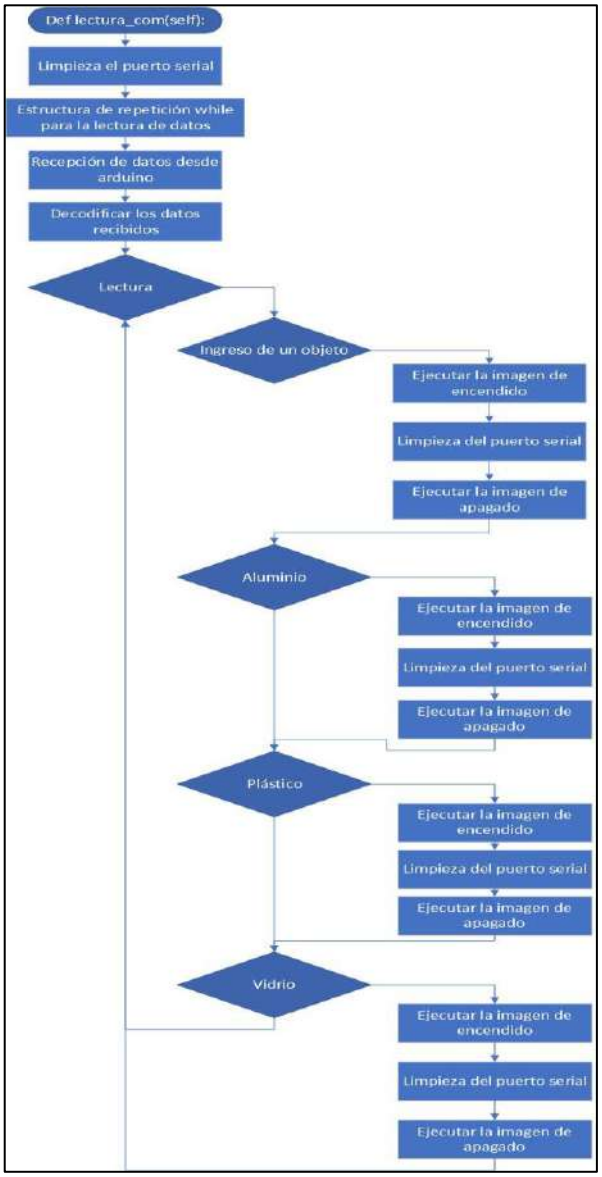


Ilustración 3-52: Función de lectura de datos

Realizado por: Romero, Cristian, 2023

- **Tabla de eventos:** Permite registrar los eventos de los dos procesos productivos, en el proceso de selección y clasificación de botellas muestra la cantidad de botellas y el tipo de material de cada una, en la envasadora lineal se muestra el tipo de objeto como la cantidad de objetos envasados. En la ilustración 3-53 se muestra la tabla de eventos de los procesos industriales.

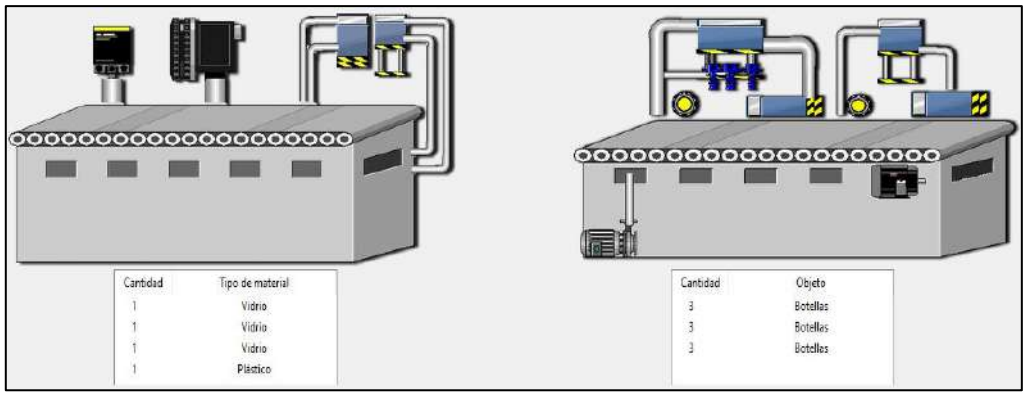


Ilustración 3-53: Función de lectura de datos desde el microcontrolador

Realizado por: Romero, Cristian, 2023

- **Base de datos:** Para la creación de la base de datos se usó SQLite, es una popular base de datos relacional de código abierto, fácil de usar y para su uso no es necesario contar con un servidor lo que la hace autónoma y puede funcionar dentro del contexto de la interface HMI. En el anexo K se encuentra el módulo para la conexión con la base de datos.
    - El primer paso para el desarrollo de la base de datos es crear un módulo que permita realizar una conexión entre la interface HMI y la base de datos. En la ilustración 3-54 se muestra la arquitectura de la base de datos.

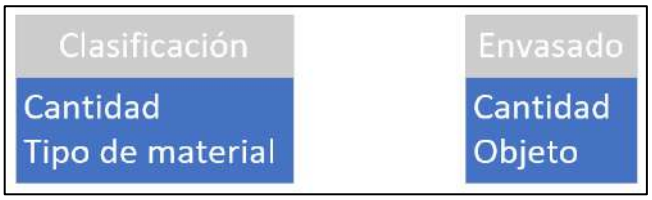


Ilustración 3-54: Arquitectura de la base de datos del sistema SCADA

Realizado por: Romero, Cristian, 2023

    - A continuación, se crea el módulo que permite crear la base de datos con sus respectivas tablas, una para cada proceso, además el módulo cuenta con funciones encargadas de eliminar y guardar los datos obtenidos desde los RTUs. En la ilustración 3-55 se muestra la base de datos en el BD Browser.

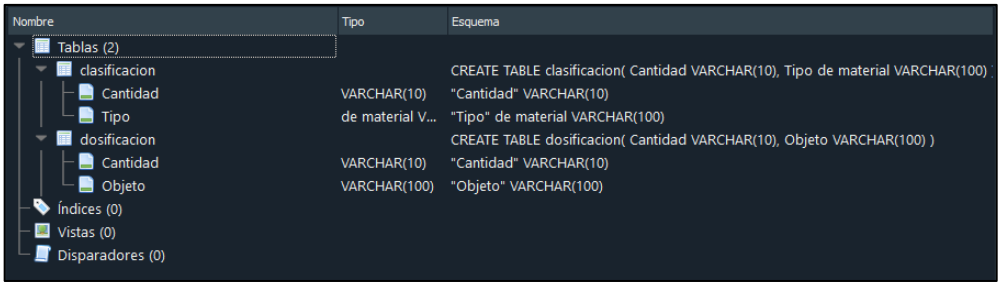


Ilustración 3-55: Base de datos desde el DB Browser

Realizado por: Romero, Cristian, 2023

- El proceso de almacenamiento se realiza de manera automática, cada que ingresa un dato a la MTU se lo envía directamente a la base de datos, con lo que respecta al proceso de crear y eliminar la base de datos, la interface HMI cuenta con un menú que permite realizar estos cambios, la ilustración 3-56 muestra el menú.

Ilustración 3-56: Menú base de datos

Realizado por: Romero, Cristian, 2023

3.7 Verificación del proceso de selección y clasificación de botellas

El primer análisis de funcionamiento y resultados se realiza en el proceso de selección y clasificación de botellas, este proceso cuenta con el sistema de reconocimiento visual que usa visión artificial, además del sistema de mando y potencia. En la ilustración 3-57 se puede observar el estado final del proceso de selección y clasificación de botellas.

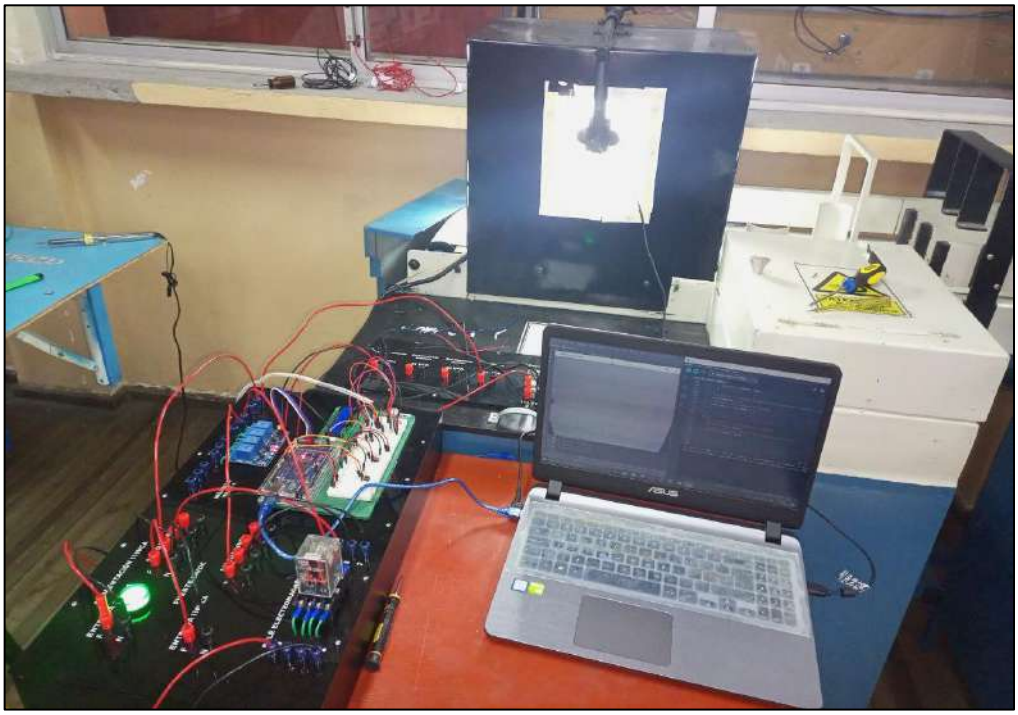


Ilustración 3-57: Proceso de selección y clasificación de objetos

Realizado por: Romero, Cristian, 2023

3.7.1 Verificación del sistema de visión artificial

Para la verificación e interpretación de los resultados obtenidos en el sistema de visión artificial, el primer atributo a poner a prueba es el tiempo de ejecución del modelo, el tiempo varía en función de la complejidad modelo, el tamaño del programa y las capacidades de procesamiento del ordenador. En la tabla 3-25 (A) se muestran los datos de veinte tomas realizadas con el fin de medir el tiempo de ejecución del modelo.

Tabla 3-25 (A): Tiempo de ejecución del modelo

Prueba	Tiempo (segundos)
1	8,18
2	8,50
3	8,06
4	8,34
5	8,58
6	8,44
7	8,31
8	8,35
9	8,11
10	8,48
11	8,41
12	8,24
13	8,25
14	8,75
15	8,15

Realizado por: Romero, Cristian, 2023

Con los datos obtenidos se puede analizar que el tiempo de respuesta promedio de la ejecución del modelo es de 8 segundos. No existen variaciones de consideración, se puede concluir que el modelo tiene tiempos de respuesta estables. En la tabla 3-25 (B) se indica el promedio y los tiempos máximos y mínimos de ejecución.

Tabla 3-25 (B): Tiempo promedio, máximo y mínimo

	Tiempo (Segundos)
Promedio	8,34
Máximo	8,75
Mínimo	8,06

Realizado por: Romero, Cristian, 2023

Dentro de la revisión de la eficiencia del sistema de reconocimiento se puede verificar la probabilidad de acierto del modelo junto con el tiempo de respuesta del mismo, para la prueba se toman en consideración los tres elementos que se procesarán. En la ilustración 3-58 se aprecia el proceso de detección junto con la probabilidad del reconocimiento.

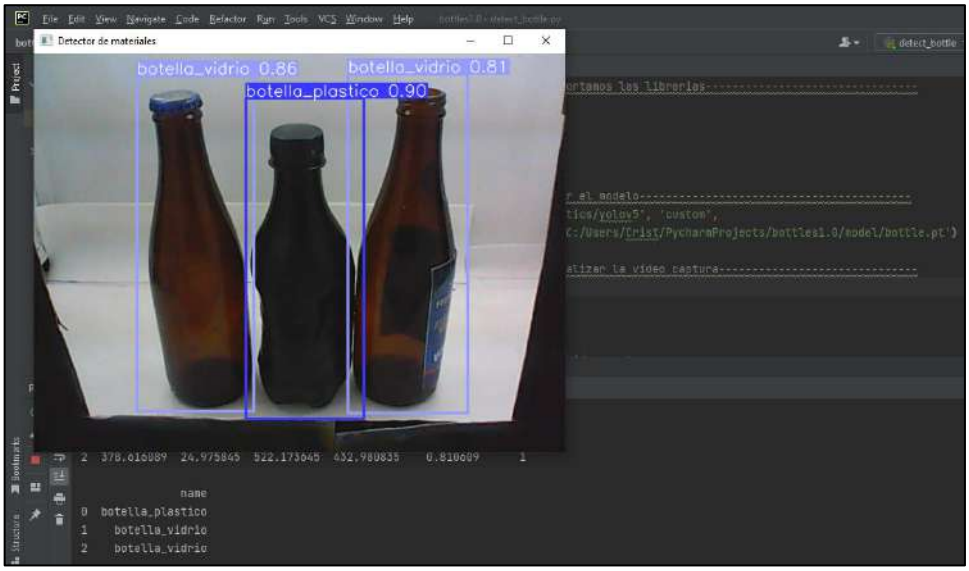


Ilustración 3-58: Detección de objetos del sistema de visión artificial

Realizado por: Romero, Cristian, 2023

En la tabla 3-26 (A) se enlisto el nivel de probabilidad junto con el tiempo de detección de cada una de las botellas a clasificar, el mayor nivel de probabilidad se define con el valor 1 y el menor 0.

Tabla 3-26 (A): Tiempo de detección y probabilidad

Material	Tiempo (segundos)	Probabilidad
Vidrio	1,85	0,74
	1,85	0,84
	1,79	0,77
	1,87	0,76
	1,80	0,87
	1,81	0,72
	1,88	0,77
	1,88	0,81
	1,87	0,78
	1,84	0,79
Plástico	1,62	0,87
	1,74	0,77
	1,78	0,78
	1,76	0,78
	1,78	0,87

	1,79	0,91
	1,82	0,84
	1,83	0,93
	1,87	0,86
	1,83	0,92
Aluminio	1,80	0,77
	1,82	0,81
	1,88	0,93
	1,82	0,90
	1,92	0,93
	1,88	0,93
	1,82	0,92
	1,85	0,90
	1,88	0,93
	1,92	0,83

Realizado por: Romero, Cristian, 2023

De los datos de la tabla 3-26 (A) se puede destacar que el tiempo de respuesta en el reconocimiento de cada uno de los objetos no varía demasiado, así como los valores de probabilidad de detección. En la tabla 3-26 (B) se muestran los valores máximos y mínimos, así como los promedios de los dos atributos analizados.

Tabla 3-26 (B): Promedio, máximo y mínimo

	Tiempo (segundos)	Probabilidad
Promedio	1,83	0,84
Máximo	1,92	0,93
Mínimo	1,62	0,72

Realizado por: Romero, Cristian, 2023

3.7.2 Tiempos de ejecución del proceso de selección y clasificación de botellas

En referencia al tiempo total de ejecución del proceso de selección y clasificación de botellas, se puede señalar que es el tiempo que tarda en realizar el proceso de clasificación completo. El proceso va desde el ingreso del objeto, su detección por parte del sistema de visión artificial y su clasificación por parte de los actuadores neumáticos, finalizando en los depósitos ya sean de plástico, vidrio o aluminio.

En la tabla 3-27 (A) se aprecian los tiempos de ejecución completos en función del tipo de material procesado. Se tomaron en cuenta los tres elementos procesados.

Tabla 3-27 (A): Tiempo de procesamiento de las botellas

Material	Tiempo (segundos)	Material	Tiempo (segundos)	Material	Tiempo (segundos)
Vidrio	12,16	Plástico	15,50	Aluminio	11,54
	12,55		15,22		11,91
	12,31		15,21		11,71
	12,84		15,24		11,62
	12,88		15,36		11,98
	12,18		15,10		11,61
	12,30		15,60		11,90
	12,45		15,55		11,85
	12,74		15,15		11,73
	12,57		15.41		11,61

Realizado por: Romero, Cristian, 2023

Se puede apreciar que existe una pequeña variación en los tiempos de procesamiento entre cada material, esto se debe al recorrido que tiene cada uno en la banda, el primer depósito es el del material aluminio, por lo tanto, su tiempo de procesamiento es menor. A continuación, está el depósito de las botellas de vidrio y al final del proceso se encuentra el depósito de las botellas de plástico que cuentan con el mayor tiempo de procesamiento.

En la tabla 3-27 (B) se presentan los datos promedio de tiempo de procesamiento, así como los valores máximos y mínimos de cada material.

Tabla 3-27 (B): Promedios, máximos y mínimos

	Tiempo (segundos) Vidrio	Tiempo (segundos) Plástico	Tiempo (segundos) Aluminio
Promedio	12,50	15,33	11,75
Máximo	12,88	15,60	11,98
Mínimo	12,16	15,10	11,54

Realizado por: Romero, Cristian, 2023

3.8 Verificación del proceso de envasado lineal

Dentro del análisis de resultados obtenidos en el proceso de envasado lineal, se pueden determinar los tiempos de ejecución total, a diferencia del proceso de clasificación no cuenta con sistemas externos de reconocimiento de objetos. En la ilustración 3-59 se aprecia el estado final del proceso de envasado lineal.

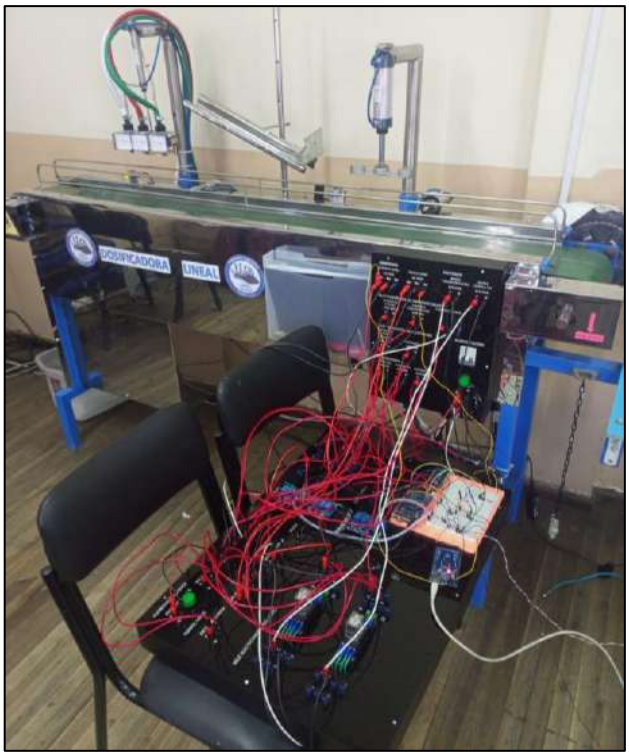


Ilustración 3-59: Proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

3.8.1 Tiempos de ejecución del proceso de envasado lineal

El proceso de envasado cuenta con un tiempo total de ejecución, definido como el tiempo que se tarda en envasar el fluido en las botellas, así como tapar las mismas. El proceso inicia desde el ingreso de los envases vacíos en el área de llenado hasta su salida del área de tapado. Las botellas son procesadas en grupos de tres unidades.

En la tabla 3-28 (A) se muestran los tiempos tomados en las pruebas de funcionamiento del proceso.

Tabla 3-28 (A): Tiempo de ejecución del proceso de envasado

Prueba	Tiempo (segundos)
1	48
2	46
3	48
4	48
5	49
6	47
7	46
8	46
9	46
10	47

11	47
12	46
13	46
14	47
15	47

Realizado por: Romero, Cristian, 2023

Se puede apreciar que el tiempo del proceso tiene una pequeña variación, en la tabla 3-28 (B) se presenta el tiempo promedio de llenado además de los tiempos máximos y mínimos en los que puede variar el proceso.

Tabla 3-28 (B): Tiempo promedio, máximo y mínimo

	Tiempo (segundos)
Promedio	46,9
Máximo	49
Mínimo	46

Realizado por: Romero, Cristian, 2023

3.9 Verificación del sistema SCADA

Para el proceso de análisis y verificación del sistema SCADA se realizan distintas pruebas, principalmente de tiempos de respuesta de la interface en el envío y recepción de datos, así como en tiempos de ejecución de la misma. El sistema SCADA envía y recibe señales desde el terminal maestro o MTU y esta a su vez envía y recibe las señales a los terminales esclavo o RTU.

3.9.1 Prueba de la interface HMI

Antes de realizar la conexión e integración de los procesos, se desarrollan pruebas simulando los procesos industriales. Para la simulación se usa un potenciómetro y una placa microcontrolador Arduino, el objetivo es enviar las señales que requiere la interface HMI del sistema SCADA generando una simulación procesos integrados. En la ilustración 3-60 se muestra el proceso de simulación de la interface HMI.

Ilustración 3-60: Simulación de los procesos industriales

Realizado por: Romero, Cristian, 2023

Una vez se determina que la interface muestra la recepción de las señales mediante los leds y la tabla de eventos, se verifica en el programa *DB Browser* si el sistema envía los datos a la base de datos en *SQLite*. En la ilustración 3-61 (A) proceso de selección y clasificación de botellas y en la ilustración 3-61 (B) proceso de envasado lineal, se muestra el almacenamiento de la información de la tabla de eventos en la base de datos.

