



UNIVERSIDAD
DE MÁLAGA



ESCUELA DE INGENIERÍAS INDUSTRIALES

DPTO. TECNOLOGÍA ELECTRÓNICA

TECNOLOGÍA ELECTRÓNICA

TRABAJO FIN DE GRADO

Sistema de manipulación robotizada de un dispensador de bebidas carbonatadas mediante visión artificial

Grado en: **INGENIERÍA ELÉCTRICA**

Autor: **MAHER BOUYMEJAN**

Tutor: **DR. FRANCISCO JOSÉ SÁNCHEZ PACHECO**

MÁLAGA, JUNIO de 2.023

Agradecimientos:

A mi familia, especialmente a mis padres y mis hermanos, que siempre han creído en mí y me han apoyado en todo momento. A Beatriz y a las demás personas que me han ofrecido su apoyo y cariño.

A mis tutores Francisco y José Ramón, por brindarme la oportunidad de realizar este proyecto tan interesante y retador, por orientarme con su experiencia y sabiduría, y por todos sus valiosos consejos que me han ayudado a crecer profesional y personalmente.

A Grupo PREMO y a todos los que forman parte de él, por acogerme con tanta amabilidad y generosidad, y por compartir conmigo sus conocimientos y su pasión por el trabajo bien hecho.

RESUMEN

En el presente trabajo de fin de grado se desarrolla un sistema para automatizar un dispensador de cerveza que incorpora un compresor propio, haciendo uso de un brazo robótico colaborativo que es guiado por un sistema de visión artificial utilizando tanto redes neuronales convolucionales como umbralización.

Con el objetivo de controlar el dispensador, se ha creado un algoritmo que supervisa el tiempo de llenado de las diferentes secciones del vaso, lo cual permite realizar análisis y la toma de decisiones para regular el proceso de dispensación.

Además, se han diseñado prototipos como un dispensador de vasos personalizados y otras piezas necesarias para el funcionamiento del sistema.

En resumen, este trabajo aborda el desarrollo de un sistema completo de dispensado de cerveza automatizado, incorporando tecnologías avanzadas como la visión artificial y la robótica colaborativa, junto con la creación de prototipos para mejorar la experiencia del usuario.

PALABRAS CLAVE

Visión artificial, Robótica, Inteligencia artificial, Redes neuronales convolucionales, análisis de datos.

ABSTRACT

In this undergraduate thesis, a system is developed to automate a beer dispenser that incorporates its own compressor, using a collaborative robotic arm guided by an artificial vision system that utilizes both convolutional neural networks and thresholding.

With the aim of controlling the dispenser, an algorithm has been created to monitor the filling time of different sections of the glass, allowing for analysis and decision-making to regulate the dispensing process.

Furthermore, prototypes have been designed, such as a dispenser for custom glasses and other necessary pieces for the system's operation.

In summary, this work addresses the development of a complete automated beer dispensing system, incorporating advanced technologies such as artificial vision and collaborative robotics, along with the creation of prototypes to enhance the user experience.

KEYWORDS

Artificial vision, Robotics, Artificial intelligence, Convolutional neural networks, Data analysis.

Índice de contenido:

1. Introducción.....	1
2. Objetivos.....	3
3. Software utilizado.....	5
4. Elementos utilizados.....	7
4.1. Dispensador de cerveza.....	7
4.2. Dispensador de vasos de plástico.....	8
4.3. Brazo robótico colaborativo.....	10
4.4. Caja de conexiones.....	11
4.5. Ordenador táctil.....	12
5. Equipamiento Utilizado.....	13
5.1. Impresoras 3D de resina.....	13
5.1.1. Resina 4K:.....	13
5.1.2. Resina Draft:.....	14
5.2. CNC (Centro de Mecanizado de Control Numérico).....	14
6. Diseño del dispensador de vasos.....	15
6.1. Base del dispensador.....	16
6.2. Separador de vasos del dispensador.....	17
6.3. Actuador lineal.....	18
7. Diseño del mecanismo para la actuación sobre grifo.....	20
8. Esquema de montaje.....	24
9. Código de programación.....	29
9.1. Interfaz gráfica.....	31
9.2. Base de datos.....	33
9.3. Visión Artificial.....	34
9.4. Código del brazo robótico.....	37
9.5. Código Arduino.....	38
9.6. Código para servir una cerveza.....	40
10. Redes neuronales.....	48
10.1. YOLO V8.....	48

10.1.1.	Funcionamiento de YOLO:	49
10.1.2.	Principales características de YOLOv8:	49
10.1.3.	Proceso de personalización de YOLOv8:	50
10.2.	Red customizada para la detección de vasos:	51
10.2.1.	Matriz de confusión	54
11.	Manual de usuario del sistema:	57
11.1.	Pantalla de inicialización:	57
11.2.	Pantalla de error al inicio.	59
11.2.1.	Error de conexión del brazo robótico:	59
11.2.2.	Error de conexión a internet:	60
11.3.	Pantalla de menú principal:	62
11.3.1.	Test.	64
11.3.2.	Analytics	66
11.3.3.	Positions	70
11.3.4.	Computer Vision	72
11.3.5.	Artificial Intelligence	74
11.3.6.	Parameters	76
11.3.7.	Start Demo	78
11.4.	Pantalla demo:	79
11.5.	Pantalla servir cerveza:	80
11.6.	Pantalla de encuesta.	83
11.7.	Pantalla de Error: Vaso no detectado:	85
11.8.	Pantalla de Error: Cerveza no detectada:	87
11.9.	Pantalla de Error: Error en el Brazo:	89
12.	Prueba de funcionamiento.	91
13.	Conclusión.	96
14.	Bibliografía	98
15.	Anexos	100
15.1.	Código de programación:	100
15.1.1.	Importación de todas las librerías necesarias (library_imports.py)	100
15.1.2.	Control del robot (Cobot.py)	102
15.1.3.	Ejecución de movimientos del robot en un hilo (bot_hilo.py)...	109
15.1.4.	Gestión de la base de datos online (database_beer.py).	136

15.1.5.	Detección de objetos mediante YOLO v8 (Detector_cup.py) ..	150
15.1.6.	Comunicación con el Arduino para el control de motores (Control_servo.py)	151
15.1.7.	Visión artificial en un hilo (cam_hilo.py)	153
15.1.8.	Interfaz de usuario y Demo (Gui_beer1.py)	174
15.1.9.	Control del Arduino (Control_arduino.ino)	242
15.2.	Lista de parámetros de configuración	245
15.2.1.	Parámetros de control del sistema (AI_conf.txt)	245
15.2.2.	Parámetros de visión computacional (cv_conf.txt)	246
15.2.3.	Parámetros de configuración inicial (Param_conf.txt)	246
15.2.4.	Posiciones del brazo robótico (Poss_conf.txt)	247
15.3.	Índice de planos	249

Índice de Ilustraciones:

Ilustración 1. Vaso de plástico transparente, utilizado en el sistema.	3
Ilustración 2. Dispensador de cerveza y grifo con compensador.	7
Ilustración 3. Dispensador de vasos personalizado.	9
Ilustración 4. Pinza del robot personalizada 1	10
Ilustración 5. Pinza del robot personalizada 2	10
Ilustración 6. Puente H utilizado (L298N)	11
Ilustración 7. Dispensador de vasos, pieza fija.	16
Ilustración 8. Dispensador de vasos, parte móvil.	17
Ilustración 9. Actuador lineal utilizado.	18
Ilustración 10. Dispensador de vasos montado	19
Ilustración 11. Acoplamiento entre la parte móvil del dispensador y el actuador lineal	19
Ilustración 12. Mecanismo para el control del grifo.	20
Ilustración 13. Adaptador biela-manivela.	21
Ilustración 14. Adaptador biela-manivela, partes.	22
Ilustración 15. Articulación angular M6	23
Ilustración 16. Dimensiones de la articulación angular.	23
Ilustración 17. Montaje del dispensador de cerveza.	24
Ilustración 18. Esquema de conexiones	25
Ilustración 19. Esquema de montaje de la mesa	27
Ilustración 20. Esquema de montaje en la parte inferior de la mesa	28
Ilustración 21. Diagrama de flujo, sistema global.	29
Ilustración 22. Diagrama de flujo, interfaz gráfica.	31
Ilustración 23. Diagrama de flujo, base de datos.	33
Ilustración 24. Diagrama de flujo, visión artificial	36
Ilustración 25. Base de datos de posiciones para configurar el compensador, Interfaz de usuario.	42
Ilustración 26. Diagrama de flujo, configuración del compensador.	42
Ilustración 27. Diagrama de flujo del sistema 1	45
Ilustración 28. Diagrama de flujo del sistema 2	46
Ilustración 29. Diagrama de flujo del sistema 3	47
Ilustración 30. Información sobre el dataset de entrenamiento.	52
Ilustración 31. Redimensionamiento de las imágenes.	52
Ilustración 32. Tabla comparativa de modelos YOLO V8	53
Ilustración 33. Matriz de confusión.	54
Ilustración 34. Prueba de funcionamiento.	56
Ilustración 35. Gráficas de seguimiento de entrenamiento.	56
Ilustración 36. Pantalla de inicio	57
Ilustración 37. Pantalla de inicio, mostrando un error.	58
Ilustración 38. Error conexión robot	59
Ilustración 39. Error conexión a internet	60
Ilustración 40. Pantalla menú principal	62
Ilustración 41. Botones de ayuda y salida.	62
Ilustración 42. Pantalla menú principal con botones de configuración.	63

Ilustración 43. Pantalla test.	64
Ilustración 44. Gráfica Apertura del grifo, del compensador, duración del sistema, duración de la cerveza en función de la cerveza	66
Ilustración 45. Gráfica. Duración de llenado de cada tramo en función de la cerveza.	67
Ilustración 46. Gráfica. Distribución de la puntuación y de los errores.	68
Ilustración 47. Gráfica, distribución de la puntuación.	69
Ilustración 48. Pantalla de posiciones del brazo robótico.....	70
Ilustración 49. Pantalla parámetros de visión computacional.....	72
Ilustración 50. Pantalla parámetros de Inteligencia artificial.....	74
Ilustración 51. Pantalla parámetros de inicio.....	76
Ilustración 52. Pantalla de espera.	78
Ilustración 53. Pantalla demo.	79
Ilustración 54. Pantalla de espera mientras se sirve la cerveza.....	80
Ilustración 55. Pantalla de encuesta.	83
Ilustración 56. Pantalla Error, vaso no detectado.....	86
Ilustración 57. Pantalla error, cerveza no detectada.	87
Ilustración 58. Advertencia, barril de cerveza vacío.....	88
Ilustración 59. Pantalla error, colisión del robot.....	89
Ilustración 60. Grafico de satisfacción del usuario.	91
Ilustración 61. Encuesta de satisfacción.	92
Ilustración 62. Gráfico de corrección del sistema.....	93
Ilustración 63. Gráficos de la duración de cada sección de llenado.....	94
Ilustración 64. Bebidas servidas.....	94
Ilustración 65. Imagen del vaso dentro de la pinza.	95
Ilustración 66. Vaso detectado mediante una CNN.....	95

1. Introducción.

En los últimos años, ha habido un creciente interés en el desarrollo de sistemas automatizados para el control de sistemas industriales. En particular, se ha investigado en el uso de brazos robóticos colaborativos, así como en el uso de la visión computacional para la detección de objetos y la mejora de la eficiencia en los procesos productivos.

El objetivo de este proyecto es investigar la capacidad de un brazo robótico colaborativo guiado por un sistema de visión computacional para controlar un sistema de dispensación de bebidas.

Se ha elegido este sistema debido a su potencial uso en ferias y eventos de robótica como demostración de las capacidades del brazo robótico de una forma visualmente atractivo y llamativo, pudiendo generar un impacto significativo en la audiencia.

Además de la implementación del brazo robótico y la cámara de visión computacional, se busca optimizar el control del flujo de la bebida mediante la utilización de algoritmos para el control de procesos, que permitan mejorar la precisión y eficiencia del sistema.

Para este proyecto se ha realizado un exhaustivo estudio de la literatura relacionada con el diseño de sistemas automatizados y su implementación en diversas áreas de aplicación. Se han explorado tecnologías y técnicas de diversos campos como la robótica colaborativa, la visión computacional, las redes neuronales, los microcontroladores, los sistemas de control, la base de datos en la nube y el análisis de datos.

Para la implementación de este proyecto, se ha diseñado y construido un prototipo funcional para el control de un dispensador de bebida, el cual cuenta con un brazo robótico colaborativo guiado por un sistema de visión artificial y un microcontrolador para el control del grifo del dispensador para asegurar un flujo preciso y eficiente de la bebida servida.

Por otro lado, para el control del brazo robótico, se ha creado un sistema de control autónomo que permite establecer la posición y la trayectoria de retroceso del robot en el caso de que colisione. Asimismo, para la detección y reconocimiento de objetos, se han utilizado técnicas avanzadas de visión computacional, tales como detección de bordes, segmentación y clasificación de objetos.

Para la gestión y almacenamiento de los datos obtenidos en el proceso de dispensación de cerveza, se ha implementado una base de datos en la nube que permite un fácil acceso y análisis de los datos.

Finalmente, se han realizado pruebas y simulaciones del sistema para validar su funcionamiento.

En resumen, este proyecto responde a un esfuerzo exhaustivo por explorar y aplicar diversas tecnologías y técnicas en la creación y desarrollo de sistemas automatizados. Con el objetivo de implementar un sistema de control de dispensadores de bebida eficiente, moderno y que resulte visualmente atractivo y llamativo.

2. Objetivos.

El objetivo del presente proyecto es crear un sistema automatizado capaz de controlar un dispensador de cerveza utilizando un brazo robótico colaborativo.

El sistema debe ser capaz de dispensar la cerveza en vasos de plásticos transparentes de 0.3 litros de capacidad.



Ilustración 1. Vaso de plástico transparente, utilizado en el sistema.

El método que se deberá utilizar para la dispersión de la cerveza será el llamado “Dos tiempos” que consta de los 3 siguientes pasos:

- Descarte del primer golpe de cerveza: consiste en desperdiciar el primer tramo de cerveza que sale al abrir el grifo, dado que esta permaneció dentro del grifo absorbiendo aire, y generara un exceso de espuma en la cerveza.

- Primer tiempo: consiste el comenzar a llenar el vaso introducción el vaso a 45 grados hasta llenar una tercera parte de la capacidad del vaso, una vez superada esa capacidad se colocará el vaso en posición vertical y cerrar el grifo cuando llegue a la capacidad deseada.
- Segundo tiempo: El objetivo de esta fase es crear la corona de espuma, esta se consigue abriendo el grifo en la posición de crema de espuma.

El dispensador utilizado posee un compresor de aire incorporado de 2.8-3.2 bar que se utiliza dióxido de carbono (CO₂) para mantener la cerveza en el barril presurizado, por lo que no es necesario utilizar un tanque de CO₂, pero por contra, genera una presión fluctuante e inestable, lo que podría afectar a la salida de la cerveza y producir una espuma excesiva. Además, el uso del compresor podría generar una mayor oxidación de la cerveza, lo que podría afectar su sabor y aroma. (D E Briggs, (2004))

El dispensador de cerveza también consta de un grifo compensador, lo que permite regular el caudal en función de la presión del barril.

Además del dispensador de cerveza el sistema debe constar de un dispensador de vasos.

Debido a la complejidad de control del dispensador de cerveza es necesario integral en el brazo robótico un sistema de visión computacional.

El sistema debe de ser de fácil montaje, para facilitar el transporte de este.

3. Software utilizado.

Para la creación de este sistema se han utilizado diversos programas, que a continuación se detallan:

- SolidWorks: Este programa se ha utilizado para el diseño 3D de todas las piezas necesarias para el sistema que se imprimieron en una impresora 3D.
- Ufactory Studio: Programa utilizado para la configuración del brazo robótico, y para la obtención de las diversas posiciones necesarias para el sistema.
- Visual Studio Code: Editor de código fuente, utilizado para depurar el código de programación.
- Arduino IDE: Entorno de desarrollo integrado de Arduino utilizado para programar el microcontrolador para el control de los motores.
- Python: Lenguaje de programación utilizado para el sistema.

También se utilizaron diversas librerías de Python, que a continuación se describen:

- Time: Es una biblioteca estándar de Python que ofrece funciones para trabajar con el tiempo. Se utiliza para medir el tiempo de ejecución del programa y para crear retrasos de tiempo.
- OpenCV: Es una biblioteca de computación visual que se utiliza principalmente para el procesamiento de imágenes y vídeo.
- Numpy: Es una biblioteca de Python para la computación científica que se utiliza para realizar operaciones matemáticas y numéricas.
- Serial: Es una biblioteca para PySerial que proporciona un interfaz para trabajar con puertos serial. Se utiliza para comunicarse con dispositivos de hardware utilizando la comunicación serial, como los microcontroladores.
- Copy: Es una biblioteca de Python que ofrece funciones para crear copias de objetos. Se utiliza para crear copias de objetos complejos que contienen datos sensibles.

- Threading: Es una biblioteca de Python que se utiliza para crear aplicaciones multihilo. Se utiliza para ejecutar diferentes tareas en paralelo.
- xArm: Es una biblioteca para programar robots manipuladores con el controlador del xArm. Se utiliza para controlar robots manipuladores desde un ordenador mediante Python.
- Sys, Os, Traceback: Son librerías de Python que se utilizan para obtener información del sistema y para manejar errores de forma adecuada.
- Random: Es una biblioteca estándar de Python que se utiliza para generar números aleatorios.
- Ultralytics: Es una biblioteca para visión computacional que facilita el entrenamiento y la evaluación de modelos de detección de objetos y segmentación de imágenes.
- PyQt5: Es una librería para diseñar aplicaciones gráficas de usuario (GUI). Permite crear ventanas, botones, cuadros de texto, entre otros elementos gráficos que se utilizan para interactuar con el usuario.
- Ast: Es una biblioteca de Python que se utiliza para analizar y compilar código Python.
- Matplotlib: Es una librería de visualización de datos en 2D que se utiliza para crear gráficos y diagramas.
- PIL: Es una biblioteca de Python para realizar operaciones con imágenes. Se utiliza para abrir y guardar imágenes, crear miniaturas, entre otras funciones.

4. Elementos utilizados.

Los elementos utilizados en este sistema son:

4.1. Dispensador de cerveza.

El dispensador utilizado es un dispensador de cerveza, modelo PYGMY 25/K de la marca Lindr, este integra un compresor aire por lo que no es necesario utilizar una botella de CO₂ y es capaz de refrigerar la bebida de 2°C a 8°C mediante un termostato mecánico con 7 posiciones siendo 0 apagado. Según el fabricante cuenta con una capacidad máxima de 25 litros por hora.



Ilustración 2. Dispensador de cerveza y grifo con compensador.

Este dispensador cuenta con un grifo que puede ser operado mediante 2 accionamientos:

- Agarradera: Es la palanca que acciona el acoplador de barril de cerveza, el tubo Extractor es empujada cuando la palanca se posiciona hacia abajo y empujando el retén que tiene el barril, haciendo que salga la cerveza por la línea de bebida. (The Best Dispensing, 2023)

- **Compensador:** es una válvula utilizada para controlar la cantidad de cerveza y espuma que sale del grifo. Su objetivo es compensar la presión y la velocidad del flujo de cerveza para que la cantidad de espuma sea la adecuada en cada vaso.

Este grifo también cuenta con una posición crema de cerveza, utilizada para obtener una capa de espuma más densa y cremosa en la parte superior del vaso. Se logra mediante el uso de una boquilla especial dentro del grifo de cerveza diseñada para crear burbujas más pequeñas y una textura más cremosa en la espuma. Para accionar este modo se usa la agarradera en sentido contrario al de apertura de cerveza, en este modelo.

Para automatizar la acción sobre la agarradera se ha optado por utilizar un servo motor, que mediante un mecanismo es capaz de regular la apertura del grifo y posicionar en el modo crema.

4.2. Dispensador de vasos de plástico.

Como resultado de no encontrar en el mercado un dispensador de vasos de plásticos adecuados, se diseñó uno en 3D mediante SolidWorks, y se imprimió usando una impresora 3D de resina.

El problema se daba porque los vasos de plástico se pegaban bastante entre ellos, lo que hacía que, al aumentar la fuerza con la que se tiraba del vaso, en vez de salir uno, salían varios, y si se disminuía la fuerza, la pinza se deslizaba por el vaso.

Para solucionar este problema se diseñó un dispensador de vasos que consta de 2 piezas una fija a un soporte metálico en forma de T, mediante el cual se fija a la mesa y una pieza móvil encargada de separar los vasos e impedir que salgan más de un vaso. Esta pieza móvil es desplazada gracias a un actuador lineal de 12v.



Ilustración 3. Dispensador de vasos personalizado.

4.3. Brazo robótico colaborativo.

El brazo robótico utilizado es el modelo xArm 6 de la marca Premo. Se trata de un robot industrial diseñado para aplicaciones de manufactura. Cuenta con seis grados de libertad y un alcance máximo de 590 mm. Puede manejar cargas útiles de hasta 5 kg, y se puede programar para operar en modo autónomo o para ser controlado remotamente. xArm 6 utiliza motores sin escobillas y una variedad de sensores para garantizar un movimiento controlado y preciso, y está diseñado para ser fácil de usar y programar.

Al xArm se le ha incorporado una pinza personalizada en el extremo final.



Ilustración 4. Pinza del robot personalizada 1



Ilustración 5. Pinza del robot personalizada 2

4.4. Caja de conexiones.

Dentro de la caja de conexiones se han fijado los elementos pequeños del sistema como el Arduino Uno y el puente H.

- Arduino Uno: Está basado en el microcontrolador ATmega328P y cuenta con 14 pines digitales de entrada/salida (6 de los cuales pueden ser utilizados como salidas PWM) y 6 pines analógicos de entrada. Además, cuenta con una conexión USB-B para programarlo y alimentarlo, y un conector de alimentación externa.
- Puente H (L298N): es un módulo para el control de motores de corriente continua (DC). Es muy útil para proyectos que requieran el control de motores bidireccionales, ya que permite controlar la dirección y la velocidad del motor a través de dos pines de control. El L298N también cuenta con protección para evitar sobrecargas o sobrecalentamiento del motor.

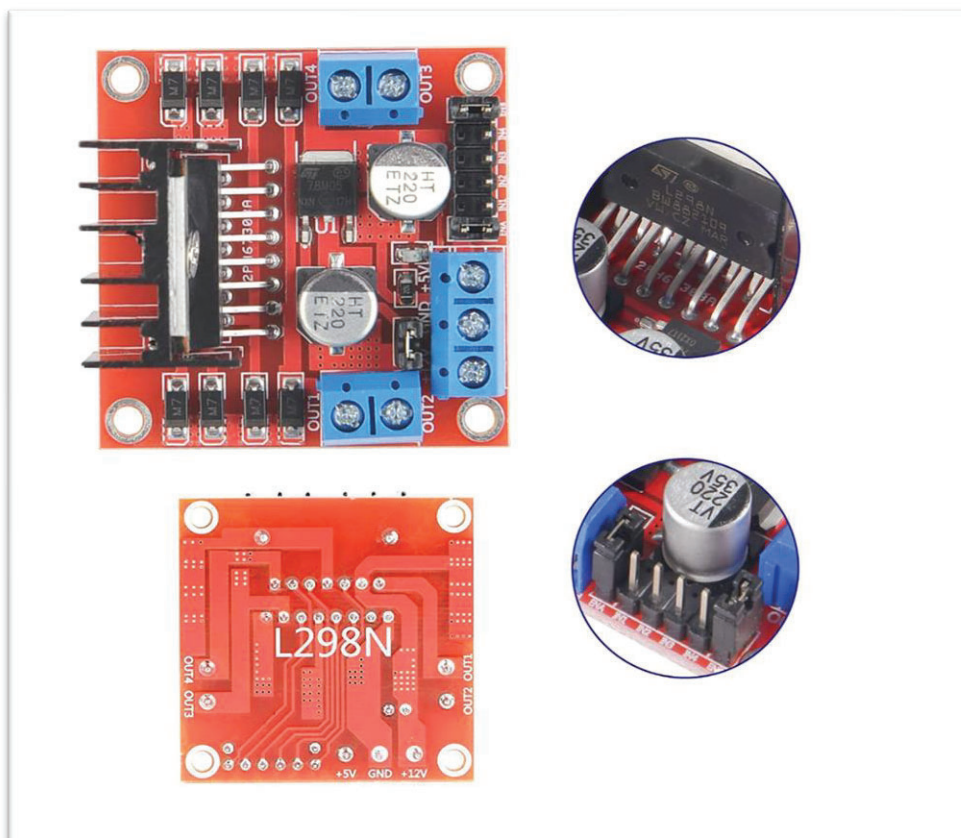


Ilustración 6. Puente H utilizado (L298N)

4.5. Ordenador táctil.

Para este proyecto se ha utilizado como cerebro del sistema una Tablet: CHUWI Hi10 X, se trata de un ordenador con pantalla táctil y sistema operativo Windows 11, 8 GB de RAM y 128GB de ROM.

El motivo por el que se escogió este PC fue por la necesidad de usar un puerto Ethernet RJ-45 para control del brazo robótico y el hecho de que sea táctil facilita la interacción del usuario con la interfaz.

Debido a que la Tablet utilizada solo cuenta con 2 puertos tipo C, y uno de ellos se utiliza para su alimentación, empleamos un “hub docking station, Tipo C” que incluye un puerto RJ45 y tres puertos USB. Este “HUB” ha permitido ampliar las capacidades de conectividad de mi sistema y poder así conectar el brazo robótico y el microcontrolador al sistema.

La Tablet actúa como el centro de control y coordinación de todas las partes involucradas, incluyendo el robot colaborativo y el Arduino que controla los motores. A través del software programado en Python, el PC se encarga de recibir datos, procesarlos y enviar comandos tanto al robot como al Arduino.

También tiene otras funciones, como el almacenamiento y análisis de datos, la comunicación con la base de datos en la nube o la interfaz de usuario para facilitar la interacción con el sistema.

5. Equipamiento Utilizado.

Durante el desarrollo de este proyecto, se requerían piezas específicas que no estaban disponibles en el mercado. Para solucionar este desafío, se optó por Diseñarlas en 3D usando SolidWorks y fabricarlas utilizar impresoras 3D y CNC.

5.1. Impresoras 3D de resina.

Las impresoras 3D de resina son dispositivos que utilizan la tecnología de estereolitografía (SLA) para crear objetos tridimensionales de alta precisión.

En lugar de utilizar el proceso de deposición de material capa por capa como en las impresoras 3D convencionales, las impresoras de resina emplean una resina líquida fotosensible que se solidifica mediante la exposición a la luz ultravioleta.

Una vez que todas las capas se han impreso y solidificado, las piezas se retiran del área de construcción y se someten a un proceso de curado para garantizar la completa polimerización de la resina. (Peck, 2023)

Tipos de resinas empleadas:

5.1.1. Resina 4K:

La resina 4K se caracteriza por su capacidad para producir objetos impresos con una superficie suave, detalles nítidos y una excelente definición. Esto la hace especialmente adecuada para aplicaciones que requieren una alta precisión y calidad de acabado.

Sin embargo, debido a su alta resolución y calidad, la resina 4K tiende a ser más costosa que otras resinas, y el tiempo de impresión puede ser más largo debido a la mayor cantidad de detalles y capas requeridas para lograr la alta resolución.

5.1.2. Resina Draft:

La resina Draft se caracteriza por tener una menor resolución y detalles menos definidos en comparación con la resina 4K. Aunque la calidad y los detalles de las impresiones pueden ser inferiores, la resina Draft es una opción popular para prototipos iniciales o piezas que no requieren una alta precisión o acabado.

Debido a su menor resolución y calidad, la resina Draft tiende a ser más asequible y el tiempo de impresión es más rápido en comparación con la resina 4K.

La resina 4K tiende a tener una mayor resistencia en comparación con la resina Draft. Esto significa que las piezas impresas con resina 4K pueden soportar mejor fuerzas o tensiones aplicadas sin sufrir deformaciones o roturas.

5.2. CNC (Centro de Mecanizado de Control Numérico)

Además de las impresoras 3D, también se empleó un sistema de CNC para fabricar piezas específicas en este proyecto. El CNC, o Centro de Mecanizado de Control Numérico, es un método de fabricación automatizada que utiliza instrucciones codificadas en un programa para controlar la operación de una máquina herramienta.

Este modelo 3D de la pieza requerida se importa a un software de programación CNC, donde se generaron las instrucciones de mecanizado necesarias. Estas instrucciones incluyeron detalles como las rutas de corte, las velocidades de avance y los tipos de herramientas a utilizar.

Una vez que se completó la programación, se carga el código en la máquina CNC y se fija el material de trabajo, aluminio, en la mesa de la máquina.

El CNC ejecuta las instrucciones de mecanizado de manera precisa y repetitiva, utilizando las herramientas adecuadas, como fresas o brocas, para dar forma a las piezas. El resultado fue la obtención de piezas de alta calidad con tolerancias ajustadas y acabados precisos. (Dassault Systèmes., 2023)

6. Diseño del dispensador de vasos.

Los vasos utilizados son vasos de plástico de 33 cl de capacidad, el principal problema de utilizar este tipo de vasos es que se adhieren entre si cuando se encuentran empaquetados.

La primera idea fue utilizar un dispensador de vasos comercial, pero debido a que utilizaba unas pretinas metálicas en forma de gancho para separar los vasos, se observaba el siguiente problema:

Al utilizar la pinza del brazo robótico para sacar de forma descendente vertical el vaso del dispensador en función de la fuerza aplicada en la pinza se sacaban uno, varios o ningún vaso. Debido a que una fuerza alta provocaba la extracción de varios vasos y una demasiado débil que se deslizase la pinza sobre el vaso, sin extraer ningún vaso. Y cuando se configuraba con una fuerza que conseguía la extracción de un vaso, no se obtuvo la repetibilidad deseada, fallando más de la mitad de las veces, ya sea extrayendo varios o ningún vaso a la vez.

Para solucionar este problema:

- Se programó el robot para realizar distintas maniobras para la extracción del vaso mediante el uso de giros, movimientos cortos, etc.
- Se rediseñó la pinza del brazo robótico para introducir una hendidura para colocar una junta tórica para aumentar la adherencia y evitar que se deslice por el vaso sin extraerlo.

Pero no se obtuvo mejoras significantes en la repetibilidad, por lo que se tomó la decisión de diseñar un dispensador de vasos personalizado.

El dispensador de vasos personalizado debe de separar los vasos entre sí, y evitar que se puedan extraer más de un vaso a la vez.

Para cumplir con las condiciones anteriores se diseñó un dispensador que consta de 2 partes, una de ella móvil que se desliza por la otra parte, para separar los vasos y permitiendo que solo se extraiga un vaso a la vez.

6.1. Base del dispensador.

La primera pieza es la base del dispensador:

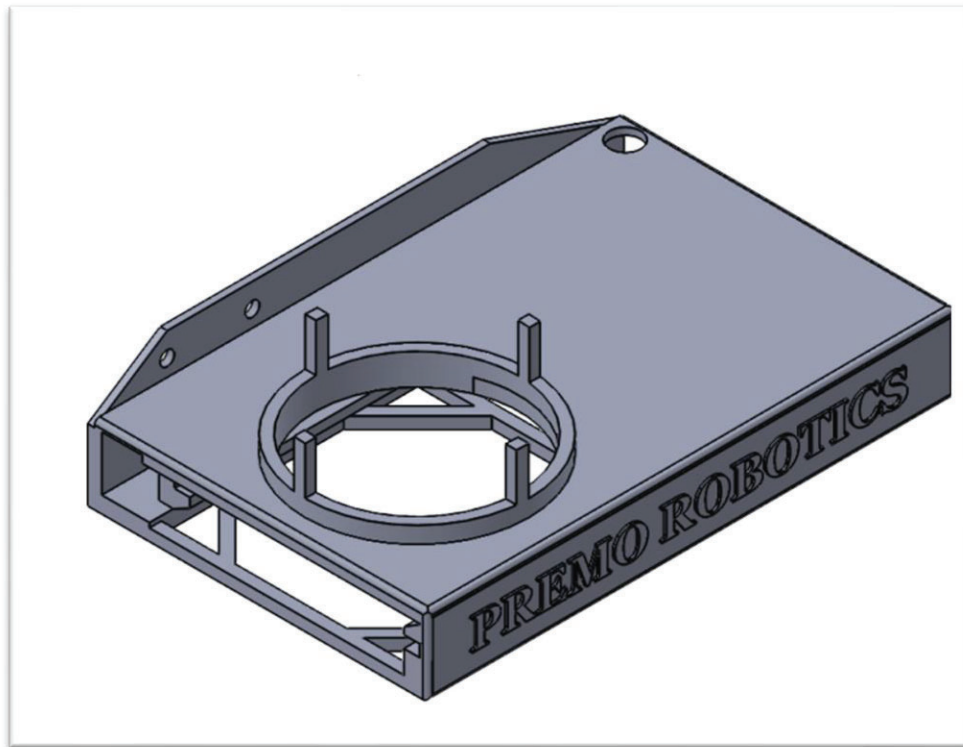


Ilustración 7. Dispensador de vasos, pieza fija.