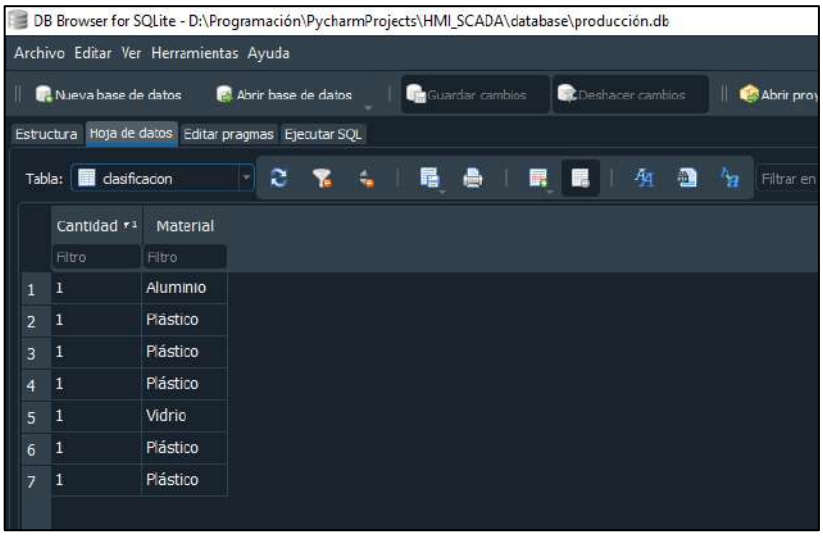


Ilustración 3-61 (A): Tabla del proceso de clasificación de botellas

Realizado por: Romero, Cristian, 2023

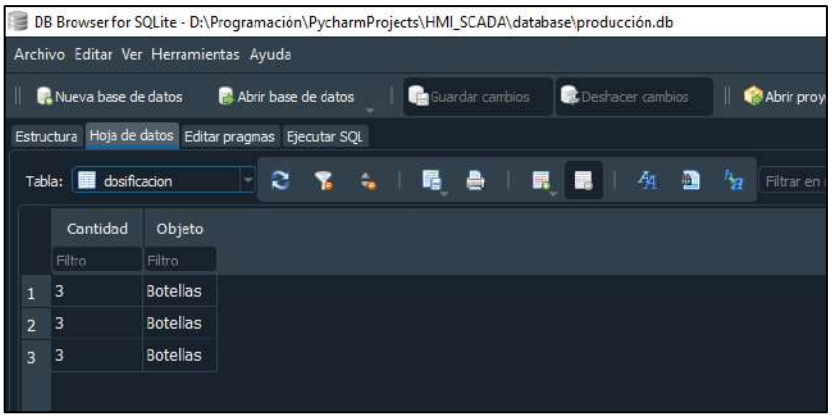


Ilustración 3-61 (B): Tabla del proceso de envasado lineal

Realizado por: Romero, Cristian, 2023

3.9.2 Integración de los procesos industriales

El sistema SCADA integra dos procesos industriales mediante el usa de una MTU o dispositivo maestro que envía y recibe datos. La MTU controla los tiempos de envío y recepción de datos a los dos dispositivos esclavos o RTU, existe un dispositivo esclavo por cada proceso. En la ilustración 3-62 se representa la integración de los procesos industriales.

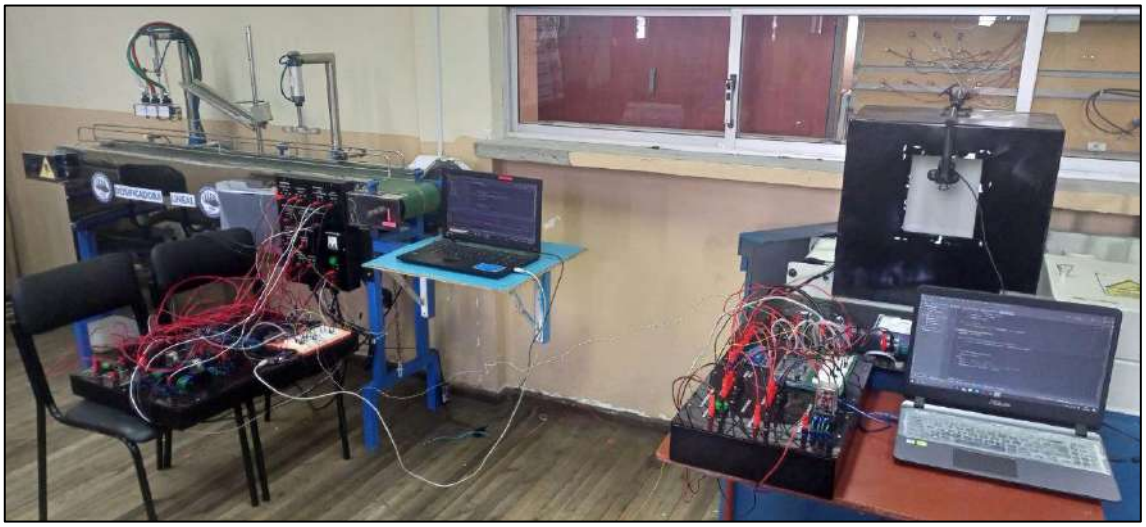


Ilustración 3-62: Integración de los procesos industriales

Realizado por: Romero, Cristian, 2023

La verificación y análisis de los resultados se realiza con los procesos integradas en el sistema SCADA, para la verificación se realizan pruebas de tiempos de ejecución, envío y recepción de datos desde la interface HMI hacia los procesos industriales.

3.9.2.1 Verificación de la ejecución de la interface HMI

Dentro del análisis y verificación de la interface HMI se realiza la toma de tiempos de ejecución, lo que permite determinar la eficiencia de la ejecución de la interface. En la tabla 3-29 (A) se aprecian los tiempos de ejecución.

Tabla 3-29 (A): Tiempo de ejecución de la interface HMI

Prueba	Tiempo (segundos)
1	1,22
2	1,23
3	1,22
4	1,22
5	1,26
6	1,22
7	1,23
8	1,28
9	1,22
10	1,22
11	1,23
12	1,22
13	1,23
14	1,22
15	1,22

Realizado por: Romero, Cristian, 2023

En los tiempos de ejecución inicial de la interface se puede aprecia que la variación entre ejecuciones es insignificante, además el tiempo que tarda la interface en ejecutarse es aceptable. En la tabla 3-29 (B) se calcula el tiempo promedio de ejecución, también el tiempo máximo y mínimo.

Tabla 3-29 (B): Tiempo promedio, máximo y mínimo

	Tiempo (segundos)
Promedio	1,23
Máximo	1,28
Mínimo	1,22

Realizado por: Romero, Cristian, 2023

3.9.2.2 Verificación de los tiempos de envío de datos

Los tiempos de envío de datos se verifican en función de los datos que se envían desde la interface a los procesos, estos datos son encendido y apagado. La interface cuenta con botones que permiten realizar estas acciones de manera fácil e intuitiva. En la tabla 3-30 (A) se presentan los datos obtenidos.

Tabla 3-30 (A): Tiempo de envío de datos desde la HMI

Proceso	Tiempo (segundos) Encendido	Tiempo (segundos) Apagado	Proceso	Tiempo (segundos) Encendido
Selección y clasificación de botellas	4,97	5,26	Envasado lineal	4,23
	4,53	5,16		4,06
	4,70	5,76		4,43
	4,68	5,45		4,79
	4,30	5,35		4,19
	4,95	5,05		4,04
	4,61	5,53		4,10
	4,59	5,46		4,57
	4,09	5,82		4,00
	4,11	5,05		4,74

Realizado por: Romero, Cristian, 2023

La variación en los tiempos de encendido y apagado dependen de la programación en cada una de las RTU, así como el número de elementos por proceso. En el caso del proceso de selección y clasificación de botellas se cuenta con una programación más pequeña al contar con menos actuadores y sensores. El proceso de envasado cuenta con un mayor número de actuadores y sensores lo que aumenta la carga de procesamiento de la RTU generando así retardos en el proceso de envío de datos.

En la tabla 3-30 (B) se resume el promedio y el tiempo máximo y mínimo por cada envío de datos a los procesos.

Tabla 3-30 (B): Tiempos promedios, máximos y mínimos

Proceso	Selección y clasificación de botellas		Envasado lineal
	Tiempo (segundos) Encendido	Tiempo (segundos) Apagado	Tiempo (segundos) Encendido
Promedio	4,55	5,39	4,32
Máximo	4,97	5,82	4,79
Mínimo	4,09	5,05	4,00

Realizado por: Romero, Cristian, 2023

3.9.2.3 Verificación de los tiempos de recepción de datos

A la hora de recibir datos, el sistema SCADA los presenta de manera visual en la interface HMI, esta presentación se realiza en la tabla de eventos y por los señalizadores visuales. Los señalizadores led visuales son los primeros indicadores en la HMI y en la tabla de eventos se presenta la finalización del proceso. En la tabla 3-31 (A) se presentan los datos de recepción de la información del proceso de selección y clasificación de botellas.

Tabla 3-31 (A): Tiempo de recepción de datos del proceso de clasificación

Dato	Tiempo (segundos)	Dato	Tiempo (segundos)
Ingreso al área de reconocimiento visual	1,07	Tipo de material	2,50
	1,40		2,90
	1,76		2,92
	1,87		2,50
	1,73		2,19
	1,38		2,67
	1,79		2,26
	1,18		2,42
	1,09		2,75
	1,38		2,99

Realizado por: Romero, Cristian, 2023

En la tabla 3-31 (B) se listan los tiempos de recepción de datos realizados en el proceso de envasado lineal.

Tabla 3-31 (B): Tiempo de recepción de datos del proceso de envasado

Dato	Tiempo (segundos)	Dato	Tiempo (segundos)
Sensor 1 Área de llenado	1,11	Sensor 2 Área de tapado	1,96
	1,53		1,40
	1,24		1,85
	1,03		1,75
	1,19		1,06
	1,06		1,38
	1,94		1,77
	1,12		1,41
	1,12		1,94
	1,03		1,22

Realizado por: Romero, Cristian, 2023

En los dos procesos la recepción del dato que tarda más en obtenerse es el tipo de material, ya para el procesamiento es necesario más sistemas, para clasificar un objeto primero se usa el sistema de reconocimiento visual, este envía información a la RTU y esta a su vez a la MTU para presentarla en el HMI.

En la tabla 3-31 (C) se presenta el resumen de tiempos de recepción de datos, promedio, tiempo máximo y mínimo.

Tabla 3-31 (C): Tiempos promedios, máximos y mínimos

Proceso	Selección y clasificación de botellas		Envasado lineal	
	Tiempo (segundos) Ingreso	Tiempo (segundos) Clasificación	Tiempo (segundos) Sensor 1	Tiempo (segundos) Sensor 2
Promedio	1,46	2,61	1,24	1,57
Máximo	1,87	2,99	1,94	1,96
Mínimo	1,07	2,19	1,03	1,06

Realizado por: Romero, Cristian, 2023

CAPÍTULO IV

4 GESTIÓN DEL PROYECTO

4.1 Cronograma de actividades

Las actividades realizadas para el desarrollo del proyecto se dividen en función del cronograma presentado en la tabla 4-1.

Tabla 4-1: Cronograma de actividades

	Septiembre	Octubre				Noviembre				Diciembre				Enero				Febrero		
	1	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3
Revisión de elementos *hardware*																				
Investigación de herramientas de *hardware* y *software open source* para control de procesos																				
Selección de las herramientas de *hardware* y *software open source*																				
Diseño de los módulos didácticos																				
Adquisición de los elementos de *hardware* y *software*																				
Construcción de los módulos didácticos																				
Desarrollo del algoritmo de visión artificial para el proceso de selección y clasificación																				
Desarrollo del algoritmo de control del proceso de selección y clasificación de botellas																				
Desarrollo del algoritmo de control del proceso de envasado lineal																				
Ensamblaje del circuito del proceso de selección y clasificación de botellas																				
Ensamblaje del circuito del proceso de envasado lineal																				
Desarrollo del algoritmo para la integración de los procesos industriales																				
Construcción de la interface HMI del sistema SCADA																				
Pruebas de funcionamiento y validación del sistema SCADA																				

Realizado por: Romero, Cristian, 2023

4.2 Recursos

4.2.1 Costos directos

Son todos los costos generados por las actividades pertenecientes al proceso de construcción. En la tabla 4-2 se presentan los valores de los costos directos generados para desarrollar el sistema SCADA.

Tabla 4-2: Costos directos

Cantidad	Materiales	Costos (dólares)
4	Bases para los módulos didácticos	260
4	Placas acrílicas para los módulos didácticos	100
3	Arduino Mega AT2560	-
3	Pantallas LCD 16x2	-
3	Protoboard	-
3	Bases para Arduino, LCD y protoboard	30
2	Fuente de alimentación de 24 VCD	20
1	Fuente de alimentación de 12 VCD para Arduino	6
4	Luces piloto	6,40
2	Módulo de 4 relés	-
1	Módulo de 8 relés	-
2	Disyuntores	-
15	Metros de cable rojo calibre N°14	6
15	Metros de cable negro calibre N°14	6
10	Metros de cable verde calibre N°14	4
198	Jack tipo banana hembra	44,55
100	Jack tipo banana macho	17,50
Total		500,45

Realizado por: Romero, Cristian, 2023

4.2.2 Costos indirectos

Son los costos que se generan por el proceso de construcción de los módulos a partir de ciertas actividades que no necesariamente pertenecen al proceso de fabricación. En la tabla 4-3 se presentan los valores generados por estos costos.

Tabla 3-3: Costos indirectos

Descripción	Costos (dólares)
Transporte	48
Imprevistos	30
Total	78

Realizado por: Romero, Cristian, 2023

4.2.3 Costos totales

Para determinar los costos totales se toman en cuenta todos los costos generados en el proceso de construcción del sistema SCADA. En la tabla 4-4 se detalla el valor total de costos en el proceso de desarrollo del tema de tesis.

Tabla 4-4: Costos totales

Descripción	Costos (dólares)
Costos directos	500,45
Costos indirectos	78
Total	578,45

Realizado por: Romero, Cristian, 2023

4.2.4 Talento humano

Cristian Romero
Ing. Marcelo Calispa
Ing. Eduardo García

4.2.5 Recursos materiales

Entre los elementos presentados en la tabla 4-2 que no cuentan con un valor en costos, existen elementos facilitados por el ingeniero Eduardo García y elementos reutilizados de los procesos integrados en el sistema SCADA.

En el desarrollo de las pruebas de validación se usaron elementos propios del proyecto, principalmente elementos de *software open source* que no incurren en costos.

CAPÍTULO V

5 CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Se realizó un estudio comparativo de las diferentes posibles herramientas que permitan habilitar los procesos industriales didácticos, además que permitan desarrollar un sistema SCADA con todas las características que conllevan. Se seleccionaron como herramienta de *hardware* a los microcontroladores Arduino, los mismo que cumple con las funciones de dispositivo RTU en el control de los procesos y MTU para el control de los dispositivos en los procesos, cabe mencionar que los dispositivos Arduino cuentan con IDE de programación para el control junto con conexiones de comunicación I2C. Como herramienta de *software* el lenguaje de programación seleccionado es Python, gracias a su versatilidad permite el desarrollo del algoritmo de visión artificial junto con el sistema SCADA y su interfaz gráfica HMI, cuenta con compatibilidad con los dispositivos *hardware*.

Se realizó el diseño CAD de los módulos de actuadores y control de los procesos de clasificación y envasado, para los módulos de actuadores se tomó en cuenta el número de dispositivos con los que cuentan los procesos y los módulos de control se crearon en función de los dispositivos de mando a usar, en el desarrollo se usó la herramienta de diseño asistido por computadora Inventor. En la construcción de la parte superior de los módulos se usó acrílico PMMA gracias a su alta resistencia a los golpes y estética, en la base se usó chapa metálica por su alta resistencia y durabilidad, el corte de las piezas y la rotulación fueron creadas por corte laser en función de las medidas de los diseños CAD.

Se diseñó el sistema SCADA en base a la integración de los dos procesos industriales que se encuentran en la carrera de Ingeniería Industrial. El primer paso es la puesta en marcha de los dos procesos usando los elementos de control *open source* seleccionados. En el proceso de selección y clasificación de botellas se desarrolló el algoritmo de visión artificial que permite la clasificación, así como el programa de control para la RTU Arduino, en el proceso de envasado lineal solo es necesario el programa de control para la RTU. El proceso de desarrollo del sistema finaliza con la programación del protocolo de comunicación I2C entre los dispositivos maestros y esclavos, así también se creó la interface HMI del sistema en conjunto con la base de datos que permite la recepción de información de los eventos de los dos procesos.

Se ejecutaron las verificaciones y comprobaciones de los distintos sistemas, desde el proceso de comunicación entre la RTU y el programa de visión artificial, la primera verificación que se da es la ejecución de la VA se da en un tiempo promedio de 8,34 segundos y el tiempo promedio de detección es de 1,83 segundos. El proceso de clasificación de botellas tiene un tiempo de

ejecución que depende de cada material, para la botella de aluminio el proceso de clasificación tarda 11,75 segundos en promedio, la botella de vidrio 12,50 segundos y la botella de plástico tarda 15,33 segundos. El proceso de envasado lineal tiene un tiempo promedio de ejecución de 46,9 segundos por un grupo de 3 botellas, para finalizar se verificó los tiempos de ejecución del sistema SCADA, los tiempos promedios de ejecución de la interface HMI son bajos, en promedio se ejecuta en 1,23 segundos y los tiempos de envío de datos son 4,75 segundos y los de recepción son de 1,72 segundos.

RECOMENDACIONES

Se recomienda realizar un análisis exhaustivo de las herramientas a usar, ya que se debe interconectar y eso se logra si son compatibles o disponen de librerías para su comunicación, además de disponer de los protocolos de comunicación necesarios en función de una topología seleccionada. Es importante tomar en cuenta el precio y la disponibilidad de los elementos de *hardware* ya que *open source* no significa gratis.

En el diseño de los módulos es necesario tener presente el número de elementos que van a formar parte del sistema, ya que los módulos van a contar con las entradas y salidas de cada uno de los procesos. La construcción se debe realizar con elementos resistentes, duraderos con sus respectivas etiquetas que faciliten el proceso de conexión.

Es importante una inducción en el uso de los módulos, ya que van a ser usados principalmente por estudiantes para sus prácticas de automatización y es importante evitar riesgos eléctricos por el mal uso, también se debe procurar el cuidado de los módulos.

Se recomienda el uso de un manual de procedimientos para el uso y conexión de los módulos, junto con el uso de los *scripts* de control desarrollados en el presente trabajo de integración curricular.

En el momento de ejecución del proceso de visión artificial es recomendable el uso de una laptop u ordenador que cuente con un procesador de gama media a alta, es recomendable el uso de un ordenador con un procesador i5 en adelante, como requerimiento mínimo el ordenador debe contar con una tarjeta gráfica dedicada de al menos 2 GB y con una RAM de 8 GB en adelante.

BIBLIOGRAFÍA

1. **AGHENTA, L.O.; & IQBAL, M.T.** "Development of an IoT Based Open Source SCADA System for PV System Monitoring". *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)* [en línea], 2021, (United State of America), pp. 1403-1408. [Consulta: 23 junio 2022] ISBN 978-1-7281-0319-8. Disponible en: https://ieeexplore.ieee.org/document/8861827/.

2. **AGUIRRE, D.** Desarrollo de un sistema SCADA para uso en pequeñas y medianas empresas [en línea]. (Trabajo de titulación). (Grado), Universidad de Piura, Facultad de Ingeniería. Piura, Perú. 2013. pp. 8-9. [Consulta: 10 de abril 2022] Disponible en: https://pirhua.udep.edu.pe/bitstream/handle/11042/1739/ING_527.pdf?sequence.

3. **ANTÓN, S.D.; FRAUNHOLZ, D.; LIPPS, C.; POHL, F.; ZIMMERMANN, M.; & SCHOTTEN, H.D.** "Two decades of SCADA exploitation: A brief history". *2017 IEEE Conference on Applications, Information and Network Security, AINS 2017* [en línea], 2017, (Malaysia) 2018-Janua, pp. 98-104., [Consulta: 25 mayo 2022]. Disponible en: http://arxiv.org/abs/1905.08902.

4. **ARDUINO**. Arduino Mega 2560 Rev3. [en línea]. [Consulta: 22 noviembre 2022]. Disponible en: https://store-usa.arduino.cc/products/arduino-mega-2560-rev3.

5. **ASHRAF, U.; & IQBAL, M.T.** "An open source SCADA for a solar water pumping system designed for Pakistani Conditions". *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* [en línea], 2021, (United State of America), pp. 1403-1408., [Consulta: 18 de junio 2022] ISBN 978-1-6654-1490-6. Disponible en: https://ieeexplore.ieee.org/document/9376009/.

6. **AV ELECTRONICS**. Módulo Relé 4 Canales. [en línea]. [Consulta: 22 enero 2022]. Disponible en: https://avelectronics.cc/producto/modulo-rele-4-canales/.

7. **BOYER, S.** *SCADA: Supervisory Control and Data Acquisition*. 3. United Stated of America: The Instrumentation, Systems, and Automation Society, 2004. ISBN 1-55617-877-8., pp. 53-56.

8. **DELGADO, J.** Sistema de tele-rehabilitación a domicilio con chatbot y microordenador de bajo coste [en línea]. (Trabajo de titulación). (Grado) Universidad de Valladolid,

Departamento de teoría de la señal y comunicaciones e ingeniería telemática. Valladolid, España. 2021. pp. 33-34. [Consulta: 23 enero 2023] Disponible en: https://uvadoc.uva.es/bitstream/handle/10324/50034/TFG-G5214.pdf?sequence=1&isAllowed=y.

9. **DOWNEY, A.** *Think Python* [en línea]. 2da. Needham-Massachusetts: Green Tea Press. [Consulta: 12 enero 2023]. Disponible en: https://www.onlineprogrammingbooks.com/think-python-2nd-edition/.

10. **GARCÍA, S.** Implementación de analizadores de protocolos de comunicaciones SPI, I2C [en línea]. (Trabajo de titulación). (Grado) Universidad Carlos III de Madrid, Escuela Politécnica Superior, Departamento de Tecnología Electrónica, Madrid, España. 2014. pp. 25-26. [Consulta: 24 noviembre 2022] Disponible en: https://e-archivo.uc3m.es/handle/10016/26320.

11. **GAVRISHEV MESHERYAKOV, D.** Deep Learning en microordenadores: aplicación con dispositivo Coral sobre Raspberry Pi [en línea]. (Trabajo de titulación). (Maestría) Universitat Politècnica de València, Escuela Técnica Superior de Ingeniería Informática, Valencia, España. 2022. pp. 14-24. [Consulta: 23 enero 2023]. Disponible en: file:///C:/Users/Crist/Downloads/Gavrishev - Deep Learning en microordenadores aplicacion con dispositivo Coral sobre Raspberry Pi.pdf.

12. **GONZÁLEZ, A.; et al.** *Técnicas y algoritmos básicos de visión artificial* [en línea]. Logroño-España: Universidad de la Rioja, 2006. [Consulta: 15 diciembre 2022]. Disponible en: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf

13. **GONZÁLEZ, J.; et al.** *Introducción al software libre* [en línea]. Barcelona-España: Fundació per a la Universitat Oberta de Catalunya, 2003. [Consulta: 5 enero 2023]. Disponible en: https://dspace-libros.metabiblioteca.com.co/handle/001/344.

14. **GORDON, C.; & DEON, R.** *Practical Modern SCADA Protocols : DNP3, 60870. 5 and Related Systems* [en línea]. Oxford-Great Britain: ELSEVIER, 2004. [Consulta: 28 junio 2022]. Disponible en: https://ebookcentral.proquest.com/lib/espoch/detail.action?docID=226682.

15. **HERRERA, J.; BARRIOS, M. & PÉREZ, S.** "Diseño e implementación de un sistema SCADA inalámbrico mediante la tecnología zigbee y arduino". *Prospectiva* [en línea], 2014,

12(2), pp. 204-205. [Consulta: 21 junio 2022]. ISSN 2216-1368. Disponible en: http://ojs.uac.edu.co/index.php/prospectiva/article/view/290.

16. **HINOJOSA, A.** *PYTHON Paso a paso* [en línea]. Madrid-España: RA-MA, S.A., 2016. [Consulta: 10 enero 2023] Disponible en: https://books.google.com.pe/books?id=Uo6fDwAAQBAJ&lpg=PA1&hl=es&pg=PA1#v=onepage&q&f=false

17. **INSTITUTO NACIONAL DE ESTADISTICAS Y CENSOS**. *Directorio de Empresas y Establecimientos 2020.* [blog]. Quito: INEC, 2021. [Consulta: 8 junio 2022] Disponible en: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Economicas/DirectorioEmpresas/Directorio_Empresas_2020/Boletin_Tecnico_DIEE_2020.pdf.

18. **MINCHALA, L.; OCHOA, E.; VELECELA, D.; ASTUDILLO, F.; & GONZALEZ, J.** "An Open Source SCADA System to Implement Advanced Computer Integrated Manufacturing". *IEEE LATIN AMERICA TRANSACTIONS* [en línea], 2016, (Ecuador) 14(12), pp. 4657-4662. [Consulta: 9 abril 2022] Disponible en: https://ieeexplore.ieee.org/abstract/document/7816994.

19. **MONTES, C.; & PLASENCIA, A.** "Elaboración e implementación de un módulo con PLC para la simulación de un proceso de clasificación de botellas para el laboratoria de la Escuela de Ingeniería Industrial de la ESPOCH" [en línea] (Trabajo de titulación). (Grado) Escuela Superior Politécnica de Chimborazo, Facultad de Mecánica, Escuela de Ingeniería Industrial. Riobamba-Ecuador. 2015. pp. 30-53. [Consulta: 9 abril 2022]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/4153

20. **MOYA, C**. Software orientado a sistemas de control HMI/Scada usando recursos libres y de código abierto, desarrollado sobre plataforma linux [en línea] (Trabajo de titulación). (Grado) Universidad San Francisco, Quito-Ecuador. 2009. pp. 12-68. [Consulta: 19 abril 2022]. Disponible en: http://repositorio.usfq.edu.ec/handle/23000/1103.

21. **OCHOA, S.; & VELECELA, E.** "Propuesta e implementación de una solución alternativa para el control, supervisión y comunicación en procesos industriales mediante programas de código libre" [en línea] (Trabajo de titulación). (Grado) Universidad de Cuenca, Facultad de Ingeniería, Escuela de Ingeniería Electrónica y Telecomunicaciones. Cuenca-Ecuador. 2016. pp. 55-56. [Consulta: 21 junio 2022]. Disponible en:

http://dspace.ucuenca.edu.ec/handle/123456789/24259

22. **PAZMIÑO, L.; & YÉPEZ, M.** "Construcción e implementación de un módulo didáctico para el proceso de llenado de envases para la Escuela de Ingeniería Industrial". [en línea] (Trabajo de titulación). (Grado) Escuela Superior Politécnica de Chimborazo, Facultad de Mecánica, Escuela de Ingeniería Industrial. Riobamba-Ecuador. 2015. pp. 5-45. [Consulta: 26 octubre 2022]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/4162

23. **PÉREZ-LÓPEZ, E.** "Los sistemas SCADA en la automatización industrial." *Revista Tecnología en Marcha* [en línea], 2015, (Costa Rica) 28(4), pp. 3-14. [Consulta: 7 abril 2022]. ISSN 2215-3241. Disponible en: https://revistas.tec.ac.cr/index.php/tec_marcha/article/view/2438.

24. **PORTALO CALERO, J.M.; GONZÁLEZ PÉREZ, I.; CALDERÓN GODOY, A.J. & CALDERÓN GODOY, M.** "Comparativa de entornos Open-Source para sistemas de supervisión aplicables a Smart Grids/Smart Micro-Grids". *XL Jornadas de Automática: libro de actas (Ferrol, 4-6 de septiembre de 2019)* [en línea], 2020, (España), pp. 428-434. [Consulta: 7 abril 2022]. ISBN 9788497497169. Disponible en: https://ruc.udc.es/dspace/bitstream/handle/2183/23752/2019_Portalo-Calero_Comparativa-entornos-open-source-sistemas-supervision-smart-grids.pdf?sequence=3&isAllowed=y.

25. **PYTHON**. *Logo de Python.* [blog]. [Consulta: 24 noviembre 2022] Disponible en: https://www.python.org/community/logos/.

26. **RAMÍREZ GOROSTIZAGA, I.** Diseño hardware y desarrollo de librerías para la implementación de una plataforma open source basada en un microcontrolador PIC como alternativa a la plataforma Arduino UNO. [en línea] (Trabajo de titulación). (Grado) Universitat de les Illes Balears, Escuela Politécnica Superior, Palma, España. 2016. pp. 13-14. [Consulta: 13 enero 2023]. Disponible en: http://dspace.uib.es/xmlui/handle/11201/151533.

27. **RASPBERRY**. *Raspberry*. [blog]. [Consulta: 23 enero 2023] Disponible en: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

28. **RODRIGUES ALVES, T.; BURATTO, M.; DE SOUZA, F.M.; & RODRIGUES, T.V.** "OpenPLC: An open source alternative to automation". *IEEE Global Humanitarian Technology Conference (GHTC 2014)* [en línea], 2014, (United State of America), pp. 585-

589. [Consulta: 12 enero 2023]. ISBN 978-1-4799-7193-0. Disponible en: http://ieeexplore.ieee.org/document/6970342/.

29. **RODRÍGUEZ, A.** *Sistemas SCADA*. 3. Barcelona-España: Marcombo, 2013. ISBN 978-607-707-406-9., pp. 10-51.

30. **SOLIS, D.** Uso de algoritmos de visión artificial para el diseño e implementación de un sistema de control de acceso a través de reconocimiento facial [en línea] (Trabajo de titulación). (Grado) Universidad Técnica Particular de Loja, Loja, Ecuador. 2017. pp. 28-30. Disponible en: https://dspace.utpl.edu.ec/bitstream/20.500.11962/21072/1/Solís Alulima Darío Javier.pdf.

31. **SQLITE.** *Características de SQLite* [blog]. [Consulta: 23 noviembre 2022]. Disponible en: https://sqlite.org/features.html.

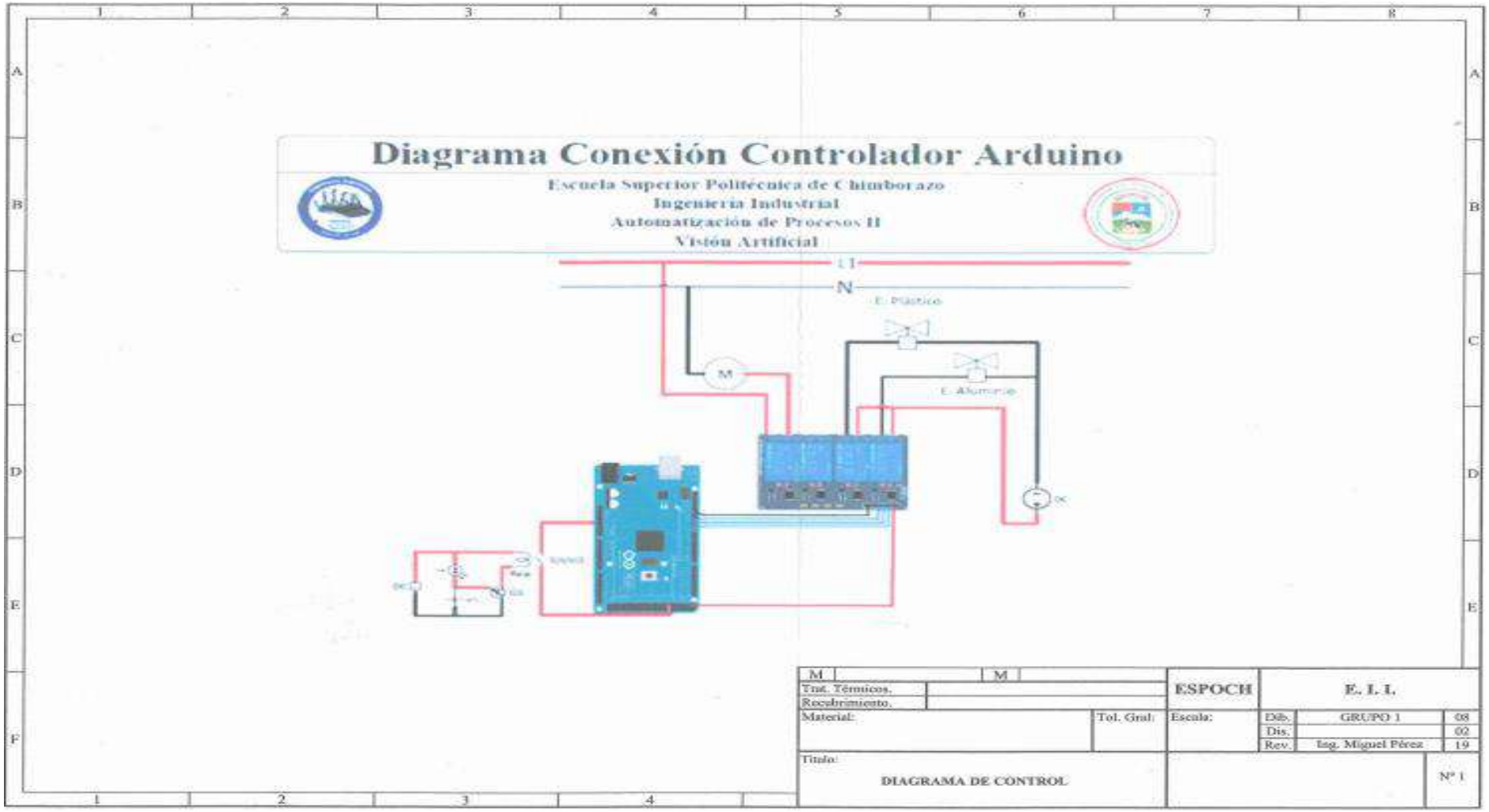
32. **SUAREZ, J.** Diseño e implementación de un sistema SCADA para automatismos, basados en hardware y software libre [en línea] (Trabajo de titulación). (Grado) Universidad tecnológica de Pereira, Pereira, Colombia. 2015. pp. 29-78. [Consulta: 20 abril 2022]. Disponible en: https://repositorio.utp.edu.co/server/api/core/bitstreams/a82c8a4c-3fe3-4bd1-b312-731574c7a37e/content.

33. **THOMAS, M. & MCDONALD, J.** *Power System SCADA and Smart Grids*. Boca Raton-United States of America: Taylor & Francis Group, 2015. ISBN 978-1-4822-2675-1., pp. 5-83.

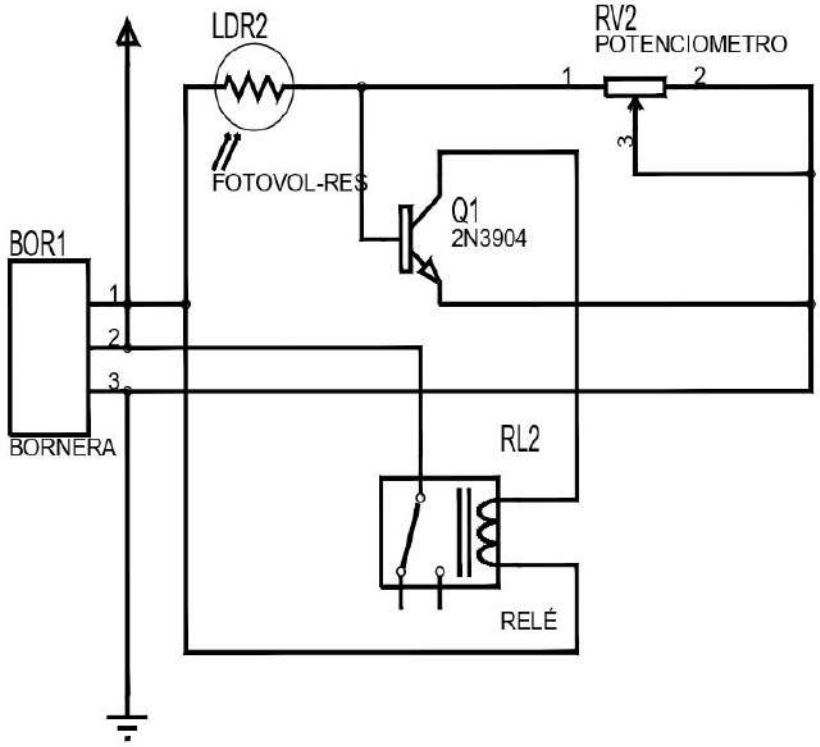
34. **TIERRA, J. & GALARZA, M.** "Implementación de un sistema de control y monitoreo en base al procedimiento de imagenes digitales en los sistemas de visión artificial aplicado al reconocimiento de la máquina selectora de botellas en el laboratorio de la Escuela de Ingeniería Industrial" [en línea] (Trabajo de titulación). (Grado) Escuela Superior Politécnica de Chimborazo, Facultad de Mecánica, Escuela de Ingeniería Industrial. Riobamba-Ecuador. 2017. pp. 7-69. [Consulta: 5 mayo 2022]. Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/9346/1/85T00483.pdf.

ANEXOS

ANEXO A: PLANOS DEL ESTADO DE SITUACIÓN ACTUAL DEL PROCESO DE SELECCIÓN Y CLASIFICACIÓN DE BOTELLAS



N°	Denominación
Q1	Transistor 2N3904
LDR2	Resistencia fotovoltaica
RV1	Potenciómetro de 10K
RL2	Relé SRD 5VCD
BOR2	Bornera de 3 entradas



	Fecha	Nombre		
Div.	13/10/20	C. Romero	Escuela Superior Politécnica de Chimborazo Facultad de Mecánica Escuela de Ingeniería Industrial	Escala
Rev.	22/10/20	E. García		
Apr.	22/10/20	M. Calispa		
Firma			Circuito sensor fotovoltaico	Hoja 1 - 1

Diagrama de Potencia

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

Escuela de Ingeniería Industrial

Automatización de Procesos II

Máquina Clasificadora deBotellas Mediante Visión Artificial

TENSIÓN 110V/AC

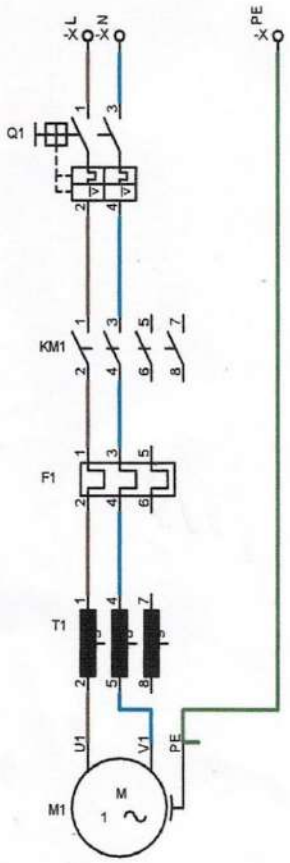
DISYUNTOR

CONTACTOR

GUARDAMOTOR

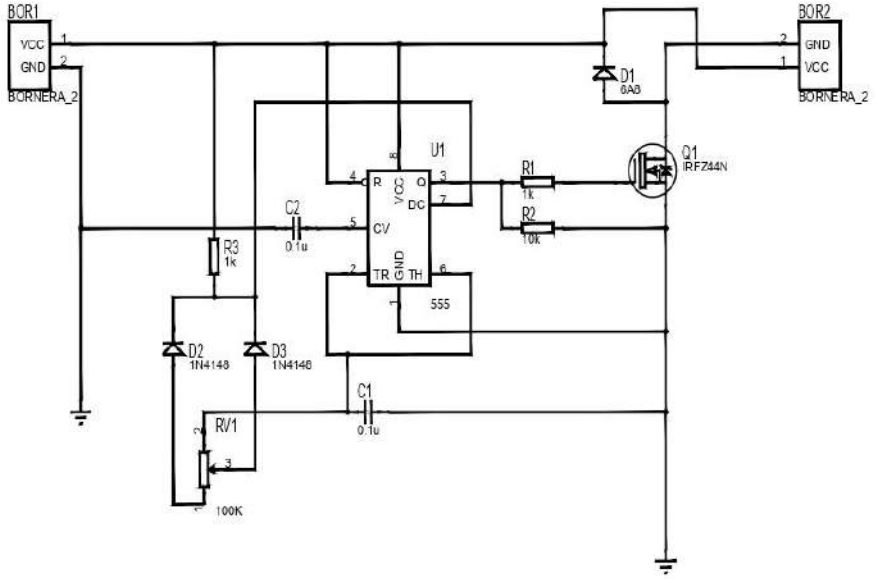
TRANSFORMADOR

MOTOREDUCTOR



	Fecha	Nombre Firmas		Entidad
Dibujado	06/02/19	GRUPO 1		ESPOCH
Comprobado	08/02/19	MIGUEL P.		
Fecha: 07-Feb-2019	Título: DIAGRAMA DE POTENCIA			Núm: 1 de 1
				Archivo: DIAGRAMA DE POTENCIA

N°	Denominación
R1	Resistencia de 1K
R2	Resistencia de 10K
R3	Resistencia de 1K
RV1	Potenciómetro de 100K
BOR1	Bornera de 2 entradas
BOR2	Bornera de 2 entradas
D1	Diodo 1N4148
D2	Diodo 1N4148
D3	Diodo 1N4148
C1	Capacitor cerámico 0,1u
C2	Capacitor cerámico 0,1u
Q1	Transistor IRFZ44N
U1	Circuito integrado 555



	Fecha	Nombre		
Div.	13/10/20	C. Romero	Escuela Superior Politécnica de Chimborazo	Escala
Rev.	22/10/20	E. García	Facultad de Mecánica	
Apr.	22/10/20	M. Calispa	Escuela de Ingeniería Industrial	
Firma			Circuito variador de velocidad del motor-reductor	Hoja 1 - 1