Se utiliza para fijar el dispensador al soporte metálico en forma de T que lo ancla a la mesa.

Posee una cavidad donde se aloja un actuador lineal que mueve la otra pieza del dispensador.

También posee un hueco donde se introducen los vasos y soportes para colocar un tubo que almacenara todos los vasos.

La parte inferior de la pieza se reforzó para aumentar la resistencia de la estructura en voladizo.

6.2. Separador de vasos del dispensador.

La segunda pieza es el separador de vasos y la parte móvil:

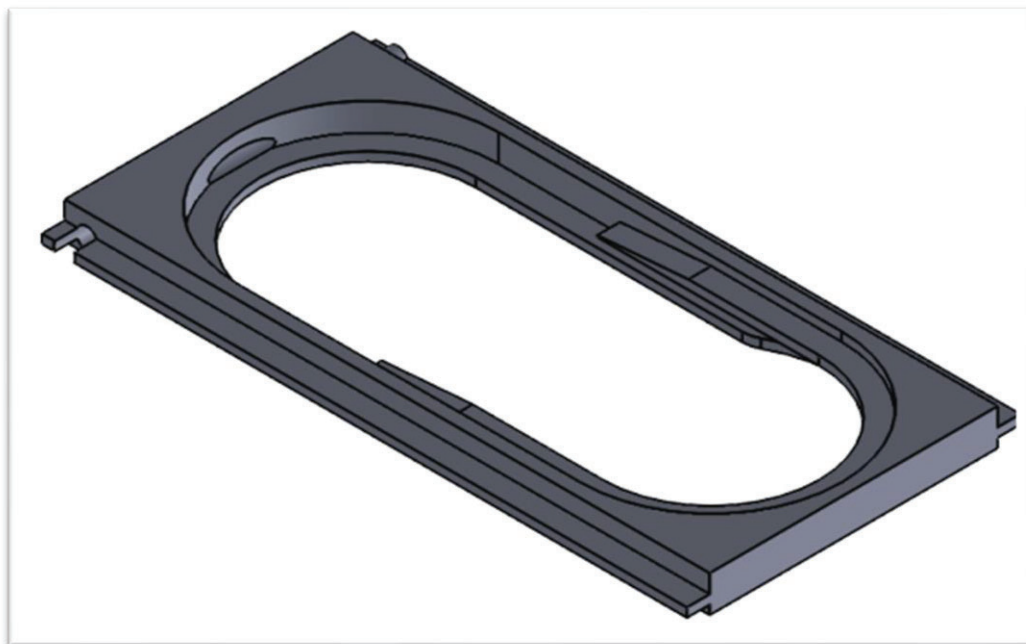


Ilustración 8. Dispensador de vasos, parte móvil.

Esta pieza está diseñada para deslizarse por el hueco dentro de la base del dispensador, y en su parte inferior cuenta con 2 secciones:

La sección superior es la encargada de separar los vasos mediante una cuña de 30 mm de largo y 3 mm de alto, dejando un hueco inferior de 3 mm donde solo cabe un vaso, impidiendo así que se puedan extraer más de un vaso a la vez.

La sección superior es la encargada de dejar pasar el vaso una vez se encuentre separado de los demás.

6.3. Actuador lineal.

Para mover la parte fija se optó por un actuador lineal de 140 mm de alcance lo que condicione el tamaño del dispensador.

Este actuador lineal es fijado mediante un tornillo y tuerca M3 a la base del dispensador por su parte fija y más grande. Y por su parte móvil es fijado al separador de vasos mediante un tornillo y 3 tuercas M3 para evitar holguras.

El movimiento del motor se gobierna mediante un microcontrolador Arduino uno y un puente H. El puente H es un circuito que permite controlar la dirección y velocidad de un motor DC. De este modo el sistema puede decidir cuándo mover el separador de vasos para dejar extraer el vaso.

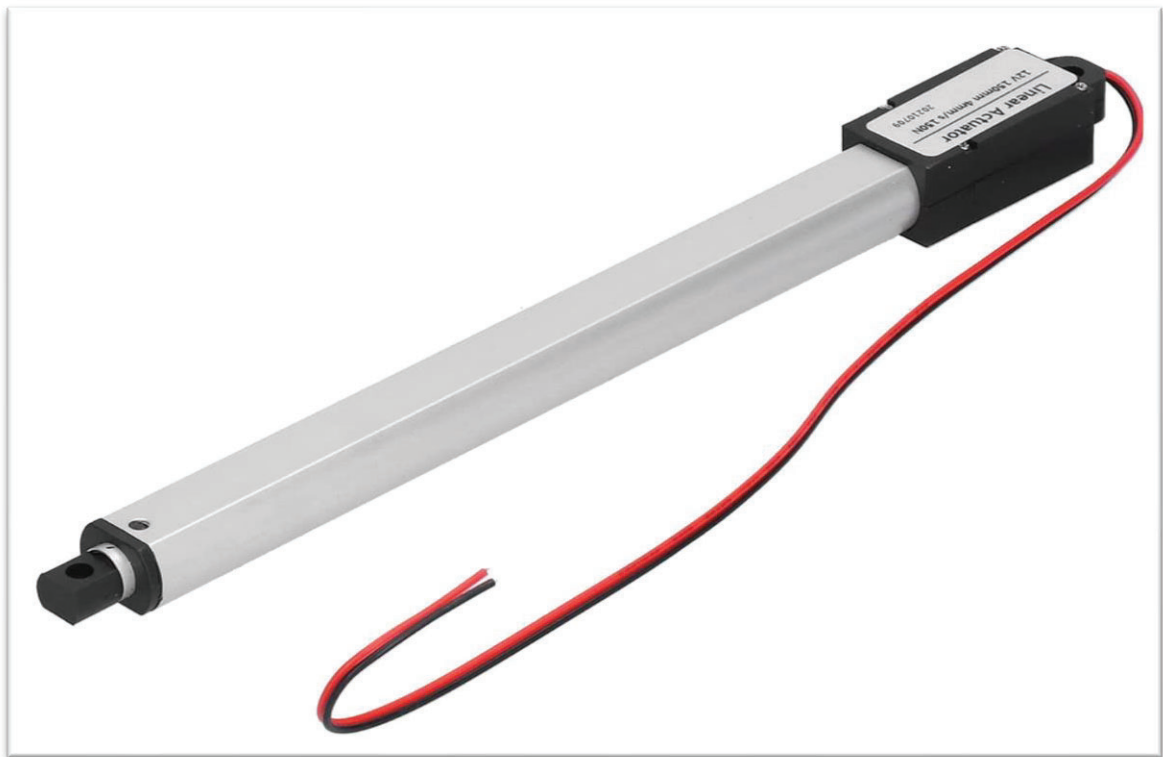


Ilustración 9. Actuador lineal utilizado.



Ilustración 10. Dispensador de vasos montado.

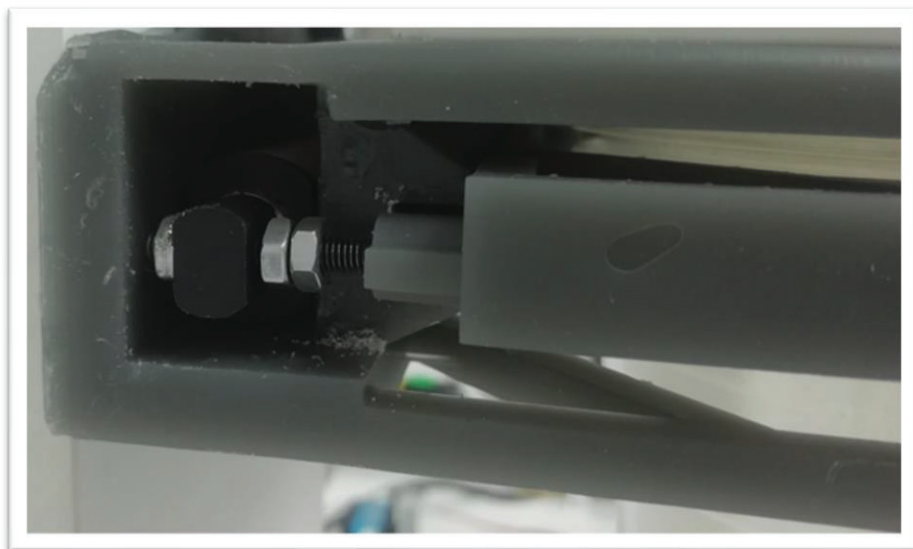


Ilustración 11. Acoplamiento entre la parte móvil del dispensador y el actuador lineal.

7. Diseño del mecanismo para la actuación sobre grifo.

Para el diseño del mecanismo para controlar la agarradera del grifo, se empleó un mecanismo de biela-manivela movido por un servo motor de 35 kg de par máximo.

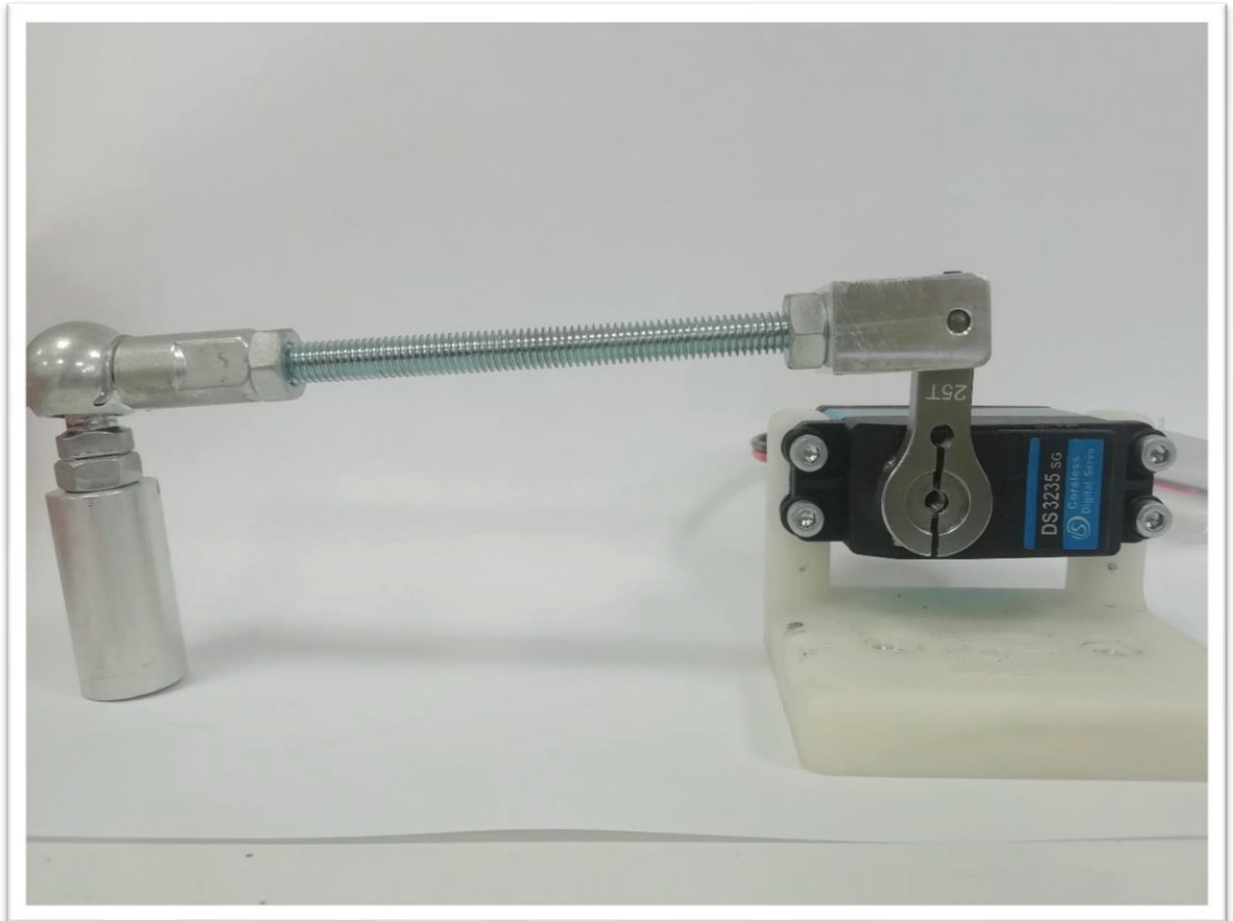


Ilustración 12. Mecanismo para el control del grifo.

El mecanismo consta de los siguientes componentes:

- 1- Servo motor: Es un motor que incluye un controlador integrado y se utiliza para controlar el movimiento angular preciso. En este caso, el servo motor se utilizará como la fuente de energía para impulsar el movimiento del mecanismo.
- 2- Soporte del servomotor, diseñado en 3D e impreso en 3D mediante una impresora 3D, el material utilizado es “4K”.

- 3- Biela: Es una barra rígida que conecta el servo motor con la siguiente parte del mecanismo. La biela transforma el movimiento de rotación del servo motor en un movimiento lineal o de desplazamiento.
- 4- Adaptador para la conexión de la biela y la manivela:

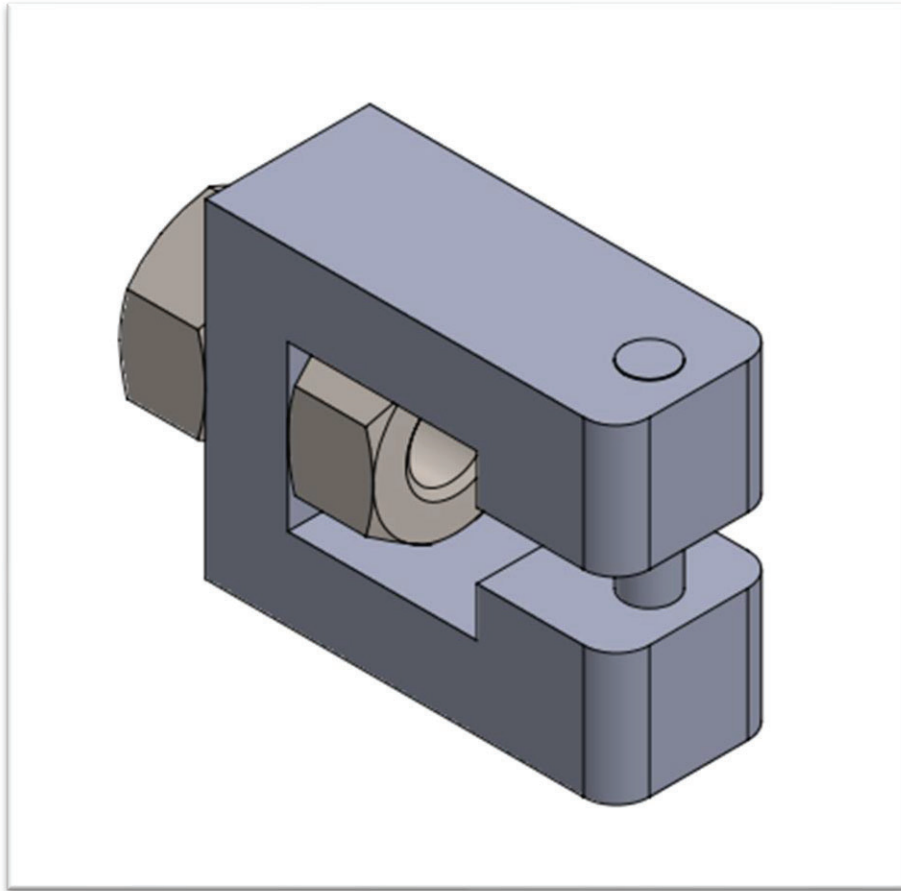


Ilustración 13. Adaptador biela-manivela.

Debido a que la biela del servo motor y la manivela siendo una varilla roscada de 6 mm de diámetro no eran compatibles, se diseñó un adaptador que pudiera unir las dos partes en una articulación.

La pieza fue diseñada en 3D y fabricada mediante una CNC (Centro de Mecanizado de Control Numérico) en aluminio.

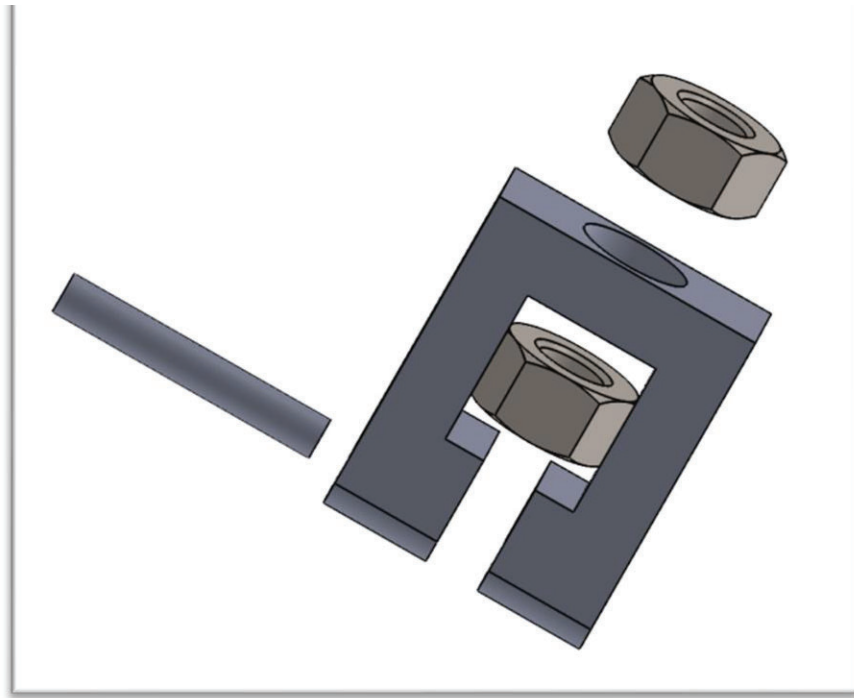


Ilustración 14. Adaptador biela-manivela, partes.

El adaptador consta de las siguientes partes:

- Pieza base, adaptador, diseñado en 3D y fabricado mediante una CNC.
- Bulón es pieza cilíndrica que se utiliza como pasador o perno para unir los dos elementos, la biela y el adaptador.
- Y también consta de 2 tuercas M6 para fijar la barrilla roscada al adaptador.

- 5- Manivela: Es una varilla roscada de 6 mm de diámetro que se utiliza como un elemento de conexión entre la biela y la siguiente parte del mecanismo. La manivela tiene una rosca externa que se enrosca en una tuerca.
- 6- Articulación angular o tornillo de rótula: Es una articulación que permite un movimiento angular y de inclinación. En este caso, se utiliza como la conexión entre la manivela y la agarradera del grifo. La articulación angular o tornillo de rótula proporciona flexibilidad y capacidad de ajuste en la conexión, permitiendo que los componentes conectados puedan moverse en diferentes direcciones y ángulos.



Ilustración 15. Articulación angular M6

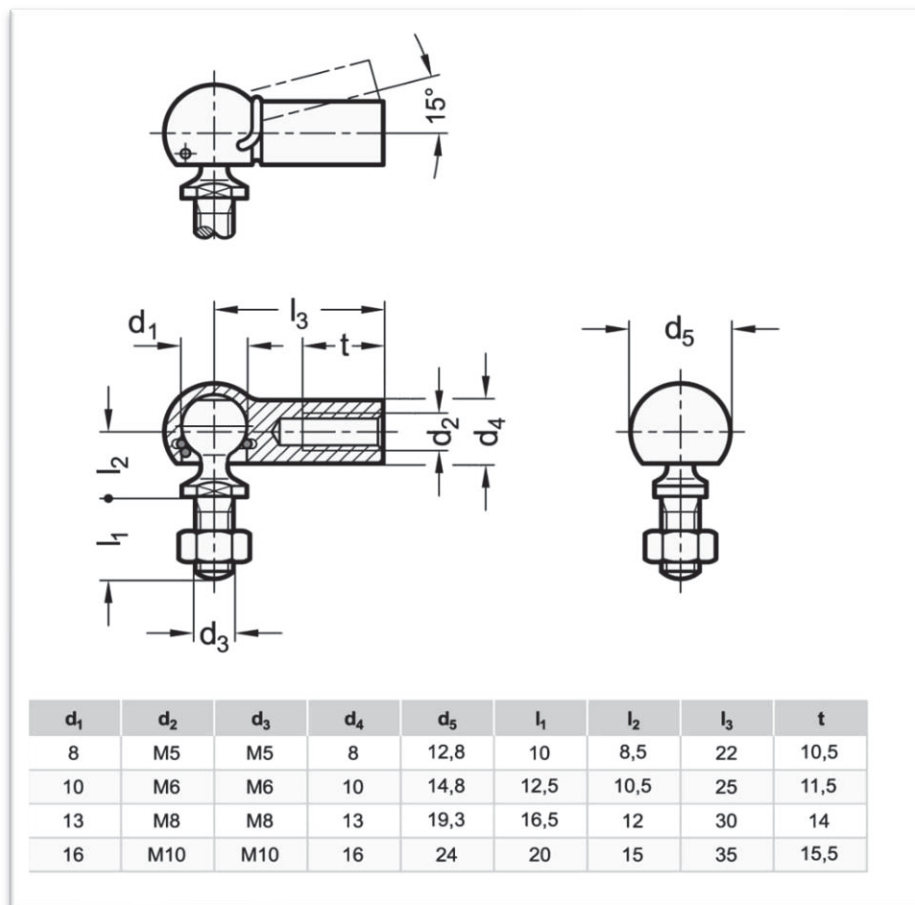


Ilustración 16. Dimensiones de la articulación angular.

- 7- Para unir la articulación angular (tornillo M6) y la agarradera del grifo (tornillo M8) se utiliza un adaptador hembra M6 - hembra M8.

8. Esquema de montaje.

Para el montaje del dispensador de cerveza, se puede seguir las instrucciones del fabricante, que se muestra el esquema a continuación:

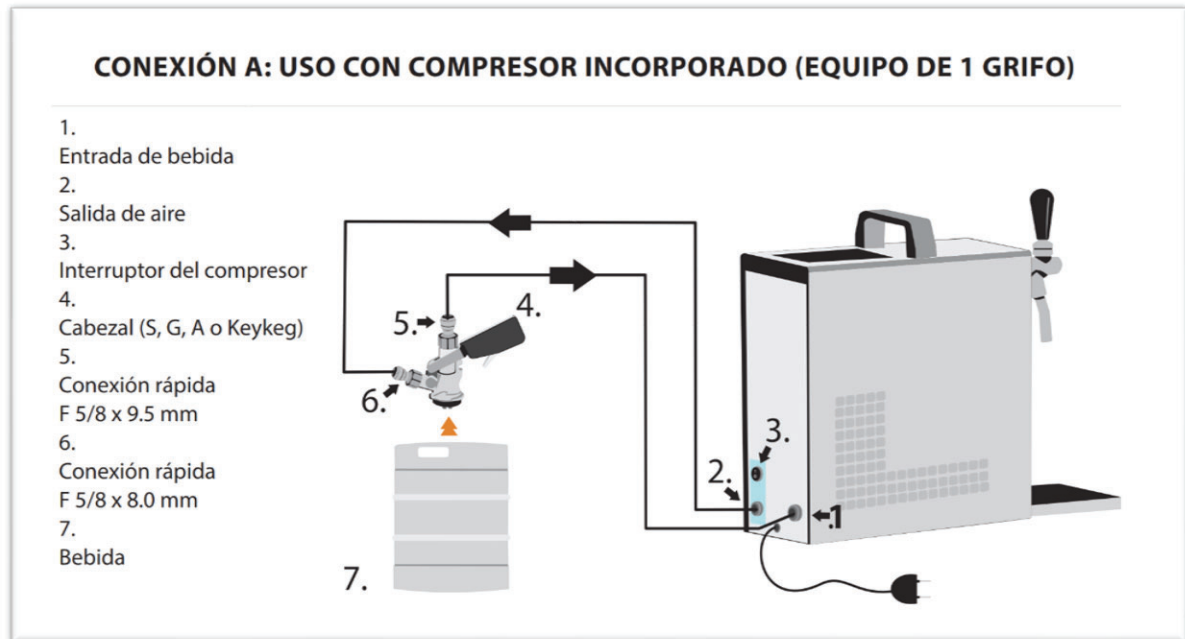


Ilustración 17. Montaje del dispensador de cerveza.

Todos los conductos deben pasar por el pasacables situado en la mesa, en la parte trasera del dispensador de cerveza.

Además, también se debe conectar las conexiones del servo motor que controla el grifo. Este consiste en un cable con 3 pines macho: Negro, Rojo y Blanco que se conectan a los 3 pines hembra de mismos colores del servo motor. Este cable pasa por el pasacables hasta llegar a la caja de conexiones anclada debajo de la mesa.

El actuador lineal del dispensador de vasos de plásticos dispone de 2 cables, Negro y Rojo que pasan por el mismo pasacables hasta llegar a la caja de conexiones.

A continuación, se muestra el esquema de la caja de conexiones:

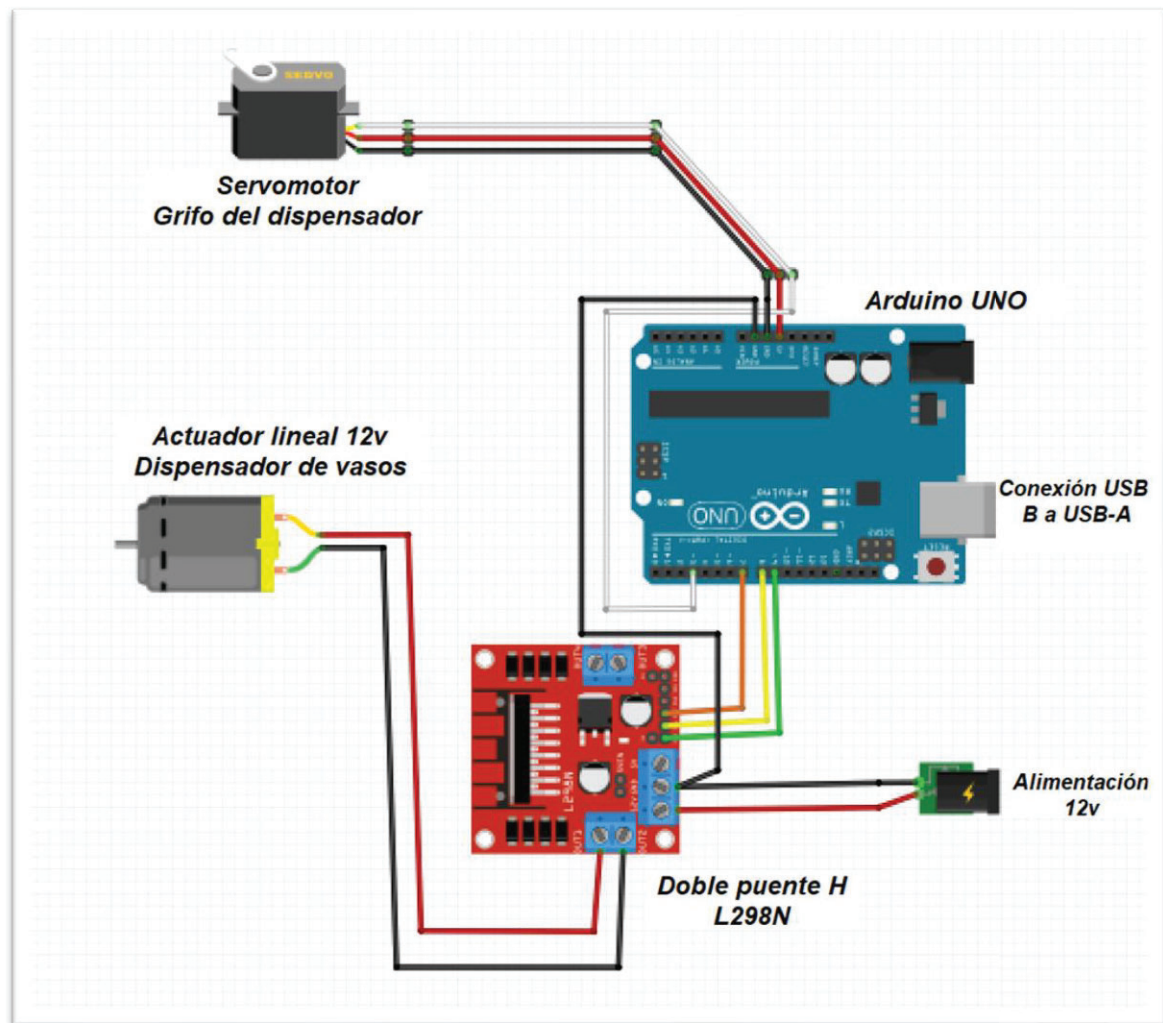


Ilustración 18. Esquema de conexiones.

En el esquema anterior se muestra el esquema de conexiones de la caja de conexiones. Dentro de la caja de conexiones están fijados un Arduino UNO y un doble puente H (L198N).

Conexiones:

- Fuente de alimentación 12 voltios al conector DC 5.5 mm situado dentro de la caja de conexiones.
 - Cable Rojo → Pin 12v (Puente H)
 - Cable negro → Pin GND (Puente H) y Pin GND (Arduino)
- Cable de conexión Arduino – PC (USB-B – USB-A).
- Conexión Servomotor al Arduino:

- Pin Rojo → Pin 5v (Arduino)
 - Pin Negro → Pin GND (Arduino)
 - Pin Blanco → Pin Digital 3 (Arduino)
- Conexión motor lineal a Puente H:
 - Cable Rojo → OUT 1
 - Cable Negro → OUT 2
- Conexión Puente H con el Arduino:
 - Pin ENA → Pin 9 (Arduino)
 - Pin IN1 → Pin 8 (Arduino)
 - Pin IN2 → Pin 7 (Arduino)

A continuación, se muestra el esquema de montaje del sistema en la mesa:

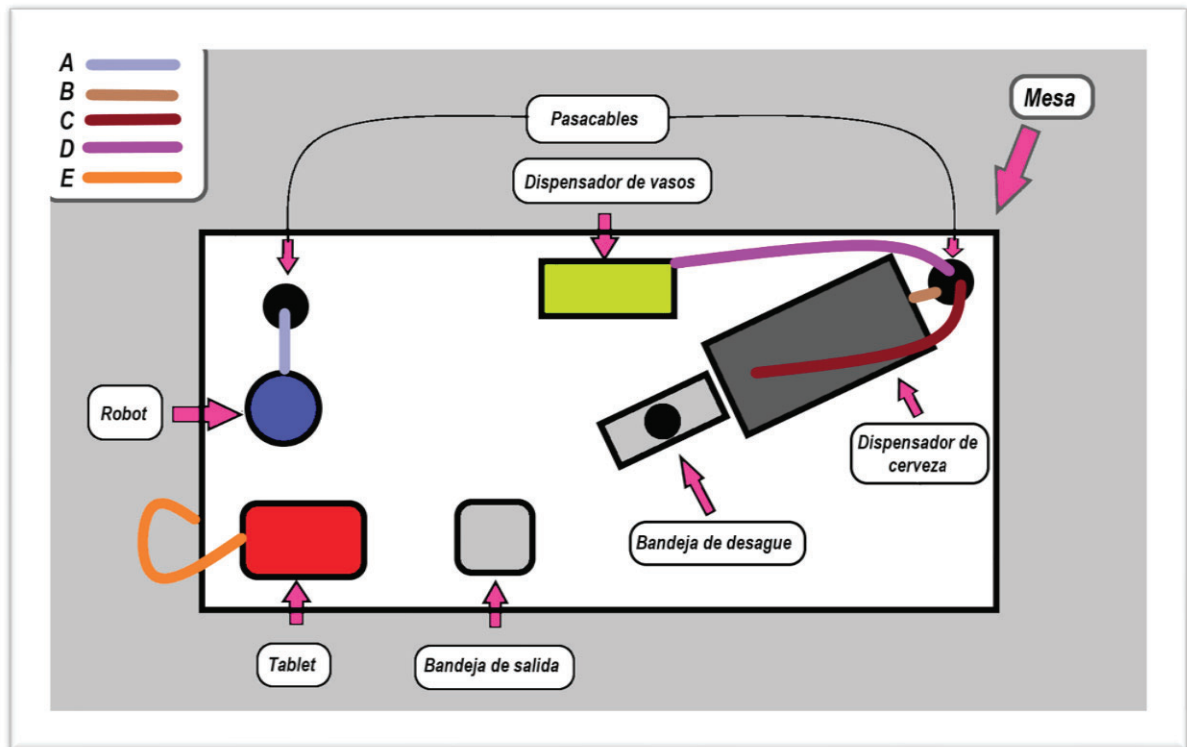


Ilustración 19. Esquema de montaje de la mesa.