Diagrama Neumático

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

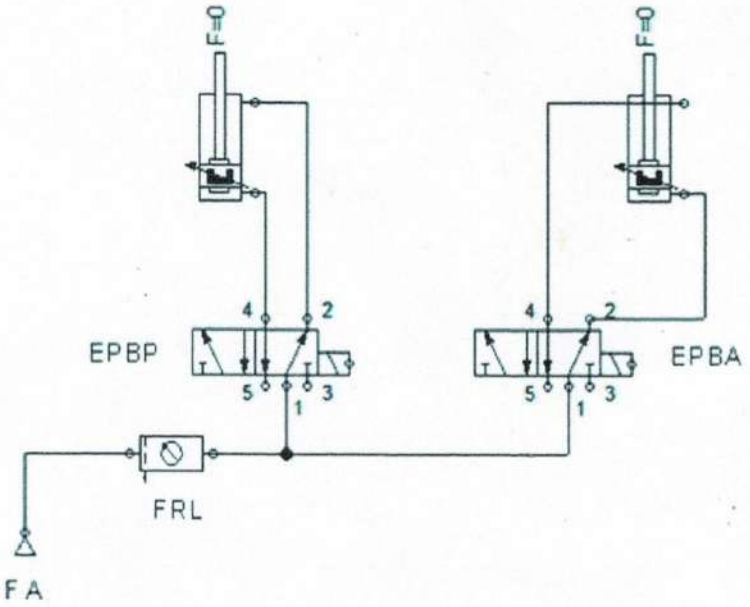
Escuela de Ingeniería Industrial

Automatización de Procesos II

Máquina clasificadora de botellas mediante visión artificial

CDE
Clasificador de botellas de plástico

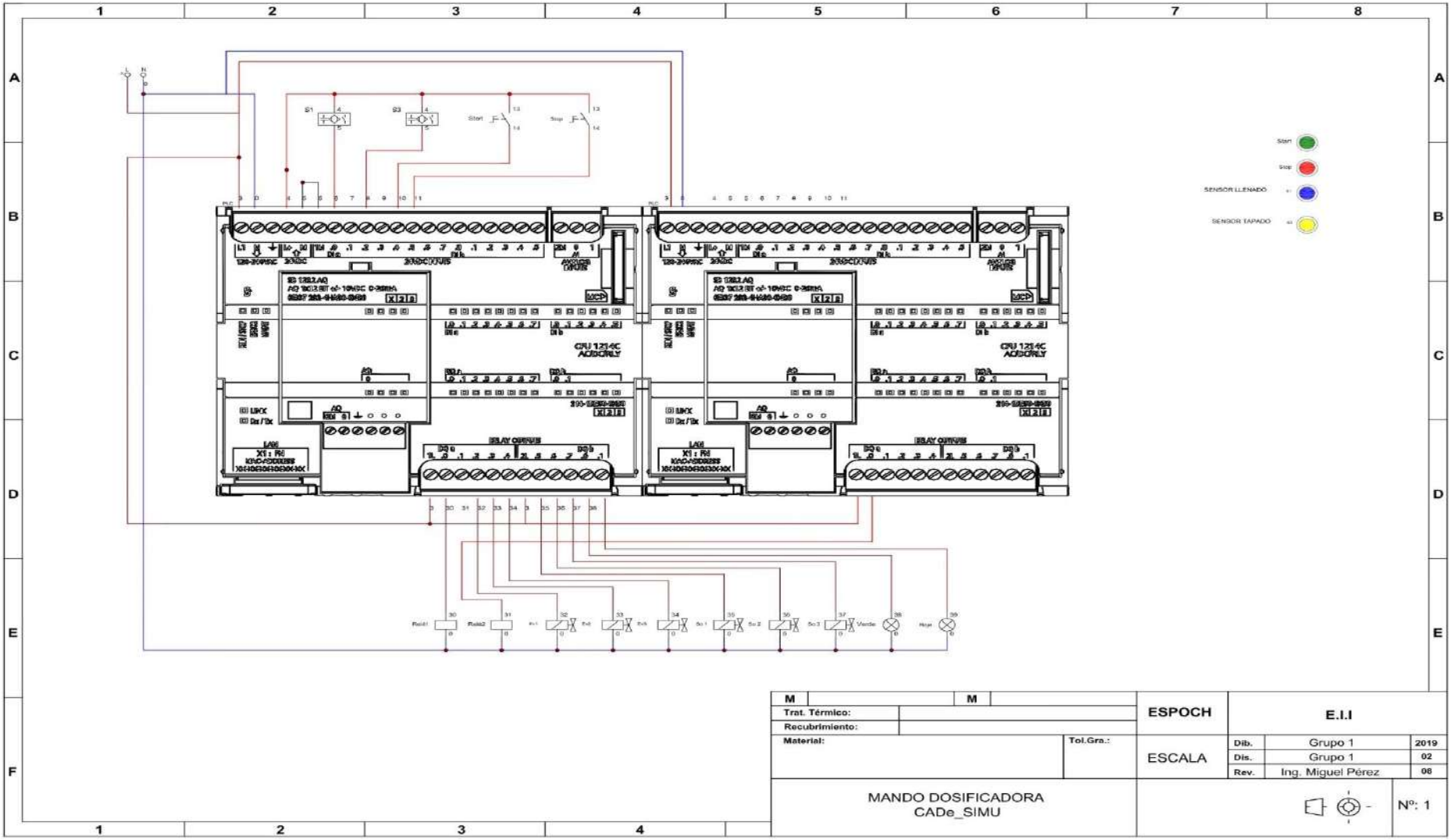
CDE
Clasificador de botellas de aluminio

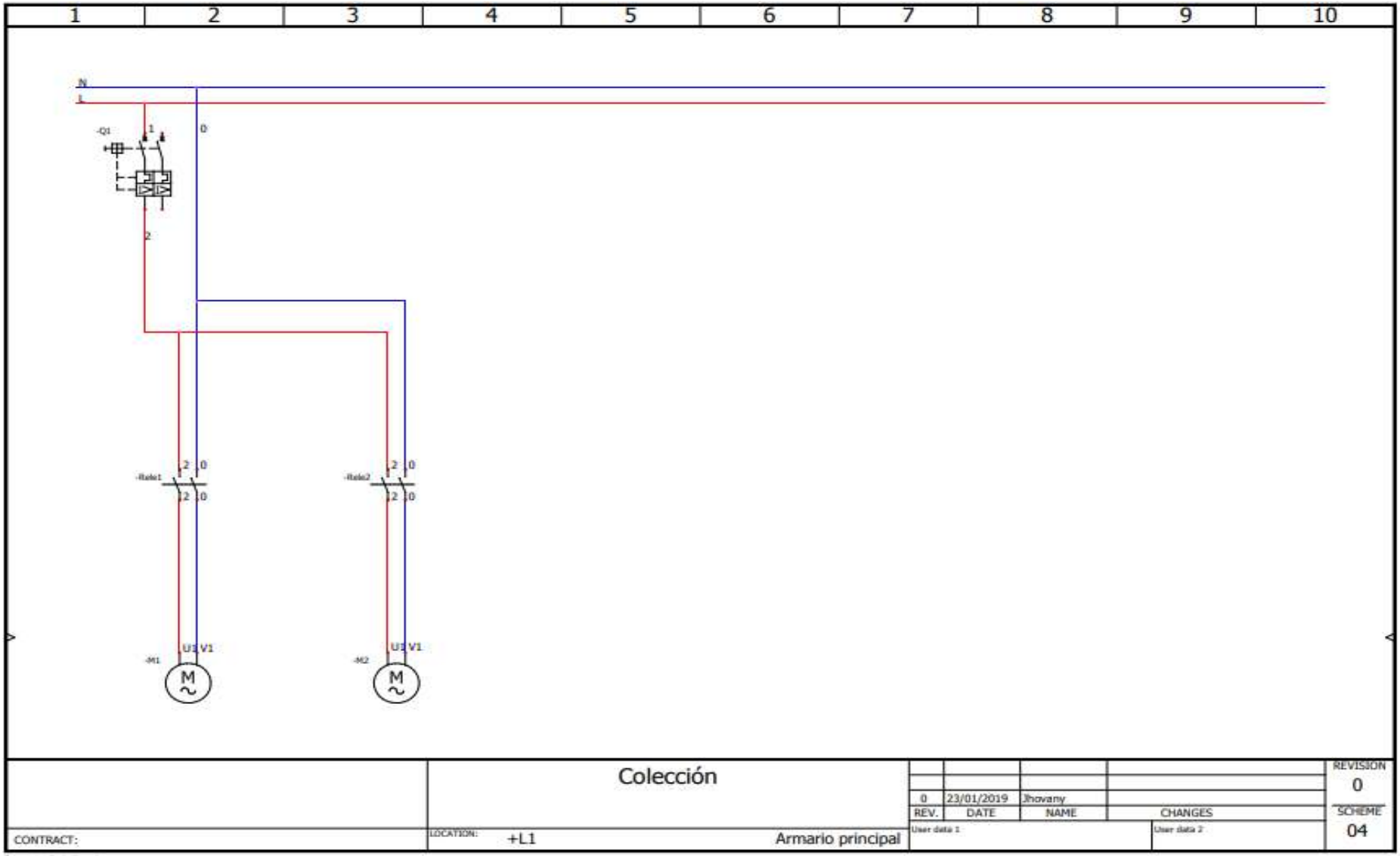


CDE: Cilindro doble efecto
EPBA: Electroválvula 5/2 para botella de aluminio
EPBP: Electroválvula 5/2 para botella de plástico
FRL: Filtro regulador lubricado (Unidad de mantenimiento)
FA: Fuente de alimentación a 7 bares

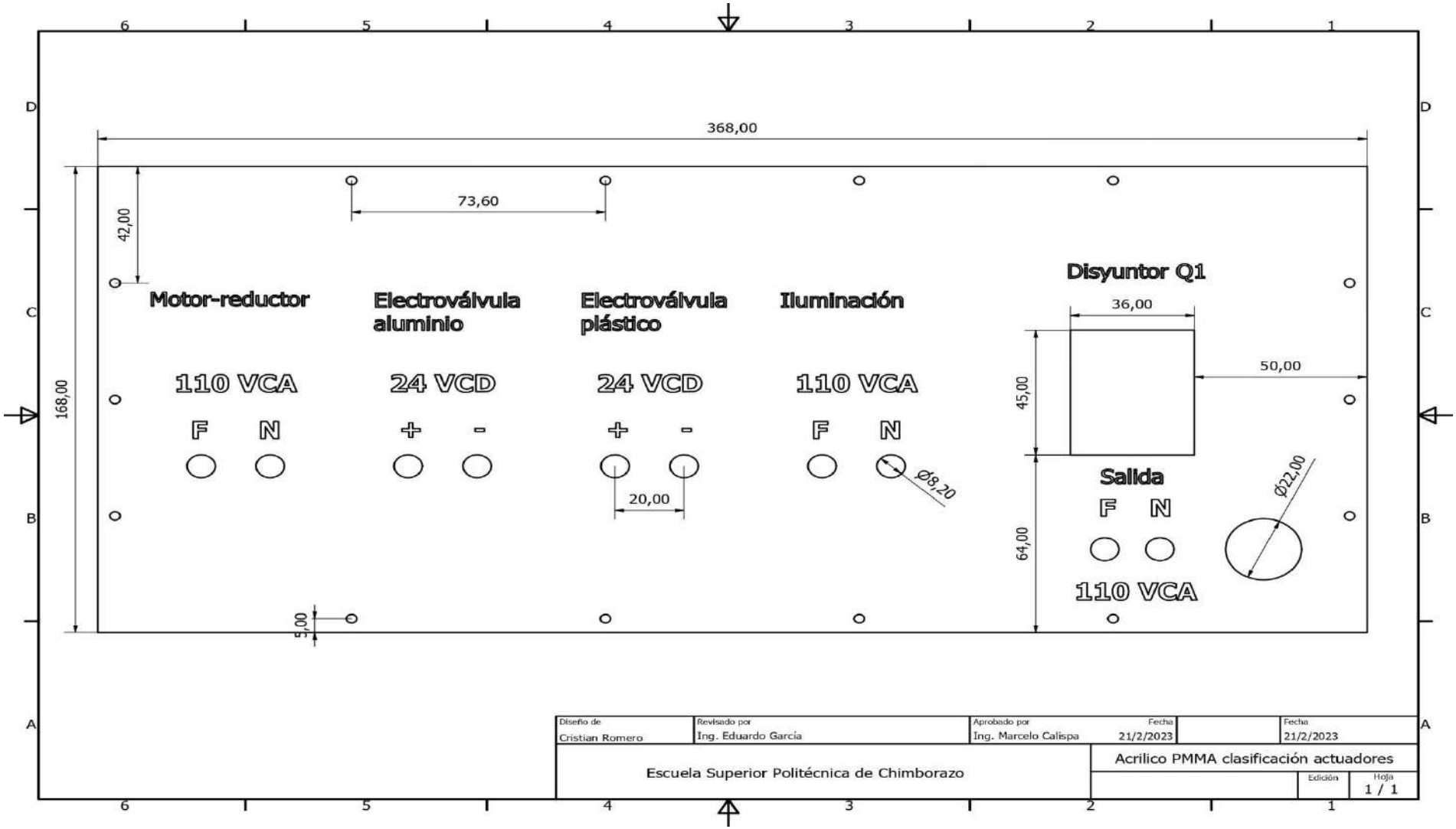
M		M		ESPOCH	E. I. I.
Trat. Térmicos.					
Recubrimiento.					
Material:		Tol. Gral:	Escala:	Dib. GRUPO 1	08
				Dis.	02
				Rev. Ing. Miguel Pérez	19
Titulo: DIAGRAMA NEUMÁTICO					Nº 2

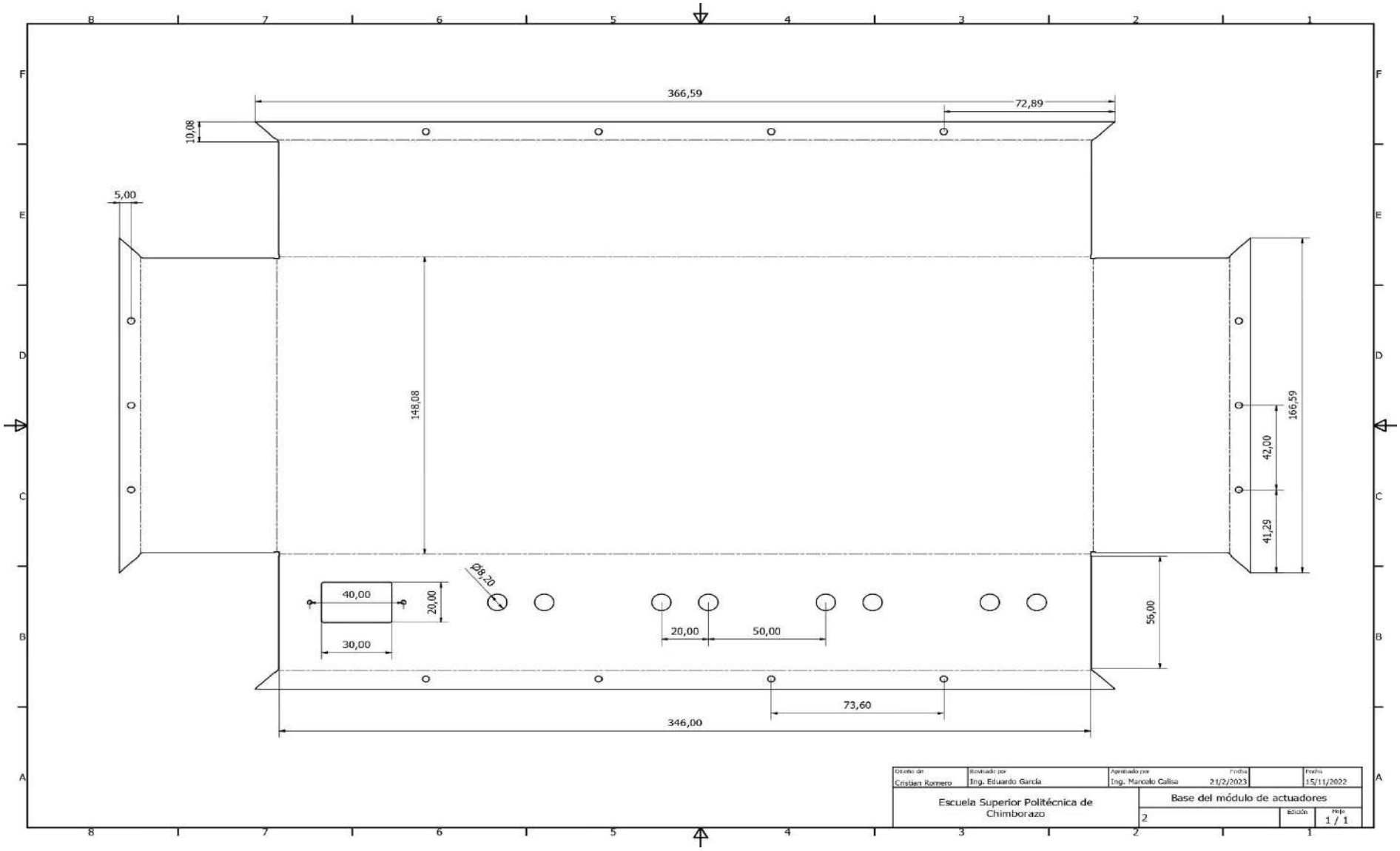
ANEXO B: PLANOS DEL ESTADO DE SITUACIÓN ACTUAL DEL PROCESO DE ENVASADO LINEAL

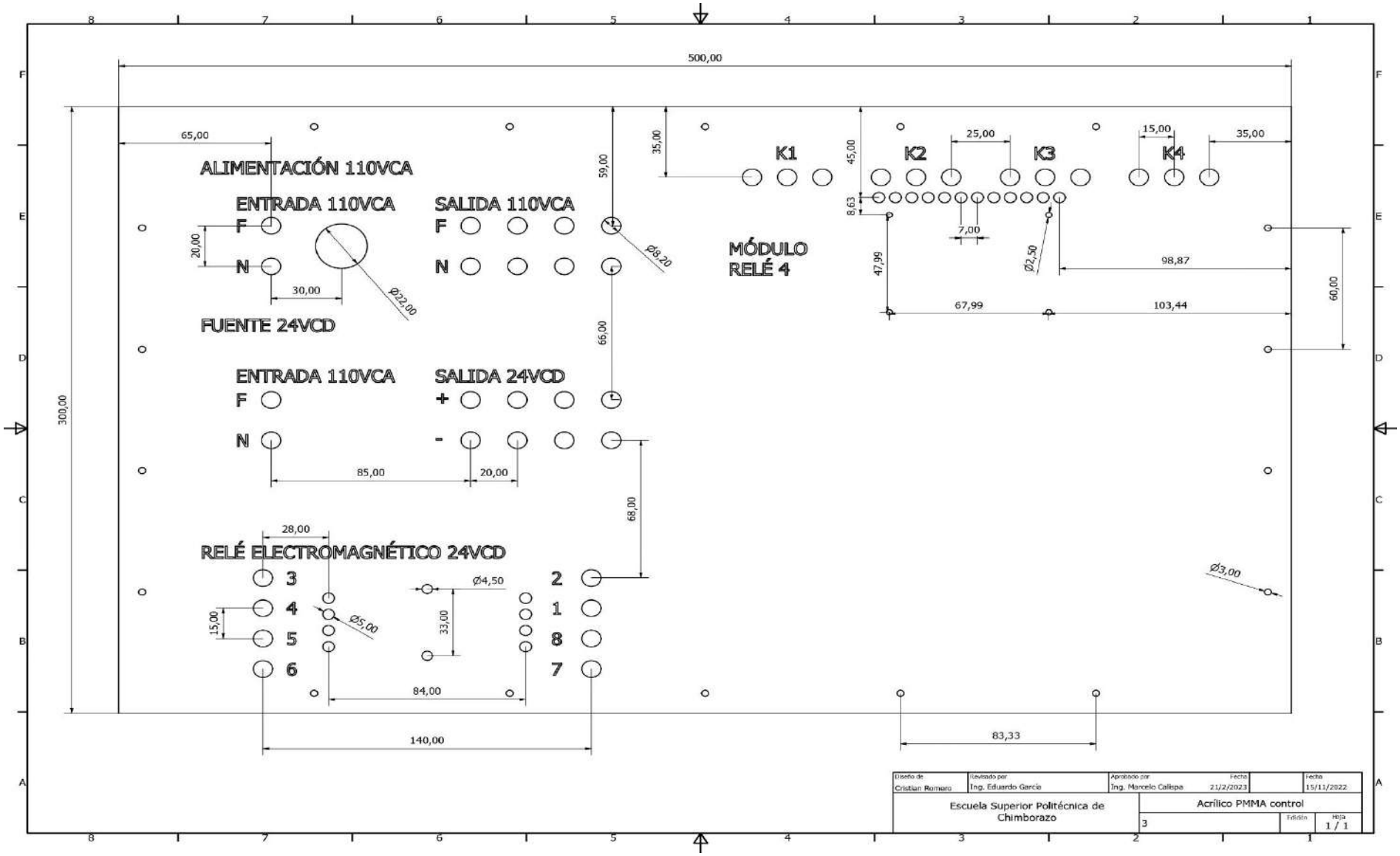


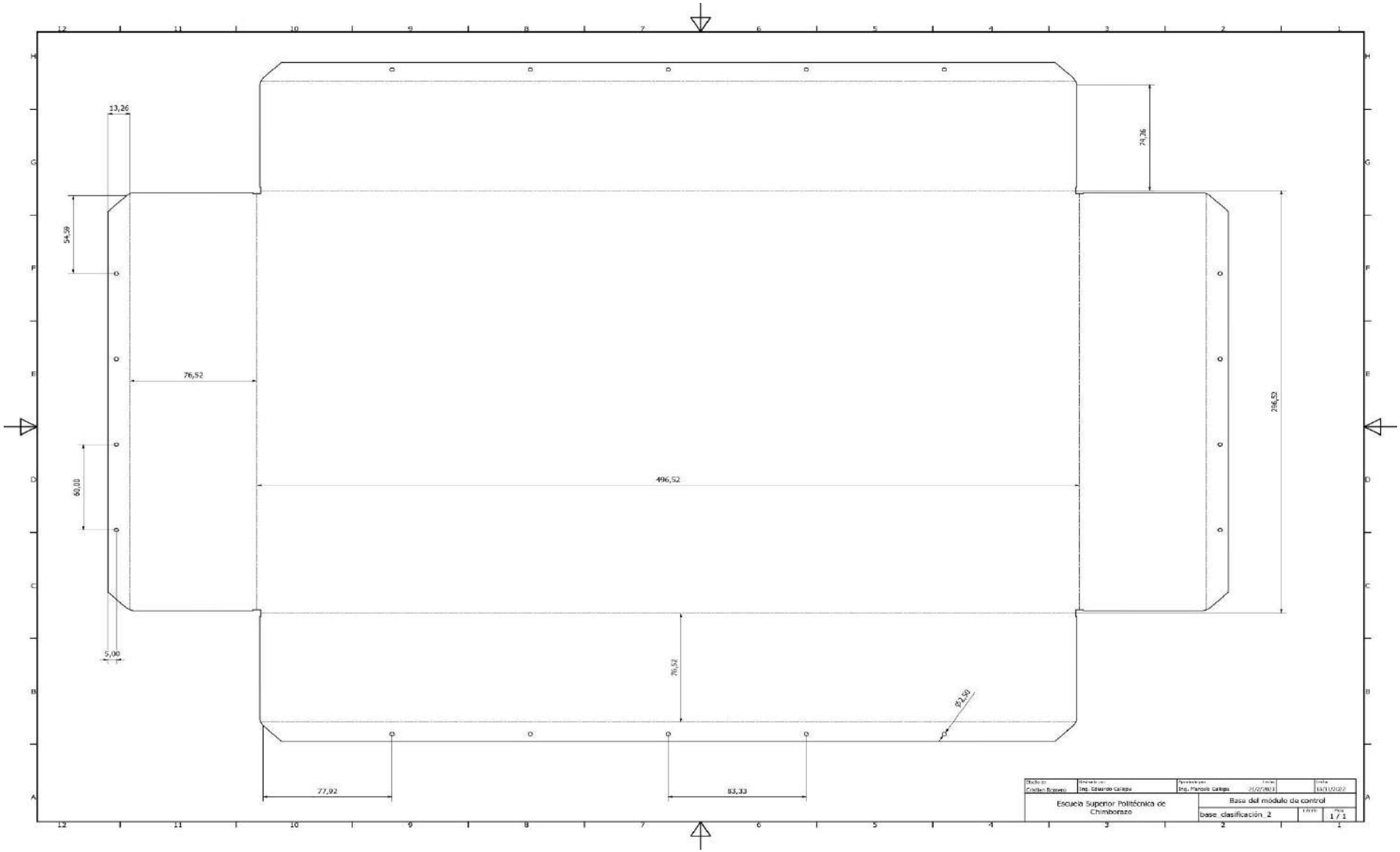

1
2
3
4
5
6
7
8
9
10
N
L
-Q1
1
0
2
-Rele1
2
0
2
0
-Rele2
2
0
2
0
-M1
U1 V1
M
-M2
U1 V1
M
Colección
CONTRACT:
LOCATION:
+L1
Armario principal
0
23/01/2019
Jhovany
REV.
DATE
NAME
CHANGES
User data 1
User data 2
REVISION
0
SCHEME
04

ANEXO C: MÓDULOS DEL PROCESO DE SELECCIÓN Y CLASIFICACIÓN DE BOTELLAS

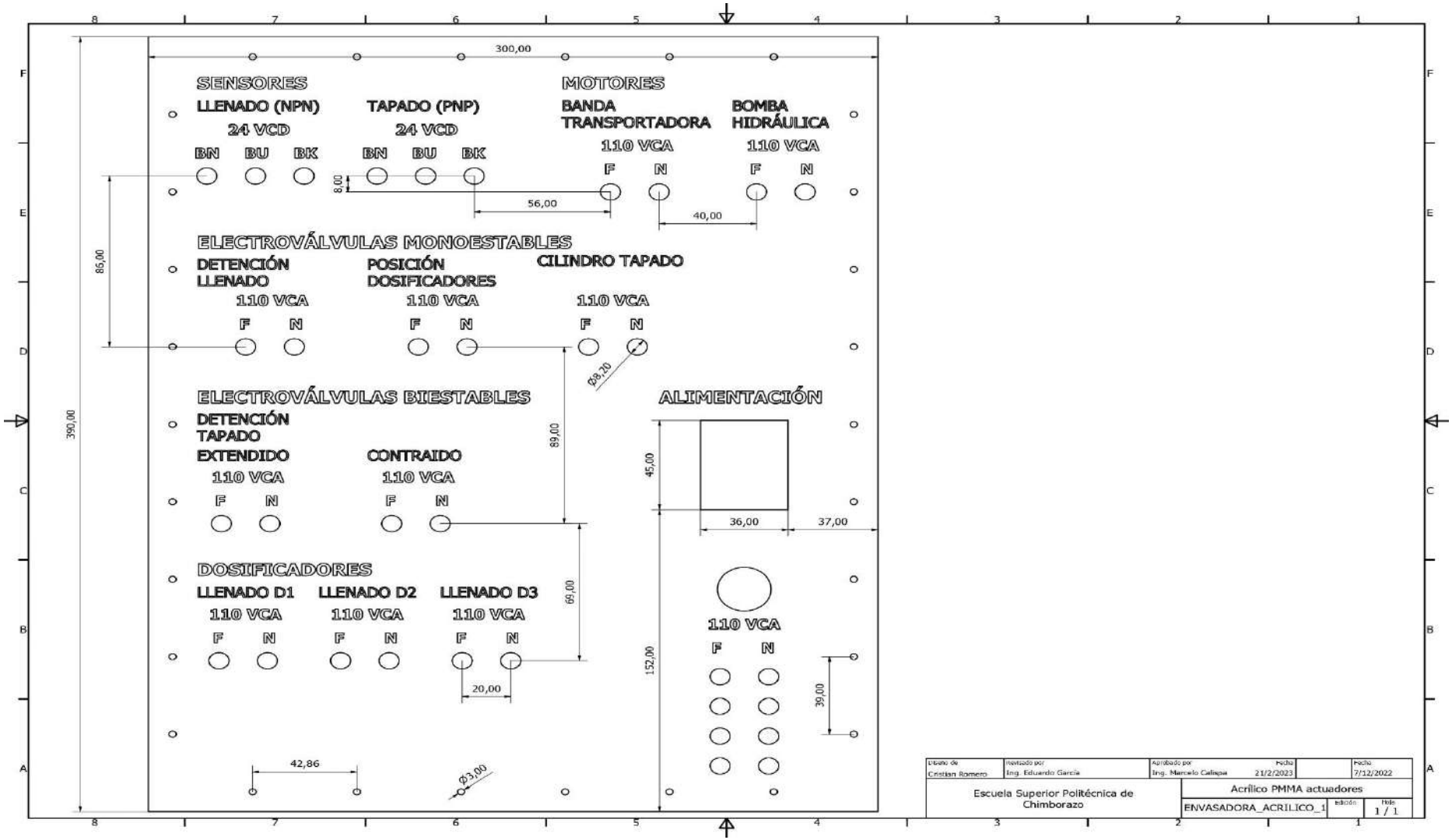


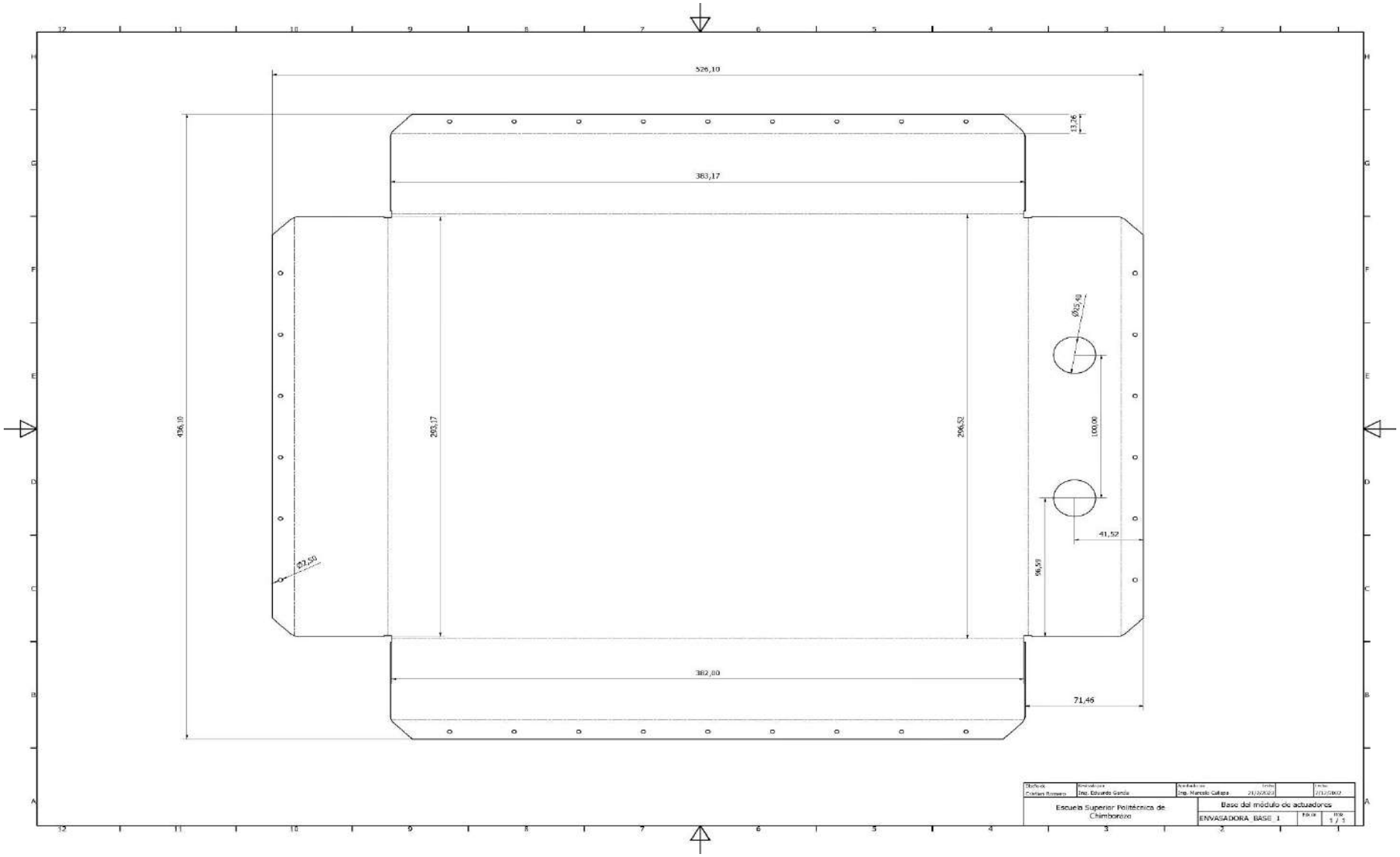

366,59
72,89
10,08
5,00
148,08
166,59
42,00
41,29
56,00
Ø8,20
40,00
20,00
30,00
20,00
50,00
73,60
346,00
Diseño de
Cristian Romero
Revisado por
Ing. Eduardo García
Aprobado por
Ing. Marcelo Calisa
Fecha
21/2/2023
Fecha
15/11/2022
Escuela Superior Politécnica de Chimborazo
Base del módulo de actuadores
2
Edición
Hoja
1 / 1

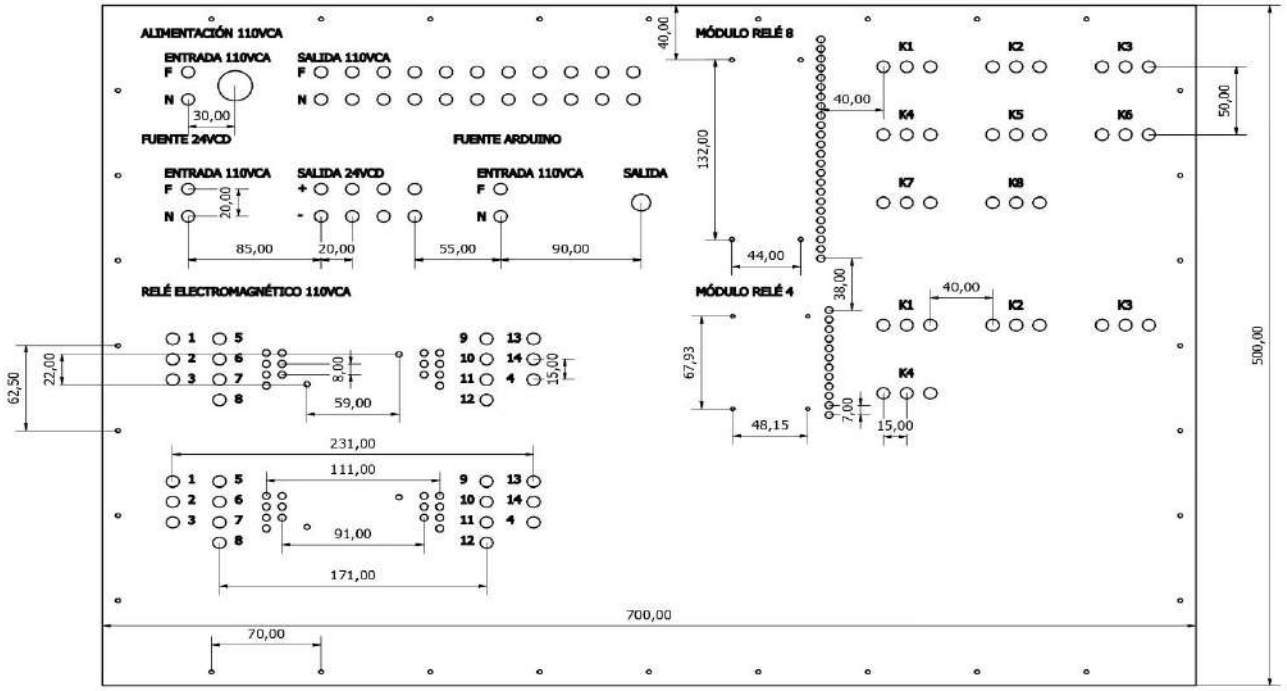

500,00
300,00
65,00
ALIMENTACIÓN 110VCA
ENTRADA 110VCA
F
N
20,00
30,00
Ø22,00
SALIDA 110VCA
F
N
59,00
Ø8,20
FUENTE 24VCD
ENTRADA 110VCA
F
N
SALIDA 24VCD
+
-
66,00
85,00
20,00
68,00
RELÉ ELECTROMAGNÉTICO 24VCD
28,00
3
4
5
6
2
1
8
7
15,00
Ø5,00
Ø4,50
33,00
84,00
140,00
35,00
K1
MÓDULO
RELÉ 4
45,00
8,63
K2
25,00
K3
15,00
K4
35,00
7,00
Ø2,50
47,99
98,87
67,99
103,44
60,00
Ø3,00
83,33
Diseño de
Cristian Romero
Revisado por
Ing. Eduardo García
Aprobado por
Ing. Marcelo Callepa
Fecha
21/2/2023
Fecha
15/11/2022
Escuela Superior Politécnica de Chimborazo
Acrílico PMMA control
3
Edición
Hoja
1 / 1


13,26
74,26
54,59
76,52
296,52
60,00
496,52
5,00
76,52
Ø2,50
77,92
83,33
Escuela Superior Politécnica de Chimborazo
Base del módulo de control
base_clasificación_2
1 / 1

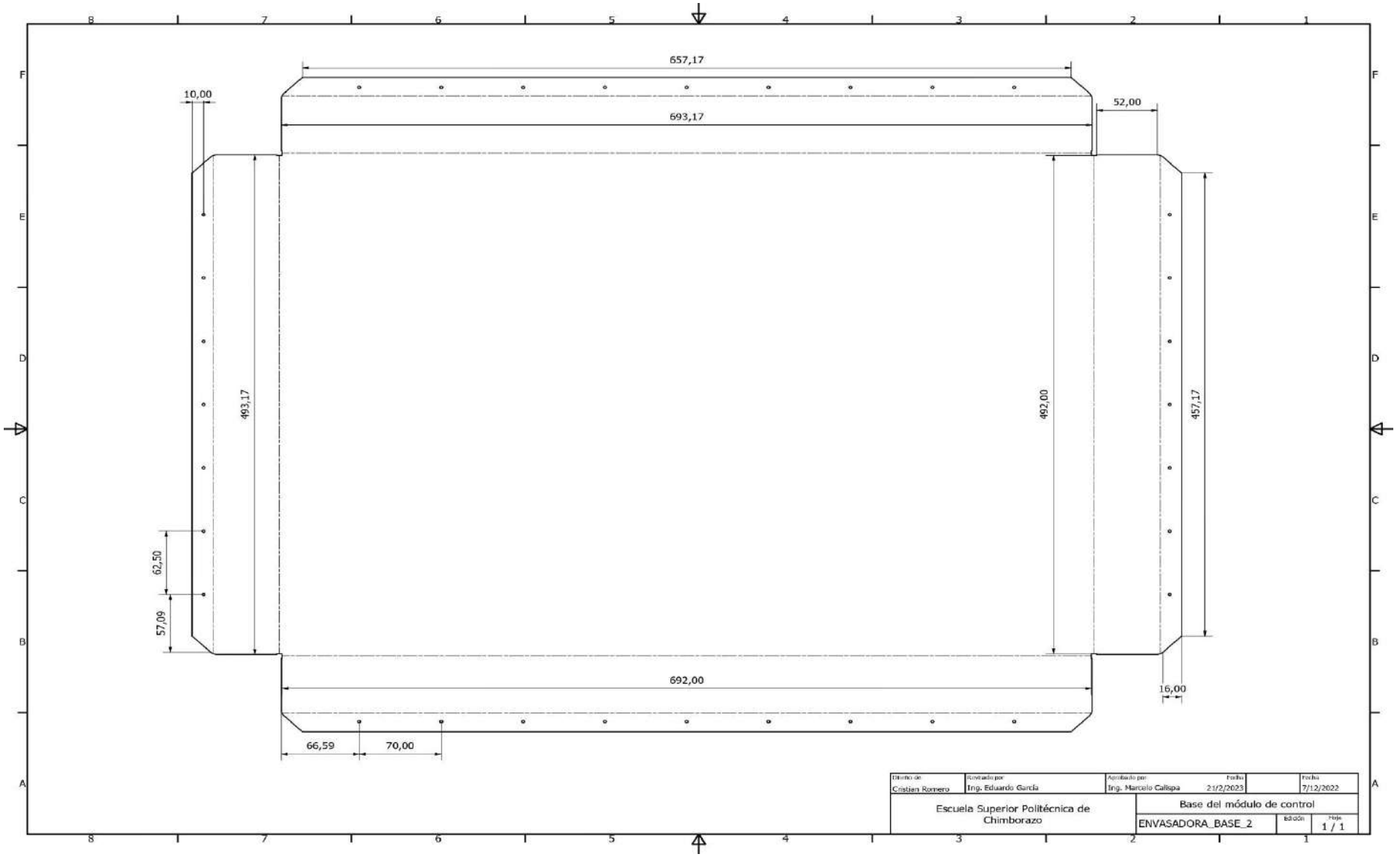
ANEXO D: MÓDULOS DEL PROCESO DE ENVASADO LINEAL




526,10
13,26
383,17
436,10
293,17
296,52
Ø9,40
100,00
41,52
96,59
Ø7,50
382,80
71,46
Escuela Superior Politécnica de Chimborazo
Base del módulo de actuadores
ENVASADORA BASE 1
1 / 1



Diseño de	Revisado por	Aprobado por	Fecha		Fecha
Cristian Romero	Ing. Eduardo García	Ing. Marcelo Calispa	21/2/2023		7/12/2022
Escuela Superior Politécnica de Chimborazo			Acrílico PMMA control		
			ENVASADORA_ACRILICO_2	Edición	Hoja 1 / 1


657,17
10,00
693,17
52,00
493,17
492,00
457,17
62,50
57,09
692,00
16,00
66,59
70,00
Diseño de
Cristian Romero
Revisado por
Ing. Eduardo García
Aprobado por
Ing. Marcelo Gallispa
Fecha
21/2/2023
Fecha
7/12/2022
Escuela Superior Politécnica de Chimborazo
Base del módulo de control
ENVASADORA_BASE_2
Edición
Hoja
1 / 1

ANEXO E: ENTRENAMIENTO DEL MODELO DE RECONOCIMIENTO VISUAL

Paso 1: Instalación de los requerimientos

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output  # to display ima
ges

print(f"Setup complete. Using torch {torch.__version__} ({torch.c
uda.get_device_properties(0).name if torch.cuda.is_available() el
se 'CPU'})")
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 9463, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 9463 (delta 15), reused 25 (delta 15), pack-reused
9436
Receiving objects: 100% (9463/9463), 9.88 MiB | 22.44 MiB/s,
done.
Resolving deltas: 100% (6574/6574), done.
/content/yolov5
     |▏                                | 636 kB 15.3 MB/s
     |▏                                | 178 kB 74.8 MB/s
     |▏                                | 1.1 MB 40.5 MB/s
     |▏                                | 49.9 MB 26 kB/s
     |▏                                | 61 kB 8.9 MB/s
     |▏                                | 138 kB 79.1 MB/s
  Building wheel for roboflow (setup.py) ... done
  Building wheel for wget (setup.py) ... done
ERROR: pip's dependency resolver does not currently take into
account all the packages that are installed. This behaviour is
the source of the following dependency conflicts.
google-colab 1.0.0 requires requests~=2.23.0, but you have
requests 2.25.1 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium
0.8.3 which is incompatible.
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have
imgaug 0.2.9 which is incompatible.
Setup complete. Using torch 1.9.0+cu102 (Tesla P100-PCIE-16GB)
```

Paso 2: Entrenamiento del modelo