En el anterior esquema de montaje se muestran las posiciones de los elementos del sistema dentro de la mesa y la colocación de los distintos cables de conexión:

- A → Cable de conexión del Robot con el controlador de este.
- B → Enchufe de alimentación del dispensador de cerveza
- C → 2 cables (rojo y negro) de conexión del Actuador lineal del dispensador de vasos a la caja de conexiones (OUT1 y OUT2) del L298N.
- D → Cable con 3 pines, de conexión del servomotor con el Arduino UNO.
- E → 2 Conexiones tipo C, que se conectarán a la Tablet, el puerto superior está destinado al cargador y el inferior al Hub USB y Ethernet.

A continuación, se muestra la colocación de los elementos en la parte posterior de la mesa:

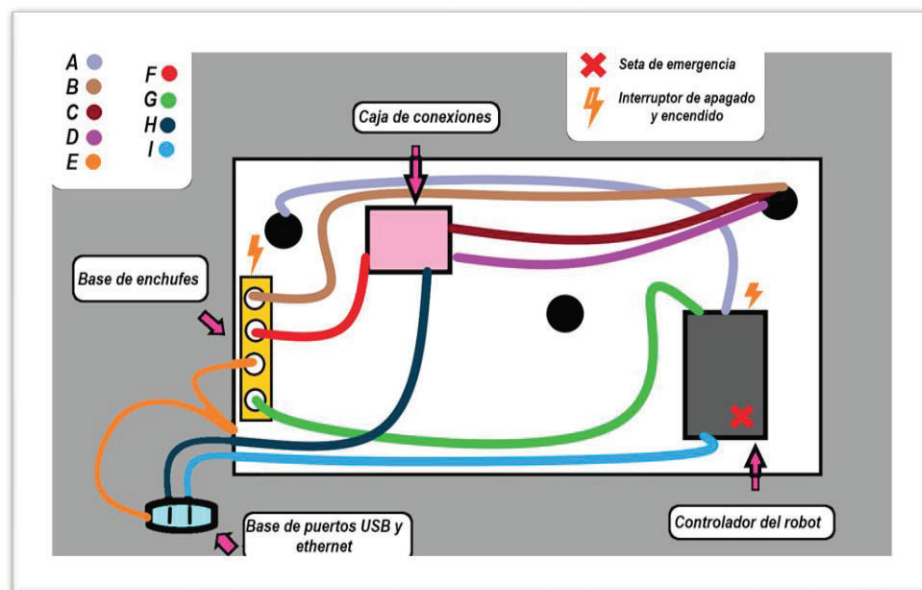


Ilustración 20. Esquema de montaje en la parte inferior de la mesa.

En el anterior esquema de montaje se muestran las posiciones de los elementos del sistema debajo de la mesa y la colocación de los distintos cables de conexión:

- A → Cable de conexión del Robot con el controlador de este.
- B → Enchufe de alimentación del dispensador de cerveza
- C → 2 cables (rojo y negro) de conexión del Actuador lineal del dispensador de vasos a la caja de conexiones (OUT1 y OUT2) del L298N.
- D → Cable con 3 pines, de conexión del servomotor con el Arduino UNO.
- E → 2 Conexiones tipo C, que se conectarán a la Tablet, el puerto superior está destinado al cargador y el inferior al Hub USB y Ethernet.
- F → Enchufe de la alimentación al conector DC 5.5 mm
- G → Enchufe de alimentación del controlador del Robot.
- H → Cable de conexión Arduino → HUB (USB-B – USB-A).

I → Cable Ethernet de conexión del controlador del Robot → HUB (RJ45 – RJ45).

9. Código de programación.

A continuación, se muestra un diagrama de bloque de la programación del sistema.

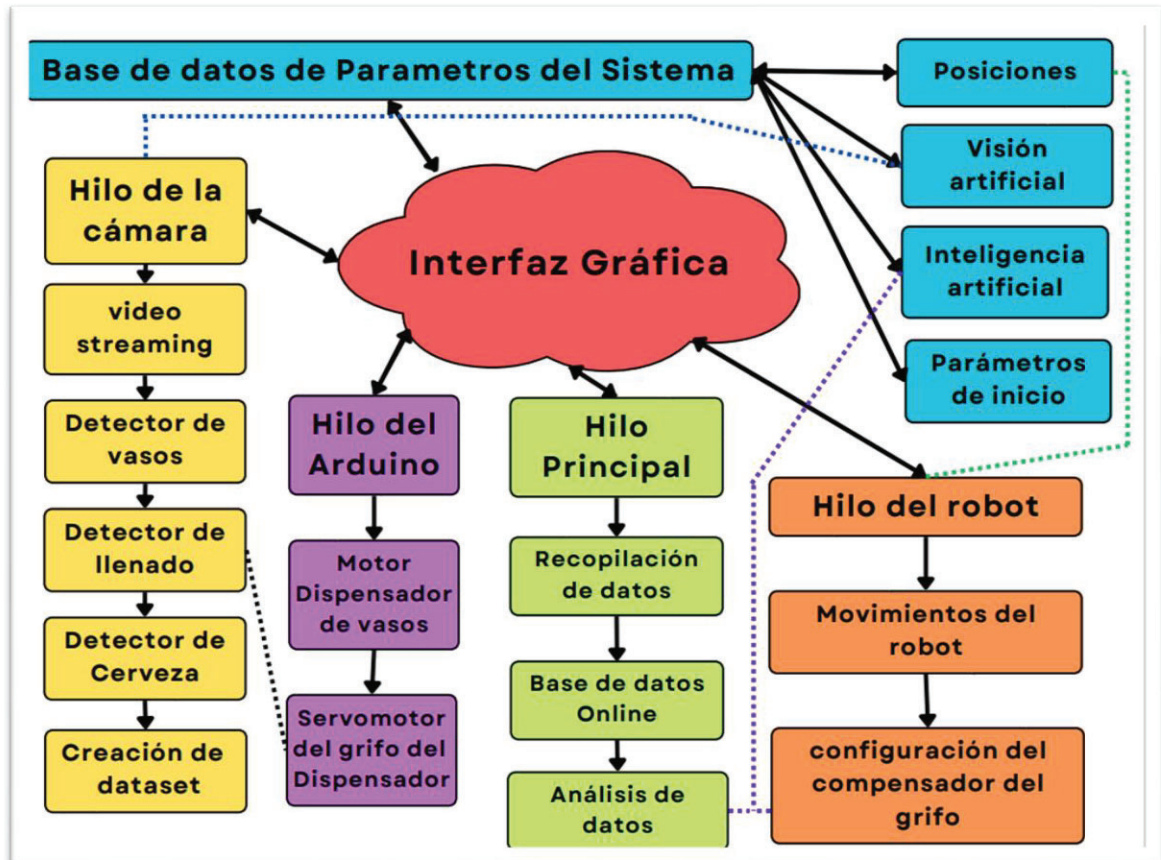


Ilustración 21. Diagrama de flujo, sistema global.

Se optó por utilizar varios hilos para no perder información mientras se ejecutan movimientos del robot, ya que, durante el movimiento, la visión artificial se paralizaba, lo que ocasionaba la pérdida de datos importantes. La implementación de hilos permitió que tanto la visión artificial como los movimientos del robot se ejecutaran simultáneamente, lo que aseguró que la información recopilada por la visión artificial no se perdiera durante el movimiento del robot. De esta manera, se logró una mayor eficiencia en la ejecución del sistema, evitando la interrupción de tareas importantes y optimizando el uso de los recursos disponibles.

La programación mediante hilos en Python permite que varias partes del programa se ejecuten de forma simultánea, lo que puede mejorar significativamente el rendimiento de una aplicación. Una ventaja importante de los hilos en Python es que son relativamente ligeros y fáciles de crear y controlar. Además, los hilos pueden compartir recursos, como la memoria del programa, lo que puede resultar en una mejor eficiencia y reducción de costos. Sin embargo, los hilos también pueden ser fuente de errores si no se gestionan adecuadamente, especialmente cuando se manejan recursos compartidos.

El sistema utiliza tres hilos adicionales además del hilo principal. Uno de ellos se encarga de ejecutar todas las tareas de visión artificial, otro se dedica a realizar todas las tareas relacionadas con el brazo robótico, y un tercer hilo se encarga del control de los motores a través del puerto serie. De esta manera, se logra una mayor eficiencia en la ejecución del sistema al permitir que cada hilo se encargue de tareas específicas y que se ejecuten simultáneamente, asegurando una tarea constante en cada uno de los componentes del sistema y evitando la interrupción de tareas críticas.

Algunas de las partes del sistema son:

9.1. Interfaz gráfica.

El sistema dispone de una interfaz gráfica programada usando la librería PyQt5, Este es el nexo de unión entre las distintas partes del sistema.

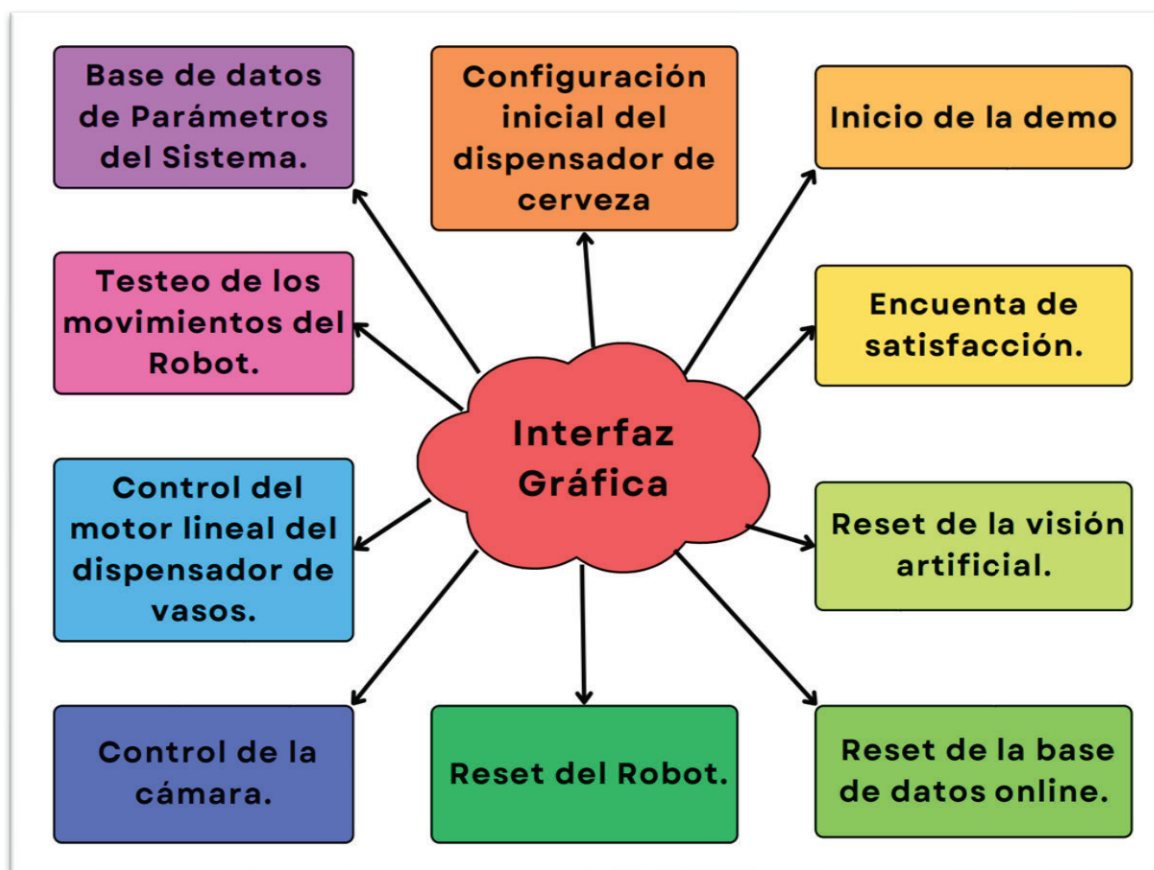


Ilustración 22. Diagrama de flujo, interfaz gráfica.

La interfaz gráfica consta con distintas pantallas que se detallan a profundidad en el capítulo del manual de usuario.

Algunas de las funciones de la interfaz son:

- Tablas de datos de distintos tipos de parámetros para poder modificar parámetros y movimientos de forma fácil y sin salir del programa. Esto proporciona gran versatilidad al programa, dado que todos los parámetros pueden ser modificados fácilmente.
- La interfaz cuenta con una ventana de testeo, en la cual se puede depurar los movimientos del robot paso a paso, además también permite abrir y cerrar el dispensador de vasos mediante un botón.

- Para el testeo del control del motor del grifo cuenta con 2 botones, uno para cerrar el grifo, en caso de emergencia, y otro que abre el grifo cuando se presiona y se cierra en el momento en el que se deja de presionar.
- Funciones para el control de la cámara.
- También cuenta con diversos botones para reiniciar las distintas partes del sistema por si fuera necesario.
- Y por último cuenta con una ventana de encuesta, donde el usuario valor su experiencia.

Entre las funciones de la interfaz se encuentran:

- Tablas de datos de distintos tipos de parámetros para modificar parámetros y movimientos fácilmente sin salir del programa, lo que proporciona gran versatilidad al programa.
- Ventana de testeo para depurar los movimientos del robot paso a paso
- Un botón que permite abrir y cerrar el dispensador de vasos.
- Dos botones para testear el control del motor del grifo, uno para cerrar el grifo en caso de emergencia y otro para abrir y cerrar el grifo mientras se presiona.
- Funciones para el control de la cámara.
- Diversos botones para reiniciar las distintas partes del sistema por si fuera necesario.
- Ventana de encuesta para que el usuario pueda valorar su experiencia.

9.2. Base de datos

El sistema cuenta con dos bases de datos, una de parámetros para configurar el sistema y una segunda base de datos en la nube para almacenar información obtenida del sistema en cada proceso de servir una cerveza, en el que el identificador único es el número de cerveza.

La base de datos utilizada es MongoDB que es una base de datos NoSQL que se puede ejecutar en local o en la nube, utilizando MongoDB Atlas.

Entre las ventajas de MongoDB se encuentran la alta escalabilidad, la capacidad para manejar datos no estructurados, la flexibilidad en el esquema, el soporte para replicación y distribución de datos, y la facilidad para integrarse con tecnologías de código abierto.

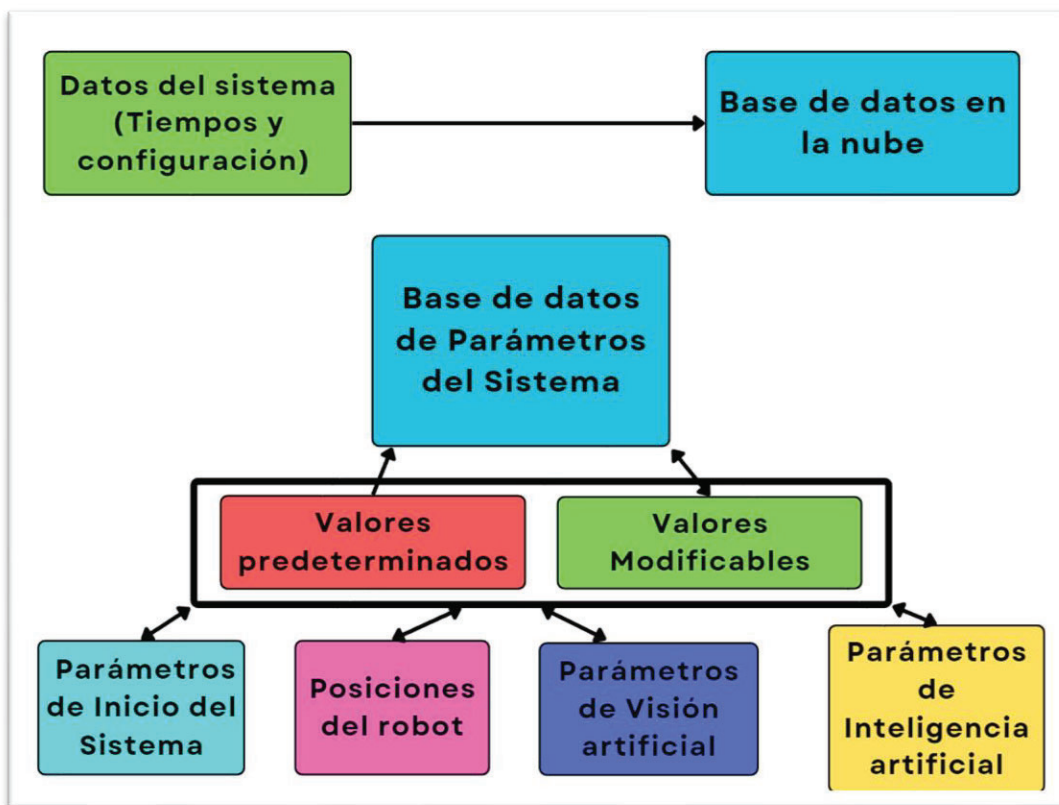


Ilustración 23. Diagrama de flujo, base de datos.

La base de datos local del sistema que almacena los parámetros de configuración, está duplicada, con el fin de conservar la configuración inicial del sistema.

9.3. Visión Artificial.

Una vez iniciado el sistema de servir una cerveza si inician los 3 hilos en paralelo.

El hilo encargado de la visión artificial es el encargado de leer en todo momento la imagen de la cámara y obtener información de ella.

El sistema está configurado para realizar capturas de la imagen en momentos claves y almacenar dichas imágenes en carpetas distintas en función del momento escogido, esto permite recopilar imágenes durante el sistema y utilizarlas posteriormente para entrenar redes neuronales con dichas imágenes. A estas carpetas llenas de fotos de un momento específico dentro del ciclo, se les llama Dataset.

Gracias a estos dataset se han creado 2 redes neuronales uno encargado de detectar la presencia de vaso dentro de la pinza, antes de abrir el grifo. Y otro que analiza los distintos volúmenes de cerveza y espuma dentro del vaso al finalizar el ciclo.

Para la detección de llenado se utiliza visión artificial convencional, mediante umbralización debido a que es necesario una reacción rápida del sistema al detectar el llenado para que no se desborde, y utilizando una red neuronal se produce un retraso en la detección de llenado.

Para la detección de llenado es necesario haber introducido anteriormente la región de interés para cada límite de llenado.

El sistema funciona de la siguiente forma:

- 1- Se toma una foto del límite de llenado del vaso vacío y se guarda en una carpeta determinada con el nombre del identificador de la cerveza.
- 2- Una vez abierto el grifo, se compara la imagen actual de la cámara con la imagen de referencia del vaso vacío, en caso de no detectar diferencias, significa que aún no ha llegado al límite y continua el bucle de búsqueda hasta detectar diferencia.
- 3- Una vez detectada diferencia entre la imagen del límite de referencia y el actuar, finaliza el bucle, se guarda dicha imagen en otra carpeta con el

nombre del identificador de la cerveza, y se obtiene el tiempo que ha tardado en alcanzar el límite.

La detección de diferencia entre las 2 imágenes depende de 2 parámetros configurables en el sistema:

- “Threshold” que es la diferencia en escala de grises de tonalidad necesaria entre una imagen de una ROI (región de interés) vacía de referencia y otra imagen de esta ROI para que se considere que se ha producido un cambio y por tanto que se ha alcanzado el límite marcado por el ROI
- y el “Theshold 2” es la cantidad de píxeles que debe cambiar para que se considere que se ha alcanzado el límite, este valor depende el tamaño de la región de interés, siempre debe ser mayor de 0 y nunca superar el número de elementos de la matriz ROI.

El sistema será más sensible para detectar que se ha alcanzado el límite marcado por la ROI, cuanto más pequeños sean los valores “Theshold” y “Theshold 2”.

El valor de “Theshold” dependerá de la diferencia de tonalidades en escala de grises entre el fondo del vaso transparente vacío y el vaso con el líquido a detectar.

El valor de “Theshold 2” depende de ruido de la imagen, dado que se utiliza para evitar falsas detecciones, por ejemplo, los producidos por reflejos.

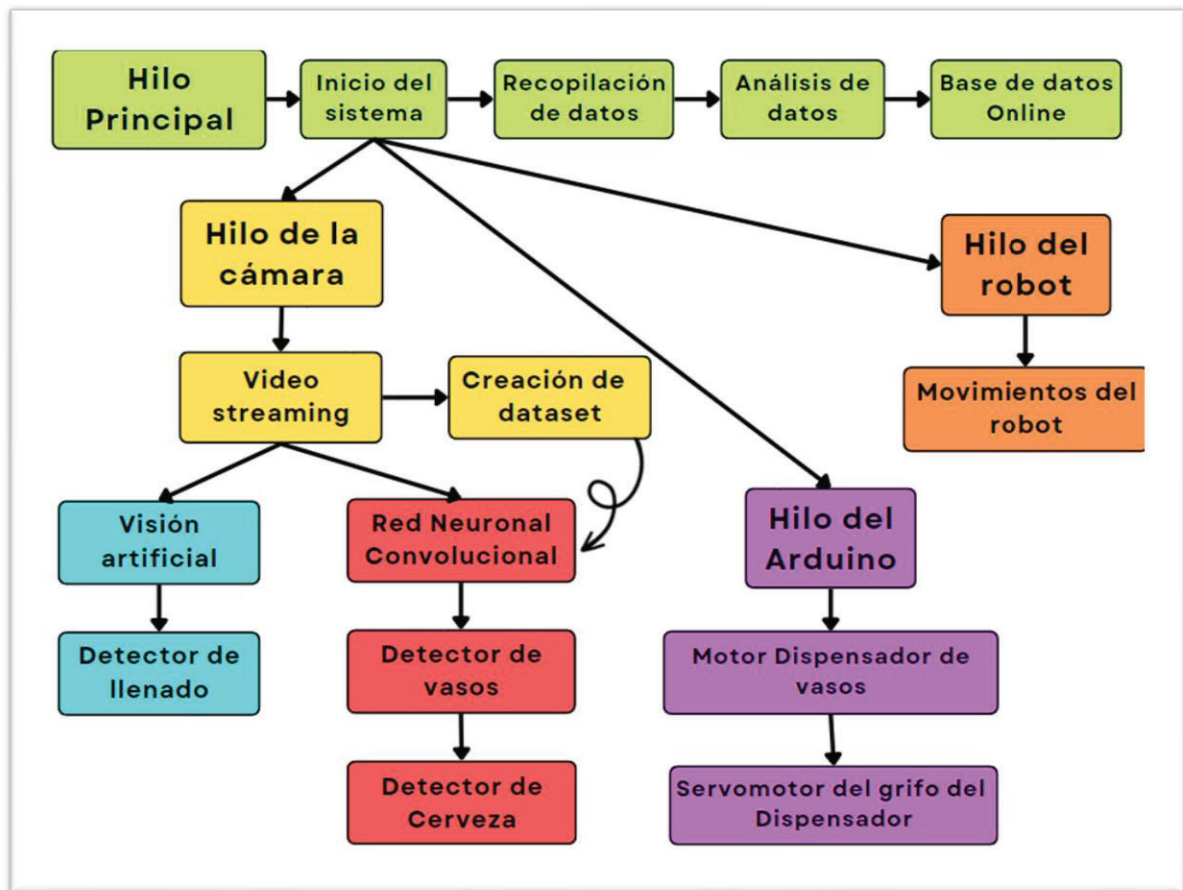


Ilustración 24. Diagrama de flujo, visión artificial.

9.4. Código del brazo robótico.

Para programar el robot se utiliza un hilo exclusivo para ello debido a las posibles pérdidas de información que se producen cuando el programa espera a que el robot finalice el movimiento.

Todas las posiciones del brazo robótico se encuentran guardadas en una base de datos y antes de producirse el movimiento lee dicho movimiento de la base de datos, pudiendo ser fácilmente modificable dichas posiciones.

Los movimientos utilizados en este sistema son los movimientos cartesianos, debido a que son más fáciles de modificar mediante la interfaz.

El código también dispone de un sistema de control autónomo que permite establecer la posición y la trayectoria de retroceso del robot en el caso de que colisione. Esto se consigue de la siguiente forma:

- 1- Detector de colisiones: cada vez que se ejecuta un movimiento se analiza si se ha producido un error durante su ejecución.
- 2- Si se produce un error todo el sistema se detiene y se cierra de emergencia el grifo.
- 3- Se obtiene la posición en el que se encuentra el brazo robótico.
- 4- Y en función de dicha posición, dependiendo si el robot lleva ya cerveza en el vaso o no, se produce el rearme el robot y el movimiento hacia la posición cero del sistema, si el robot posee un vaso con cerveza, antes de ir a dicha posición deja el vaso en un lugar seguro.

El brazo robótico también controla el grado de apertura del compensador del grifo de cerveza, esto lo hace teniendo guardados las posiciones de la palanca cerrada y abierta completamente, y mediante interpolación se obtiene la posición deseada en función del grado de apertura deseada en porcentaje.

9.5. Código Arduino.

Para el control del grifo de cerveza se utiliza un servomotor y para el control del dispensador de vasos se utiliza un motor lineal controlado comuna señal PWM.

El código de configuración de estos motores se realiza en Arduino IDE y se guarda dentro del Arduino, y mediante el puerto serie se controla las distintas funciones a utilizar para activar los motores y dirigirse a la posición deseada.

Este código también se ejecuta en un hilo específico para garantizar el cierre de emergencia del grifo en caso de error en el robot.

El código permite controlar el servo motor y el actuador lineal utilizando comandos recibidos a través del puerto serial. Dependiendo del valor recibido, se realizarán diferentes acciones en los dispositivos.

El código utiliza la biblioteca "Servo" para controlar un servo motor y un puente H para controlar un actuador lineal. A continuación, se detalla el funcionamiento de cada parte del código:

1- Importar la biblioteca y declarar variables:

- Se importa la biblioteca "Servo.h" que permite controlar los servos motores.
- Se declara una instancia de la clase "Servo" llamada "servo1" para controlar el servo motor.

2- Declarar pines:

- Se declaran las variables para los pines del puente H que se utilizarán para controlar el actuador lineal.
- "EnaPin" es el pin de habilitación del puente H (ENA).
- "In1Pin" y "in2Pin" son los pines de control de dirección del puente H (IN1 e IN2).

3- Configuración inicial (función setup):

- Se establecen los pines del puente H como salidas.
- Se conecta el servo motor al pin 3.
- Se inicia la comunicación serial a una velocidad de 9600 baudios.
- Se mueve el servo motor a una posición inicial de 20 grados (Grifo cerrado).
- Se configuran los pines del puente H para cerrar el actuador lineal durante 5 segundos.

4- Bucle principal (función loop):

- Se verifica si hay datos disponibles en el puerto serial.
- Se lee un valor entero desde el puerto serial y se guarda en la variable "angle".
- Si el valor de "angle" está entre 0 y 70, se mueve el servo motor a ese ángulo.
- Si el valor de "angle" es 800, se configuran los pines del puente H para que el actuador lineal se cierre durante 5 segundos y luego se detenga.
- Si el valor de "angle" es 900, se configuran los pines del puente H para que el actuador lineal se abra durante 4 segundos y luego se detenga.

9.6. Código para servir una cerveza.

Una vez que se presione el botón de inicio de la demo, comienza el proceso de servir una cerveza, constando de los siguientes pasos:

- 1- El primer paso es la obtención de los parámetros de entrada del sistema de la base de datos actualizada. Los parámetros de entrada son el grado de apertura del grifo de cerveza [%] y de la apertura del compensador [%].
- 2- Una vez obtenidos los parámetros de entrada expresados en porcentaje se procede a transformarlos en unidades que puedan ser utilizados por el microcontrolador.

- a. La apertura del grifo de cerveza se configura mediante un servo motor controlado por un Arduino. Para mover el dicho servomotor se debe enviar la posición en grado (de 0 a 270) que se desea mover.
Según la configuración del mecanismo para configurar el grifo se usan estos rangos:

- Posición de grifo cerrado: 20 grados
- Apertura de la cerveza: rango utilizado (20, 70] grados, siendo 20°, grifo cerrado y 70 grados, abierto por completo.
- La posición de crema de cerveza: 0 grados.

Se emplea una función para obtener el grado de servomotor a partir del porcentaje de entrada.

- b. La configuración de la válvula del compensador se realiza mediante el brazo robótico, dado que no se conoce la posición en la que se encuentra la válvula, dado que puede ser manipulada por el público, es necesario primero cerrar la válvula (subiéndola desde abajo) mediante la pinza y una vez situada en la posición cero bajar dicha válvula en función del grado da apertura que se desee.

Hay que tener en cuenta que todas las posiciones del brazo robótico son modificables y se encuentran dentro de una base de datos.

Para realizar esta configuración son necesarios 5 posiciones cartesianas del brazo robótico, que se procede a describirlas a continuación:

- pos_flow_0: Posición intermedia para evitar colisiones con el dispensador.
- pos_flow_1: Posición para situar dentro de la pinza la válvula del compensador.
- pos_flow_2: Posición que se utiliza para que se produzca un movimiento vertical que levante la válvula (con la pinza inferior), cerrándola y dejándola posicionada.
- pos_flow_3: Una vez cerrada la válvula, se realiza un movimiento vertical hacia abajo, con el fin de situar la pinza superior en la posición cero de la válvula.
- pos_flow_4: Esta posición se utiliza para obtener la coordenada “z” máxima que implicaría la apertura máxima de la válvula.

Extrayendo la coordenada z de las posiciones pos_flow_3 y pos_flow_4, obtenemos el rango máximo (en mm) de bajada para abrir la válvula al completo.

Mediante el rango z obtenido y el parámetro de entrada de apertura del compensador en forma de porcentaje, se puede calcular por mediante proporcionalidad la distancia vertical que debe descender el robot para configurar la válvula a la posición deseada.

En el caso de que el dispensador cambie de posición, solo haría falta actualizar la posiciones en la base de datos para configurar el sistema, lo que lo hace muy flexible.


PREMO BEER




Robot positions by motion group

Save
Default

Home Go cup Exit cup Go beer Mov 30° Mov 0° Exit beer Exit table Exit table 2 Regulate flow

	Poss	Speed	Acceleration
Initial	[190.3, 335.1, 296.3, -88.9, 2.1, 29.3]	200.0	4000.0
Inside the regulator	[179.0, 486.6, 332.6, -91.1, 0.4, 31.6]	200.0	4000.0
Gripper flow	500.0		
Close regulator with the lower clamp	[179.0, 486.6, 379.8, -91.1, 0.4, 31.6]	200.0	4000.0
Top clamp in the zero position of the regulator	[177.7, 491.0, 325.0, -91.0, 0.3, 31.9]	200.0	4000.0
Top clamp in the 100% position of the regulator	[177.7, 491.0, 293.1, -91.0, 0.3, 31.9]	200.0	4000.0

All the robot's movements are Cartesian.
 Example of accepted format for movements: [179.0, 486.6, 379.8, -91.1, 0.4, 31.6]
 Example of accepted format for Griper: 40

Ilustración 25. Base de datos de posiciones para configurar el compensador, Interfaz de usuario.