```
!python train.py --img 416 --batch 16 --epochs 150 --
data {dataset.location}/data.yaml --weights yolov5s.pt --cache
```

```
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
train: weights=yolov5s.pt, cfg=, data=/content/datasets/American-Mushrooms-1/data.yaml, hyp=data/hyps/hyp.scratch.yaml, epochs=150, batch_size=16, imgsz=416, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, evolve=None, bucket=, cache=ram, image_weights=False, device=, multi_scale=False, single_cls=False, adam=False, sync_bn=False, workers=8, entity=None, project=runs/train, name=exp, exist_ok=False, quad=False, linear_lr=False, label_smoothing=0.0, upload_dataset=False, bbox_interval=-1, save_period=-1, artifact_alias=latest, local_rank=-1, freeze=0, patience=100
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v5.0-471-g2993c3f torch 1.9.0+cu102 CUDA:0 (Tesla P100-PCIE-16GB, 16280.875MB)

hyperparameters: lr0=0.01, lrf=0.2, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 🚀 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://github.com/ultralytics/yolov5/releases/download/v5.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 107MB/s]

Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
```

```
  1                -1  1        18560  models.common.Conv
[32, 64, 3, 2]
  2                -1  1        18816  models.common.C3
[64, 64, 1]
  3                -1  1        73984  models.common.Conv
[64, 128, 3, 2]
  4                -1  3       156928  models.common.C3
[128, 128, 3]
  5                -1  1       295424  models.common.Conv
[128, 256, 3, 2]
  6                -1  3       625152  models.common.C3
[256, 256, 3]
  7                -1  1      1180672  models.common.Conv
[256, 512, 3, 2]
  8                -1  1       656896  models.common.SPP
[512, 512, [5, 9, 13]]
  9                -1  1      1182720  models.common.C3
[512, 512, 1, False]
 10                -1  1       131584  models.common.Conv
[512, 256, 1, 1]
 11                -1  1            0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 12            [-1, 6]  1            0  models.common.Concat
[1]
 13                -1  1       361984  models.common.C3
[512, 256, 1, False]
 14                -1  1        33024  models.common.Conv
[256, 128, 1, 1]
 15                -1  1            0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 16            [-1, 4]  1            0  models.common.Concat
[1]
 17                -1  1        90880  models.common.C3
[256, 128, 1, False]
 18                -1  1       147712  models.common.Conv
[128, 128, 3, 2]
 19           [-1, 14]  1            0  models.common.Concat
[1]
```

```
 20                -1  1    296448  models.common.C3
[256, 256, 1, False]
 21                -1  1    590336  models.common.Conv
[256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat
[1]
 23                -1  1   1182720  models.common.C3
[512, 512, 1, False]
 24          [17, 20, 23]  1     18879  models.yolo.Detect
[2, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90,
156, 198, 373, 326]], [128, 256, 512]]
Model Summary: 283 layers, 7066239 parameters, 7066239 gradients,
16.4 GFLOPs

Transferred 356/362 items from yolov5s.pt
Scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 59 weight, 62 weight (no
decay), 62 bias
albumentations: version 1.0.3 required by YOLOv5, but version
0.1.12 is currently installed
train: Scanning '/content/datasets/American-Mushrooms-
1/train/labels' images and labels...123 found, 0 missing, 0 empty,
0 corrupted: 100% 123/123 [00:00<00:00, 1919.15it/s]
train: New cache created: /content/datasets/American-Mushrooms-
1/train/labels.cache
train: Caching images (0.1GB ram): 100% 123/123 [00:00<00:00,
540.50it/s]
val: Scanning '/content/datasets/American-Mushrooms-
1/valid/labels' images and labels...5 found, 0 missing, 0 empty, 0
corrupted: 100% 5/5 [00:00<00:00, 114.99it/s]
val: New cache created: /content/datasets/American-Mushrooms-
1/valid/labels.cache
val: Caching images (0.0GB ram): 100% 5/5 [00:00<00:00, 241.22it/s]
[W pthreadpool-cpp.cc:90] Warning: Leaking Caffe2 thread-pool after
fork. (function pthreadpool)
Plotting labels...
[W pthreadpool-cpp.cc:90] Warning: Leaking Caffe2 thread-pool after
fork. (function pthreadpool)
```

```
autoanchor: Analyzing anchors... anchors/target = 3.45, Best
Possible Recall (BPR) = 1.0000
Image sizes 416 train, 416 val
Using 2 dataloader workers
Logging results to runs/train/exp

Starting training for 150 epochs...
Epoch   gpu_mem       box       obj       cls    labels  img_size
     0/149     1.52G    0.1167   0.02613   0.03485        42
416: 100% 8/8 [00:03<00:00,  2.07it/s]
               Class     Images     Labels          P          R
mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 11.33it/s]
                 all          5         14     0.0337      0.478
0.0324    0.00404

Epoch   gpu_mem       box       obj       cls    labels  img_size
   149/149     1.52G   0.02392   0.01369 0.0007764        41
416: 100% 8/8 [00:01<00:00,  6.98it/s]
               Class     Images     Labels          P          R
mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 27.48it/s]
                 all          5         14      0.996          1
0.995      0.813

150 epochs completed in 0.065 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model Summary: 224 layers, 7056607 parameters, 0 gradients, 16.3
GFLOPs
               Class     Images     Labels          P          R
mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 20.76it/s]
                 all          5         14      0.995          1
0.995      0.813
                 CoW          5          5      0.991          1
0.995      0.848
          chanterelle          5          9          1          1
0.995      0.777
Results saved to runs/train/exp
```

Paso 3: Descargar el modelo

```
#export your model's weights for future use
from google.colab import files
files.download('./runs/train/exp/weights/best.pt')
```

ANEXO F: ALGORITMO DE RECONOCIMIENTO VISUAL

Algoritmo de reconocimiento visual

```
#---------------------------Importamos las librerias----------------
----------------
#Librerias para arduino
import time
import serial
#Librerias para detección de objetos
import torch
import cv2
import numpy as np
import pandas

#---------------------------Comunición serial con arduino-----------
----------------
#Iniciar la comunicación con el puerto serial
arduino = serial.Serial('COM4', 9600, timeout = 1)
#Tiempo de espera
time.sleep(1)

#---------------------------Leer el modelo--------------------------
---------------
model = torch.hub.load('ultralytics/yolov5', 'custom',
                       path =
'C:/Users/Crist/PycharmProjects/bottle2/model/botella.pt')

#---------------------------Realizar la vídeo captura---------------
----------------
cap = cv2.VideoCapture(1)

#Empezar el while true
while True:
    #Realizar la lectura de la vídeo captura
    ret, frame = cap.read()

    #Realizar las detecciones
    detect = model(frame)

    #Vector de datos referente a la captura de imagenes
    info = detect.pandas().xyxy[0]
    #print(info)

    #Acceder a la información
    info2 = detect.pandas().xyxy[0].to_dict(orient = "records")

    #Preguntar si hay información
    if len(info2) != 0:
        #Crear un for
        for result in info2:
            material = result['name']
            #Preguntar que tipo de material es el analizado
            if material == "metal":
                #Enviar dato del material a arduino
                arduino.write(b'A')
                #Imprimir el tipo de material
                print("El material es aluminio")

            elif material == "plastico":
                #Enviar dato del material a arduino
                arduino.write(b'P')
```

```
                print("El material es plástico")
            elif material == "vidrio":
                #Enviar dato del material a arduino
                arduino.write(b'V')
                print("El material es vidrio")

    #Mostramos FPS
    cv2.imshow("Detector de materiales",
np.squeeze(detect.render()))

    #Leer el teclado
    t = cv2.waitKey(1)
    if t == 27:
        break

cap.release()
cv2.destroyAllWindows()
arduino.close()
```

ANEXO G: INTERFAZ DEL ALGORITMO DE VISIÓN ARTIFICIAL

Script de ejecución de la interfaz gráfica

```
#-----------------------------------------Importar librerias------
--------------------------
#Importar la libreria creada para la ventana
from GUI_vision import GUI_v
#Importar la libreria visual TKinter
from tkinter import Tk

#-----------------------------------------Función de la ventana---
--------------------------
def main():
    raiz = Tk()
    raiz.wm_title("Máquina selectora y clasificadora de botellas")
    app = GUI_v(raiz)
    app.mainloop()

if __name__ == "__main__":
    main()
```

Módulo de ejecución de funciones

```
#--------------------------Importamos las librerias----------------
----------------
#Librerias para arduino
import time
import serial
#Librerias para detección de objetos
import torch
import cv2
import numpy as np
import pandas

#-----------------------------------------------------Importar
librerias-----------------------------------------------
#Librería de elementos visuales
from tkinter import Frame, Label, PhotoImage, Button
#Libreria para el uso de imágenes
import cv2
from PIL import Image, ImageTk
import subprocess

#Clase del frame que va en la ventana
class GUI_v(Frame):
    #Función de construcción
    def __init__(self, master = None):
        #Llamar a la súper clase Frame para la creación de un frame
dando valores de altura y ancho
        super().__init__(master, height=500, width=900)
        #Almacenar la variable master de la clase padre en la
variable master de la clase hija
        self.master = master
        #Definir la ubicación del frame creado
        self.pack()
        #Llamar a la función de elementos visuales
        self.clasificador()

    #Función para iniciar el script de reconocimiento visual
    def iniciar(self):
```

```
        inicio = time.time()
        #----------------------------Comunición serial con arduino---
------------------------
        #Iniciar la comunicación con el puerto serial
        arduino = serial.Serial('COM4', 9600, timeout=1)
        #Tiempo de espera
        time.sleep(1)

        #----------------------------Leer el modelo-----------------
-----------------------
        model = torch.hub.load('ultralytics/yolov5', 'custom',
                               path='model/botella.pt')

        #----------------------------Realizar la vídeo captura------
------------------------
        cap = cv2.VideoCapture(1)

        #Empezar el while true
        while True:
            #Realizar la lectura de la vídeo captura
            ret, frame = cap.read()

            #Realizar las detecciones
            detect = model(frame)

            #Vector de datos referente a la captura de imagenes
            info = detect.pandas().xyxy[0]
            #print(info)

            #Acceder a la información
            info2 =
detect.pandas().xyxy[0].to_dict(orient="records")

            #Preguntar si hay información
            if len(info2) != 0:
                #Crear un for
                for result in info2:
                    material = result['name']
                    #Preguntar que tipo de material es el analizado
                    if material == "metal":
                        #Enviar dato del material a arduino
                        arduino.write(b'A')
                        #Imprimir el tipo de material
                        print("El material es aluminio")

                    elif material == "plastico":
                        #Enviar dato del material a arduino
                        arduino.write(b'P')
                        print("El material es plástico")
                        fin = time.time()
                        print(fin - inicio)
                    elif material == "vidrio":
                        #Enviar dato del material a arduino
                        arduino.write(b'V')
                        print("El material es vidrio")


            #Mostramos FPS
            cv2.imshow("Detector de materiales",
np.squeeze(detect.render()))
```

```
            #Leer el teclado
            t = cv2.waitKey(1)
            if t == 27:
                break

        cap.release()
        cv2.destroyAllWindows()
        arduino.close()

    #Función para cerrar todas las ventanas
    def finalizar(self):
        #Cerrar la ventana
        cv2.destroyAllWindows()
        #Cerrar la pantalla
        self.master.destroy()


    #Función de los elementos visuales
    def clasificador(self):


        #Etiqueta con el primer encabezado
        self.lbl1 = Label(self, text="ESCUELA SUPERIOR POLITÉCNICA
DE CHIMBORAZO", font=('Verdana', 20))
        self.lbl1.place(x=80, y=0)
        #Etiqueta con el segundo encabezado
        self.lbl2 = Label(self, text="FACULTAD DE MECÁNICA",
font=('Verdana', 20))
        self.lbl2.place(x=260, y=40)
        #Etiqueta con el tercero encabezado
        self.lbl3 = Label(self, text="CARRERA DE INGENIERÍA
INDUSTRIAL", font=('Verdana', 20))
        self.lbl3.place(x=170, y=80)
        #Etiqueta con el cuarto encabezado
        self.lbl3 = Label(self, text="Máquina selectora y
clasificadora de botellas", font=('Verdana', 14))
        self.lbl3.place(x=220, y=120)
        #Etiqueta con el quinto encabezado
        self.lbl3 = Label(self, text="Laboratorio de electricidad y
electrónica", font=('Verdana', 14))
        self.lbl3.place(x=245, y=150)

        #Imagen 1
        #Redimensionamiento de la imagen
        self.ima1 =
Image.open('img/Escudo_de_la_Escuela_Superior_Politécnica_de_Chimbor
azo.png')
        self.ima1 = self.ima1.resize((220, 220), Image.ANTIALIAS)
        #Colocar la imagen
        self.ima2 = ImageTk.PhotoImage(self.ima1)
        self.lblima1 = Label(self, image= self.ima2)
        self.lblima1.place(x=40, y=200)

        #Imagen 2
        #Colocar la imagen
        self.img = PhotoImage(file='img/imageedit_4_8675083414-
removebg-preview.png')
        self.lbl_img = Label(self, image=self.img)
        self.lbl_img.place(x=630, y=200)
```

```
        #Botones
        #Botón de inicio
        #Importar la imagen para el botón
        self.img_ini = PhotoImage(file='img/Inicio.png')
        #Creación del botón
        self.inicio = Button(self, text='Iniciar',
image=self.img_ini, height='40', width='200', command=self.iniciar)
        #Posición del botón
        self.inicio.place(x=350, y=220)
        #Botón de fin
        #Importar la imagen para el botón
        self.img_fin = PhotoImage(file='img/Finalizar.png')
        #Creación del botón
        self.final = Button(self, text='Finalizar',
image=self.img_fin, height='40', width='200',
command=self.finalizar)
        # Posición del botón
        self.final.place(x=350, y=300)
```

ANEXO H: ALGORITMO DE CONTROL DEL PROCESO DE SELECCIÓN Y CLASIFICACIÓN DE BOTELLAS

Script de control del proceso IDE Arduino

```
/------------------------------MÁQUINA SELECTORA Y CLASIFICADORA DE BOTELLAS---------------
//------------------------------Librerias-----------------------------------------------
//Libreria para el uso de la pantalla LCD
#include <LiquidCrystal.h>
//Libreria para el uso del bus de comunicación I2C
#include <Wire.h>

//------------------------------Declaración de pines y variables------------------------
//------------------------------Botones de arranque y paro------------------------------
//Pin de arranque
const int arranque = 3;
//Pin de paro
const int paro = 2;
//Variables de arranque y paro
//Variable de arranque del proceso
int inicio;
//Variable de enclavamiento
int enclavamiento;
//Variable de paro del proceso
int fin;
//Variable de encendido enviada desde el maestro
volatile int on_master;
//------------------------------Sensor de ingreso---------------------------------------
//Pin digital del sensor de ingreso
const int sensor = 53;
//Pin digital del laser de ingreso
const int laser = 51;
//Variable de ingreso del proceso
int ingreso;
//------------------------------Actuadores----------------------------------------------
//Pin de iluminación
const int luz = 36;
//Pin del motor-reductor
const int motor = 34;
//Pin de la electroválvula para el aluminio
const int eva = 32;
//Pin de la electroválvula para el plástico
const int evp = 30;
```

```
//------------------------------Contadores--------------------------
----------------------
//Contador del total de botellas
int cont_total = 0;
//Contador de botellas de aluminio
int cont_aluminio = 0;
//Contador de botellas de plástico
int cont_plastico = 0;
//Contador de botellas de vidrio
int cont_vidrio = 0;

//------------------------------Comunicación con Python-------------
----------------------
//Variable de almacenamiento de los datos obtenidos desde Python
char py;
//Variable para envio de datos al maestro
volatile int objeto;

//------------------------------LCD---------------------------------
----------------------
//Pines para la conexión de la pantalla LCD
const int rs = 24, en = 26, d4 = 40, d5 = 42, d6 = 44, d7 = 46;
//Pines declarados para la pantalla LCD
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

//------------------------------Ejecución del estado inicial del
proceso--------------------
void setup() {
  //----------------------------Protocolos de comunicación----------
----------------------
  //Inicio de la comunicación serial
  Serial.begin(9600);
  //Inicio de la comunicación I2C (Esclavo 1)
  Wire.begin(1);
  //Llamado de funciones de requerimiento y recepción de información
  //Función de requerimiento
  Wire.onRequest(EventoPeticion);
  //Función de recepción
  Wire.onReceive(EventoRecepcion);
  //Inicio de la pantalla LCD
  lcd.begin(16, 2);
  //----------------------------Definición de los pines-------------
----------------------
  //Pin digital del botón de arranque
  pinMode(arranque, INPUT);
  //Pin digital del botón de paro
  pinMode(paro, INPUT);
  //Pin digital del sensor fotoeléctrico
  pinMode(sensor, INPUT);
```

```
  //Pin digital del laser de ingreso
  pinMode(laser, OUTPUT);
  //Pin digital de la iluminación
  pinMode(luz, OUTPUT);
  //Pin digital del motor-reductor de la banda transportadora
  pinMode(motor, OUTPUT);
  //Pin digital de la electroválvula para el aluminio
  pinMode(eva, OUTPUT);
  //Pin digital de la electroválvula para el plástico
  pinMode(evp, OUTPUT);
  //-----------------------------Estado inicial de los pines-----------
----------------------
  //Estado inicial del laser de ingreso
  digitalWrite(laser, HIGH);
  //Estado inicial de la iluminación
  digitalWrite(luz, HIGH);
  //Estado inicial del motor-reductor
  digitalWrite(motor, HIGH);
  //Estado inicial de la electrovávula para el aluminio
  digitalWrite(eva, HIGH);
  //Estado incial de la electroválvula para el plástico
  digitalWrite(evp, HIGH);
}

//------------------------------Ejecución de la secuencia del proceso-
-----------------------
void loop() {
  //Lectura del pin digital del botón de arranque
  inicio = digitalRead(arranque);
  //Lectura del pin digital del botón de paro
  fin = digitalRead(paro);
  //Preguntar si existe una señal de inicio de secuencia
  if (inicio == 1 || on_master == 1){
    //Cambiar el estado de la variable enclavamiento
    enclavamiento = 1;
  }
  //Preguntar si existe una señal de paro de seciencia
  if (fin == 1 || on_master == 2){
    //Cambiar el estado de la variable enclavamiento
    enclavamiento = 0;
  }
  //Preguntar si la variable enclavamiento se activo
  if (enclavamiento == 1){
    //Colocar el cursor en la calumna 0, línea 0
    lcd.setCursor(0, 0);
    //Muestra del mensaje en la pantalla LCD
    lcd.print("ENCENDIDO");
    //Limpieza de la variable que recibe los datos desde Python
    py = "";
```

```
    //Encendido del láser de ingreso
    digitalWrite(laser, LOW);
    //Encendido de la iluminación de la máquina
    digitalWrite(luz, LOW);
    //Encendido del motor-reductor de la banda transportadora
    digitalWrite(motor, LOW);
    //Tiempo de espera para la lectura del sensor
    delay(500);
    //Lectura del pin digital del sensor de ingreso
    ingreso = digitalRead(sensor);
    //Preguntar si existe detección en el ingreso de la máquina
    if (ingreso == 0){
      //Almacenamiento del dato de ingreso en la máquina al MTU
      objeto = 3;
      //Limpieza de la variable que recibe los datos desde Python
      py = "";
      //Tiempo de espera para el apagado de la banda
      delay(1150);
      //Limpieza de la variable de envío de datos a la MTU
      objeto = 0;
      //Apagado de la banda transportadora
      digitalWrite(motor, HIGH);
      //Tiempo de espera de la banda apagada
      delay(3500);
      //Limpieza de la variable que recibe los datos desde
Python
      py = "";
      //Preguntar si se obtuvo datos de la comunicación serial con el
programa de reconocimiento visual
      while (Serial.available()){
        //Almacenar los datos obtenidos de la comunicación serial con
el programa de reconocimeinto visual
        py = Serial.read();
        //Limpieza del puerto serial
        Serial.flush();
      }
      //Esperar a que termine la transmisión de datos desde el programa
de reconocimiento visual
      Serial.flush();
      //Determinar el tipo de objeto que ingresó: Aluminio (A),
plástico (P), Vidrio (V)
      //Preguntar si el objeto es de
aluminio
      if (py == 'A'){
        //Almacenamiento de datos del tipo de material para si envío a
la MTU
        objeto = 4;
        //Encendido de la banda transportadora
        digitalWrite(motor, LOW);
```

```
        //Tiempo de espera de la banda encendida
        delay(1900);
        //Encendido de la electroválvula para el aluminio
        digitalWrite(eva, LOW);
        //Apagado de la banda transportadora
        digitalWrite(motor, HIGH);
        //Tiempo de espera de la banda apagada
        delay(2000);
        //Apagado de la electroválvula para el aluminio
        digitalWrite(eva, HIGH);
        //Encendido de la banda transportadora
        digitalWrite(motor, LOW);
        //Contador de latas de aluminio
        cont_aluminio = cont_aluminio + 1;
        //Limpieza de la variable que recibe los datos desde
Python
        py = "";
        //Limpieza de la variable de envío de datos a la MTU
        objeto = 0;
      }
      //Preguntar si el objeto es de plástico
      if (py == 'P'){
        //Almacenamiento de datos del tipo de material para si envío a
la MTU
        objeto = 5;
        //Encendido de la banda transportadora
        digitalWrite(motor,LOW);
        //Tiempo de espera de la banda encendida
        delay(1400);
        //Apagado de la banda transportadora
        digitalWrite(motor, HIGH);
        //Encendido de la electroválvula para el plástico
        digitalWrite(evp, LOW);
        //Tiempo de espera de la banda apagada
        delay(2000);
        //Apagado de la electroválvula para el plástico
        digitalWrite(evp, HIGH);
        //Encendido de la banda transportadora
        digitalWrite(motor, LOW);
        //Contador de botellas de plástico
        cont_plastico = cont_plastico + 1;
        //Limpieza de la variable que recibe los datos desde
Python
        py = "";
        //Limpieza de la variable de envío de datos a la MTU
        objeto = 0;
      }
      if (py == 'V'){
```

```
        //Almacenamiento de datos del tipo de material para si envío a
la MTU
        objeto = 6;
        //Contador de botellas de vidrio
        cont_vidrio = cont_vidrio + 1;
        //Limpieza de la variable que recibe los datos desde
Python
        py = "";
        //Limpieza de la variable de envío de datos a la MTU
        objeto = 0;
      }
      //Actualizar la variable de enclavamiento para que se repita el
proceso
      enclavamiento = 1;
      //Incremento del contador de botellas
      cont_total = cont_total + 1;
    }
  }
  else{
    //Limpieza de la pantalla LCD
    lcd.clear();
    //Colocar el cursor en la calumna 0, línea 0
    lcd.setCursor(0, 0);
    //Muestra del mensaje en la pantalla LCD
    lcd.print("APAGADO");
    //Apagado del laser de ingreso
    digitalWrite(laser, HIGH);
    //Apagado de la iluminación de la máquina
    digitalWrite(luz, HIGH);
    //Apagado del motor-reductor de la banda transportadora
    digitalWrite(motor, HIGH);
  }

}
//-------------------------------------------Funciones de comunicación
I2C-------------------------------
//Función de envio de información desde el esclavo al maestro una vez
hecha la petición
void EventoPeticion() {
  Wire.write(objeto);
}

//Función de recepción de ordenes desde el maestro (MTU)
void EventoRecepcion(int on) {
  while (Wire.available() > 0) {
    on_master = Wire.read();
  }
}
```

ANEXO I: ALGORITMO DE CONTROL DEL PROCESO DE ENVASADO LINEAL

Script de control del proceso IDE Arduino

```
//Programa final de funcionamiento de la máquina envasadora, con un
contador de envases
//Librerias para el uso del bus de comunicación I2C
#include <Wire.h>

//------------------Declaración de las variables y los pines-----
//------------------Botones de arranque y paro-------------------
//Botón de arranque
const int arranque = 3;
//Botón de paro
const int paro = 2;
//Variable de arranque del proceso
int inicio;
//Variable de enclavamiento
int enclavamiento;
//------------------------Sensores-------------------------------------
//Sensor capacitivo 1 ingreso al área de llenado
const int S1 = 52;
//Contador de botellas en el ingreso al llenado
int cont1;
//Variable de ingreso del sensor S1
int ingreso1;
//Sensor capacitivo 2 ingreso al área de tapado
const int S2 =50;
//Variable de ingreso del sensor S2
int ingreso2;
//Contador de botellas en el ingreso al tapado
int cont2;
//Variable para envío de datos a la MTU (Global)
volatile int objeto;
//Variable para la recepción de datos desde la MTU (Global)
volatile int on_master;
//-------------------------Actuadores---------------------------------
//Motor de la banda transportadora
const int Q00 = 32;
//Bomba hidráulica
const int Q01 = 34;
//Cilindro de detención del área de llenado
const int Q03 = 36;
//Cilindro de posición de los dosificadores
const int Q04 = 38;
//Dosificadores
//Dosificador 1
const int Q05 = 24;
//Dosificador 2
const int Q09 = 26;
```

```
//Dosificador 3
const int Q10 = 28;
//Cilindro de detención del área de tapado extendido
const int Q06 = 42;
//Cilindro de detención del área de tapado contraido
const int Q07 = 44;
//Cilindro de tapado
const int Q08 = 40;

//-------------------------Inicio de los parametros del programa-------
void setup() {
  //-----------------------Protocolos de comunicación------------------
  //Inicio de la comunicación serial
  Serial.begin(9600);
  //Inicio de la comunicación I2C (Esclavo 2)
  Wire.begin(2);
  //Llamado de las funciones de requerimiento y recepción de
información
  //Función de requerimiento
  Wire.onRequest(EventoPeticion);
  //Función de recepción
  Wire.onReceive(EventoRecepcion);
  //Declaración de la funcionalidad de los pines
(ingreso=INPUT/salida=OUTPUT)
  //Botón de arranque
  pinMode(arranque, INPUT);
  //Botón de paro
  pinMode(paro, INPUT);
  //Sensor capacitivo 1 ingreso al área de llenado
  pinMode(S1, INPUT);
  //Sensor capacitivo 2 ingreso al área de tapado
  pinMode(S2, INPUT);
  //Motor banda transportadora
  pinMode(Q00, OUTPUT);
  //Bomba hidráulica
  pinMode(Q01, OUTPUT);
  //Cilindro de detención del área de llenado
  pinMode(Q03, OUTPUT);
  //Cilindro de posición de los dosificadores
  pinMode(Q04, OUTPUT);
  //Dosificadores
  //Dosificador 1
  pinMode(Q05, OUTPUT);
  //Dosificador 2
  pinMode(Q09, OUTPUT);
  //Dosificador 3
  pinMode(Q10, OUTPUT);
  //Cilindro de detención del área de tapado extendido
  pinMode(Q06, OUTPUT);
```

```
  //Cilindro de detención del área de tapado contraido
  pinMode(Q07, OUTPUT);
  //Cilindro de tapado
  pinMode(Q08, OUTPUT);
  //----------------------Estado inicial de los pines de salida-------
  //Motor banda transportadora (desactivado)
  digitalWrite(Q00, HIGH);
  //Cilindro de detención del área de llenado (desactivado)
  digitalWrite(Q03, HIGH);
  //Cilindro de posición de los dosificadores (desactivado)
  digitalWrite(Q04, HIGH);
  //Dosificadores (desactivados)
  //Dosificador 1
  digitalWrite(Q05, HIGH);
  //Dosificador 2
  digitalWrite(Q09, HIGH);
  //Dosificador 3
  digitalWrite(Q10, HIGH);
  //Bomba hidráulica (desactivado)
  digitalWrite(Q01, HIGH);
  //Cilindro de detención del área de tapado extendido (desactivado)
  digitalWrite(Q06, HIGH);
  //Cilindro de detención del área de tapado contraido (desactivado)
  digitalWrite(Q07, HIGH);
  //Cilindro de tapado (desactivado)
  digitalWrite(Q08, HIGH);
}



//------------Bucle de repetición de la secuencia de la máquina--------
void loop() {
  //Lectura del pin digital del botón de arranque
  inicio = digitalRead(arranque);
  //Preguntar si existe activación en el botón de encendido
  if (inicio == 1 || on_master == 3){
    //Cambiar el estado de la variable de enclavamiento
    enclavamiento = 1;
  }
  //Pregunta si el proceso está activado
  if (enclavamiento == 1){
    //Encendido del motor de la banda
    digitalWrite(Q00, LOW);
    //Cilindro de detención del área de llenado (activado)
    digitalWrite(Q03, LOW);
    //Cilindro de detención del área de tapado contraido (activado)
    digitalWrite(Q07, LOW);
    //----------------------Área de llenado----------------------
    //Ciclo de repetición para contar el ingreso de las botellas
```

```
    while(cont1 < 3) {
      //Almacenamiento de datos del sensor de ingreso
      ingreso1 = digitalRead(S1);
      //Tiempo de espera para la lectura
      delay(310);
      //Preguntar si existen datos en la variable de ingreso
      if(ingreso1 != 0){
        //Incrementar el contador en 1
        cont1 = cont1 + 1;
      }
    }
    //Variable para el envio de datos de ingreso a la MTU
    objeto = 7;
    //Si se cumplen las condiciones anteriores realizar la secuencia de
acciones de llenado (impresión de estado)
    if (cont1 == 3){
      //Apagado de la banda
      digitalWrite(Q00, HIGH);
      //Bajar el cilindro de los dosificadores
      digitalWrite(Q04, LOW);
      //Tiempo de bajada del cilindro
      delay(1000);
      //Encendido de la bomba hidráulica
      digitalWrite(Q01, LOW);
      //Apertura de los dosificadores
      digitalWrite(Q05, LOW);
      digitalWrite(Q09, LOW);
      digitalWrite(Q10, LOW);
      //Tiempo de espera de llenado
      delay(10000);
      //Apagado de la bomba hidráulica
      digitalWrite(Q01, HIGH);
      //Desactivar los dosificadores
      digitalWrite(Q05, HIGH);
      digitalWrite(Q09, HIGH);
      digitalWrite(Q10, HIGH);
      //Subir el cilindro de los dosificadores
      digitalWrite(Q04, HIGH);
      //Tiempo de espera de subida de los cilindros
      delay(1000);
      //Desactivación del cilindro de detención del área de llenado
      digitalWrite(Q03, HIGH);
      //Tiempo de espera de encendido de la banda
      delay(1000);
      //Encendido de la banda
      digitalWrite(Q00, LOW);
      //Cilindro de detención del área de tapado contraido (activado)
      digitalWrite(Q07, LOW);
      //Valor inicial del contador
```

```
      cont1 = 0;
      //Limpieza de envío de datos
      objeto = 0;
    }

    //--------------------Área de tapado-----------------------
    //Ciclo de repetición para contar el ingreso de las botellas
    while (cont2 < 3) {
      //Almacenamiento de datos del sensor de ingreso
      ingreso2 = digitalRead(S2);
      //Tiempo de espera para la lectura
      delay(300);
      //Preguntar si existen datos en la variable de ingreso
      if(ingreso2 != 0){
        //Incrementar el contador en 1
        cont2 = cont2 + 1;
        }
      //Impresión de los datos del contador
      Serial.println(cont2);
    }
    //Variable para el envio de datos de ingreso a la MTU
    objeto = 8;
    //Si se cumple realizar la secuencia de acciones de tapado
    if (cont2 == 3){
      Serial.println("Se detectó una botella en el área de
tapado");
      //Activación del cilindro de detención del área de tapado
      digitalWrite(Q06, LOW);
      digitalWrite(Q07, HIGH);
      //Tiempo de espera para apagado
      delay(300);
      //Apagado de la banda
      digitalWrite(Q00, HIGH);
      //Bajada del cilindro de tapado
      digitalWrite(Q08, LOW);
      //Tiempo de espera de bajada
      delay(2000);
      //Subida del cilindro de tapado
      digitalWrite(Q08, HIGH);
      //Desactivación del cilindro de detención del área de tapado
      digitalWrite(Q06, HIGH);
      digitalWrite(Q07, LOW);
      //Tiempo de espera subida
      delay(2000);
      //Encendido de la banda
      digitalWrite(Q00, LOW);
      //Valor inicial del contador
      cont2 = 0;
      //Limpieza de envío de datos
```

```
      objeto = 0;
    }
    enclavamiento = 1;
  }
  if(enclavamiento == 0) {
    //Motor banda transportadora (activado)
    digitalWrite(Q00, HIGH);
    //Cilindro de detención del área de llenado (activado)
    digitalWrite(Q03, HIGH);
    //Cilindro de posición de los dosificadores (desactivado)
    digitalWrite(Q04, HIGH);
    //Dosificadores (desactivado)
    //Dosificador 1
    digitalWrite(Q05, HIGH);
    //Dosificador 2
    digitalWrite(Q09, HIGH);
    //Dosificador 3
    digitalWrite(Q10, HIGH);
    //Bomba hidráulica (desactivado)
    digitalWrite(Q01, HIGH);
    //Cilindro de detención del área de tapado extendido (desactivado)
    digitalWrite(Q06, HIGH);
    //Cilindro de detención del área de tapado contraido (activado)
    digitalWrite(Q07, HIGH);
    //Cilindro de tapado (desactivado)
    digitalWrite(Q08, HIGH);
  }
}

//----------------------Funciones de comunicación I2C---------------
//Función de envio de información desde el esclavo al maestro una vez
hecha la petición
void EventoPeticion() {
  Wire.write(objeto);
}
//Función de recepción de ordenes desde el maestro (MTU)
void EventoRecepcion(int on) {
  while (Wire.available() > 0) {
    on_master = Wire.read();
  }
}
```

ANEXO J: ALGORITMO DE CONTROL DEL DISPOSITIVO MAESTRO

Script de control del proceso IDE Arduino

```
//--------------------------------MTU (MAESTRO)-------------------------
//--------------------------------Librerias-----------------------------
//Libreria para la comunicación I2C
#include <Wire.h>

//--------------------------------Variables-----------------------------
int on;
int on1;
int i;
int i1;
int objeto;
int objeto1;

//-------------------------Ejecución del estado inicial del proceso---
void setup() {
  //Inicio de comunicación del bus I2C
  Wire.begin();
  //Inicio de la comunicación serial
  Serial.begin(9600);

}

//---------------------------Ejecución repetitiva del maestro----------
void loop() {
  //----------------------Envío de datos a los esclavos----------------
  //Envío de datos al esclavo 1
  //Inicio de transmisión de datos
  Wire.beginTransmission(1);
  //Preguntar si existe información en el puerto serial
  if (Serial.available() > 0) {
    //Iniciar el contador para la toma de datos
    i = 0;
    //Estructura de repetición para la toma de un solo dato
    while (i < 1) {
      //Almacenamiento del dato que llegó desde el puerto serial
      on = Serial.parseInt();
      //Impresión en el puerto serial del dato almacenado
      Serial.println(on);
      //Aumentar el contador
      i = i + 1;
    }
  }
  Wire.write(on);
  Wire.endTransmission();

  //Envío de datos al esclavo 2
```

```
//Inicio de transmisión de datos
Wire.beginTransmission(2);
//Preguntar si existe información en el puerto serial
if (Serial.available() > 0) {
  //Iniciar el contador para la toma de datos
  i1 = 0;
  //Estructura de repetición para la toma de un solo dato
  while (i1 < 2) {
    //Almacenamiento del dato que llegó desde el puerto serial
    on1 = Serial.parseInt();
    //Impresión en el puerto serial del dato almacenado
    Serial.println(on1);
    //Aumentar el contador
    i1 = i1 + 1;
  }
}
Wire.write(on1);
Wire.endTransmission();

//--------------------Recepción de datos de los esclavos-------------
//Requerimientos de los bytes al esclavo 1
Wire.requestFrom(1, 1);
//Preguntar si existen datos desde el bus de comunicación I2C
while (Wire.available() > 0) {
  //Recepción de los datos desde el esclavo 1
  objeto = Wire.read();
}
Serial.println(objeto);

//Requerimientos de los bytes al esclavo 2
Wire.requestFrom(2, 1);
//Preguntar si existen datos desde el bus de comunicación I2C
while (Wire.available() > 0) {
  //Recepción de los datos desde el esclavo 1
  objeto1 = Wire.read();
}
Serial.println(objeto1);}
```

ANEXO K: ALGORITMO DE LA INTERFAZ HMI

Módulo HMI_modulo

```
#-----------------------------------------------------Librerías y
módulos----------------------------------------
#Libreria de ventanas y efectos
from tkinter import Frame, Label, Button, PhotoImage, ttk, Menu
#Libreria para el uso de imagenes
from PIL import Image, ImageTk
#Librerias para la comunicación con arduino
import time
import serial
#Libreria para ejecutar distintos hilos
import threading
#Libreria para la base de datos
from eventos import crear_tabla, borrar_tabla, Clasificar,
guardar_clasificar, Envasado, guardar_envasado

#Función de la barra de menú
def barra_menu(master):
    barra_menu = Menu(master)
    master.config(menu=barra_menu, width=300, height=300)

    menu_comunicacion = Menu(barra_menu, tearoff=0)
    barra_menu.add_cascade(label="Comunicación Serial",
menu=menu_comunicacion)

    menu_comunicacion.add_command(label="Iniciar la comunicación",
command=HMI.iniciar_com)
    menu_comunicacion.add_command(label="Cerrar la comunicación",
command=HMI.cerrar_com)

    menu_bd = Menu(barra_menu, tearoff=0)
    barra_menu.add_cascade(label="Base de datos", menu=menu_bd)
    menu_bd.add_command(label="Crear registro en la BD",
command=crear_tabla)
    menu_bd.add_command(label="Eliminar registro en la BD",
command=borrar_tabla)


#Clase del frame que irá dentro de la ventana
class HMI(Frame):
    #Función de construcción
    def __init__(self, master = None):
        #Llamar a la súper clase Frame para la creación de un frame
dando valores de altura y ancho
        super().__init__(master, height=700, width=1500)
        #Almacenar la variable master de la clase padre en la
variable master de la clase hija
        self.master = master
        #Definir la ubicación del frame creado
        self.pack()
        #llamar a la función de elementos visuales de la HMI
        self.visual()
        #Llamar a la función para la comunicación serial
        self.iniciar_com()


    #Función para el inicio de la comunicación serial
    def iniciar_com(self):
        # Iniciar comunicación serial con el microcontrolador
```

```
arduino
        self.arduinom1 = serial.Serial('COM6', 9600, timeout=1)
        # Tiempo de espera
        time.sleep(1)


    #Función para cerrar la comunicación
    def cerrar_com(self):
        self.arduinom1.close()

    #Función de evento de encendido del proceso de selección y
clasificación de botellas
    def encender_maq1(self):
        #Imagen de luz encendida
        #Redimensionamiento de la imagen
        self.im_led_on = Image.open('img/led_on_out.png')
        self.im_led_on = self.im_led_on.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led_onc = ImageTk.PhotoImage(self.im_led_on)
        self.lblimledon = Label(self, image=self.im_led_onc)
        self.lblimledon.place(x=100, y=270)
        #Envío de datos a Arduino
        self.arduinom1.write(b'1')
        #Variable para iniciar la lectura
        self.lectura = True
        #Hilo para ejecutar el proceso de lectura desde el sensor
        self.hilo1 = threading.Thread(target=self.lectura_com)
        #Ejecución del proceso de lectura de datos
        self.hilo1.start()



    #Función de evento de apagado del proceso de selección y
clasificación de botellas
    def apagar_maq1(self):
        #Imagen de luz apagada
        #Redimensionamiento de la imagen
        self.im_led_off = Image.open('img/led_off_out.png')
        self.im_led_off = self.im_led_off.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led_offc = ImageTk.PhotoImage(self.im_led_off)
        self.lblimledoff = Label(self, image=self.im_led_offc)
        self.lblimledoff.place(x=100, y=270)
        #Envío de datos a Arduino
        self.arduinom1.write(b'2')
        # Variable para iniciar la lectura
        self.lectura = False

    #Función de lectura de datos de la máquina 1
    def lectura_com(self):
        time.sleep(1.0)
        #Limpiar el puerto serial
        self.arduinom1.reset_input_buffer()
        try:
            while (self.lectura):
                s1 = self.arduinom1.readline()
                s1 = s1.decode('ascii', 'strict').strip()
                #Detección de un objeto en el ingreso de la máquina
selectora y clasificadora de botellas
```

```
                if (s1 == '3'):
                    #Ingreso de objetos en la máquina
                    #Imagen de luz encendida
                    #Redimensionamiento de la imagen
                    self.im_led_in =
Image.open('img/led_on_out.png')
                    self.im_led_in = self.im_led_in.resize((60, 60),
Image.ANTIALIAS)
                    # Posición de la imagen
                    self.im_led_inc =
ImageTk.PhotoImage(self.im_led_in)
                    self.lblimledin = Label(self,
image=self.im_led_inc)
                    self.lblimledin.place(x=250, y=270)
                    #Tiempo de espera
                    time.sleep(4)
                    #Limpiar el puerto serial
                    s1 = self.arduinom1.reset_input_buffer()
                    #Imagen de luz apagada 2
                    #Redimensionamiento de la imagen
                    self.im_ledout =
Image.open('img/led_off_out.png')
                    self.im_ledout = self.im_ledout.resize((60, 60),
Image.ANTIALIAS)
                    #Posición de la imagen
                    self.im_led2out =
ImageTk.PhotoImage(self.im_ledout)
                    self.lblimledout = Label(self,
image=self.im_led2out)
                    self.lblimledout.place(x=250, y=270)
                #Detección de una botella de aluminio
                if (s1 == '4'):
                    #Detección de una botella de aluminio
                    #Encendido del led simulado
                    self.imon = Image.open('img/miniledon.JPG')
                    self.imon = self.imon.resize((30, 15),
Image.ANTIALIAS)
                    self.imon1 = ImageTk.PhotoImage(self.imon)
                    self.lblimalon = Label(self, image=self.imon1)
                    self.lblimalon.place(x=480, y=265)
                    #Datos insertados
                    self.tabla1.insert("", 0, text="1",
values=("Aluminio"))
                    #Guaradar en la base de datos
                    maq_clasifica = Clasificar("1", "Aluminio")
                    guardar_clasificar(maq_clasifica)
                    #Tiempo de espera
                    time.sleep(4.0)
                    #Limpiar el puerto serial
                    s1 = self.arduinom1.reset_input_buffer()
                    #Apagado del led simulado
                    self.imoffal = Image.open('img/miniledoff.JPG')
                    self.imoffal = self.imoffal.resize((30, 15),
Image.ANTIALIAS)
                    self.imoffal1 = ImageTk.PhotoImage(self.imoffal)
                    self.lblimal = Label(self, image=self.imoffal1)
                    self.lblimal.place(x=480, y=265)
                #Detección de una botella de plástico
                if (s1 == '5'):
                    #Detección de una botella de plástico
                    #Encendido del led simulado
```