A continuación, se muestra el diagrama de bloques para la configuración del compensador.

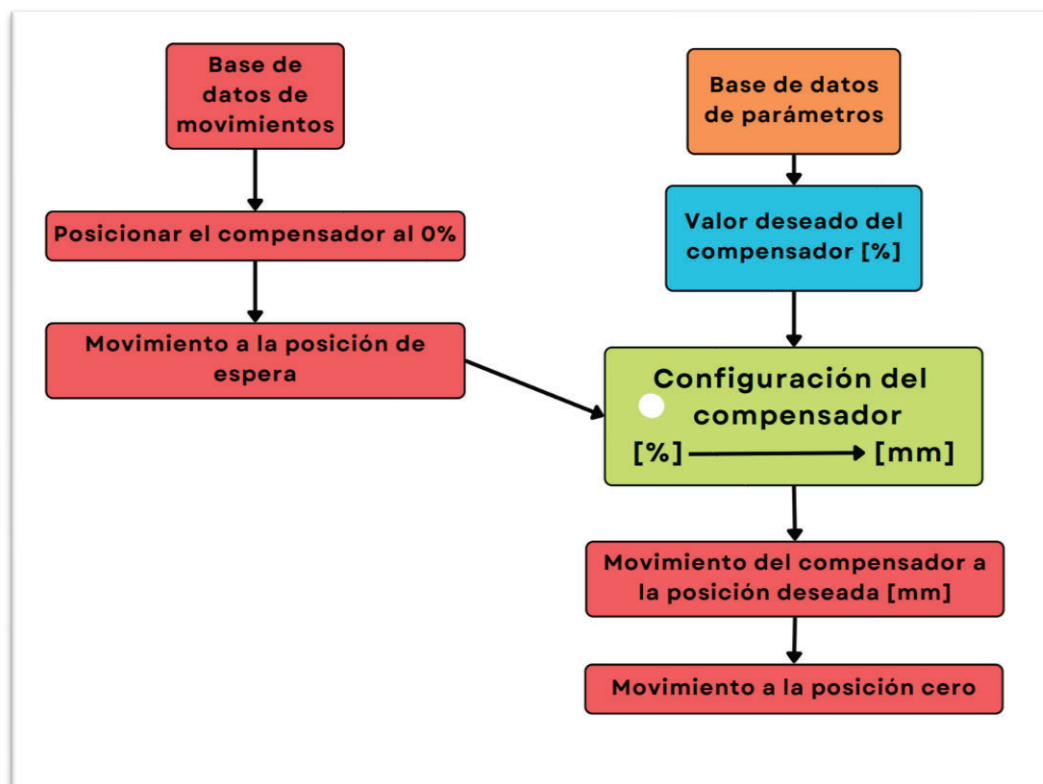


Ilustración 26. Diagrama de flujo, configuración del compensador.

- 3- Una vez configurado el compensador el brazo robótico vuelve a su posición cero, a la espera de que se presione el botón de servir cerveza.
- 4- Al presionar dicho botón, se activa la apertura del dispensador de vasos a la vez que el brazo robótico que se mueve hacia él, las velocidades tanto del actuador lineal del dispensador de vasos y del robot están ajustados para minimizar los tiempos de espera.

El sistema cuenta con un detector de colisiones del robot, este comprueba después de cada movimiento si los sensores del robot han detectado una colisión, si se detecta una colisión:

- Se cierra el grifo de cerveza.
 - Y se paraliza el sistema hasta que se presione el botón de “Reset”.
 - Una vez presionado este botón, el robot se rearma y conociendo la posición en la que se encuentra, se dirige hacia la posición cero, en dos posibles rutas, dependiendo si la pinza lleva vaso o no.
 - Una vez en la posición cero se puede reiniciar el sistema.
- 5- A continuación, el robot utiliza la pinza para sujetar el vaso y tirar de él hacia abajo.
 - 6- Una vez extraído el vaso, se abre y se cierra la pinza para colocar el vaso en la posición deseada.
 - 7- Posteriormente se activa la detección de vasos mediante una red neuronal, este proceso paraliza el sistema de 3 a 6 segundos para comprobar la presencia de vasos. Y la realización de una captura de la imagen de la cámara que será guardada en una carpeta determinada.
 - 8- Una vez detectado el vaso, se procede a cerrar el dispensador de vasos a la vez que el robot se dirige a posicionar el vaso a 45 grados, pero fuera del alcance del grifo de cerveza para prepararse para el descarte.

8-Bis- En caso de no detectarse vaso, el sistema detectará que se ha producido un fallo, y se procederá a:

- Cerrar el dispensador de vasos.
- Mover el robot de vuelta a la posición cero.
- Y volver a inicial el sistema.

- 9- El descarte: Para realizar el descarte se abre el grifo de cerveza en la posición determinada en función del parámetro de entrada de apertura de grifo [%], y 1 segundo después se realiza el movimiento del brazo robótico, colocando en vaso con una inclinación de 45 grados debajo del grifo.
- 10-Detección de llenado: una vez colocado el brazo robótico, se activa la detección de llenado para el primer límite, este consiste en la comparación de la región de interés con el vaso vacío y el actual, y en cuanto se produce un cambio superior al umbral escogido y personalizable, se considera que se ha superado el límite.
- a. Si se detecta que se ha superado el límite, se procede a realizar el siguiente movimiento del robot posicionando el vaso a 30 grados con la horizontal y activándose el detector de llenado para el 2 límite.
 - b. Si la salida de la bebida no se produce o es demasiado lenta, se detecta al exceder el tiempo máximo de duración de la detección de llenado. Una vez detectado este error se cierra el grifo de cerveza y se analiza los parámetros de entrada.
 - i. Si los parámetros de entrada (apertura del grifo y del compensador) se encuentran al máximo, es decir al 100%, y a pesar de ello no se produce la salida de la bebida, se considera que el barril está vacío, apareciendo un mensaje en pantalla indicándolo y realizando el movimiento de retorno a la posición inicial del robot.
 - ii. En el caso de que los parámetros de entrada no se encuentren al máximo se incrementan ambos un 10%, siempre que no supere el 100% y se reinicia el sistema.

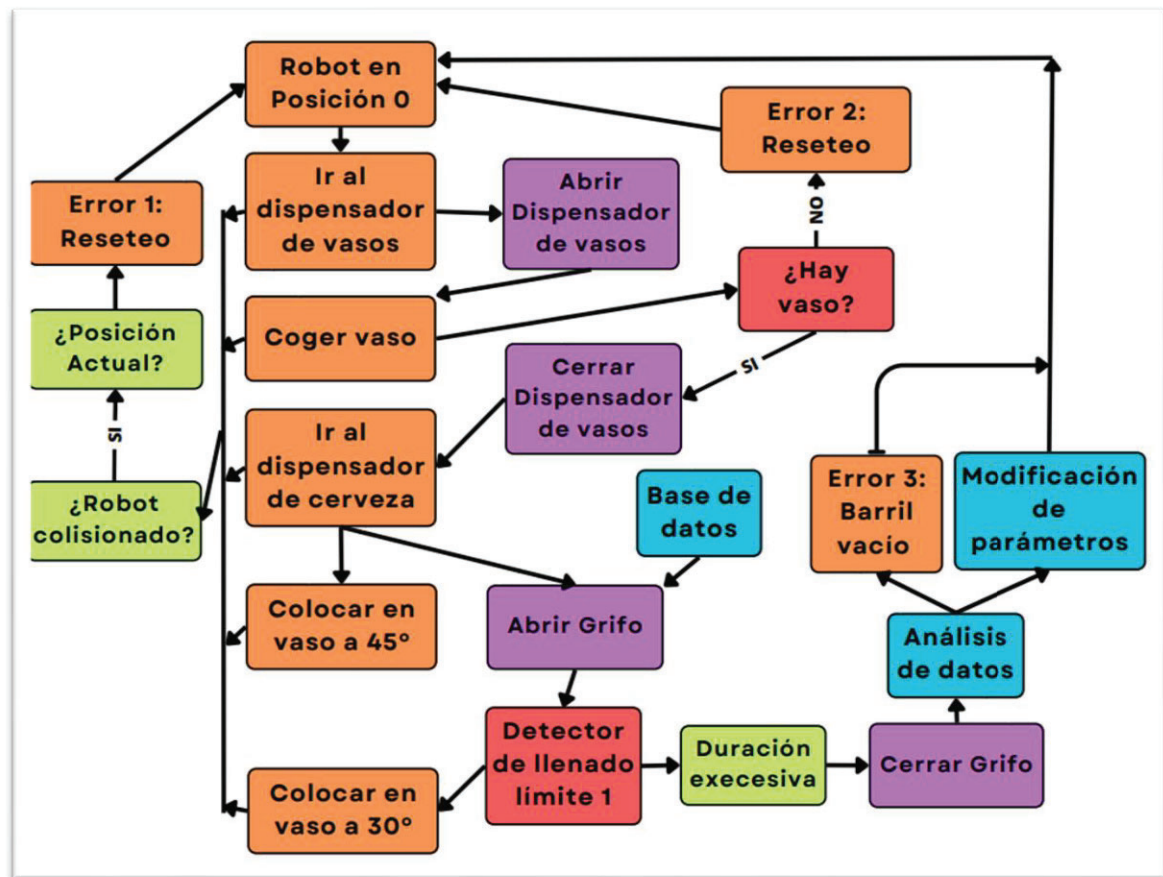


Ilustración 27. Diagrama de flujo del sistema 1.

- 11-Una vez detectado el llenado del primer límite, el brazo robótico se mueve, recolocando el vaso a 30 grados con la horizontal, mientras sigue abierto el grifo. Una vez concluido el movimiento se activa la detección de llenado del límite 2.
- 12-La detección de llenado de límite 2 es similar al anterior, salvo que cambia la posición de la región de interés y que la detección de duración excesiva tiene el objetivo de finalizar el proceso en el caso de que se produzca un error al no detectar el llenado.
- 13-Una vez detectado el llenado del límite 2, el robot se mueve, recolocando el vaso a 0 grados con la horizontal, mientras continúa el grifo abierto. Una vez concluido el movimiento se inicia la detección de llenado del límite 3.
- 14- La detección de llenado del límite 3 es similar al anterior, salvo que al finalizar se cierra el grifo.
- 15-Una vez cerrado el grifo se mueve el mismo a la posición de crema de cerveza. Y comienza la detección de llenado del límite 4.

- 16-Una vez finalizado el llenado del límite 4, se cierra el grifo.
- 17-A continuación, se realizan una serie de movimientos para colocar el vaso relleno de bebida en la bandeja de recogida.
- 18-Una vez dejado el vaso en la bandeja de recogida, el robot se desplaza a una posición más alta donde tomara una foto de la bebida para almacenarla y analizarla posteriormente.

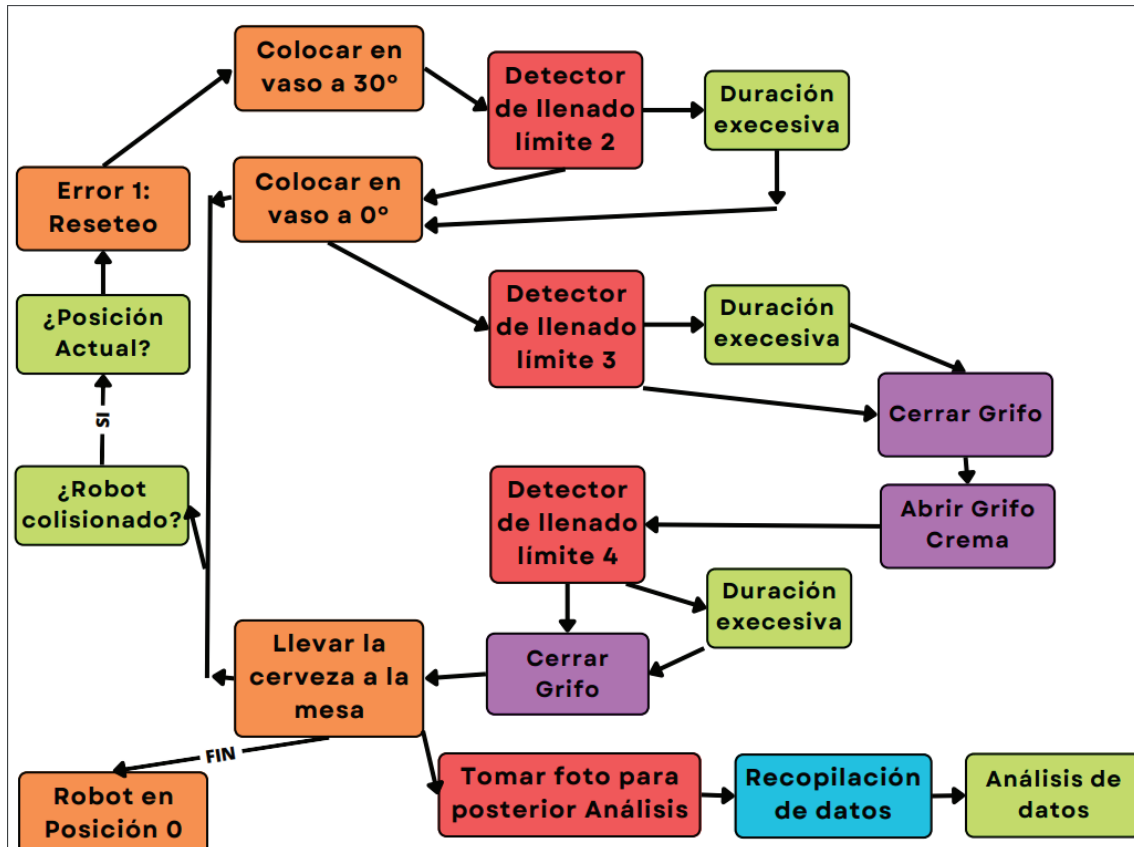


Ilustración 28. Diagrama de flujo del sistema 2.

- 19- Una vez finalizado el proceso, se procede a analizar todos los datos recopilados durante el proceso de servir la bebida, con el fin de modificar o no los parámetros de entrada del sistema.
- Si las distintas duraciones se encuentran dentro del rango óptimo, no se modificarán los parámetros.
 - Si las distintas duraciones superan el límite máximo de duración, implica que el flujo de la bebida es bajo y por tanto se modifican los parámetros aumentando el valor del parámetro de apertura del compensador, en el caso de que se alcance el máximo valor de dicho parámetro se aumentara el valor de apertura del grifo.

- c. Si las distintas duraciones no superan el límite mínimo de duración, implica que el flujo de la bebida es demasiado alto y por tanto, se modifican los parámetros, disminuyendo el valor del parámetro de apertura del compensador, en el caso de que se alcance el mínimo valor de dicho parámetro se disminuye el valor de apertura del grifo.

En el caso de que sea necesario modificar el parámetro de apertura del compensador, el brazo robótico modificara la apertura de la válvula del compensador antes de volver a la posición cero y finalizar el ciclo.

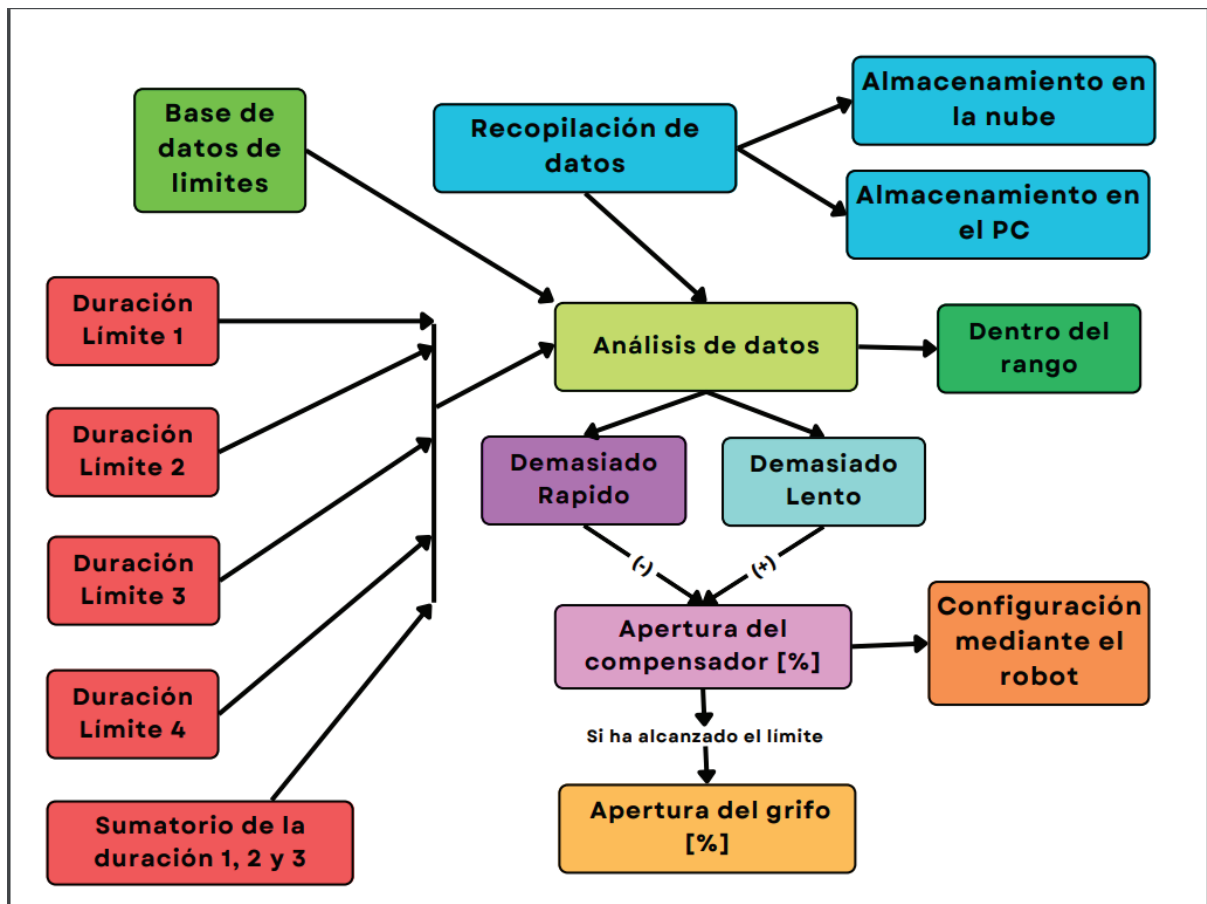


Ilustración 29. Diagrama de flujo del sistema 3.

10. Redes neuronales.

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo de arquitectura de redes neuronales profundas especialmente diseñadas para el procesamiento de datos en forma de imágenes o datos espaciales. Son ampliamente utilizadas en tareas de visión por computadora, como reconocimiento de objetos, detección de características y clasificación de imágenes.

Una red neuronal convolucional está compuesta por varias capas, incluyendo capas de convolución, capas de pooling y capas completamente conectadas. Estas capas se combinan para formar una arquitectura capaz de extraer características relevantes de una imagen y aprender a reconocer patrones específicos.

La capa de convolución es la parte central de una CNN. Consiste en un conjunto de filtros (kernels) que se deslizan sobre la imagen de entrada para extraer características locales mediante operaciones de convolución. Cada filtro detecta diferentes características, como bordes, texturas o formas, y genera mapas de características.

La capa de pooling reduce el tamaño espacial de los mapas de características obtenidos mediante la convolución. Esto ayuda a reducir la dimensionalidad y la cantidad de parámetros en la red, además de proporcionar cierta invariancia a las traslaciones y mejorando la eficiencia computacional.

10.1. YOLO V8

YOLO (You Only Look Once, Solo mira una vez) es una arquitectura de red neuronal convolucional diseñada específicamente para la detección en tiempo real de objetos en imágenes. A diferencia de otros enfoques de detección que requieren múltiples pasadas sobre la imagen, YOLO realiza la detección y localización de objetos en una sola pasada.

YOLO divide la imagen en una cuadrícula y asigna a cada celda de la cuadrícula cajas delimitadoras y probabilidades para diferentes clases de objetos. A través de múltiples capas convolucionales y capas completamente conectadas, YOLO aprende a reconocer y localizar objetos en la imagen de entrada. (Kelleher, 2018)

YOLO v8 (versión 8) es una versión mejorada y actualizada de la arquitectura YOLO, diseñada para la detección de objetos en tiempo real. A continuación, se explica cómo funciona YOLOv8, sus principales características y el proceso para personalizarlo.

10.1.1. Funcionamiento de YOLO:

- 1- Preprocesamiento de la imagen: La imagen de entrada se ajusta al tamaño requerido por la red neuronal y se normaliza.
- 2- Extracción de características: La imagen preprocesada se pasa a través de una serie de capas convolucionales que extraen características a diferentes escalas y niveles de abstracción.
- 3- Detección y localización de objetos: A partir de las características extraídas, YOLOv8 utiliza múltiples capas de detección para predecir las cajas delimitadoras y las probabilidades de clase de los objetos presentes en la imagen. Estas capas de detección se encargan de capturar objetos de diferentes tamaños y aspectos.
- 4- Supresión de no máximos: Se aplica una supresión de no máximos para eliminar las detecciones redundantes y conservar las detecciones más relevantes y precisas.
- 5- Postprocesamiento: Las detecciones finales se ajustan a las dimensiones originales de la imagen y se proporcionan como resultados de detección.
(Joseph Redmon)

10.1.2. Principales características de YOLOv8:

- 1- Detección en tiempo real: YOLOv8 es capaz de realizar detecciones en tiempo real, lo que lo hace adecuado para aplicaciones en tiempo real, como sistemas de vigilancia o vehículos autónomos.

- 2- Arquitectura basada en Darknet: YOLOv8 utiliza la biblioteca Darknet, que es una implementación en código abierto de redes neuronales convolucionales escrita en C y CUDA.
- 3- Detección de múltiples objetos: YOLOv8 puede detectar y clasificar múltiples objetos en una sola imagen, incluso si se superponen entre sí.
- 4- Alto rendimiento: YOLOv8 ha sido optimizado para lograr un rendimiento rápido en la detección de objetos, incluso en dispositivos con recursos limitados.
- 5- Generalización y adaptabilidad: YOLOv8 puede detectar objetos de diferentes clases y se puede adaptar a diferentes dominios y aplicaciones mediante la personalización de los datos de entrenamiento.

10.1.3. Proceso de personalización de YOLOv8:

- 1- Recopilación de datos de entrenamiento: se debe recopilar un conjunto de imágenes etiquetadas con las clases de objetos que deseas detectar. Cada imagen debe tener asociadas las coordenadas de las cajas delimitadoras y las etiquetas de clase correspondientes.
- 2- Configuración del archivo de configuración: ajuste de los parámetros y la configuración del archivo "yolov8.cfg" para adaptarlo a la tarea de detección. Esto incluye ajustar el número de clases, los tamaños de las capas, las tasas de aprendizaje, entre otros.
- 3- Entrenamiento de la red neuronal: Utilizando las imágenes etiquetadas, se entrena la red neuronal YOLOv8 ajustando los pesos de los diferentes componentes de la red. Esto implica ejecutar el proceso de entrenamiento en un conjunto de imágenes y ajustar iterativamente los pesos mediante técnicas de optimización, como el descenso de gradiente estocástico.
- 4- Prueba y evaluación: Una vez entrenada la red, debes probarla en un conjunto de datos de prueba para evaluar su desempeño y precisión. Esto implica ejecutar la red en imágenes de prueba y comparar las detecciones realizadas con las anotaciones de las imágenes. (Ultralytics, 2023)

- 5- Ajuste y refinamiento: Si la precisión de detección no es satisfactoria, se puede ajustar y refinar los parámetros y la configuración de la red, o bien recopilar más datos de entrenamiento para mejorar el rendimiento.

10.2. Red customizada para la detección de vasos.

Para customizar una red neuronal convolucional para la detección de vasos de plásticos dentro de la pinza, se utilizan un conjunto de 90 imágenes (dataset) tomadas por la cámara situada en la pinza del brazo robótico.

Para reunir el dataset se programa el robot para que extraiga vasos del dispensador y haga fotos en bucle.

Una vez obtenido el dataset, se etiquetan manualmente las cajas delimitadoras y las etiquetas de clase correspondientes.

Debido a que la red neuronal debe indicar la presencia o no de vasos y dichos vasos son transparentes y difíciles de apreciar con la resolución de la cámara, Se introducen en el cada set imágenes de la pinza sin vasos.

Para el preprocesamiento y el etiquetado del dataset se utiliza en este proyecto la herramienta Roboflow.

De las 90 imágenes, 67 imágenes son positivas, con vasos y 23 imágenes son negativas, sin vaso.

Del dataset de 90 imágenes, no se emplean todas para el entrenamiento, sino que solo se entrenan la red con el 78% (con 70 imágenes) y el resto se emplean para el proceso de validación y testeo de la red neuronal convolucional.

✓	Source Images	Images: 90 Classes: 1 Unannotated: 0
✓	Train/Test Split	Training Set: 70 images Validation Set: 10 images Testing Set: 10 images
✓	Preprocessing	Auto-Orient: Applied Resize: Stretch to 640×640

Ilustración 30. Información sobre el dataset de entrenamiento.

Además, se añaden herramientas de preprocesamiento, como el redimensionamiento de las imágenes a 640x640 píxeles.

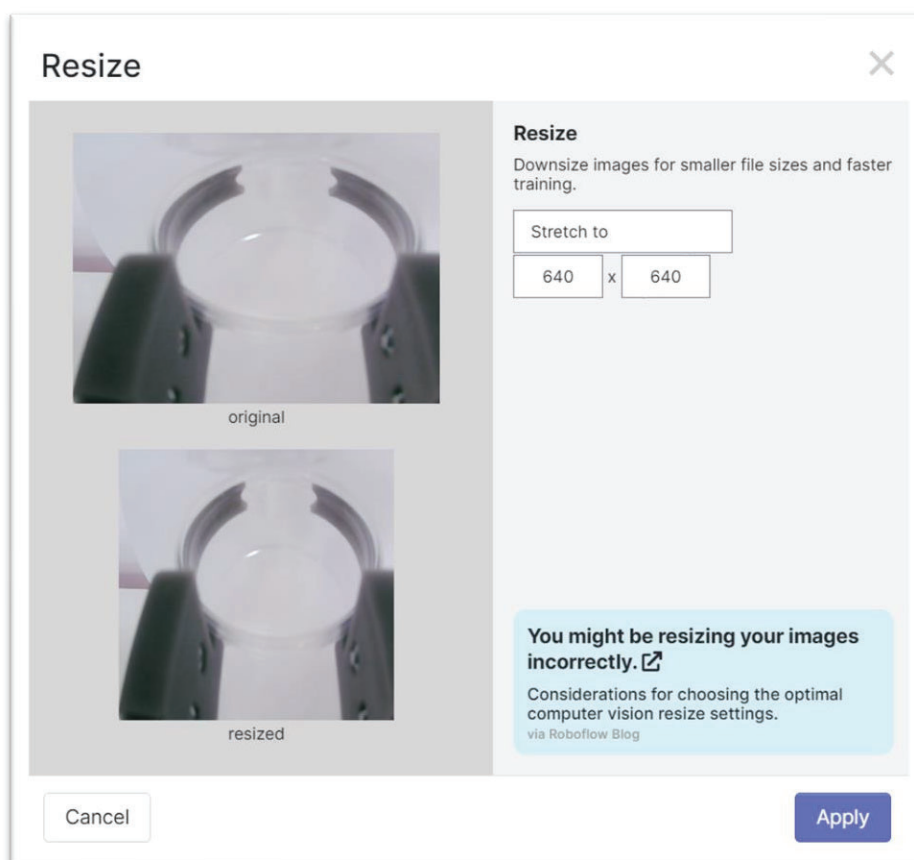


Ilustración 31. Redimensionamiento de las imágenes.

Para el entrenamiento de la red se utiliza la herramienta Google Colab debido a que proporciona recursos de cómputo gratuitos, incluyendo CPU y GPU. Esto es especialmente beneficioso para entrenar modelos de detección de objetos como YOLO, que requieren una cantidad significativa de recursos computacionales.

Para entrenar un modelo de detección de objetos YOLOv8 se utiliza la siguiente función en Python.

```
“!yolo task=detect, mode=train, model=yolov8x.pt,  
data={dataset.location}/data.yaml, epochs=25, imgsz=640, plots=True”
```

El comando se divide en diferentes parámetros que se explican a continuación:

- Yolo: es el comando para ejecutar la tarea de detección de objetos.
- Task = detect: indica que la tarea es detectar objetos.
- Mode=train: indica que el modelo se está entrenando.
- Model=yolov8x.pt: indica el modelo que se está utilizando para el entrenamiento.

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Ilustración 32. Tabla comparativa de modelos YOLO V8

El modelo YOLOv8x.pt es una variante del modelo YOLOv8 que utiliza una arquitectura más grande y compleja para lograr un mejor rendimiento. En general, el modelo YOLOv8x.pt es más preciso que otros modelos YOLOv8, pero también es más lento.

- Data={dataset.location}/data.yaml: indica la ubicación del conjunto de datos personalizado.
- Epochs=25: indica la cantidad de épocas que se utilizarán para entrenar el modelo. Las épocas son una medida de la cantidad de veces que el modelo ve el conjunto de datos completo durante el entrenamiento, en otras palabras, una época es una iteración completa a través de todo el conjunto de datos.
- Imgsz=640: indica el tamaño de las imágenes que se utilizarán para entrenar el modelo.
- plots=True: indica si se deben generar gráficos durante el entrenamiento.

10.2.1. Matriz de confusión

Una vez obtenida la red customizada, procedemos a evaluar el rendimiento de un modelo mediante la matriz de confusión.

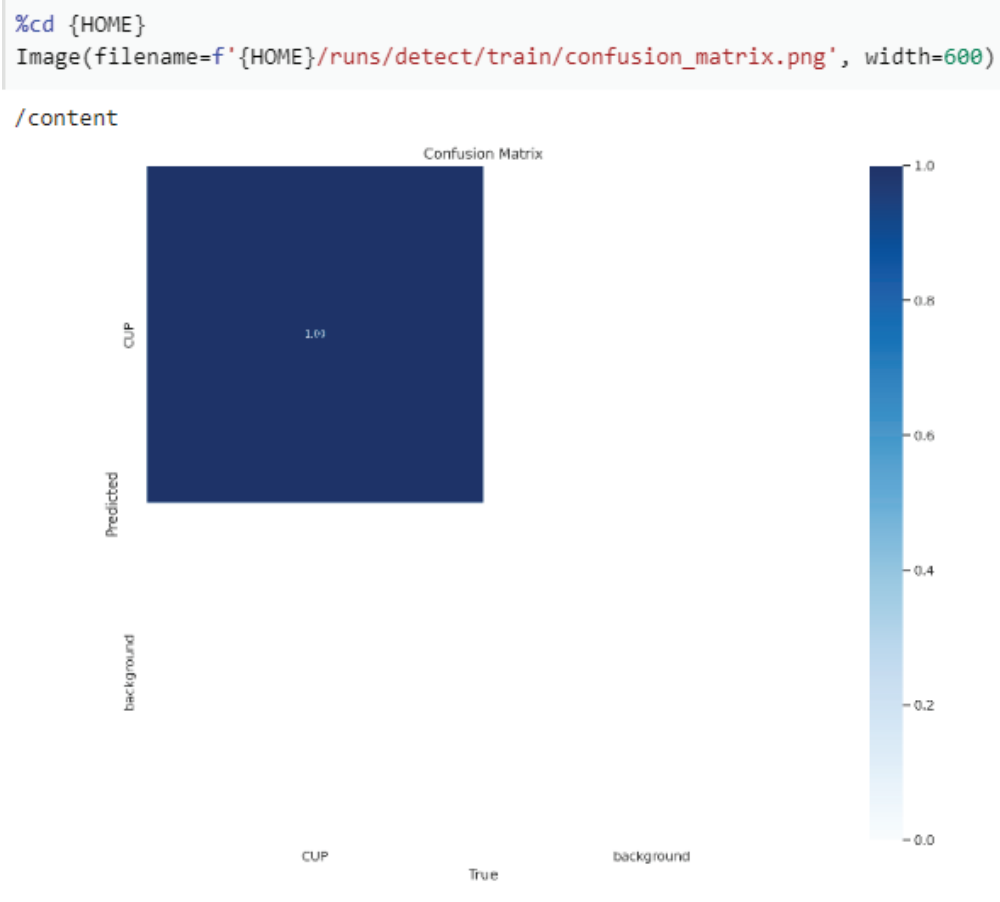


Ilustración 33. Matriz de confusión.

Dado que el modelo solo detecta una clase llamada CUP (vaso) se simplifica mucho la matriz.

De la matriz podemos obtener la siguiente información:

- Verdaderos positivos (True Positives, TP): indica el número de objetos que el modelo ha detectado y clasificado correctamente como positivos. Es decir, el modelo identificó correctamente la presencia de los objetos de interés.
- Falsos negativos (False Negatives, FN): indica el número de objetos reales que el modelo no ha detectado o ha clasificado incorrectamente como negativos. Es decir, el modelo no logró identificar la presencia de los objetos de interés.
- Falsos positivos (False Positives, FP): indica el número de regiones que el modelo ha detectado incorrectamente como objetos positivos, pero en realidad no contienen objetos de interés. Estas regiones se consideran falsas alarmas.
- Verdaderos negativos (True Negatives, TN): en el contexto de la detección de objetos, el concepto de verdaderos negativos no es tan relevante. Representaría la ausencia de objetos de la clase a detectar y generalmente no se considera, ya que el enfoque principal está en detectar los objetos de interés.

A continuación, se calcula la precisión y la tasa de detección de la red, aunque no es necesario debido a que el modelo no ha fallado.

- Precisión de la detección:

$$Precisión = \frac{TP}{TP + FP} = \frac{1}{1 + 0} = 1$$

- La tasa de detección:

$$Tasa\ de\ detección = \frac{TP}{TP + FN} = \frac{1}{1 + 0} = 1$$

Estas métricas son útiles para medir la capacidad del modelo para detectar correctamente los objetos de la clase objetivo.



Ilustración 34. Prueba de funcionamiento.

A continuación, se muestran las gráficas de seguimiento de entrenamiento, Estas gráficas representan diferentes métricas y medidas relacionadas con el rendimiento del modelo durante el proceso de entrenamiento.

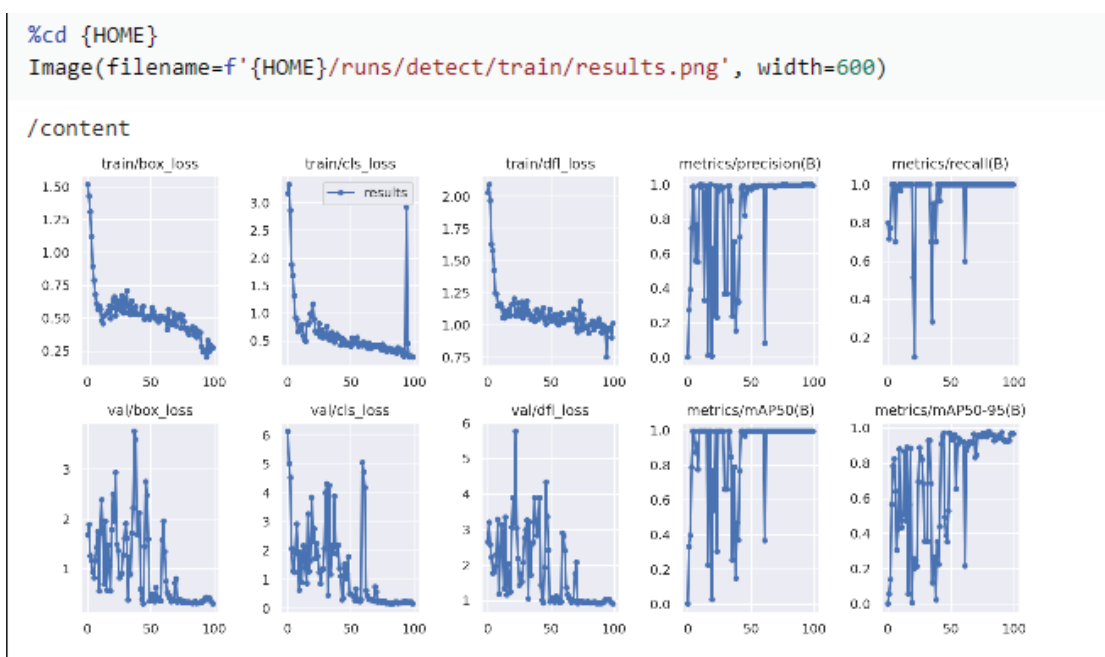


Ilustración 35. Gráficas de seguimiento de entrenamiento.

11. Manual de usuario del sistema.

11.1. Pantalla de inicialización:

Al iniciar el programa, aparecerá una ventana que ocupara la pantalla al completo, en la pantalla se mostrara el logotipo del sistema junto a una barra de progreso, justo debajo de la barra de progreso, aparecerán mensajes informado del estado del sistema y advirtiéndolo de errores detectados.

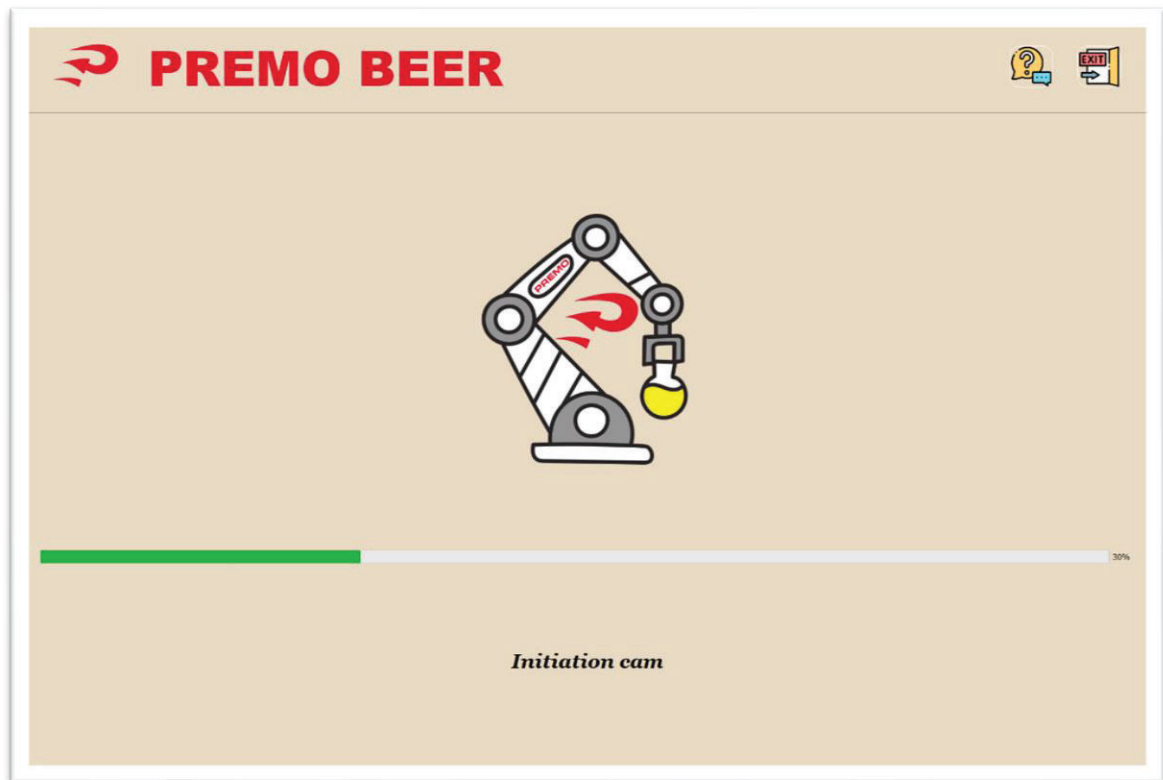


Ilustración 36. Pantalla de inicio.



Ilustración 37. Pantalla de inicio, mostrando un error.

11.2. Pantalla de error al inicio.

En el caso de que se produjera un error al iniciarse el sistema, aparecerá una ventana, informando del tipo de error y recomendando posibles soluciones.

11.2.1. Error de conexión del brazo robótico:

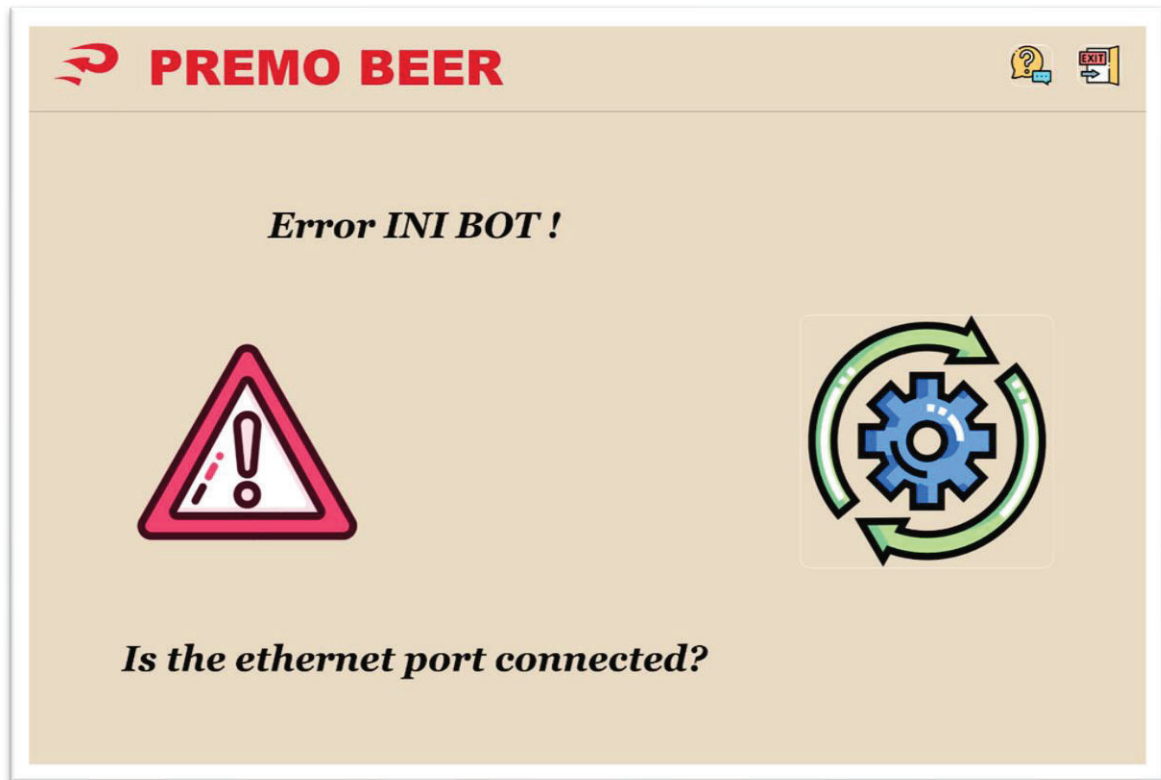


Ilustración 38. Error conexión robot

Este error tiene 3 posibles causas:

- No está conectado el puerto Ethernet rj45, que conecta el brazo robótico con el sistema.
- El brazo robótico está apagado, bien por no estar alimentado a la corriente o por el interruptor del controlador situado debajo de la mesa, está en apagado.
- La IP del brazo robótico utilizado es incorrecta. Para solucionarlo, se puede modificar el parámetro de IP siguiendo los siguientes pasos:
 - Presione el botón “EXIT” situado en la parte superior derecha de la pantalla.

- Aparecerá el menú principal, presione el botón “Settings” para que se muestren los botones de configuración.
- A continuación, presione el botón “Parameters” para entrar dentro de la ventana de modificación de los parámetros de inicio.
- Modifique en la tabla el valor de “IP BOT”, con la dirección IP del brazo robótico utilizado, que se muestra en la placa de características del controlador.
- Presionar “Save” para guardar los cambios.
- Y por último reiniciar el sistema presionando 2 veces seguidas el botón “Exit”.

Una vez solucionado el problema, presionar el botón “reset” (cuyo símbolo es un engranaje y dos flechas en forma de círculo) para inicial el brazo robótico.

11.2.2. Error de conexión a internet:

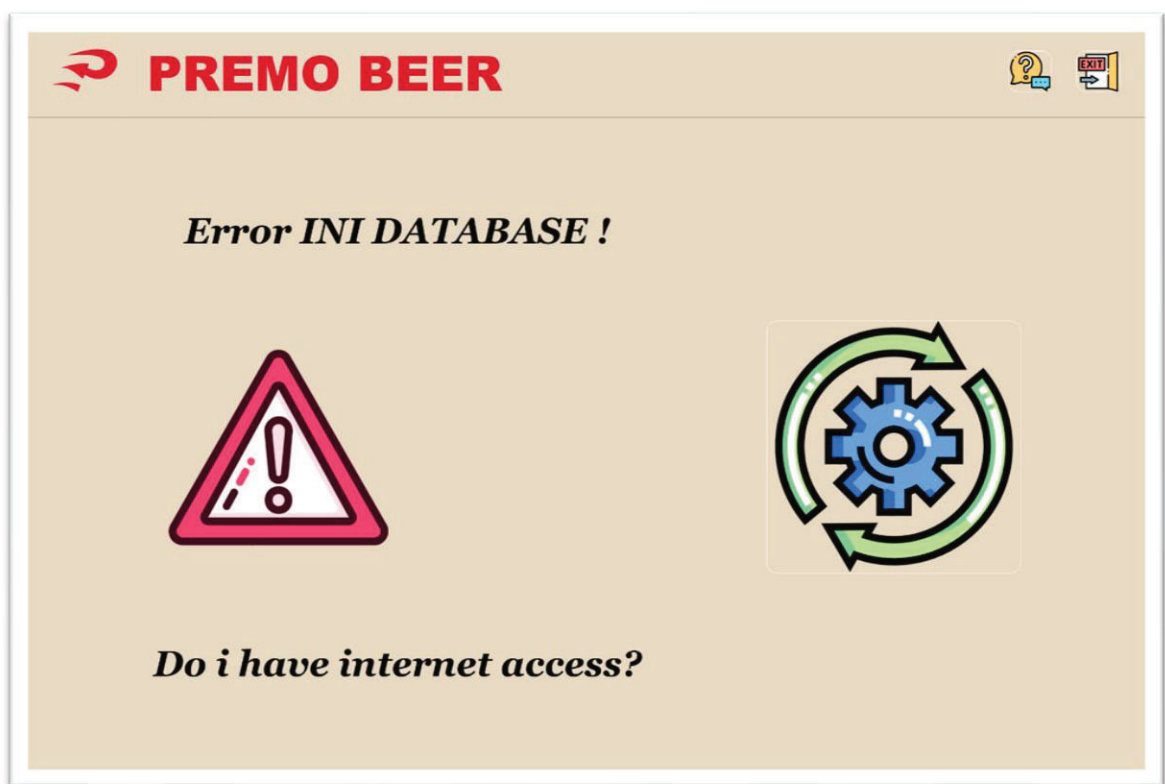


Ilustración 39. Error conexión a internet

El sistema utiliza una base de datos en la nube, por lo que no es posible utilizarla sin internet.

El sistema es capaz de funcionar sin internet y siempre almacenará la recopilación de datos dentro de la carpeta llamada: Analysys seguida de la fecha de inicio del sistema ("Analysys_17-04-2023-_17_10_39"), esta carpeta será creada por el sistema en su inicio en el mismo directorio donde se encuentra la aplicación.

No obstante, la función "Analytics" encargada de mostrar gráficos del comportamiento del sistema, no está implementada para funcionar sin internet.

Una vez solucionado el problema, presionar el botón "Reset" (cuyo símbolo es un engranaje y dos flechas en forma de círculo) para conectarse a internet.

Si desea continuar sin internet, presione el botón "EXIT".

11.3. Pantalla de menú principal:

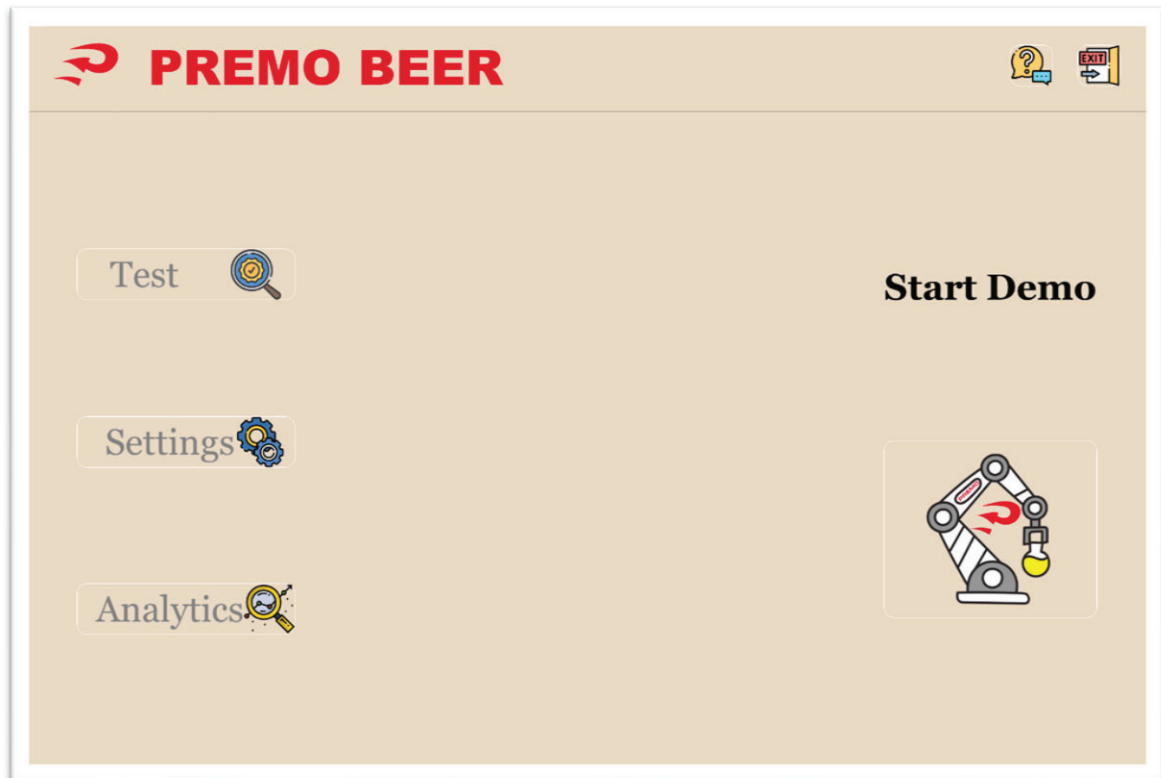


Ilustración 40. Pantalla menú principal.

Una vez inicializado el sistema aparecerá una pantalla con diversos botones que a continuación se describen:

En la parte superior derecha se sitúan 2 botones que son constantes en todas las ventanas.

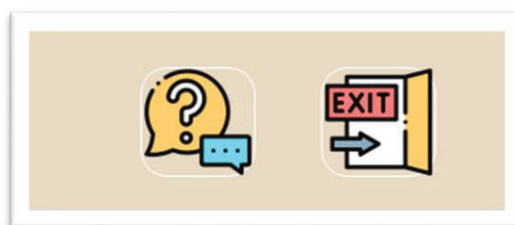


Ilustración 41. Botones de ayuda y salida.

El de la izquierda es el botón de ayuda, presionándolo se abrirá un archivo pdf de ayuda.

El botón de la derecha es el botón “Exit” o de salida, en caso de situarse en él menú principal, apagará el sistema, y en el caso de encontrarse en otra ventana, llevará al menú principal.

Presionando el botón “Settings” se muestra una columna más de botones para configurar los parámetros del sistema más fácilmente. Presionando otra vez el botón, la columna de botones se ocultará.

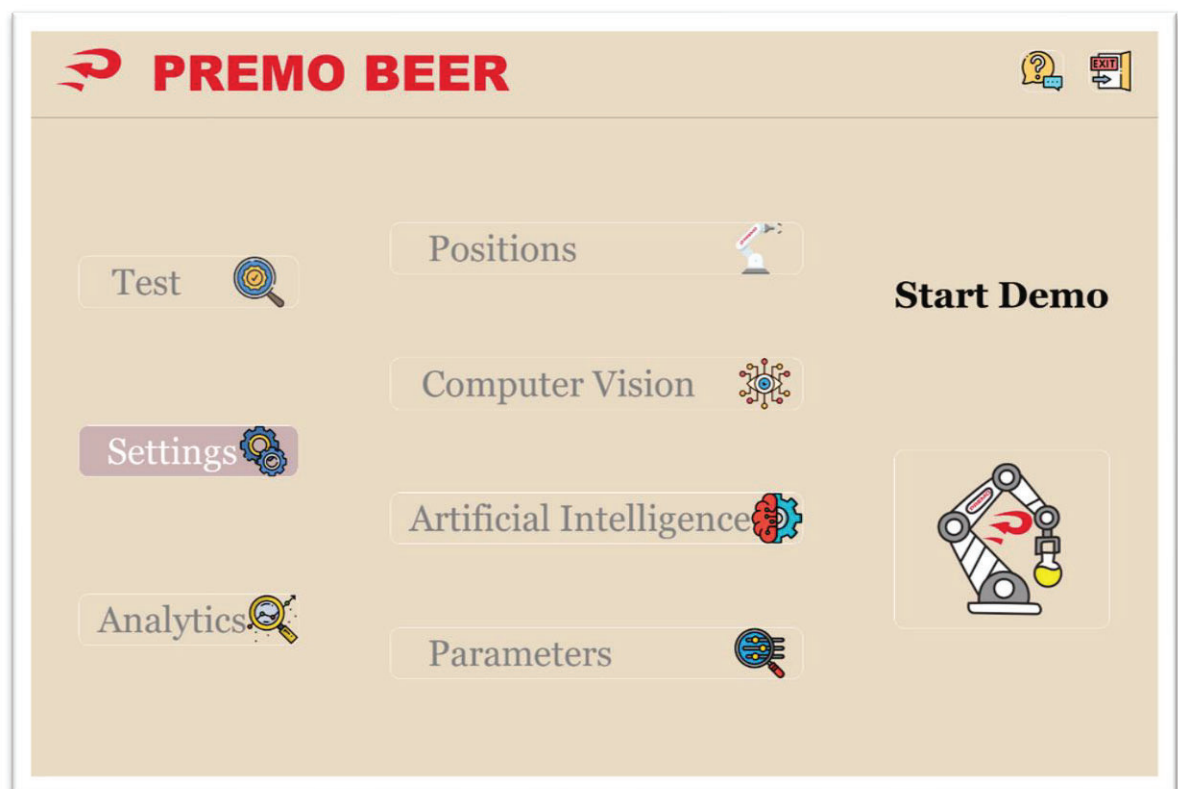


Ilustración 42. Pantalla menú principal con botones de configuración.

A continuación, se detallan los demás botones:

11.3.1. Test.

Al presionar el botón “Test” aparecerá la siguiente pantalla, consta de 3 columnas de botones. El objetivo de esta pantalla es comprobar el funcionamiento del sistema

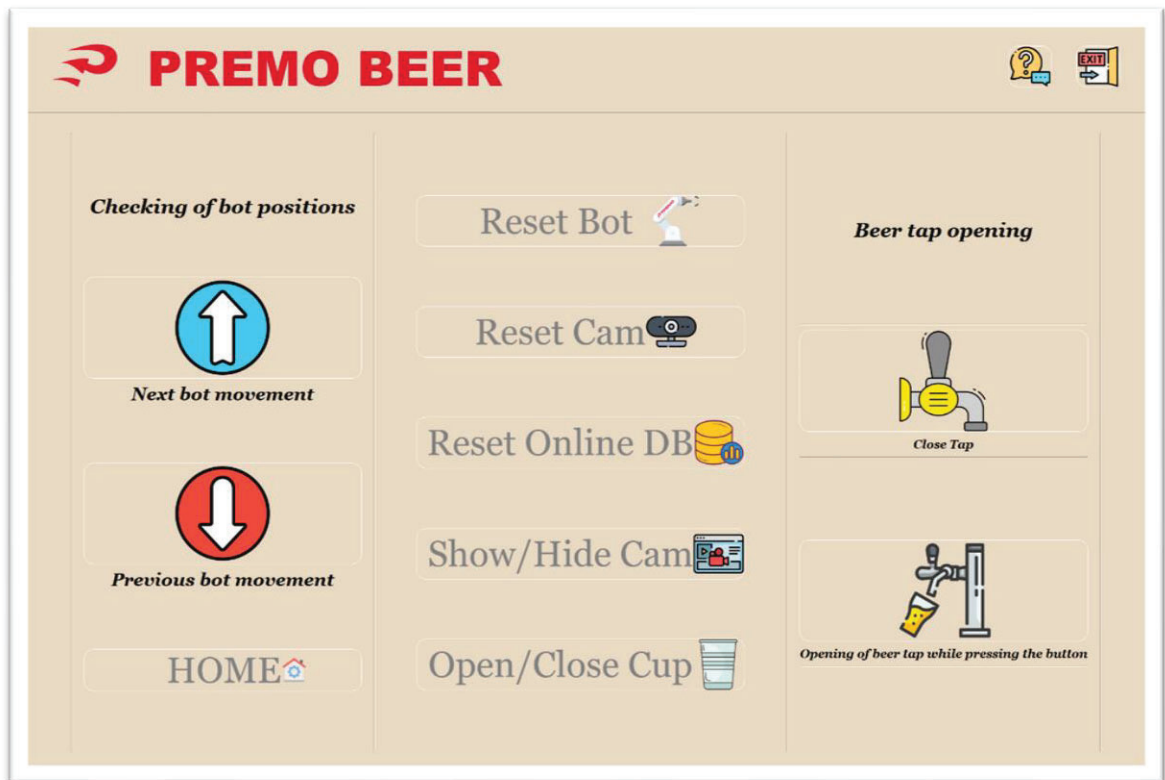


Ilustración 43. Pantalla test.

La primera columna de botones es para comprobar todas las posiciones del brazo robótico en orden.

- El botón de la flecha azul mueve el robot a la siguiente posición, pudiendo volver a presionarlo hasta realizar todo el recorrido.
- El botón de la flecha roja mueve el robot a la anterior posición, pudiendo volver a presionarlo hasta situarse en la posición cero u home.
- El botón “Home”, mueve el robot a la posición cero u home, tenga en cuenta que no debe de utilizarse si existe algún obstáculo entre la

posición actual del brazo robótico y la “home”, dado que se seguirá la trayectoria más corta sin tener en cuenta los posibles obstáculos.

Estas funciones son muy útiles para comprobar las posiciones del brazo robótico y para comprobar que están bien situados todos los elementos, como el dispensador de cerveza y el dispensador de vasos.

La segunda columna de botones posee botones para reiniciar los sistemas y para comprobar funciones.

- El botón “Reset Bot” se emplea para reiniciar el brazo robótico en caso de que se produzca un error de colisión o de IP del brazo robótico incorrecta.
- El botón “Reset Cam” se emplea para reiniciar la cámara y el Arduino, en el caso de introducir índices de puertos incorrectos en la ventana de configuración de parámetros iniciales.
- El botón “Reset Online DB” se emplea para reiniciar la conexión a internet y conectarse a la base de datos en la nube.
- El botón “Show/Hide Cam” se emplea para mostrar u ocultar una ventana que retransmite el video en directo de la cámara.
- El botón “Open/Close Cup” se emplea para abrir o cerrar el dispensador de vasos de plásticos, dependiendo de la posición en la que se encuentre el dispensador de vasos.

En la tercera y última columna, se encuentran 2 botones para actuar sobre el grifo del dispensador de cerveza.

El botón “Close Tap” se emplea para cerrar el grifo de cerveza en el caso de que este permaneciera abierto por error.

El botón “Opening of beer tap while pressing the button” se emplea para abrir el grifo de cerveza mientras se mantiene pulsado el botón y en el momento que se deja de presionar el botón, se cierra el grifo. Este botón es muy útil para eliminar la primera cerveza muy espumada al conectar un barril nuevo al sistema.

11.3.2. Analytics

Al presionar este botón se abrirán diversas ventanas con diversos gráficos que muestran el funcionamiento del sistema y la variación de los parámetros del sistema en función del sistema.

La primera ventana de análisis de datos muestra 2 gráficos donde el eje “x” es el número de identificación de cada cerveza servida hasta el momento, lo que muestra la evolución de los parámetros para cada cerveza.

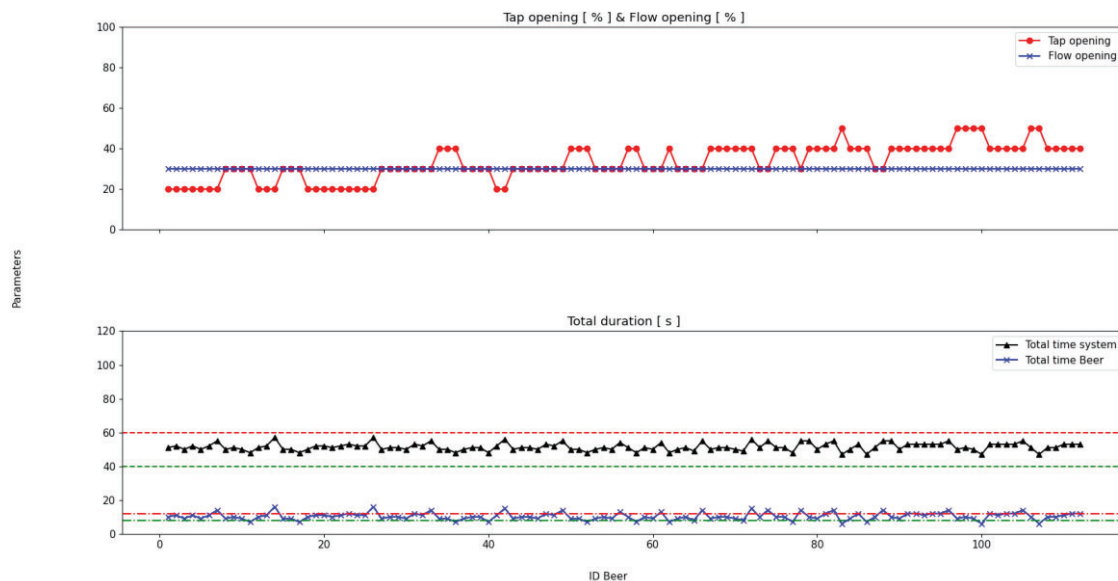


Ilustración 44. Gráfica Apertura del grifo, del compensador, duración del sistema, duración de la cerveza en función de la cerveza

El primer gráfico muestra la evolución de los parámetros de entrada del sistema, los porcentajes de apertura del compensador y del grifo de la cerveza.

El segundo gráfico representa la duración del sistema completo de servir una cerveza (línea negra) y la duración total del grifo abierto (línea azul), y sus respectivos límites máximos en rojo y mínimos en verde.

Se puede apreciar que cuando se produce un error que impide servir la cerveza, como puede ser colisión del robot o ausencia de vasos, dichos tiempos pasan a ser cero, apareciendo caídas a cero en el gráfico.

También se aprecia que cuando la duración del sistema infringe los límites se modifican los valores de entrada para mantener dichos tiempos dentro del rango establecido en la tabla de parámetros de IA.