```
                    self.imonpt = Image.open('img/miniledon.JPG')
                    self.imonpt = self.imonpt.resize((30, 15),
Image.ANTIALIAS)
                    self.imon1pt = ImageTk.PhotoImage(self.imonpt)
                    self.lblimpton = Label(self, image=self.imon1pt)
                    self.lblimpton.place(x=480, y=290)
                    #Datos insertados
                    self.tabla1.insert("", 0, text="1",
values=("Plástico"))
                    #Guardar en la base de datos
                    maq_clasifica = Clasificar("1", "Plástico")
                    guardar_clasificar(maq_clasifica)
                    #Tiempo de espera
                    time.sleep(4.0)
                    #Limpiar el puerto serial
                    s1 = self.arduinom1.reset_input_buffer()
                    #Apagado del led simulado
                    self.imoffpt = Image.open('img/miniledoff.JPG')
                    self.imoffpt = self.imoffpt.resize((30, 15),
Image.ANTIALIAS)
                    self.imoffpt1 = ImageTk.PhotoImage(self.imoffpt)
                    self.lblimpt = Label(self, image=self.imoffpt1)
                    self.lblimpt.place(x=480, y=290)
                #Detección de una botella de vidrio
                if (s1 == '6'):
                    #Detección de una botella de vidrio
                    #Encendido del led simulado
                    self.imonvr = Image.open('img/miniledon.JPG')
                    self.imonvr = self.imonvr.resize((30, 15),
Image.ANTIALIAS)
                    self.imon1vr = ImageTk.PhotoImage(self.imonvr)
                    self.lblimvron = Label(self, image=self.imon1vr)
                    self.lblimvron.place(x=480, y=315)
                    #Datos insertados
                    self.tabla1.insert("", 0, text="1",
values=("Vidrio"))
                    #Guardar en la base de datos
                    maq_clasifica = Clasificar("1", "Vidrio")
                    guardar_clasificar(maq_clasifica)
                    #Tiempo de espera
                    time.sleep(4.0)
                    #Limpiar el puerto serial
                    s1 = self.arduinom1.reset_input_buffer()
                    #Apagado del led simulado
                    self.imoffvr = Image.open('img/miniledoff.JPG')
                    self.imoffvr = self.imoffvr.resize((30, 15),
Image.ANTIALIAS)
                    self.imoffvr1 = ImageTk.PhotoImage(self.imoffvr)
                    self.lblimvr = Label(self, image=self.imoffvr1)
                    self.lblimvr.place(x=480, y=315)
            except TypeError:
                pass

        #Función de encendido de la máquina 2
        def encender_maq2(self):
            #Imagen de luz encendida
            #Redimensionamiento de la imagen
            self.im_led_onm2 = Image.open('img/led_on_out.png')
            self.im_led_onm2 = self.im_led_onm2.resize((60, 60),
Image.ANTIALIAS)
            #Posición de la imagen
```

```
        self.im_led_oncm2 = ImageTk.PhotoImage(self.im_led_onm2)
        self.lblimledonm2 = Label(self, image=self.im_led_oncm2)
        self.lblimledonm2.place(x=785, y=270)
        #Envío de datos a Arduino
        self.arduinom1.write(b'3')
        #Variable para iniciar la lectura
        self.lecturam2 = True
        # Hilo para ejecutar el proceso de lectura desde el sensor
        self.hilo2 = threading.Thread(target=self.lectura_com1)
        # Ejecución del proceso de lectura de datos
        self.hilo2.start()

    #Función de apagado de la máquina 2
    def apagar_maq2(self):
        #Imagen de luz apagada
        #Redimensionamiento de la imagen
        self.im_led_offm2 = Image.open('img/led_off_out.png')
        self.im_led_offm2 = self.im_led_offm2.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led_offcm2 = ImageTk.PhotoImage(self.im_led_offm2)
        self.lblimledoffm2 = Label(self, image=self.im_led_offcm2)
        self.lblimledoffm2.place(x=785, y=270)
        #Envío de datos a Arduino
        self.arduinom1.write(b'4')
        #Variable para iniciar la lectura
        self.lecturam2 = False

    #Función de lectura de datos de la máquina 2
    def lectura_com1(self):
        time.sleep(1.0)
        self.arduinom1.reset_input_buffer()
        try:
            while (self.lecturam2):
                s2 = self.arduinom1.readline()
                s2 = s2.decode('ascii', 'strict').strip()
                #Detección del ingreso al área de envasado
                if (s2 == '7'):
                    # Ingreso de objetos en la máquina
                    # Imagen de luz encendida
                    # Redimensionamiento de la imagen
                    self.im_led_inm2 =
Image.open('img/led_on_out.png')
                    self.im_led_inm2 = self.im_led_inm2.resize((60,
60), Image.ANTIALIAS)
                    # Posición de la imagen
                    self.im_led_incm2 =
ImageTk.PhotoImage(self.im_led_inm2)
                    self.lblimledinm2 = Label(self,
image=self.im_led_incm2)
                    self.lblimledinm2.place(x=940, y=270)
                    # Tiempo de espera
                    time.sleep(7)
                    s2 = self.arduinom1.reset_input_buffer()
                    # Imagen de luz apagada 2
                    # Redimensionamiento de la imagen
                    self.im_ledoutm2 =
Image.open('img/led_off_out.png')
                    self.im_ledoutm2 = self.im_ledoutm2.resize((60,
60), Image.ANTIALIAS)
                    # Posición de la imagen
```

```
                    self.im_led2outm2 =
ImageTk.PhotoImage(self.im_ledoutm2)
                    self.lblimledoutm2 = Label(self,
image=self.im_led2outm2)
                    self.lblimledoutm2.place(x=940, y=270)
                #Detección del ingreso al área de envasado
                if (s2 == '8'):
                    # Detección de una botella de aluminio
                    # Encendido del led simulado
                    self.imonm21 = Image.open('img/led_on_out.png')
                    self.imonm21 = self.imonm21.resize((60, 60),
Image.ANTIALIAS)
                    self.imon1m21 = ImageTk.PhotoImage(self.imonm21)
                    self.lblimalonm21 = Label(self,
image=self.imon1m21)
                    self.lblimalonm21.place(x=1125, y=270)
                    # Datos insertados
                    self.tabla2.insert("", 0, text="3",
values=("Botellas"))
                    # Guardar en la base de datos
                    maq_envasado = Envasado("3", "Botellas")
                    guardar_envasado(maq_envasado)
                    # Tiempo de espera
                    time.sleep(3)
                    s2 = self.arduinom1.reset_input_buffer()
                    # Apagado del led simulado
                    self.imoffalm21 =
Image.open('img/led_off_out.png')
                    self.imoffalm21 = self.imoffalm21.resize((60,
60), Image.ANTIALIAS)
                    self.imoffa1m21 =
ImageTk.PhotoImage(self.imoffalm21)
                    self.lblimalm21 = Label(self,
image=self.imoffa1m21)
                    self.lblimalm21.place(x=1125, y=270)
            except TypeError:
                pass

    #Función para la creación de los elementos visuales de la HMI
    def visual(self):
        #Frame para las etiquetas de la ventana
        frame1 = Frame(self, height=140, width=1500)
        frame1.place(x=0, y=0)

        #Etiquetas de título de la ventana
        self.lbl1 = Label(frame1, text="ESCUELA SUPERIOR POLITÉCNICA
DE CHIMBORAZO", font=('Verdana', 14))
        self.lbl1.place(x=440, y=0)
        self.lbl2 = Label(frame1, text="FACULTAD DE MECÁNICA",
font=('Verdana', 14))
        self.lbl2.place(x=550, y=30)
        self.lbl3 = Label(frame1, text="CARRERA DE INGENIERÍA
INDUSTRIAL", font=('Verdana', 14))
        self.lbl3.place(x=500, y=60)
        self.lbl4 = Label(frame1,
                          text="TRABAJO DE TITULACIÓN:
IMPLEMENTACIÓN DE UN SISTEMA SCADA PARA LA INTEGRACIÓN DE PROCESOS
INDUSTRIALES BASADO EN UN AMBIENTE DE PROGRAMACIÓN OPEN SOURCE",
                          font=('Verdana', 10))
        self.lbl4.place(x=60, y=90)
        self.lbl5 = Label(frame1, text="AUTOR: CRISTIAN ROMERO",
```

```
font=('Verdana', 10))
        self.lbl5.place(x=580, y=110)

        #Imagen 1
        #Redimensionamiento de la imagen
        self.ima1 =
Image.open('img/Escudo_de_la_Escuela_Superior_Politécnica_de_Chimbor
azo.png')
        self.ima1 = self.ima1.resize((80, 80), Image.ANTIALIAS)
        #Colocar la imagen
        self.ima2 = ImageTk.PhotoImage(self.ima1)
        self.lblima1 = Label(self, image=self.ima2)
        self.lblima1.place(x=300, y=5)

        #Imagen 2
        #Redimensionamiento de la imagen
        self.ima0 = Image.open('img/imageedit_4_8675083414-removebg-
preview.png')
        self.ima0 = self.ima0.resize((80, 80), Image.ANTIALIAS)
        #Colocar la imagen
        self.img = ImageTk.PhotoImage(self.ima0)
        self.lbl_img = Label(self, image=self.img)
        self.lbl_img.place(x=1000, y=5)

        #-----------------------------------------Proceso de
selección y clasificación de botellas---------------------
        #Frame para la máquina selectora y clasificadora de botellas
        frame2 = Frame(self, height=560, width=680)
        frame2.place(x=0, y=140)
        #Etiqueta para la máquina selectora y clasificadora de
botellas
        self.lbl5 = Label(frame2, text="Proceso de selección y
clasificación de botellas", font=('Verdana', 14))
        self.lbl5.place(x=90, y=0)
        #Botones
        #Botón de inicio
        #Importar la imagen para el botón
        self.img_ini = PhotoImage(file='img/Inicio.png')
        #Creación del botón
        self.inicio = Button(frame2, text='Iniciar',
image=self.img_ini, height='40', width='200',
command=self.encender_maq1)
        #Posición del botón
        self.inicio.place(x=100, y=40)
        #Botón de fin
        #Importar la imagen para el botón
        self.img_fin = PhotoImage(file='img/Finalizar.png')
        #Creación del botón
        self.final = Button(frame2, text='Finalizar',
image=self.img_fin, height='40', width='200',
command=self.apagar_maq1)
        #Posición del botón
        self.final.place(x=330, y=40)

        #Imagen SCADA
        #Redimensionamiento de la imagen
        self.im_scb = Image.open('img/CAPTURA.png')
        self.im_scb = self.im_scb.resize((550, 250),
Image.ANTIALIAS)
        #Colocar la imagen
        self.im_scb1 = ImageTk.PhotoImage(self.im_scb)
```

```
        self.lbl_im_maq1 = Label(frame2, image=self.im_scb1)
        self.lbl_im_maq1.place(x=30, y=200)

        #Frame de los datos
        frame21 = Frame(frame2, height=100, width=500,
relief='raised', border=1)
        frame21.place(x=80, y=100)
        #Etiqueta de los datos de ingreso
        self.lblsi0 = Label(frame21, text="Proceso",
font=('Verdana', 11))
        self.lblsi0.place(x=20, y=0)
        self.lblsi = Label(frame21, text="Sensor de ingreso",
font=('Verdana', 11))
        self.lblsi.place(x=130, y=0)
        self.lblsi1 = Label(frame21, text="Tipo de material",
font=('Verdana', 11))
        self.lblsi1.place(x=320, y=0)
        self.lblsia = Label(frame21, text="Aluminio",
font=('Verdana', 11))
        self.lblsia.place(x=320, y=24)
        self.lblsip = Label(frame21, text="Plástico",
font=('Verdana', 11))
        self.lblsip.place(x=320, y=49)
        self.lblsiv = Label(frame21, text="Vidrio", font=('Verdana',
11))
        self.lblsiv.place(x=320, y=74)

        #Etiqueta del led
        #Imagen de luz apagada 1
        #Redimensionamiento de la imagen
        self.im_led = Image.open('img/led_off_out.png')
        self.im_led = self.im_led.resize((60, 60), Image.ANTIALIAS)
        # Posición de la imagen
        self.im_led0 = ImageTk.PhotoImage(self.im_led)
        self.lblimled = Label(self, image=self.im_led0)
        self.lblimled.place(x=100, y=270)
        #Imagen de luz apagada 2
        #Redimensionamiento de la imagen
        self.im_led1 = Image.open('img/led_off_out.png')
        self.im_led1 = self.im_led1.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led2 = ImageTk.PhotoImage(self.im_led1)
        self.lblimled1 = Label(self, image=self.im_led2)
        self.lblimled1.place(x=250, y=270)

        #Imagenes de la sección de clasificación
        #Redimensionamiento de la imagen
        self.imoff1 = Image.open('img/miniledoff.JPG')
        self.imoff1 = self.imoff1.resize((30,15), Image.ANTIALIAS)
        self.imoff = ImageTk.PhotoImage(self.imoff1)
        #Aluminio
        self.lblimal = Label(self, image=self.imoff)
        self.lblimal.place(x=480, y=265)
        #Plástico
        self.lblimpl = Label(self, image=self.imoff)
        self.lblimpl.place(x=480, y=290)
        #Vidrio
        self.lblimvr = Label(self, image=self.imoff)
        self.lblimvr.place(x=480, y=315)
```

```
        #Tabla de datos
        self.tabla1 = ttk.Treeview(frame2, columns=("col1"))
        #Columnas
        self.tabla1.column("#0", width=70, anchor='center')
        self.tabla1.column("col1", width=200, anchor='center')
        #Encabezados
        self.tabla1.heading("#0", text="Cantidad", anchor='center')
        self.tabla1.heading("col1", text="Tipo de material",
anchor='center')
        #Posición de la tabla
        self.tabla1.place(x=160, y= 450)

        #--------------------Proceso de envasado lineal--------------
-------------------------------------
        #Frame para la máquina envasadora lineal
        frame3 = Frame(self, height=560, width=700)
        frame3.place(x=680, y=140)
        #Etiqueta para la máquina envasadora lineal
        self.lbl6 = Label(frame3, text="Proceso de envasado lineal",
font=('Verdana', 14))
        self.lbl6.place(x=220, y=0)

        #Botones
        #Botón de inicio
        #Importar la imagen para el botón
        self.img_ini1 = PhotoImage(file='img/Inicio.png')
        #Creación del botón
        self.inicio1 = Button(frame3, text='Iniciar',
image=self.img_ini1, height='40', width='200',
command=self.encender_maq2)
        #Posición del botón
        self.inicio1.place(x=120, y=40)
        #Botón de fin
        #Importar la imagen para el botón
        self.img_fin1 = PhotoImage(file='img/Finalizar.png')
        #Creación del botón
        self.final1 = Button(frame3, text='Finalizar',
image=self.img_fin1, height='40', width='200',
command=self.apagar_maq2)
        #Posición del botón
        self.final1.place(x=350, y=40)

        #Frame de los datos
        frame31 = Frame(frame3, height=100, width=500,
relief='raised', border=1)
        frame31.place(x=80, y=100)

        #Etiqueta de los datos de ingreso
        self.lblsi20 = Label(frame31, text="Proceso",
font=('Verdana', 11))
        self.lblsi20.place(x=20, y=0)
        self.lblsi21 = Label(frame31, text="Área de llenado",
font=('Verdana', 11))
        self.lblsi21.place(x=150, y=0)
        self.lblsi23 = Label(frame31, text="Área de tapado",
font=('Verdana', 11))
        self.lblsi23.place(x=340, y=0)

        #Etiqueta del led
        #Imagen de luz apagada 1
        #Redimensionamiento de la imagen
```

```
        self.im_led20 = Image.open('img/led_off_out.png')
        self.im_led20 = self.im_led.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led21 = ImageTk.PhotoImage(self.im_led20)
        self.lblimled20 = Label(self, image=self.im_led21)
        self.lblimled20.place(x=785, y=270)
        #Imagen de luz apagada 2
        #Redimensionamiento de la imagen
        self.im_led22 = Image.open('img/led_off_out.png')
        self.im_led22 = self.im_led22.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led23 = ImageTk.PhotoImage(self.im_led22)
        self.lblimled21 = Label(self, image=self.im_led23)
        self.lblimled21.place(x=940, y=270)
        #Imagen de luz apagada 3
        #Redimensionamiento de la imagen
        self.im_led32 = Image.open('img/led_off_out.png')
        self.im_led32 = self.im_led32.resize((60, 60),
Image.ANTIALIAS)
        #Posición de la imagen
        self.im_led33 = ImageTk.PhotoImage(self.im_led32)
        self.lblimled22 = Label(self, image=self.im_led33)
        self.lblimled22.place(x=1125, y=270)

        #Tabla de datos
        self.tabla2 = ttk.Treeview(frame3, columns=("col1"))
        #Columnas
        self.tabla2.column("#0", width=70, anchor='center')
        self.tabla2.column("col1", width=200, anchor='center')
        #Encabezados
        self.tabla2.heading("#0", text="Cantidad", anchor='center')
        self.tabla2.heading("col1", text="Objeto", anchor='center')
        #Posición de la tabla
        self.tabla2.place(x=160, y=450)

        #Imagen SCADA
        #Redimensionamiento de la imagen
        self.im_scb2 = Image.open('img/Captura1.png')
        self.im_scb2 = self.im_scb2.resize((550, 245),
Image.ANTIALIAS)
        # Colocar la imagen
        self.im_scb3 = ImageTk.PhotoImage(self.im_scb2)
        self.lbl_im_maq2 = Label(frame3, image=self.im_scb3)
        self.lbl_im_maq2.place(x=40, y=200)
```

Módulo conexionbd

```
#Importar la libreria para la conexión con la base de datos
import sqlite3

#Clase para la conexíon de la base de datos
class DB_SCADA:
    #Función de construcción
    def __init__(self):
        self.base_datos = 'database/producción.db'
        self.conexion = sqlite3.connect(self.base_datos)
        self.cursor = self.conexion.cursor()

    def cerrar(self):
        self.conexion.commit()
        self.conexion.close()
```

Módulo eventos

```
#Importar módulo
from conexiondb import DB_SCADA
from tkinter import messagebox

def crear_tabla():
    #Conexión con la base de datos
    conexion = DB_SCADA()
    sql = '''
        CREATE TABLE clasificacion(
            Cantidad VARCHAR(10),
            Material VARCHAR(100)
        )
        '''
    sql1 = '''
        CREATE TABLE dosificacion(
            Cantidad VARCHAR(10),
            Objeto VARCHAR(100)
        )
        '''
    try:
        conexion.cursor.execute(sql)
        conexion.cursor.execute(sql1)
        conexion.cerrar()
        titulo = "Crear registro"
        mensaje = "Se creo la tabla en la base de datos"
        messagebox.showinfo(titulo, mensaje)
    except:
        titulo = "Crear registro"
        mensaje = "La tabla ya está creada"
        messagebox.showwarning(titulo, mensaje)

def borrar_tabla():
    conexion = DB_SCADA()

    sql = 'DROP TABLE clasificacion'

    sql1 = 'DROP TABLE dosificacion'

    try:
        conexion.cursor.execute(sql)
        conexion.cursor.execute(sql1)
```

```
        conexion.cerrar()
        titulo = "Borrar registro"
        mensaje = "La tabla de la base de datos ha sido borrada
exitosamente"
        messagebox.showinfo(titulo, mensaje)
    except:
        titulo = "Borrar registro"
        mensaje = "No existe ninguna tabla"
        messagebox.showinfo(titulo, mensaje)

#Clase para crear el objeto clasificación
class Clasificar:
    #Función de construcción
    def __init__(self, cantidad, material):
        self.cantidad = cantidad
        self.material = material

    def __str__(self):
        return f"clasificacion[{self.cantidad}, {self.material}]"

def guardar_clasificar(clasificacion):
    conexion = DB_SCADA()

    sql = f"""INSERT INTO clasificacion(Cantidad, Material)
            VALUES ('{clasificacion.cantidad}',
'{clasificacion.material}')
            """

    try:
        conexion.cursor.execute(sql)
        conexion.cerrar()
    except:
        titulo = "Conexión al registro"
        mensaje = "No se ha creado la tabla en la base de datos"
        messagebox.showerror(titulo, mensaje)

#Clase paraa crear el objeto envasado
class Envasado:
    #Función de construcción
    def __init__(self, cantidad3, objeto):
        self.cantidad3 = cantidad3
        self.objeto = objeto

    def __str__(self):
        return f"dosificacion[{self.cantidad3}, {self.objeto}]"

def guardar_envasado(dosificacion):
    conexion = DB_SCADA()

    sql = f"""INSERT INTO dosificacion(Cantidad, Objeto)
            VALUES ('{dosificacion.cantidad3}',
'{dosificacion.objeto}')
            """

    try:
        conexion.cursor.execute(sql)
        conexion.cerrar()
    except:
        titulo = "Conexión al registro"
        mensaje = "No se ha creado la tabla en la base de datos"
        messagebox.showerror(titulo, mensaje)
```

Módulo main

```
#------------------------------Librerias y módulos-----------------
----------------
#Libreria para ejecutar la HMI
import time
from tkinter import Tk
#Libreria del frame HMI
from HMI_modulo import HMI, barra_menu



#------------------------------Función de ejecución de la ventana--
----------------
#Función para la ejecución de la ventana
def main():
    inicio = time.time()
    #Creación de la ventana
    raiz = Tk()
    #Título de la ventana
    raiz.wm_title("HMI SCADA")
    barra_menu(raiz)
    #Llamar a la clase que contiene el frame
    app = HMI(raiz)
    fin = time.time()
    print(fin - inicio)
    #Ejecutar el frame del módulo
    app.mainloop()

if __name__ == "__main__":
    main()
```