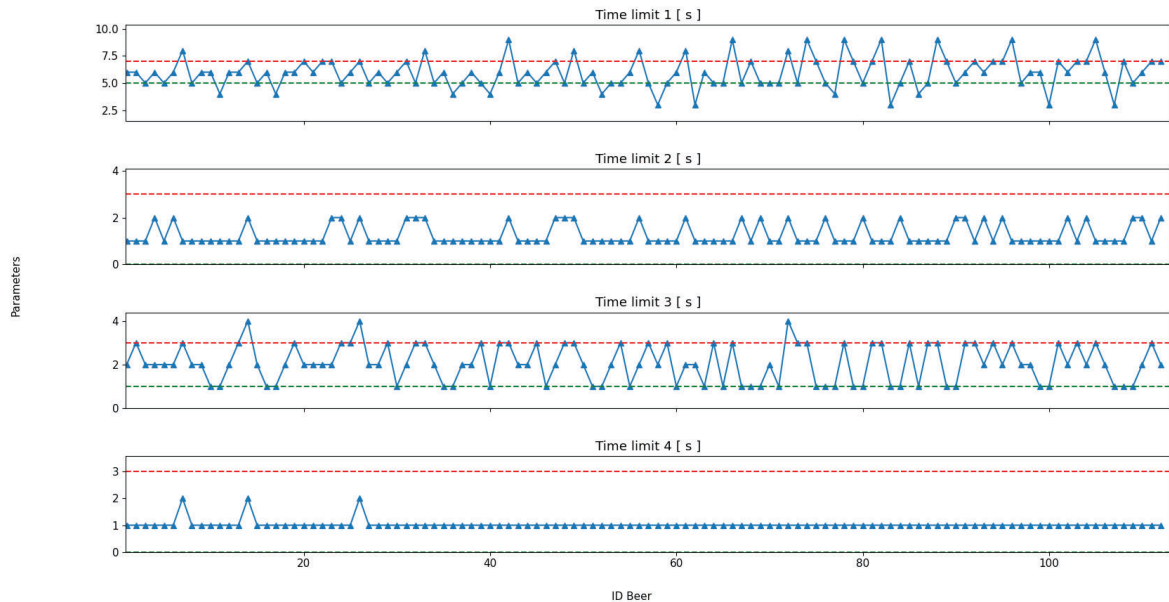


Ilustración 45. Gráfica. Duración de llenado de cada tramo en función de la cerveza.

En esta segunda ventana, se muestra 4 gráficos, que representan la duración de llenado de cada tramo con sus respectivos tiempos límites.

Se puede apreciar que cuando un tiempo excede un límite, se produce una corrección que hace que en la siguiente cerveza se corrija el valor.

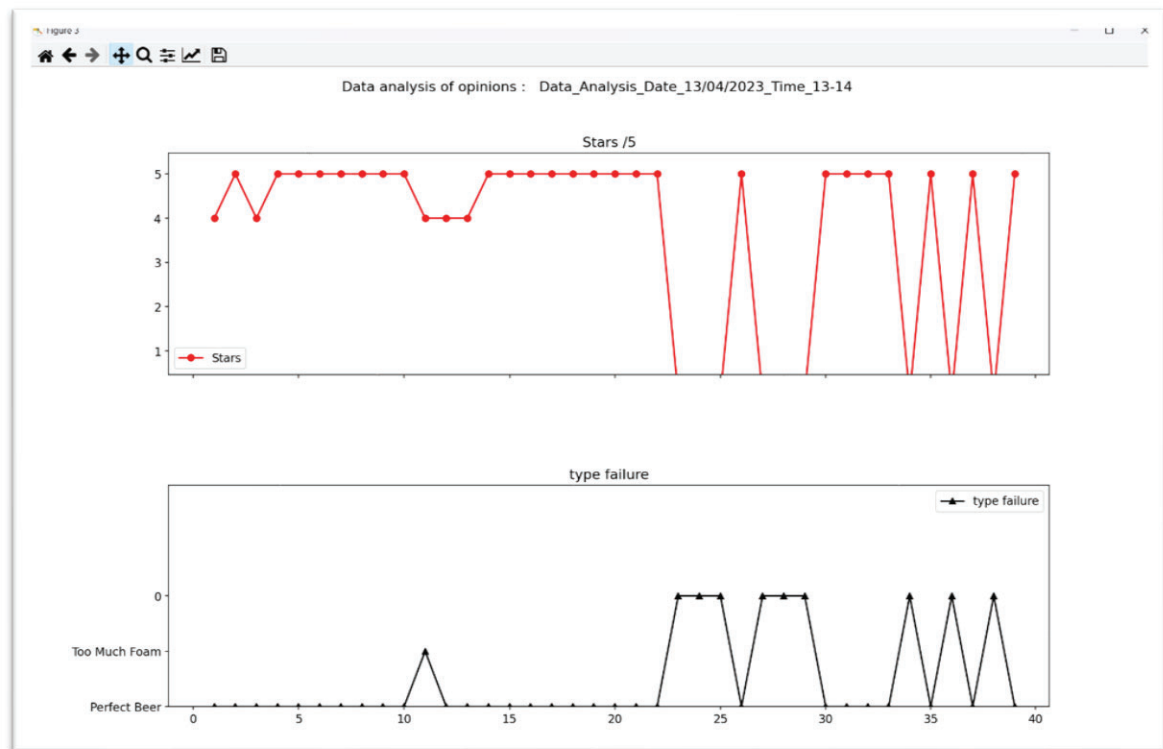


Ilustración 46. Gráfica. Distribución de la puntuación y de los errores.

En esta tercera ventana se muestra la distribución de la puntuación, cuyo máximo es 5 y mínimo 1, cuando la gráfica marca 0 implica que el usuario no ha realizado la encuesta o se ha producido un error que impidió servir la cerveza.

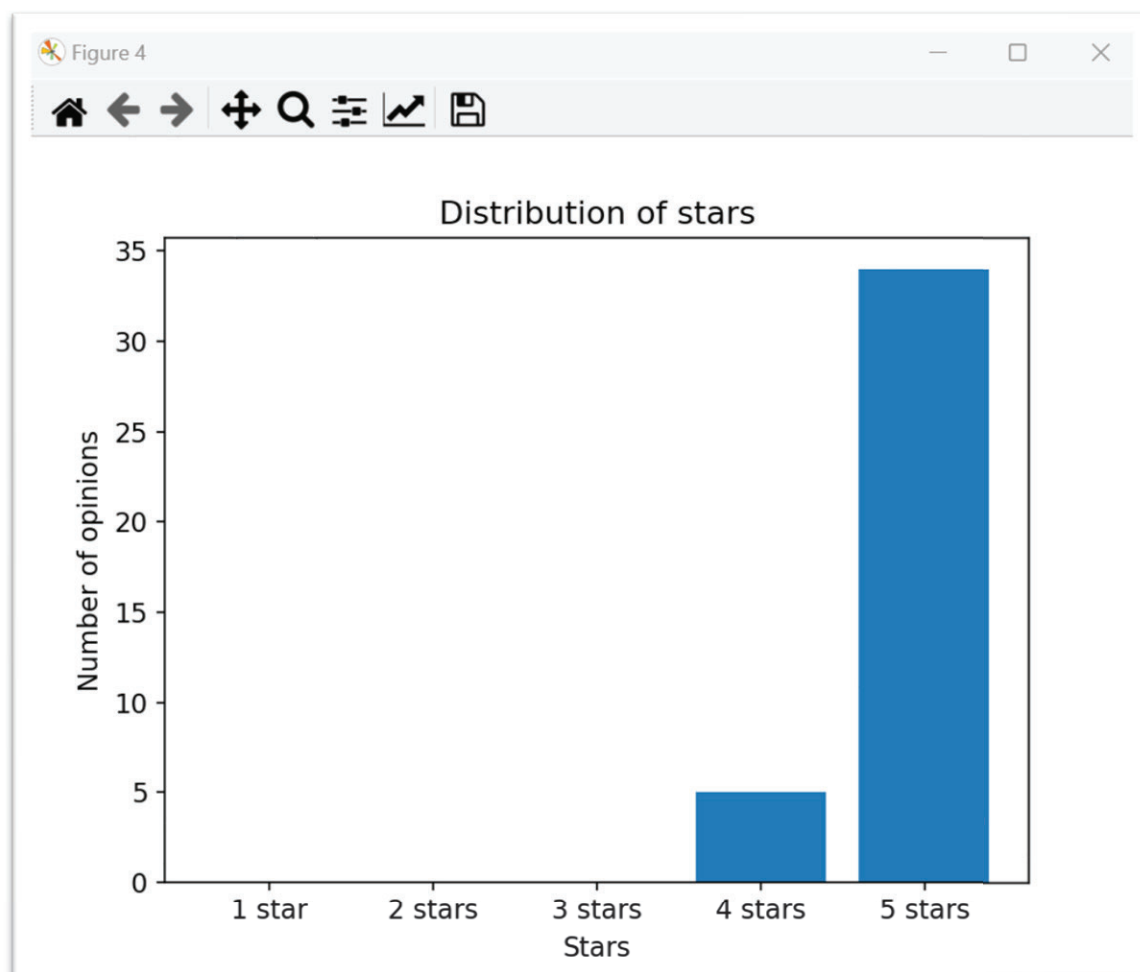


Ilustración 47. Gráfica, distribución de la puntuación.

En esta cuarta y última gráfica muestra un gráfico de barras con la distribución del grado de satisfacción del usuario plasmada en la encuesta.

11.3.3. Positions

Al presionar el botón de configuración de posiciones del brazo robótico aparece la siguiente ventana.

The screenshot shows a web interface for a robot named 'PREMO BEER'. The title is 'Robot positions by motion group'. There are two buttons: 'Save' (green) and 'Default' (red). Below these are tabs for different motion groups: Home, Go cup, Exit cup, Go beer, Mov 3P, Mov 0P, Exit beer, Exit table, Exit table 2, and Regulate flow. The 'Home' tab is selected, showing a table with columns: Pos, Speed, and Acceleration. The table has two rows: 'Home' and 'Gripper'. The 'Home' row has values: [79.229, 7.134, 5.386, -86.6, -130.8], 100.0, and 2000.0. The 'Gripper' row has the value 'OPEN'. Below the table, there is a note: 'All the robot's movements are Cartesian.' followed by two examples of accepted formats for movements and gripper states.

	Pos	Speed	Acceleration
Home	[79.229, 7.134, 5.386, -86.6, -130.8]	100.0	2000.0
Gripper	OPEN		

All the robot's movements are Cartesian.
Example of accepted format for movements: [179.0, 486.6, 379.8, -91.1, 0.4, 31.6]
Example of accepted format for Griper: 40

Ilustración 48. Pantalla de posiciones del brazo robótico.

En esta ventana, como en todas las ventanas de configuración de parámetros aparecen dos botones, debajo del botón “exit” y de ayuda:

- El botón “Save” se emplea para guardar los parámetros que se modifican en la tabla de parámetros.
- El botón “Default” se emplea para recuperar los parámetros por defecto iniciales del sistema.

Dado que el sistema posee 22 posiciones cartesianas del brazo robótico y 5 posiciones de la pinza, se organizan los movimientos en 10 pestañas, cada una de ellas posee una tabla con un grupo de movimientos. Cada movimiento está enumerado y brevemente descrito.

La tabla consta de 4 columnas, la primera de ellas se emplea para enumerar y nombrar el movimiento, la segunda columna es la posición, tenga en cuenta que en caso de un movimiento cartesiano posee formato de lista y en caso de una posición de la pinza posee formato de número flotante. La tercera y cuarta columna se tratan de la velocidad y de la aceleración del movimiento, este solo es necesario para los movimientos cartesianos, siendo su formato flotante.

No se recomienda modificar los valores de velocidad y aceleración debido a en desplazamientos cortos, una excesiva velocidad y aceleración puede provocar que salte el colisionador del brazo robótico.

El caso de querer ajustar una posición, se puede modificar el valor de la posición, teniendo en cuenta que los valores de la lista son las coordenadas cartesianas del brazo robótico y sus unidades son los mm los 3 primeros elementos y los grados los otros 3 elementos: [Eje x, Eje y, Eje z, Roll (Balanceo), Pitch (Cabeceo), Yaw (Guiñada)].

Es muy importante respetar el formato de todos los datos.

11.3.4. Computer Vision

Al presionar el botón de configuración de los parámetros de visión computacional aparece la siguiente ventana.

	Value
ROI 1	(251, 200, 17, 12)
ROI 2	(271, 139, 18, 16)
ROI 3	(336, 419, 22, 16)
ROI 4	(350, 389, 22, 21)
Threshold	20
Threshold 2	6

Ilustración 49. Pantalla parámetros de visión computacional.

En la tabla se puede modificar las posiciones las regiones de interés, pero también se puede volver a seleccionarlos de forma manual con el ratón cada una de las regiones de interés, presionando su respectivo botón azul, al presionarlo aparecerá una ventana emergente con una captura de la cámara, se deberá en dicha captura seleccionar la región de interés deseada y darle al botón “Enter” del teclado para guardar la nueva región de interés (ROI).

Se recomienda que las “ROI” sean lo más pequeña y precisa posible y se evite zonas reflectantes de luz o con efecto espejo.

Para el detector de llenado son necesarios 2 parámetros el “Threshold” que es la diferencia en escala de grises de tonalidad necesaria entre una imagen de una ROI vacía de referencia y otra imagen de la misma ROI para que se considere que se ha producido un cambio y por tanto, que se ha alcanzado el límite

marcado por el ROI y el “Theshold 2” es la cantidad de píxeles que debe cambiar para que se considere que se ha alcanzado el límite, este valor depende el tamaño de la región de interés, siempre debe ser mayor de 0 y nunca superar el número de elementos de la matriz ROI.

El sistema será más sensible para detectar que se ha alcanzado el límite marcado por la ROI, cuanto más pequeños sean los valores “Theshold” y “Theshold 2”.

El valor de “Theshold” dependerá de la diferencia de tonalidades en escala de grises entre el fondo del vaso transparente vacío y el vaso con el líquido a detectar.

El valor de “Theshold 2” depende de ruido de la imagen, dado que se utiliza para evitar falsas detecciones, por ejemplo, los producidos por reflejos.

11.3.5. Artificial Intelligence

Al presionar el botón de configuración de los parámetros de Inteligencia artificial aparece la siguiente ventana.

	Value
Lower time limit duration of the complete system	50.0
Upper time limit duration of the complete system	80.0
Lower time limit duration of open tap (including foam time)	5.0
Upper time limit duration of open tap (including foam time)	20.0
Lower filling limit time 1	3.0
Upper filling limit time 1	10.0
Lower filling limit time 2	1.0
Upper filling limit time 2	5.0
Lower filling limit time 3	2.0
Upper filling limit time 3	6.0
Lower filling limit time 4	1.0
Upper filling limit time 4	3.0

Please respect data formatting

Ilustración 50. Pantalla parámetros de Inteligencia artificial.

En dicha ventana aparece, una tabla con el valor máximo y mínimo de duración del sistema y de los distintos tramos.

Cuando dichas duraciones estén fuera del rango establecido, el sistema corregirá automáticamente los parámetros de entrada como el grado de apertura del grifo del dispensador y/o el grado de apertura del compensador grifo del dispensador, para que las distintas duraciones permanezcan dentro del rango introducido.

Las duraciones medidas son:

- Duración del sistema completo: mide el tiempo que se tarda en servir una cerveza.
- Duración que permanece el grifo abierto: es el tiempo que tarda en llenarse el vaso al completo.

- Duración del límite 1: es el tiempo que tarda en llenarse el vaso hasta el límite establecido por la ROI 1, este proceso de llenado se realiza con el vaso a 45 grados con la horizontal.
- Duración del límite 2: es el tiempo que tarda en llenarse el vaso hasta el límite establecido por la ROI 2, este proceso de llenado se realiza con el vaso a 30 grados con la horizontal.
- Duración del límite 3: es el tiempo que tarda en llenarse el vaso hasta el límite establecido por la ROI 3, este proceso de llenado se realiza con el vaso a 0 grados con la horizontal, es decir, verticalmente.
- Duración del límite 4: es el tiempo que tarda en llenarse el vaso con el grifo en posición de crema de cerveza hasta el límite establecido por la ROI 4, este proceso de llenado se realiza con el vaso verticalmente.

11.3.6. Parameters

Al presionar el botón de configuración de los parámetros aparece la siguiente ventana.



	Value
Camera Index	0
Port Arduino	COM3
IP Bot	192.168.1.203
Initial position of the flow regulator	50.0
Initial position of tap opening	20.0

Please respect data formatting: 'COM4' and '192.168.1.203'

Ilustración 51. Pantalla parámetros de inicio.

En dicha ventana aparece, una tabla con los parámetros iniciales del sistema.

Estos parámetros son:

- Índice del puerto cámara, por defecto es 0, pero si se desea añadir otra cámara, se puede modificar.
- Puerto “COM” al que está conectado el Arduino, este cambia, en función del Arduino utilizado.
- Dirección IP del brazo robótico, si se desea utilizar otro brazo robótico u otro controlador, se debe cambiar por la dirección IP del nuevo controlador.
- Grado de apertura inicial del compensador del grifo del dispensador, este valor está ponderado del 0 al 100%, siendo 0%, completamente cerrado y 100% completamente abierto. Se recomienda que sea 20% y que el

propio sistema de inteligencia artificial lo configure en función de la presión del barril.

- Grado de apertura inicial del grifo del dispensador, este valor está ponderado del 0 al 100%, siendo 0%, completamente cerrado y 100% completamente abierto. Se recomienda que sea 20% y que el propio sistema de inteligencia artificial lo configure en función de la presión del barril.

11.3.7. Start Demo

Al presionar el botón, el sistema se configura según los parámetros introducidos en la tabla de los parámetros iniciales del sistema.

Por lo que se activará el brazo robótico que se moverá hacia el dispensador para actuar sobre el compensador del grifo, colocándolo en la posición deseada la función del porcentaje introducido en la tabla de los parámetros iniciales.

Mientras el robot se esté moviendo, en la pantalla aparecerá el logotipo del sistema y una barra de progreso, debajo de ella aparecerá mensajes informando del estado del sistema.

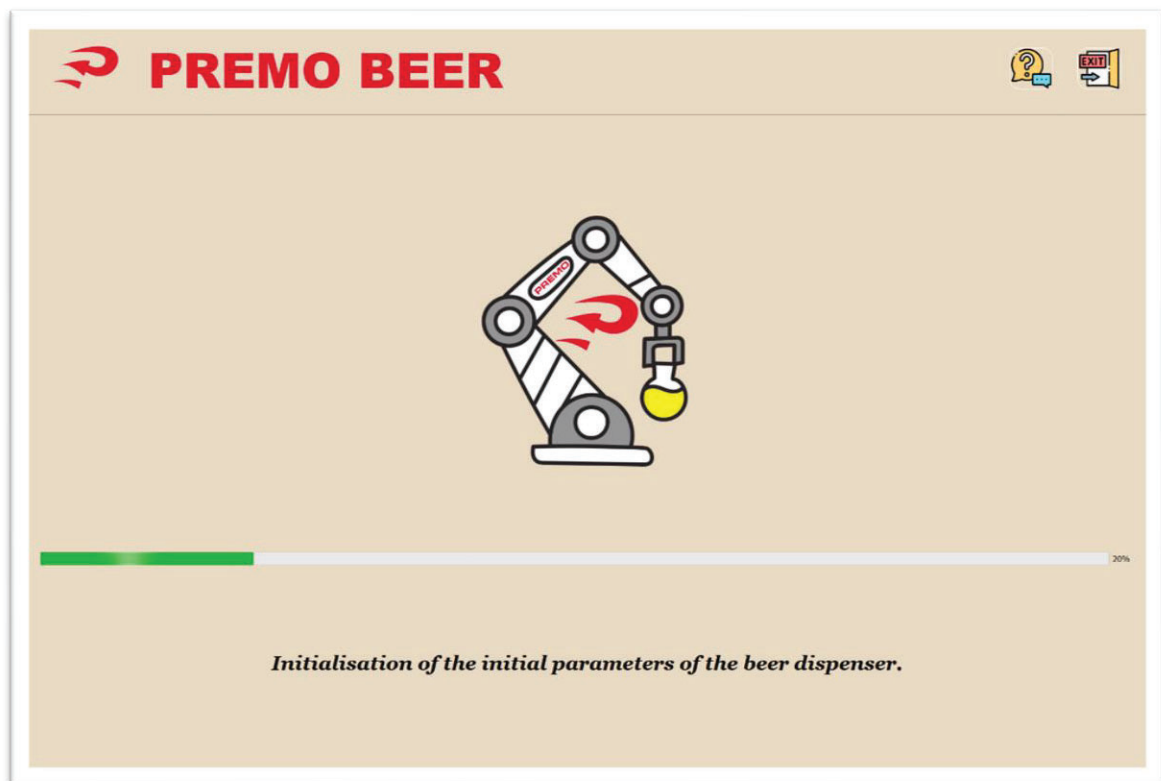


Ilustración 52. Pantalla de espera.

Al finalizar el brazo robótico, volverá a la posición cero y aparecerá la ventana demo.

11.4. Pantalla demo:

Esta pantalla posee un único botón cuyo icono es el logotipo del sistema, que al presionarlo dará inicio al proceso de servir una cerveza.

Además de dicho botón aparecen los botones de ayuda y “Exit” presentes en todas las pantallas.



Ilustración 53. Pantalla demo.

Una vez presionado dicho botón aparecerá la ventana servir cerveza.

11.5. Pantalla servir cerveza:

Esta pantalla consta de una barra de progreso y justo debajo de ella aparecerán mensajes informando del estado del sistema.



Ilustración 54. Pantalla de espera mientras se sirve la cerveza.

El proceso de servir una cerveza consta de los siguientes pasos:

- 1- Abre una ventana emergente donde se muestra el video en directo de la Cámara.
- 2- Apertura del dispensador de vasos.
- 3- Bloque de movimientos del brazo robótico para sujetar el vaso.
- 4- Bloque de movimientos del brazo robótico para sacar el vaso del dispensador de vasos.
- 5- Captura de la imagen para crear un dataset (conjunto de imágenes) y comprobación mediante una red neuronal basada en YOLO v8 de la presencia o no del vaso dentro de la pinza del brazo robótico.

En el caso de que no se detecte la presencia a vaso, el sistema se detendrá y aparecerá la pantalla de “Error no cup”, en el caso contrario,

- es decir, si se detectara la presencia de vaso dentro de la pinza, continuara el sistema el proceso.
- 6- Cerrar dispensador de vasos.
 - 7- Bloque de movimientos del brazo robótico para acercarse al dispensador de cerveza.
 - 8- Descarte: se abrirá el grifo de cerveza un segundo antes de iniciar en movimiento de colocar el vaso a 45 grados dentro del chorro de líquido de salida del grifo.
 - 9- Inicio de la función de llenado del límite 1 (Primer tiempo): una vez superado el límite de llenado 1, finaliza la función (permaneciendo el grifo abierto). Si la función permanece operativa 3 veces el límite máximo de duración del llenado 1, se cerrará el grifo y se activará la ventana "Error no beer".
 - 10- Movimiento del brazo robótico para colocar el vaso a 30 grados.
 - 11- Inicio de la función de llenado del límite 2: una vez superado el límite de llenado 2, finaliza la función (permaneciendo el grifo abierto). Si la función permanece operativa 2 veces el límite máximo de duración del llenado 2, se cierra la función (cerrándose el grifo).
 - 12- Movimiento del brazo robótico para colocar el vaso verticalmente.
 - 13- Inicio de la función de llenado del límite 3: una vez superado el límite de llenado 3, finaliza la función, cerrándose el grifo. Si la función permanece operativa 2 veces el límite máximo de duración del llenado 3, se cierra la función (cerrándose el grifo).
 - 14- Inicio de la función de llenado del límite 4 (Segundo tiempo): en esta función la apertura del grifo será en la posición de crema de cerveza, una vez superado el límite de llenado 4, finaliza la función, cerrándose el grifo. Si la función permanece operativa 2 veces el límite máximo de duración del llenado 4, se cierra la función (cerrándose el grifo).
 - 15- Bloque de movimientos del brazo robótico para colocar el vaso en la bandeja de salida.
 - 16- Bloque de movimientos del brazo robótico para soltar el vaso y posicionarse para enfocar el vaso con la Cámara.
 - 17- Captura de la imagen para crear un dataset con los resultados.

- 18-Recopilación de datos obtenidos del sistema, almacenamiento de datos tanto en una base de datos en la nube, como en el propio PC.
- 19-Análisis de los datos y corrección de los parámetros del sistema. En caso de ser necesario modificar el parámetro del compensador, el brazo robótico iniciará la maniobra para colocarlo en la nueva configuración.
- 20-Bloque de movimientos del brazo robótico para volver a la posición cero del sistema.
- 21-Ocultar la ventana emergente donde se muestra el video en directo de la Cámara.
- 22-Finaliza la función, apareciendo la pantalla de encuesta.

En caso de que se presione el botón de emergencia, o salte el colisionador del brazo robótico, se detendrá el proceso, cerrándose el grifo y apareciendo la ventana “Colisión del robot”.

11.6. Pantalla de encuesta.

Una vez terminada de servir la cerveza, aparecerá en pantalla una ventana con una pequeña encuesta, para que el usuario pueda valorar su experiencia y el sistema obtenga una realimentación.

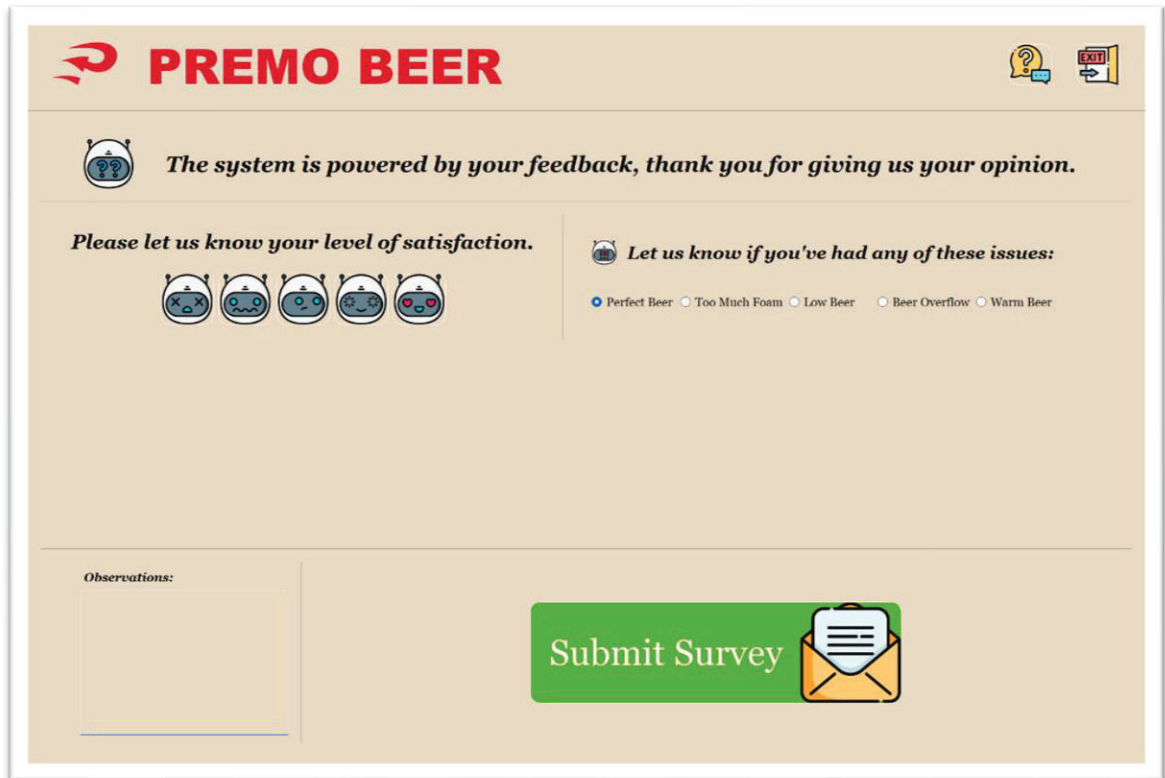


Ilustración 55. Pantalla de encuesta.

Esta encuesta cuenta con 3 preguntas opcionales:

- Valoración de la satisfacción del usuario con el sistema: la puntuación es del 1 al 5, siendo 1, poco satisfecho, representado por un emoticono con los ojos en forma de x y siendo 5, muy satisfecho, representado por un emoticono con corazones en los ojos.
- Informe de posibles fallos: se le pide al usuario que marque uno de los errores más comunes que aparecen en la lista en caso de que se produzca.

Estos errores son, Exceso de espuma, poca cerveza, desbordamiento de la cerveza o cerveza caliente. En caso de no producirse ninguno de

estos errores, se marca la opción de cerveza perfecta (opción marcada por defecto).

- La encuesta también cuenta con la opción de introducir un comentario. Como está pensado para ejecutarse en una Tablet táctil, al tocar una vez en la caja de texto se abrirá el teclado en pantalla, para escribir con mayor facilidad.

Una vez rellenada la encuesta se puede presionar el botón “Submit Survey” para enviar la encuesta y volver a la pantalla de demo, y poder pedir otra cerveza.

En caso de que no se desee rellenar la encuesta, se puede presionar el botón “Submit Survey” o “Exit”, sin necesidad de contestar a ninguna pregunta y poder salir de dicha pantalla.

11.7. Pantalla de Error: Vaso no detectado:

Esta pantalla de error aparece cuando se detecta que se ha producido un error al dispensar el vaso, por lo que la pinza no cogió ningún vaso, puede producirse por diversos motivos:

- Ausencia de vasos en el dispensador de vasos.
- Mal funcionamiento del dispensador de vasos: Se debe comprobar que el motor lineal del dispensador de vasos está alimentado a 12 voltios y conectado al Arduino correctamente.
- Deslizamiento de la pinza: este se produce cuando los vasos están muy pegados entre ellos y la pinza a pesar de cogerlo bien, no es capaz de desengancharlo y la pinza se resbala por la superficie deslizante del vaso de plástico.

Si este error se produce reiteradamente, comprobar que esté bien posicionado el dispensador de vasos y si es necesario modificar las posiciones de brazo robótico, aumentando el cierre de la pinza (“Gripper 2” en la tabla de posiciones “Exit cup”) para ejercer más fuerza sobre el vaso.

También se puede modificar la posición 4 del brazo robótico, situado en la tabla “Exit cup”, para provocar más torsión en el vaso.

La detección de este error se realiza mediante una red neuronal convolucional customizada para la detección en específico de un tipo de vaso de plástico transparente, dentro de unas pinzas específicas que lleva el robot, basada en YOLO v8.

El cambio de tipo de vaso o tipo de pinza pueden provocar que la red neuronal convolucional customizada necesite actualizarse, con un nuevo dataset.



Ilustración 56. Pantalla Error, vaso no detectado.

Una vez el sistema detecté que no hay vaso dentro de la pinza del brazo robótico, el sistema se detiene y aparece el error en pantalla.

Además del error, aparece un botón de “reset” simbolizado con un engranaje y dos flechas circulares. Al presionarlo el brazo robótico volverá a la posición cero de forma controlada, se cerrará el dispensador de vasos y volverá a aparecer la pantalla Demo.

11.8. Pantalla de Error: Cerveza no detectada:

Este error se produce cuando el tiempo de llenado del primer tramo del vaso supera 3 veces el tiempo máximo de dicho tramo. Se puede dar por ausencia de cerveza o por un flujo demasiado lento.



Ilustración 57. Pantalla error, cerveza no detectada.

Una vez el sistema detecté dicho error, el sistema se detiene, se cierra el grifo de cerveza y aparece el error en pantalla.

Además del error, aparece un botón de “reset” simbolizado con un engranaje y dos flechas circulares. Al presionarlo, el brazo robótico llevará el vaso con cuidado hasta la bandeja de entrega de cerveza, dado que es posible que contenga algo de líquido. Una vez dejado el vaso, el sistema analizará porque se ha podido producir dicho fallo y actuara en consonancia.

Las posibles causas son pueden ser:

- Mala configuración del dispensador de cerveza, el sistema modificará los parámetros del dispensador, ya sea el grado de apertura del grifo o del compensador.
- Si dichos parámetros del dispensador están al máximo (100%), el error se debe a que el barril de cerveza se ha terminado y está vacío, por lo que el sistema mandara un mensaje de advertencia para que se reemplace el barril por uno nuevo.

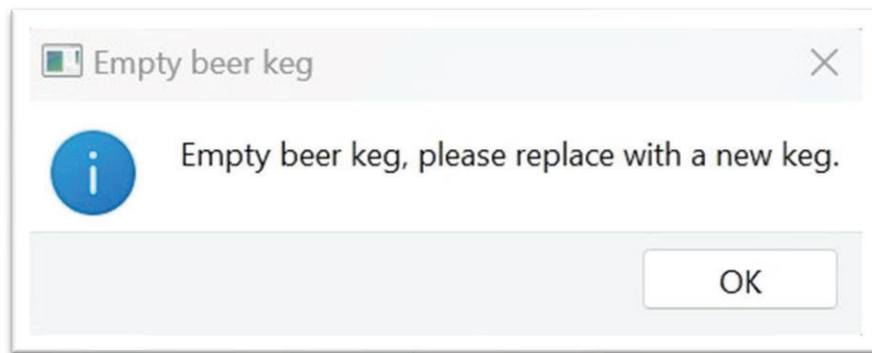


Ilustración 58. Advertencia, barril de cerveza vacío.

Una vez aplicado las soluciones, el brazo robótico volverá a la posición cero de forma controlada y volverá a aparecer la pantalla Demo.

11.9. Pantalla de Error: Error en el Brazo:

Este error se produce al detectar el error de parada de emergencia del brazo robótico, este se puede deber a 3 motivos:

- Se ha presionado la seta de emergencia: Antes de reiniciar el sistema, se debe rearmar la seta de emergencia.
- Se ha detectado una colisión del brazo robótico: si este error se reitera sin motivo aparente, se debe bajar el grado de sensibilidad ante colisiones del brazo robótico.
- Por un exceso de velocidad, este error se da en movimientos cortos con mucha velocidad y aceleración, por lo que la solución sería disminuir la velocidad y la aceleración en esos movimientos o modificar el movimiento para que se produzca mayor desplazamiento.

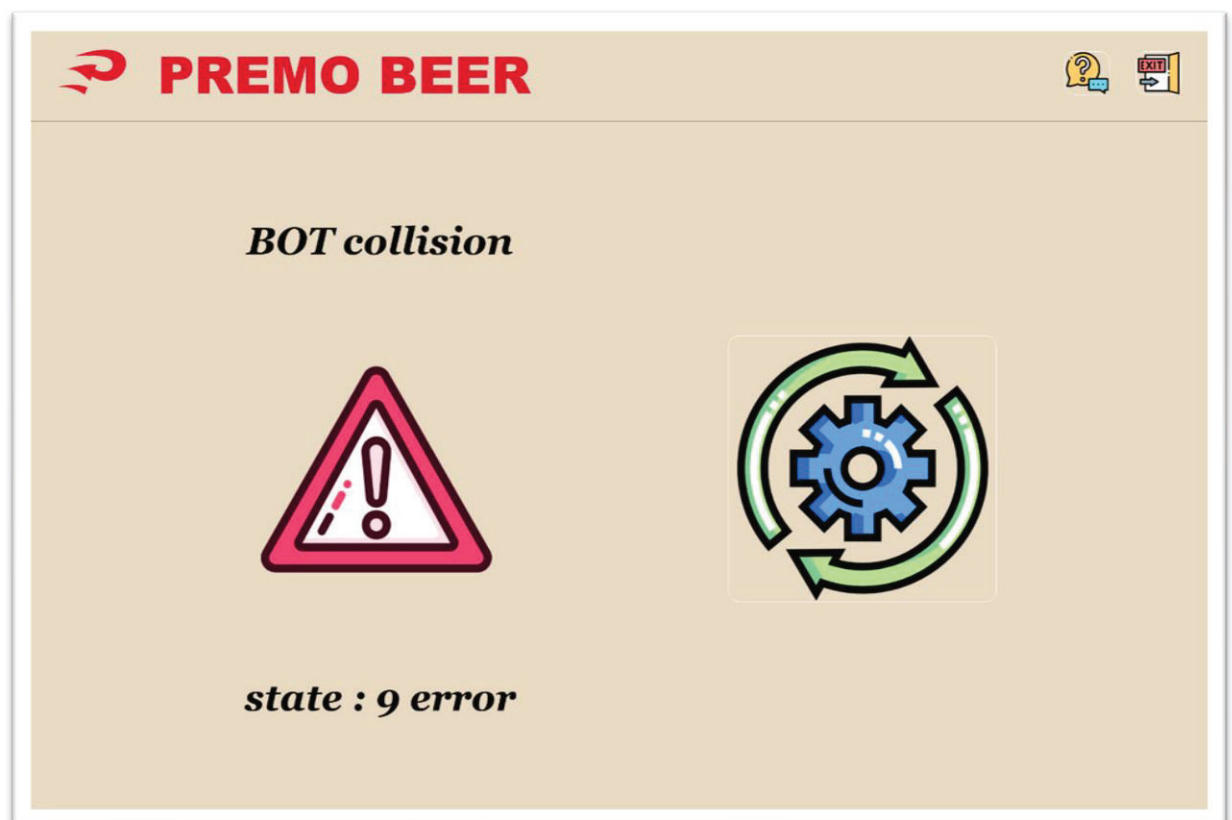


Ilustración 59. Pantalla error, colisión del robot

Una vez el sistema detecté dicho error, el sistema se detiene, se cierra el grifo de cerveza y aparece el error en pantalla.

Además del tipo de error, se muestra el código del movimiento en el que se ha producido el error, por lo que el sistema conoce la posición donde se ha detenido el brazo robótico, diferenciando entre posiciones con líquido en el vaso y sin líquido en el vaso, esto se debe a que al presionar el botón de “reset” (simbolizado con un engranaje y dos flechas circulares) el brazo robótico se rearma y vuelve a la posición cero del brazo robótico, pudiendo seguir 2 trayectorias:

- Sin líquido en el vaso: vuelve a la posición 0, realizando los mismos movimientos, pero en sentido contrario, “Volviendo sobre sus pasos”.
- Con líquido en el vaso: coloca primero el vaso en la bandeja de entrega y a continuación va a la posición cero.

Una vez el brazo robótico vuelve a la posición cero, aparece la pantalla demo para volver a pedir una cerveza.

12. Prueba de funcionamiento.

La prueba de funcionamiento se realiza durante 3 días, dejando que los usuarios interaccionen con la interfaz y completando la encuesta de satisfacción una vez servida la bebida. Además, se forzó a que el sistema cometiese errores como colisiones o falta de vasos, y así comprobar si el sistema detectaba errores.

Durante la prueba de funcionamiento, el sistema sirvió un total de 112 vasos de cerveza. Obteniendo un muy buen grado de satisfacción de los usuarios, tal como se puede apreciar en el gráfico de distribución de la puntuación obtenida de la encuesta de satisfacción.

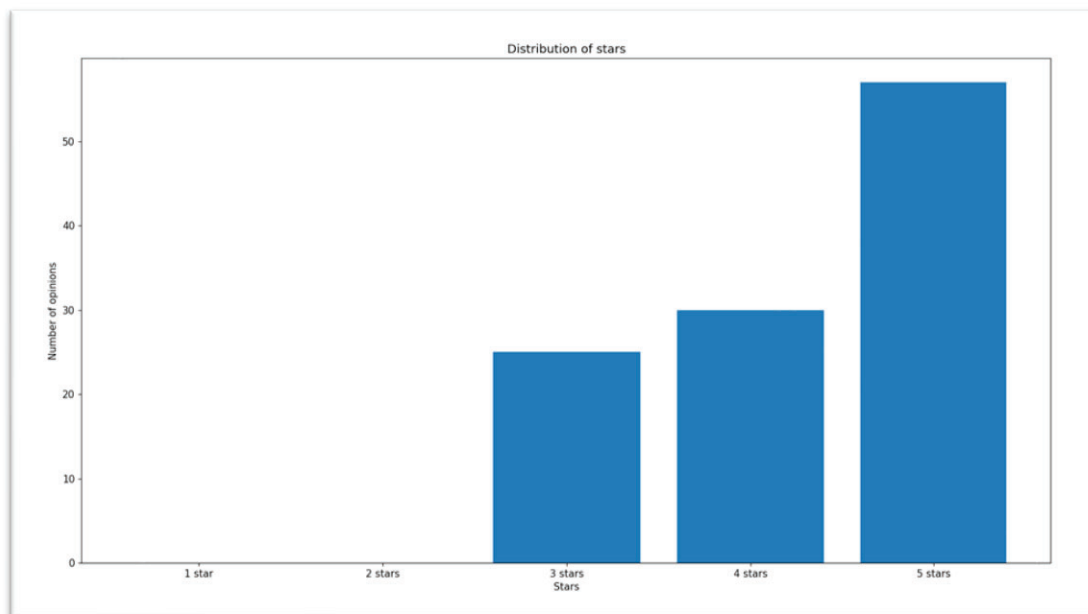


Ilustración 60. Grafico de satisfacción del usuario.

Los datos se obtienen de la encuesta de satisfacción del usuario, donde se pide que valore del 1 al 5, la satisfacción con la bebida servida.

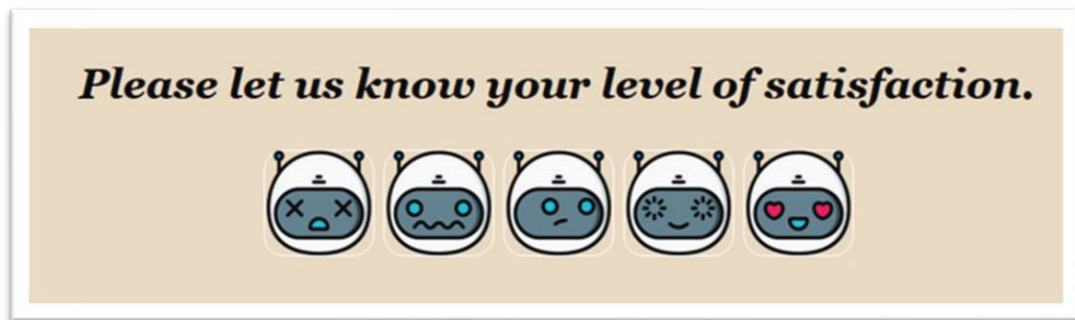
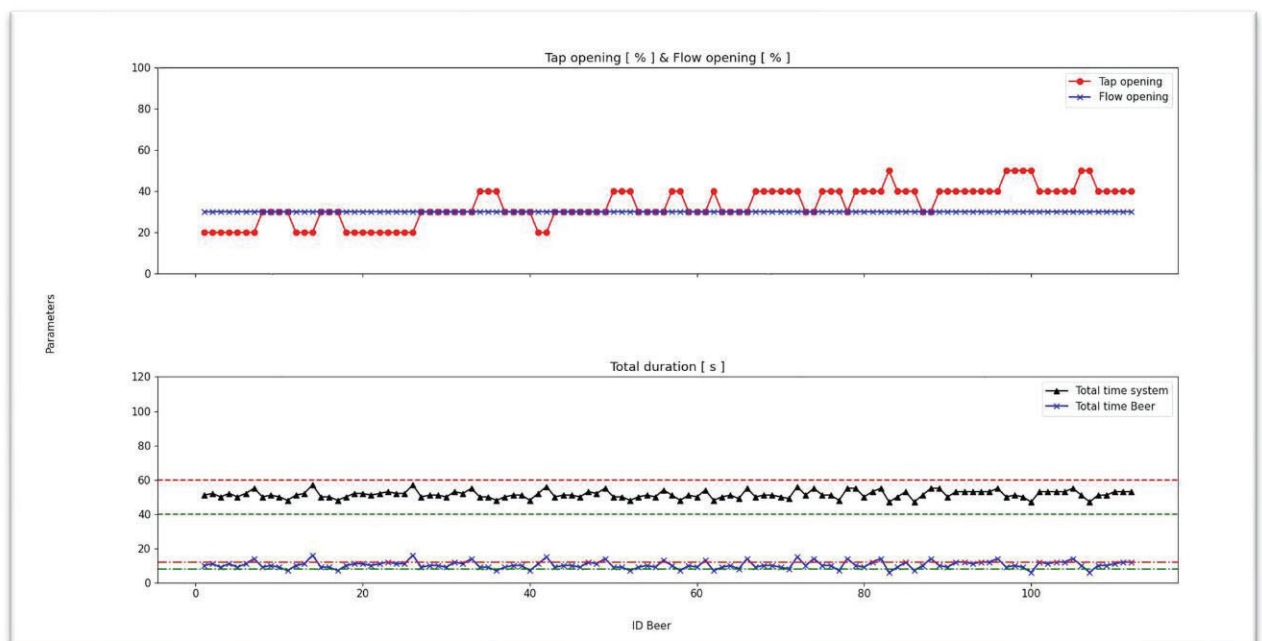


Ilustración 61. Encuesta de satisfacción.

A continuación, se muestran gráficos de funcionamiento del sistema:



En el eje X, se representa el número de identificación de cada bebida servida.

En la gráfica superior se representa la evolución de los parámetros de entrada, la apertura del grifo y apertura del compensador, representado en tanto por cien.

En la gráfica inferior se representan la evolución de las duraciones total del sistema y la del grifo abierto, en segundos.

Como se puede apreciar en la gráfica inferior, la línea negra, que representa la duración del sistema completo, la duración siempre se mantiene por debajo del minuto, siendo la duración más alta de 57 segundos y la mínima de 47 segundos.

En la línea azul del gráfico inferior, representa la duración del grifo abierto para cada servicio, en ella se puede apreciar que cuando se rebasa algún límite en esta línea, se produce una corrección en los parámetros de entrada para el siguiente servicio, para corregir el sistema y que la duración vuelva a estar en rango correcto.

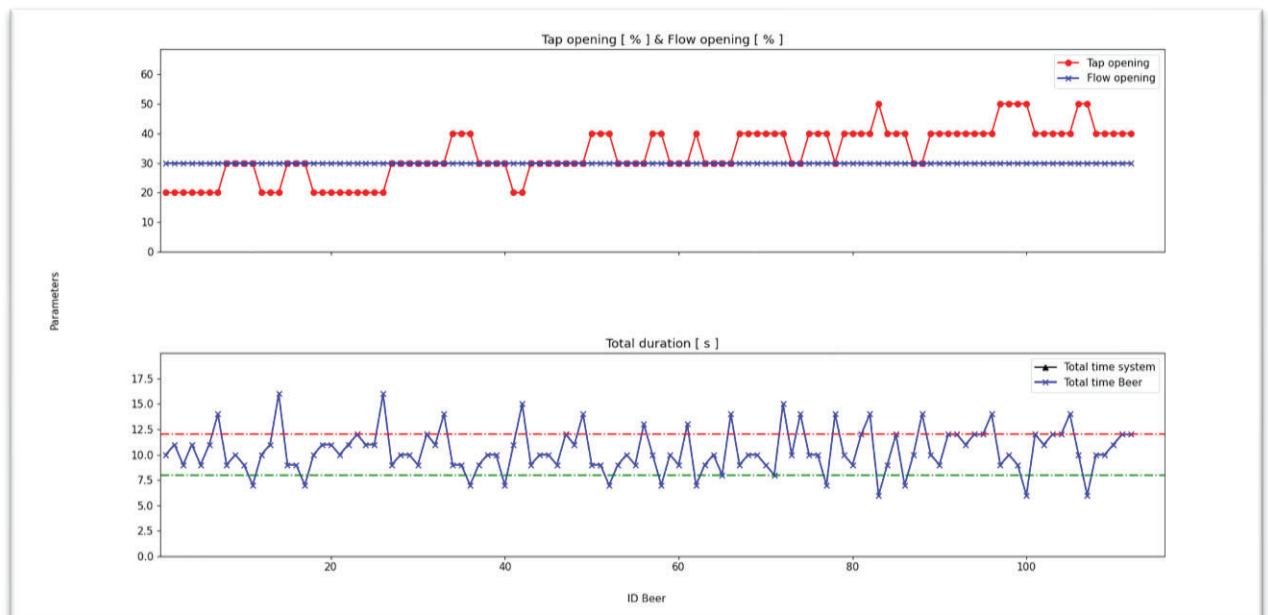


Ilustración 62. Gráfico de corrección del sistema.

A continuación de muestra el gráfico de la duración de cada sección de llenado para cada servicio, en él se puede apreciar que cuando se supera el límite algún límite se produce una corrección en el sistema que hace que vuelva a estar dentro del rango correcto.

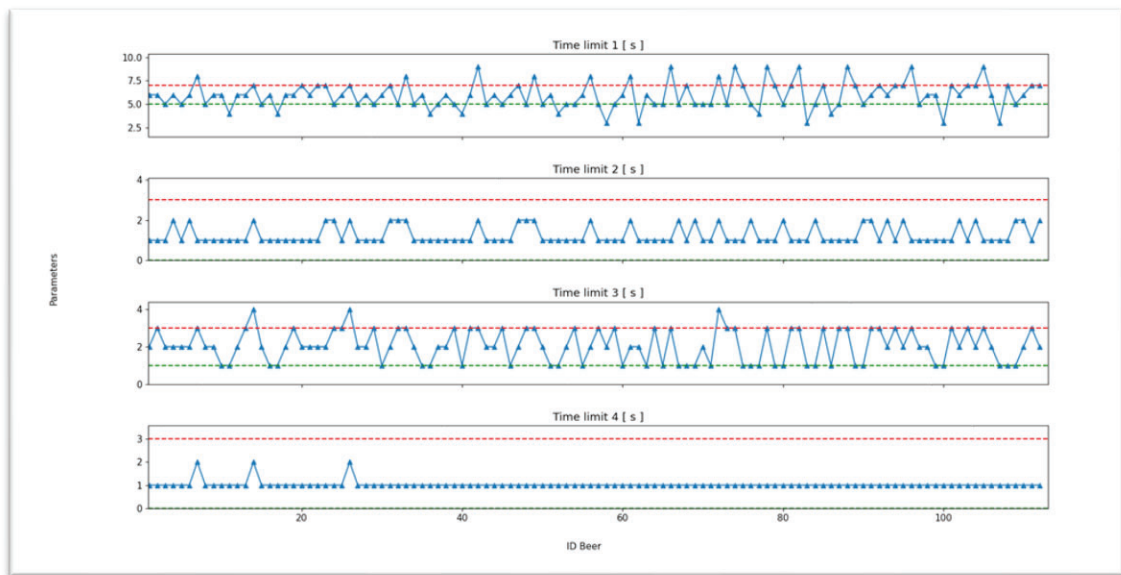


Ilustración 63. Gráficos de la duración de cada sección de llenado.

A continuación, se muestran algunas fotografías tomadas por el sistema de cada cerveza.



Ilustración 64. Bebidas servidas.

Por último, se muestran la detección del vaso mediante una red neuronal convolucional (CNN).



Ilustración 65. Imagen del vaso dentro de la pinza.

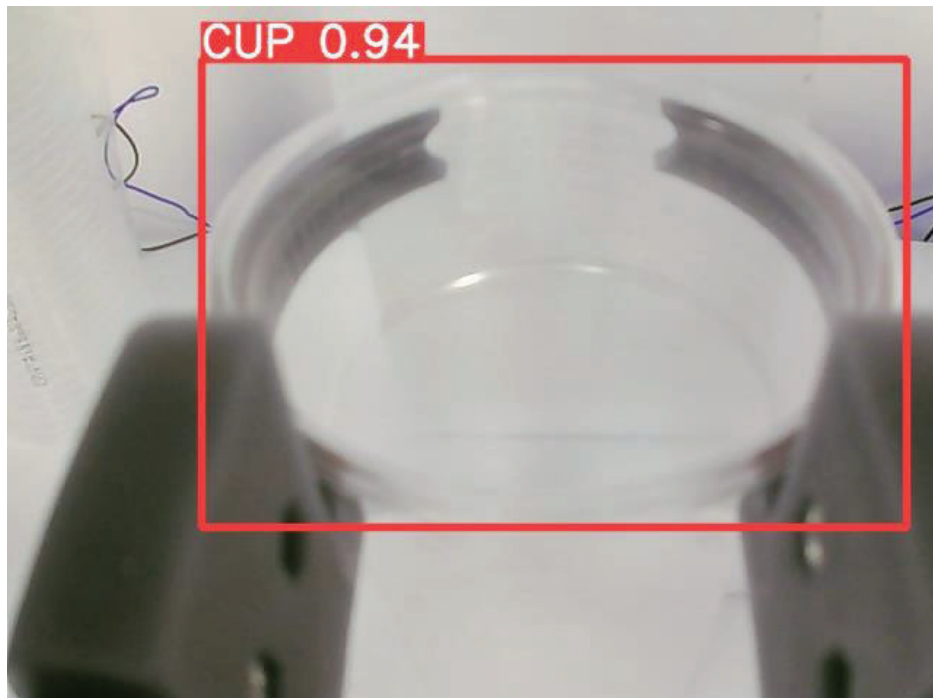


Ilustración 66. Vaso detectado mediante una CNN.

13. Conclusión.

Tras la prueba de funcionamiento, se ha comprobado que es posible controlar un dispensador fluctuante debido a la presión mediante un sistema de visión artificial.

Se ha comprobado la gran eficiencia de las redes neuronales convolucionales para la detección de objetos difíciles, como un vaso transparente.

La red neuronal fue capaz de detectar el vaso transparente y de no detectarlo cuando no estaba presente, acertando el 100% de las veces.

En contra, se ha observado que el uso de redes neuronales en video streaming (directo) ralentiza el sistema, provocando que el sistema se detenga durante 3 a 5 segundos para la detección del vaso. Dado que se trata de un sistema de seguridad y de eficiencia, ya que proceder a servir la bebida sin vaso podría causar muchos daños y un desperdicio de recursos, se considera que el tiempo de espera es aceptable.

Debido a este tiempo de latencia, se optó por la tecnología de umbralización para la detección de llenado, consiguiendo una reacción inmediata a la detección de llenado.

El uso de multihilos solucionó muchos errores, puesto que se perdía información crucial al producirse cualquier error. Y permitió que los distintos programas se comunicaran entre ellos, y que cuando uno de ellos falla, no afecta a los demás.

La base de datos tuvo un papel importante en el control del sistema, debido a que a partir del histórico de los datos se pudo diseñar límites de control para la toma de decisiones y el control del flujo del dispensador mediante el compensador.

Se evita manipular la palanca del grifo, en vista de que un aumento excesivo de su apertura provoca un aumento significativo de la velocidad, y un aumento de esta provoca un exceso de espuma y un tiempo menor de reacción del sistema de visión artificial.

La creación de una interfaz intuitiva y que dispone de secciones para la personalización de todos los datos de entrada del sistema, incluyendo las posiciones del brazo robótico, facilita mucho la puesta a punto del sistema, dado que, si es necesario hacer un cambio, ya sea una posición de brazo robótico, un límite de configuración o un cambio de robot, se puede realizar dentro de la interfaz, sin tener que modificar el código fuente de Python.

Las impresoras 3D para la fabricación de prototipos como el dispensador de vasos fue una herramienta muy útil, gracias a que permitió la fabricación de piezas personalizadas de forma rápida y precisa.

Se concluye que es posible crear un sistema que manipule y controle un dispensador de bebidas carbonatadas mediante un compresor propio que provoca fluctuaciones irregulares en el flujo de la bebida, mediante un sistema robotizado, utilizando un brazo robótico colaborativo guiado por un sistema de visión artificial usando tanto redes neuronales convoluciones como umbralización.

14. Bibliografía

- Bootcamp AI. (2020). *Intro a las redes neuronales convolucionales*. Obtenido de <https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8>
- D E Briggs, P. A. ((2004)). *Brewing: science and practice*. Elsevier.
- Dassault Systèmes. (17 de febrero de 2023). *Mecanizado CNC, ¿qué es y cómo funciona?*. Obtenido de <https://www.3ds.com/es/make/guide/process/cnc-machining>
- Harris, C. R. (s.f.). *Array programming with NumPy*. Obtenido de <https://doi.org/10.1038/s41586-020-2649-2>
- Hunter, J. D. (2023). *Matplotlib: A 2D graphics environment. Computing in Science & Engineering*. Obtenido de <https://doi.org/10.1109/MCSE.2007.55>
- Joseph Redmon, S. D. (s.f.). You Only Look Once: Unified Real-Time Object Detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 779-788).
- Kelleher, J. D. (2018). *Deep learning and neural networks: concepts, methodologies, tools, and applications*. CRC Press.
- OpenCV. (s.f.). *Open source computer vision library*. Obtenido de <https://opencv.org/>
- Peck, J. (15 de enero de 2023). *How does a Resin 3D Printer Work?*. Obtenido de All3DP: <https://all3dp.com/2/how-does-a-resin-3d-printer-work/>
- PyQt5. (2023). *The Qt Company*. Obtenido de <https://www.riverbankcomputing.com/software/pyqt/>
- PySerial. (2023). *PySerial*. Obtenido de <https://pyserial.readthedocs.io/en/latest/pyserial.html>

Redmon, J. &. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv.

Redmon, J. D. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition.

The Best Dispensing. (1 de abril de 2023). Obtenido de Cómo solucionar los 8 principales problemas en tu dispensador de cerveza: <http://www.thebestdispensing.com/post/89/39/como-solucionar-los-8-principales-problemas-en-tu-dispensador-de-cerveza>

Ultralytics. (13 de abril de 2023). *Ultralytics YOLOv8 Docs*. Obtenido de <https://docs.ultralytics.com/>

15. Anexos

15.1. Código de programación.

15.1.1. Importación de todas las librerías necesarias (library_imports.py)

```
"""
# File name: library_imports
# Developer: Maher Bouymejan
# Contact: maherbouymejan@gmail.com
# Creation date: 10/04/2023
# Description: Import all the libraries need for the system.
# Python version: 3.9.0
# Version: 0
"""

#Library to control the time
import time
import datetime

#Library to create the artificial vision
import cv2 #pip install opencv-python

#Library for numerical calculations
import numpy as np
import math

#Library to communicate with the microcontroller via serial port
import serial #pip install pyserial

import copy

#Library for running programs in parallel
import threading

#Library of the cobot xArm
"""
# xArm-Python-SDK: https://github.com/xArm-Developer/xArm-Python-SDK
# git clone git@github.com:xArm-Developer/xArm-Python-SDK.git
# cd xArm-Python-SDK
# python setup.py install
"""

try:
    from xarm.tools import utils
except:
    pass
from xarm import version
```

```

from xarm.wrapper import XArmAPI

#Library to interact with the Python interpreter and the operating system.
import sys
import os

#Library for working with error traces
import traceback

#Library to generate pseudo-random numbers
import random

#YOLO v8
from ultralytics import YOLO #pip install ultralytics

#Library to create the user interface
import tkinter as tk
from tkinter import *
from PIL import Image, ImageTk
#QT5 for GUI pip install pyqt5
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QMainWindow, QApplication, QTableWidgetItem
from PyQt5.uic import loadUi
from PyQt5.QtGui import QDesktopServices, QImage, QPixmap
from PyQt5.QtCore import Qt, QUrl, QTimer

import ast

#Library for working with databases
import pymongo #pip install pymongo

#Library to create graphs

import matplotlib #pip install matplotlib
import matplotlib.pyplot as plt
matplotlib.use('Qt5Agg') # o TkAgg

```

15.1.2. Control del robot (Cobot.py)

```
"""
# File name: Cobot
# Developer: Maher Bouymejan
# Contact: maherbouymejan@gmail.com
# Creation date: 10/04/2023
# Description: Create functions to control the cobot.
# Python version: 3.9.0
# Version: 0
"""

from library_imports import *

class Robot_Arm():
    def __init__(self, IP) -> None:
        self.IP = IP
        self.Inicializacion()

    def Inicializacion(self):

        def pprint(*args, **kwargs):
            try:
                stack_tuple = traceback.extract_stack(limit=2)[0]
                print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ' '.join(map(str, args))))
            except:
                print(*args, **kwargs)

        pprint('xArm-Python-SDK Version:{}'.format(version.__version__))

        self.arm = XArmAPI(self.IP)
        self.arm.clean_warn()
        self.arm.clean_error()
        self.arm.motion_enable(True)
        self.arm.set_mode(0)
        self.arm.set_state(0)
        time.sleep(1)

        variables = {}
        self.params = {'speed': 300, 'acc': 6000, 'angle_speed': 20, 'angle_acc':
500, 'events': {}, 'variables': variables, 'callback_in_thread': True, 'quit': False}

        # Register error/warn changed callback
        def error_warn_change_callback(data):
            if data and data['error_code'] != 0:
                self.params['quit'] = True
                pprint('err={}, quit {}'.format(data['error_code']))
```

```

self.arm.release_error_warn_changed_callback(error_warn_change_callback)

self.arm.register_error_warn_changed_callback(error_warn_change_callback)

# Register state changed callback
def state_changed_callback(data):
    if data and data['state'] == 4:
        if self.arm.version_number[0] >= 1 and self.arm.version_number[1] >=
1 and self.arm.version_number[2] > 0:
            self.params['quit'] = True
            pprint('state=4, quit')
            self.arm.release_state_changed_callback(state_changed_callback)
self.arm.register_state_changed_callback(state_changed_callback)

# Register counter value changed callback
if hasattr(self.arm, 'register_count_changed_callback'):
    def count_changed_callback(data):
        if not self.params['quit']:
            pprint('counter val: {}'.format(data['count']))
        self.arm.register_count_changed_callback(count_changed_callback)

# Register connect changed callback
def connect_changed_callback(data):
    if data and not data['connected']:
        self.params['quit'] = True
        pprint('disconnect, connected={}, reported={},
quit'.format(data['connected'], data['reported']))

self.arm.release_connect_changed_callback(error_warn_change_callback)
self.arm.register_connect_changed_callback(connect_changed_callback)

def reset(self):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ''.join(map(str, args))))
        except:
            print(*args, **kwargs)

    pprint('xArm-Python-SDK Version:{}'.format(version.__version__))

    self.arm.clean_warn()
    self.arm.clean_error()
    self.arm.motion_enable(True)
    self.arm.set_mode(0)
    self.arm.set_state(0)
    time.sleep(1)

```

```

variables = {}
self.params = {'speed': 300, 'acc': 6000, 'angle_speed': 20, 'angle_acc':
500, 'events': {}, 'variables': variables, 'callback_in_thread': True, 'quit': False}

# Register error/warn changed callback
def error_warn_change_callback(data):
    if data and data['error_code'] != 0:
        self.params['quit'] = True
        pprint('err={}, quit'.format(data['error_code']))

self.arm.release_error_warn_changed_callback(error_warn_change_callback)

self.arm.register_error_warn_changed_callback(error_warn_change_callback)

# Register state changed callback
def state_changed_callback(data):
    if data and data['state'] == 4:
        if self.arm.version_number[0] >= 1 and self.arm.version_number[1] >=
1 and self.arm.version_number[2] > 0:
            self.params['quit'] = True
            pprint('state=4, quit')
            self.arm.release_state_changed_callback(state_changed_callback)
self.arm.register_state_changed_callback(state_changed_callback)

# Register counter value changed callback
if hasattr(self.arm, 'register_count_changed_callback'):
    def count_changed_callback(data):
        if not self.params['quit']:
            pprint('counter val: {}'.format(data['count']))
        self.arm.register_count_changed_callback(count_changed_callback)

# Register connect changed callback
def connect_changed_callback(data):
    if data and not data['connected']:
        self.params['quit'] = True
        pprint('disconnect, connected={}, reported={},
quit'.format(data['connected'], data['reported']))

self.arm.release_connect_changed_callback(error_warn_change_callback)
self.arm.register_connect_changed_callback(connect_changed_callback)

def Robot_Movement_True(self, Posicion:list, mov_speed, mov_acc):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ' '.join(map(str, args))))
        except:

```

```

        print(*args, **kwargs)

    if self.arm.error_code == 0 and not self.params['quit']:
        code = self.arm.set_position(*Posicion, speed=mov_speed,
mvacc=mov_acc, radius=-1.0, wait=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_position, code={}'.format(code))

    return self.arm.error_code,self.params['quit']

def Robot_Movement_False(self, PosicionF:list, mov_speed, mov_acc):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ''.join(map(str, args))))
        except:
            print(*args, **kwargs)

    if self.arm.error_code == 0 and not self.params['quit']:
        code = self.arm.set_position(*PosicionF, speed=mov_speed,
mvacc=mov_acc, radius=-1.0, wait=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_position, code={}'.format(code))

    return self.arm.error_code,self.params['quit']

def Robot_Movement_Eje(self, descenso, mov_speed, mov_acc):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ''.join(map(str, args))))
        except:
            print(*args, **kwargs)

    if self.arm.error_code == 0 and not self.params['quit']:
        code = self.arm.set_position(z=-descenso, speed=mov_speed,
mvacc=mov_acc, relative=True, wait=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_position, code={}'.format(code))

    return self.arm.error_code,self.params['quit']

```

```

def Robot_Gripper_open(self):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ''.join(map(str, args))))
        except:
            print(*args, **kwargs)

    if self.arm.error_code == 0 and not self.params['quit']:
        code = self.arm.set_gripper_position(850, wait=True, speed=5000,
auto_enable=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_gripper_position, code={}'.format(code))

    return self.arm.error_code,self.params['quit']

def Robot_Gripper_Close(self):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ''.join(map(str, args))))
        except:
            print(*args, **kwargs)

    if self.arm.error_code == 0 and not self.params['quit']:
        code = self.arm.set_gripper_position(0, wait=True, speed=5000,
auto_enable=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_gripper_position, code={}'.format(code))

    return self.arm.error_code,self.params['quit']

def Robot_Gripper_Pos(self, apertura):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}] {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ''.join(map(str, args))))
        except:
            print(*args, **kwargs)

    if self.arm.error_code == 0 and not self.params['quit']:

```

```

        code = self.arm.set_gripper_position(apertura, wait=True, speed=5000,
auto_enable=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_gripper_position, code={}'.format(code))

    return self.arm.error_code,self.params['quit']

def hello(self):

    # saludo
    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}]' .format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())), stack_tuple[1], ' '.join(map(str, args))))
        except:
            print(*args, **kwargs)

    if not self.params['quit']:
        self.params['angle_speed'] = 160
    if not self.params['quit']:
        self.params['acc'] = 8000

    if self.arm.error_code == 0 and not self.params['quit']:
        code = self.arm.set_position(*[105.7, 363.3, 715.1, -2.0, -0.1, -178.3],
speed=self.params['speed'], mvacc=self.params['acc'], radius=5, wait=True)
        if code != 0:
            self.params['quit'] = True
            pprint('set_position, code={}'.format(code))

    for i in range(int(4)):
        if self.params['quit']:
            break
        if self.arm.error_code == 0 and not self.params['quit']:
            code = self.arm.set_servo_angle(angle=[74.0, -18.6, -68.8, 180.7,
55.8, 107.8], speed=self.params['angle_speed'],
mvacc=self.params['angle_acc'], wait=False, radius=5)
            if code != 0:
                self.params['quit'] = True
                pprint('set_servo_angle, code={}'.format(code))
        if self.arm.error_code == 0 and not self.params['quit']:
            code = self.arm.set_servo_angle(angle=[74.0, -18.6, -68.8, 180.7,
129.0, 107.8], speed=self.params['angle_speed'],
mvacc=self.params['angle_acc'], wait=False, radius=5)
            if code != 0:
                self.params['quit'] = True
                pprint('set_servo_angle, code={}'.format(code))

```

```

        if self.arm.error_code == 0 and not self.params['quit']:
            code = self.arm.set_servo_angle(angle=[76.5, -19.6, -68.8, 180.7, 92.9,
103.1], speed=self.params['angle_speed'], mvacc=self.params['angle_acc'],
wait=False, radius=5.0)
            if code != 0:
                self.params['quit'] = True
                pprint('set_servo_angle, code={}'.format(code))

        return self.arm.error_code, self.params['quit']

def Apagar_Robot(self):

    def pprint(*args, **kwargs):
        try:
            stack_tuple = traceback.extract_stack(limit=2)[0]
            print('[{}][{}]' .format(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time()))), stack_tuple[1], ' '.join(map(str, args))))
        except:
            print(*args, **kwargs)

    def error_warn_change_callback(data):
        if data and data['error_code'] != 0:
            self.params['quit'] = True
            pprint('err={}, quit {}'.format(data['error_code']))

    self.arm.release_error_warn_changed_callback(error_warn_change_callback)

    def state_changed_callback(data):
        if data and data['state'] == 4:
            if self.arm.version_number[0] >= 1 and self.arm.version_number[1] >=
1 and self.arm.version_number[2] > 0:
                self.params['quit'] = True
                pprint('state=4, quit')
                self.arm.release_state_changed_callback(state_changed_callback)

    def count_changed_callback(data):
        if not self.params['quit']:
            pprint('counter val: {}'.format(data['count']))

    if hasattr(self.arm, 'release_count_changed_callback'):
        self.arm.release_count_changed_callback(count_changed_callback)

    self.arm.release_error_warn_changed_callback(state_changed_callback)
    self.arm.release_state_changed_callback(state_changed_callback)

    self.arm.release_connect_changed_callback(error_warn_change_callback)

```

15.1.3. Ejecución de movimientos del robot en un hilo (bot_hilo.py)

"""

File name: bot_hilo

Developer: Maher Bouymejan

Contact: maherbouymejan@gmail.com

Creation date: 10/04/2023

Description: Functions for executing bot movement in a thread.

Python version: 3.9.0

Version: 0

"""

from library_imports import *

from Cobot import Robot_Arm

class HiloCobot(threading.Thread):

def __init__(self, IP):

threading.Thread.__init__(self)

self.IP=IP

self.Arm = Robot_Arm(IP)

self.estado=0

self.detener_hilo = False

self.ini_poss_bot()

def run(self):

```

if not self.detener_hilo:

    code_error,stop = self.Arm.Robot_Gripper_open()

    code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)


def ini_poss_bot(self):

    """

    #reads the coordinates of the movements,

    #in order to update itself.

    """

    global pos_0, pos_1, pos_2, pos_3, pos_4, pos_5, pos_6, pos_7, pos_8,
pos_9, pos_10, pos_11, pos_12, pos_13, pos_14, pos_15, pos_16,
pos_flow_0, pos_flow_1, pos_flow_2, pos_flow_3, pos_flow_4, gri_vaso1,
gri_vaso2, gri_vaso3, gri_vaso4, gri_flow, speed_mov, acc_mov, speed_mov1,
acc_mov1, speed_mov2, acc_mov2, speed_flow, acc_flow

    # Open the configuration file

    with open('txt_param\Poss_conf.txt', 'r') as f:

        config_lines = f.readlines()

    # Read table values from the file

    pos_0 = ast.literal_eval(config_lines[0].split('=')[1].strip())

    pos_1 = ast.literal_eval(config_lines[1].split('=')[1].strip())

    pos_2 = ast.literal_eval(config_lines[2].split('=')[1].strip())

    pos_3 = ast.literal_eval(config_lines[3].split('=')[1].strip())

    pos_4 = ast.literal_eval(config_lines[4].split('=')[1].strip())

```

```

pos_5 = ast.literal_eval(config_lines[5].split('=')[1].strip())
pos_6 = ast.literal_eval(config_lines[6].split('=')[1].strip())
pos_7 = ast.literal_eval(config_lines[7].split('=')[1].strip())
pos_8 = ast.literal_eval(config_lines[8].split('=')[1].strip())
pos_9 = ast.literal_eval(config_lines[9].split('=')[1].strip())
pos_10 = ast.literal_eval(config_lines[10].split('=')[1].strip())
pos_11 = ast.literal_eval(config_lines[11].split('=')[1].strip())
pos_12 = ast.literal_eval(config_lines[12].split('=')[1].strip())
pos_13 = ast.literal_eval(config_lines[13].split('=')[1].strip())
pos_14 = ast.literal_eval(config_lines[14].split('=')[1].strip())
pos_15 = ast.literal_eval(config_lines[15].split('=')[1].strip())
pos_16 = ast.literal_eval(config_lines[16].split('=')[1].strip())
pos_flow_0 = ast.literal_eval(config_lines[17].split('=')[1].strip())
pos_flow_1 = ast.literal_eval(config_lines[18].split('=')[1].strip())
pos_flow_2 = ast.literal_eval(config_lines[19].split('=')[1].strip())
pos_flow_3 = ast.literal_eval(config_lines[20].split('=')[1].strip())
pos_flow_4 = ast.literal_eval(config_lines[21].split('=')[1].strip())

gri_vaso1=float(config_lines[22].split('=')[1].strip())
gri_vaso2=float(config_lines[23].split('=')[1].strip())
gri_vaso3=float(config_lines[24].split('=')[1].strip())
gri_vaso4=float(config_lines[25].split('=')[1].strip())
gri_flow=float(config_lines[26].split('=')[1].strip())

```

```
speed_mov = float(config_lines[27].split('=')[1].strip())  
acc_mov = float(config_lines[28].split('=')[1].strip())  
speed_mov1=float(config_lines[29].split('=')[1].strip())  
acc_mov1=float(config_lines[30].split('=')[1].strip())  
speed_mov2=float(config_lines[31].split('=')[1].strip())  
acc_mov2=float(config_lines[32].split('=')[1].strip())  
speed_flow=float(config_lines[33].split('=')[1].strip())  
acc_flow=float(config_lines[34].split('=')[1].strip())
```

```
def End_bot(self):  
  
    self.detener_hilo = True  
  
    self.Arm.Apagar_Robot()
```

```
def reset(self):  
  
    #Function to reset the bot  
  
    self.ini_poss_bot()  
  
    self.Arm.reset()  
  
    self.estado=0  
  
    self.detener_hilo = False
```

```
def reset_mov(self, a):  
  
    #Function to reset the bot and back to where it came from
```

```

self.ini_poss_bot()

self.Arm.reset()

self.estado=0

self.detener_hilo = False


if not self.detener_hilo:

    n=a

    while n >=0:

        if 0 <= n <= 9:

            if n==9:

                code_error,stop = self.Arm.Robot_Gripper_open()

                n=n-1

            elif n==8:

                code_error,stop = self.Arm.Robot_Gripper_open()

                n=n-1

            elif n==7:

                code_error,stop =
self.Arm.Robot_Movement_True(pos_5,speed_mov,acc_mov)

                n=n-1

            elif n==6:

                code_error,stop =
self.Arm.Robot_Movement_True(pos_4,speed_mov,acc_mov)

                n=n-1

            elif n==5:

```

```

        code_error,stop = self.Arm.Robot_Gripper_open()

        n=n-1

    elif n==4:

        code_error,stop = self.Arm.Robot_Gripper_open()

        n=n-1

    elif n==3:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_3,speed_mov,acc_mov)

        n=n-1

    elif n==2:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_2,speed_mov,acc_mov)

        n=n-1

    elif n==1:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_1,speed_mov,acc_mov)

        n=n-1

    if n==0:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)

        break

    else:

        print("Error reset")

        break

```

```

def go_glass(self): # Estado de 0 a 4

    #movement from the home position to holding the cup inside the dispenser

    self.ini_poss_bot()

    self.estado = 0

    if self.detener_hilo:

        return False, self.estado

    pos_list = [pos_1, pos_2, pos_3, gri_vaso1]

    for pos in pos_list:

        code_error, stop = self.Arm.Robot_Gripper_Pos(pos) if pos in
[gri_vaso1] else self.Arm.Robot_Movement_False(pos, speed_mov1,
acc_mov1)

        if code_error != 0 or stop:

            return False, self.estado

        self.estado += 1

    self.completed = self.estado == 4

    return self.completed, self.estado

def exit_glass(self): #estado 5 a 9

    # movement from holding the cup

```

```

self.ini_poss_bot()

self.estado = 4

if self.detener_hilo:

    return False, self.estado

pos_list = [gri_vaso2, pos_4, pos_5, gri_vaso3, gri_vaso4]

for pos in pos_list:

    code_error, stop = self.Arm.Robot_Gripper_Pos(pos) if pos in
[gri_vaso2, gri_vaso3, gri_vaso4] else self.Arm.Robot_Movement_False(pos,
speed_mov, acc_mov)

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1

self.completed = self.estado == 9

return self.completed , self.estado

def go_beer(self): #estado 10 a 12

    # Movement to bring the cup to the beer dispenser.

    self.ini_poss_bot()

    self.estado = 9

```

```

if self.detener_hilo:

    return False, self.estado


pos_list = [pos_6, pos_7]

for pos in pos_list:

    code_error, stop = self.Arm.Robot_Movement_False(pos, speed_mov,
acc_mov) if pos in [pos_6] else self.Arm.Robot_Movement_True(pos,
speed_mov, acc_mov)

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1


self.completed = self.estado == 11


return self.completed , self.estado


def mov_45g(self): #estado 13

    #Movement of the cup to bring it to 30 degrees

    self.ini_poss_bot()

    self.estado = 11

    if self.detener_hilo:

        return False, self.estado

```

```
code_error, stop = self.Arm.Robot_Movement_True(pos_8, speed_mov,
acc_mov)
```

```
if code_error != 0 or stop:
```

```
    return False, self.estado
```

```
self.estado += 1
```

```
self.completed = self.estado == 12
```

```
return self.completed , self.estado
```

```
def mov_30g(self): #estado 13
```

```
    #Movement of the cup to bring it to 30 degrees
```

```
    self.ini_poss_bot()
```

```
    self.estado = 12
```

```
    if self.detener_hilo:
```

```
        return False, self.estado
```

```
code_error, stop = self.Arm.Robot_Movement_True(pos_9, speed_mov,
acc_mov)
```

```
if code_error != 0 or stop:
```

```
    return False, self.estado
```

```
self.estado += 1
```

```
self.completed = self.estado == 13
```

```
return self.completed , self.estado
```

```
def mov_0g(self): #estado 14
```

```
    #Movement of the cup to bring it to 0 degrees
```

```
    self.ini_poss_bot()
```

```
    self.estado = 13
```

```
    if self.detener_hilo:
```

```
        return False, self.estado
```

```
        code_error, stop = self.Arm.Robot_Movement_True(pos_10, speed_mov,  
acc_mov)
```

```
        if code_error != 0 or stop:
```

```
            return False, self.estado
```

```
        self.estado += 1
```

```
        self.completed = self.estado == 14
```

```
        return self.completed, self.estado
```

```
def exit_beer(self): #estado 15 a 18
```

```
    # Movement to place the glass on the table
```

```
    self.ini_poss_bot()
```

```

self.estado = 14

if self.detener_hilo:

    return False, self.estado


pos_list = [pos_11, pos_12, pos_13]

for pos in pos_list:

    code_error, stop = self.Arm.Robot_Movement_False(pos, speed_mov2,
acc_mov2) if pos in [pos_11, pos_12] else
self.Arm.Robot_Movement_True(pos, speed_mov, acc_mov)

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1


if self.estado == 17:

    code_error, stop = self.Arm.Robot_Gripper_open()

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1


self.completed = self.estado == 18


return self.completed , self.estado

```

```

def exist_table(self): #estado 19 20

    # Movement to take a photo of the beer

    self.ini_poss_bot()


    self.estado = 18

    if self.detener_hilo:

        return False, self.estado


    pos_list = [pos_14, pos_15]

    for pos in pos_list:

        code_error, stop = self.Arm.Robot_Movement_False(pos, speed_mov,
acc_mov) if pos in [pos_14] else self.Arm.Robot_Movement_True(pos,
speed_mov, acc_mov)

        if code_error != 0 or stop:

            return False, self.estado

        self.estado += 1


    self.completed = self.estado == 20


    return self.completed , self.estado


def exist_table2(self): #estado 21


    self.ini_poss_bot()

```

```

self.estado = 20

if self.detener_hilo:

    return False, self.estado

    code_error, stop = self.Arm.Robot_Movement_False(pos_16, speed_mov,
acc_mov)

if code_error != 0 or stop:

    return False, self.estado

self.estado += 1

self.completed = self.estado == 21

return self.completed , self.estado

def state_flow_Poss(self, state_flow_rec, pos_flow_i, pos_flow_s):
    """
    Function whose input value is a percentage,
    and calculates the value in millimetres that the bot must move
    to configure the valve according to the percentage.
    """

    Zmin = pos_flow_i[2]

    Zmax = pos_flow_s[2]

    flow_range = Zmin - Zmax

```

```

if not 0 <= state_flow_rec <= 100:

    raise ValueError("Invalid flow percentage")

dist_mm = round((flow_range*state_flow_rec)/100, 1)

if dist_mm > flow_range:

    dist_mm = flow_range

    print("Flow rate exceeds maximum")

elif dist_mm < 0:

    dist_mm = 0

    print("Flow rate cannot be negative")

print(f"For a flow rate of {state_flow_rec}, distance is z= {dist_mm} mm
(Bot)")

return dist_mm

def Regulate_flow(self,state_flow_rec): #estado 22 a 30

    #Function for the bot to set the flow valve using a percentage as input
    value.

    self.ini_poss_bot()

    self.estado = 21

```

```

if self.detener_hilo:

    return False, self.estado

if 0 <= state_flow_rec <= 100:

    z_flow=self.state_flow_Poss(state_flow_rec, pos_flow_3, pos_flow_4)

    code_error,stop =
self.Arm.Robot_Movement_False(pos_flow_0,speed_flow,acc_flow)

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1

    code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_1,speed_flow,acc_flow)

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1

    code_error,stop = self.Arm.Robot_Gripper_Pos(gri_flow)

    if code_error != 0 or stop:

        return False, self.estado

    self.estado += 1

```

```

        code_error,stop =
self.Arm.Robot_Movement_False(pos_flow_2,speed_flow,acc_flow)

        if code_error != 0 or stop:

            return False, self.estado

        self.estado += 1


        state_flow=0

        print("Pressure regulator at : " , state_flow , " % ")

```

```

        code_error,stop =
self.Arm.Robot_Movement_False(pos_flow_3,speed_flow,acc_flow)

        if code_error != 0 or stop:

            return False, self.estado

        self.estado += 1

```

```

        code_error,stop = self.Arm.Robot_Movement_Eje( z_flow,
speed_flow,acc_flow)

        if code_error != 0 or stop:

            return False, self.estado

        self.estado += 1

```

```

        state_flow=state_flow_rec

        print("Pressure regulator at : " , state_flow , " % ")

```

```

code_error,stop = self.Arm.Robot_Gripper_open()

if code_error != 0 or stop:

    return False, self.estado

self.estado += 1


code_error,stop =
self.Arm.Robot_Movement_False(pos_flow_1,speed_flow,acc_flow)

if code_error != 0 or stop:

    return False, self.estado

self.estado += 1


code_error,stop =
self.Arm.Robot_Movement_False(pos_flow_0,speed_flow,acc_flow)

if code_error != 0 or stop:

    return False, self.estado

self.estado += 1


else:

    print("Error porcentaje de flowmeter")

    return False, self.estado


self.completed = self.estado == 30


return self.completed , self.estado

```

```

def hello(self):

    self.estado = 40

    if self.detener_hilo:

        return False, self.estado


code_error,stop =self.Arm.hello()

if code_error != 0 or stop:

    return False, self.estado


self.estado += 1


self.completed = self.estado == 41

return self.completed , self.estado


def go_home(self): #estado 21 o 30 32

    #Bot movement home

    self.ini_poss_bot()

    self.estado = 30


if self.detener_hilo:

    return False, self.estado

```

```

        code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)

        if code_error != 0 or stop:

            return False, self.estado

        self.estado += 1

code_error,stop = self.Arm.Robot_Gripper_open()

if code_error != 0 or stop:

    return False, self.estado

self.estado += 1

self.completed = self.estado == 32

return self.completed , self.estado

def roi1(self):

    #function to check positions

    self.ini_poss_bot()

code_error,stop = self.Arm.Robot_Gripper_Pos(gri_vaso3)

code_error,stop = self.Arm.Robot_Gripper_Pos(gri_vaso4)

```

```

        code_error,stop =
self.Arm.Robot_Movement_True(pos_7,speed_mov,acc_mov)

        code_error,stop =
self.Arm.Robot_Movement_True(pos_8,speed_mov,acc_mov)

def roi1_end(self):

    #function to check positions

    self.ini_poss_bot()

    code_error,stop =
self.Arm.Robot_Movement_True(pos_7,speed_mov,acc_mov)

    code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)

    code_error,stop = self.Arm.Robot_Gripper_open()


def roi2(self):

    #function to check positions

    self.ini_poss_bot()

    code_error,stop =
self.Arm.Robot_Movement_True(pos_9,speed_mov,acc_mov)


def roi2_end(self):

    #function to check positions

    self.ini_poss_bot()


    code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)

```

```
code_error,stop = self.Arm.Robot_Gripper_open()
```

```
def roi3(self):
```

```
    #function to check positions
```

```
    self.ini_poss_bot()
```

```
    code_error,stop =
```

```
self.Arm.Robot_Movement_True(pos_10,speed_mov,acc_mov)
```

```
def roi3_end(self):
```

```
    #function to check positions
```

```
    self.ini_poss_bot()
```

```
    code_error,stop =
```

```
self.Arm.Robot_Movement_True(pos_7,speed_mov,acc_mov)
```

```
    code_error,stop =
```

```
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)
```

```
    code_error,stop = self.Arm.Robot_Gripper_open()
```

```
def roi4(self):
```

```
    #function to check positions
```

```
    self.ini_poss_bot()
```

```
    code_error,stop =
```

```
self.Arm.Robot_Movement_True(pos_10,speed_mov,acc_mov)
```

```

def roi4_end(self):

    #function to check positions

    self.ini_poss_bot()

    code_error,stop =
self.Arm.Robot_Movement_True(pos_7,speed_mov,acc_mov)

    code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)

    code_error,stop = self.Arm.Robot_Gripper_open()


def mov_poss(self,n):

    #function to check positions

    self.ini_poss_bot()

    if 0 <= n <= 32:

        if n==0:

            code_error,stop =
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)

            elif n==1:

                code_error,stop =
self.Arm.Robot_Movement_True(pos_1,speed_mov,acc_mov)

                elif n==2:

                    code_error,stop =
self.Arm.Robot_Movement_True(pos_2,speed_mov,acc_mov)

                    elif n==3:

```

```

        code_error,stop =
self.Arm.Robot_Movement_True(pos_3,speed_mov,acc_mov)

    elif n==4:

        code_error,stop = self.Arm.Robot_Gripper_Pos(gri_vaso1)

    elif n==5:

        code_error,stop = self.Arm.Robot_Gripper_Pos(gri_vaso2)

    elif n==6:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_4,speed_mov,acc_mov)

    elif n==7:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_5,speed_mov,acc_mov)

    elif n==8:

        code_error,stop = self.Arm.Robot_Gripper_Pos(gri_vaso3)

    elif n==9:

        code_error,stop = self.Arm.Robot_Gripper_Pos(gri_vaso4)

    elif n==10:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_6,speed_mov,acc_mov)

    elif n==11:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_7,speed_mov,acc_mov)

    elif n==12:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_8,speed_mov,acc_mov)

```

```

elif n==13:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_9,speed_mov,acc_mov)

elif n==14:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_10,speed_mov,acc_mov)

elif n==15:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_11,speed_mov,acc_mov)

elif n==16:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_12,speed_mov,acc_mov)

elif n==17:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_13,speed_mov,acc_mov)

elif n==18:

    code_error,stop = self.Arm.Robot_Gripper_open()

elif n==19:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_14,speed_mov,acc_mov)

elif n==20:

    code_error,stop =
self.Arm.Robot_Movement_True(pos_15,speed_mov,acc_mov)

elif n==21:

```

```

        code_error,stop =
self.Arm.Robot_Movement_True(pos_16,speed_mov,acc_mov)

    elif n==22:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_0,speed_mov,acc_mov)

    elif n==23:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_1,speed_mov,acc_mov)

    elif n==24:

        code_error,stop = self.Arm.Robot_Gripper_Pos(gri_flow)

    elif n==25:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_2,speed_mov,acc_mov)

    elif n==26:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_3,speed_mov,acc_mov)

    elif n==27:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_4,speed_mov,acc_mov)

    elif n==28:

        code_error,stop = self.Arm.Robot_Gripper_open()

    elif n==29:

        code_error,stop =
self.Arm.Robot_Movement_True(pos_flow_1,speed_mov,acc_mov)

    elif n==30:

```

```
        code_error,stop =  
self.Arm.Robot_Movement_True(pos_flow_0,speed_mov,acc_mov)  
  
    elif n==31:  
  
        code_error,stop =  
self.Arm.Robot_Movement_True(pos_0,speed_mov,acc_mov)  
  
    elif n==32:  
  
        code_error,stop = self.Arm.Robot_Gripper_open()  
  
else:  
  
    print("Error mov_poss(self,n) ")
```

15.1.4. Gestión de la base de datos online (database_beer.py).

```
"""
# File name: Database_beer
# Developer: Maher Bouymejan
# Contact: maherbouymejan@gmail.com
# Creation date: 10/04/2023
# Description: function to store data in a cloud database and in a .txt file,
also analyses the data by means of graphs.
# Python version: 3.9.0
# Version: 0
"""

from library_imports import *

class MongoDBManager:
    def __init__(self, connection_string):
        self.client = pymongo.MongoClient(connection_string)
        matplotlib.use('Qt5Agg')

    #name of the collection for each start of the programme
    fecha_actual = time.strftime("%d-%m-%Y-")
    hora_actual = time.strftime("%H_%M_%S")

    self.Folder_data= "Analysys_" + str(fecha_actual) + "_" +str(hora_actual)
    self.Folder_path_data = os.path.join(os.getcwd(), self.Folder_data)

    if not os.path.exists(self.Folder_path_data):
        os.makedirs(self.Folder_path_data)

    #-----
    self.database_name_opinions="opinions"
    self.collection_name_opinions="opinions"+str(fecha_actual)+str(hora_actual)

    # Crear la base de datos "beer parameters"
```

```

self.db_opinions = self.client[self.database_name_opinions]
# Crear la colección "beer_info" en la base de datos "beer parameters"
self.opinions_col = self.db_opinions[self.collection_name_opinions]

# Crear el índice único para el campo "id"
self.opinions_col.create_index("id", unique=True)

#-----

self.database_name_beer="beer_parameters"
self.collection_name_beer="beer_info"+str(fecha_actual)+str(hora_actual)

# Crear la base de datos "beer parameters"
self.db_beer = self.client[self.database_name_beer]
# Crear la colección "beer_info" en la base de datos "beer parameters"
self.beer_col = self.db_beer[self.collection_name_beer]

# Crear el índice único para el campo "id"
self.beer_col.create_index("id", unique=True)
#-----

#-----

self.database_name_info="info"
self.collection_name_info="info"+str(fecha_actual)+str(hora_actual)

# Crear la base de datos "beer parameters"
self.db_info = self.client[self.database_name_info]
# Crear la colección "beer_info" en la base de datos "beer parameters"
self.info_col = self.db_info[self.collection_name_info]

# Crear el índice único para el campo "id"
self.info_col.create_index("id", unique=True)

#-----

#parametros para configurar el tamaño de las graficas

```

```

self.hspace_plot=0.5
self.fig_width=15
self.fig_height=8

self.num_lines=8

#rango del eje "y" para graficas ponderadas
self.y_min=0
self.y_max=100
#rango del eje "y" para graficas en segundos totales
self.yss_min=0
self.yss_max=120
#rango del eje "y" para graficas en segundos totales beer
self.yssb_min=0
self.yssb_max=50
#rango del eje "y" para graficas en segundos limites
self.y_min=0
self.y_max=20

#limites min y maximos tiempos totales beer

self.ini_AI()

self.file_path = os.path.join(self.Folder_path_data, "beer_data.txt")
self.file_mode = "a" # append mode

self.file_path_opinions = os.path.join(self.Folder_path_data, "opinions_data.txt")
self.file_mode_opinions = "a" # append mode

self.file_path_info = os.path.join(self.Folder_path_data, "info_data.txt")
self.file_mode_info = "a" # append mode

def ini_AI(self):
# Abrir el archivo de configuración

```

```
with open('txt_param\Ai_conf.txt', 'r') as f:
    config_lines = f.readlines()
```

```
# Leer los valores de la tabla desde el archivo
self.limt_min00 = float(config_lines[0].split('=')[1].strip())
self.limt_max00 = float(config_lines[1].split('=')[1].strip())
self.limt_min0 = float(config_lines[2].split('=')[1].strip())
self.limt_max0 = float(config_lines[3].split('=')[1].strip())
self.limt_min1 = float(config_lines[4].split('=')[1].strip())
self.limt_max1 = float(config_lines[5].split('=')[1].strip())
self.limt_min2 = float(config_lines[6].split('=')[1].strip())
self.limt_max2 = float(config_lines[7].split('=')[1].strip())
self.limt_min3 = float(config_lines[8].split('=')[1].strip())
self.limt_max3 = float(config_lines[9].split('=')[1].strip())
self.limt_min4 = float(config_lines[10].split('=')[1].strip())
self.limt_max4 = float(config_lines[11].split('=')[1].strip())
```

```
def insert_opinions(self, id, date, stars, failure_type, failure, opinion, others):
    # Insertar un documento en la colección "beer_info"
    self.opinions_col.insert_one({
        "id": id,
        "date": date,
        "stars": stars,
        "failure type": failure_type,
        "Failure": failure,
        "opinion": opinion,
        "Others": others
    })
```

```
# Guardar los datos en el archivo
with open(self.file_path_opinions, self.file_mode_opinions) as f:
    f.write(f"{id},{date},{stars},{failure},{opinion},{others}\n")
```

```

def insert_info(self, id,
date,info_full,info1,info2,info3,info4,Bot_Action,Tap_Action, others):
# Insertar un documento en la colección "beer_info"
self.info_col.insert_one({
    "id": id,
    "Date": date,
    "Duration full": info_full,
    "Duration limit 1": info1,
    "Duration limit 2": info2,
    "Duration limit 3": info3,
    "Duration limit 4": info4,
    "Bot Action": Bot_Action,
    "Tap action": Tap_Action,
    "Others": others
})

# Guardar los datos en el archivo
with open(self.file_path_info, self.file_mode_info) as f:
f.write(f"{id},{date},{info_full},{info1},{info2},{info3},{info4},{Bot_Action},{Tap_Action},{others}\n")

def insert_beer_parameters(self, id, date, tap_opening, flow_opening,
start_time, end_time, total_duration, beer_time, time_limit1, time_limit2,
time_limit3,time_limit4, others):
# Insertar un documento en la colección "beer_info"
self.beer_col.insert_one({
    "id": id,
    "Date": date,
    "Tap Opening": tap_opening,
    "Flow Opening": flow_opening,
    "Start Time": start_time,
    "End Time": end_time,
    "Total Duration": total_duration,
    "Beer Time": beer_time,

```

```

"time limit 1": time_limit1,
"time limit 2": time_limit2,
"time limit 3": time_limit3,
"time limit 4": time_limit4,
"others": others
})

```

```

# Guardar los datos en el archivo

```

```

with open(self.file_path, self.file_mode) as f:
f.write(f"{id},{date},{tap_opening},{flow_opening},{start_time},{end_time},{total_d
uration},{beer_time},{time_limit1},{time_limit2},{time_limit3},{time_limit4},{others}
\n")

```

```

def beer_data_analysis(self):
db = self.client[self.database_name_beer]
collection = db[self.collection_name_beer]
cursor = collection.find({})

```

```

# Crear diccionarios para almacenar los datos de cada variable

```

```

tap_opening = {}
flow_opening = {}
total_duration = {}
time_beer = {}
time_limit1 = {}
time_limit2 = {}
time_limit3 = {}
time_limit4 = {}

```

```

# Iterar sobre los documentos de la colección y almacenar los datos en los
diccionarios

```

```

for document in cursor:
id = document["id"]
tap_opening[id] = document["Tap Opening"]
flow_opening[id] = document["Flow Opening"]

```

```

total_duration[id] = document["Total Duration"]

time_beer[id] = document["Beer Time"]
time_limit1[id] = document["time limit 1"]
time_limit2[id] = document["time limit 2"]
time_limit3[id] = document["time limit 3"]
time_limit4[id] = document["time limit 4"]

# Crear una figura con varios subplots, uno por cada variable
hora_actual = time.strftime("%H:%M:%S")
#fig, axs = plt.subplots(6, sharex=True, sharey=False)
fig0, axs0 = plt.subplots(2, 1, sharex=True, figsize=(self.fig_width,
self.fig_height*2))

fig0.suptitle(" Data analysis of : " + self.collection_name_beer + " , Time : "
+ hora_actual )

fig0.subplots_adjust(hspace=self.hspace_plot)

# Configurar los subplots con los datos de cada variable
axs0[0].plot(list(tap_opening.keys()), list(tap_opening.values()), marker = 'o',
color='red', label='Tap opening')
axs0[0].plot(list(flow_opening.keys()), list(flow_opening.values()), marker = 'x',
color='blue', label='Flow opening')
axs0[0].set_title('Tap opening [ % ] & Flow opening [ % ]')
axs0[0].set_ylim(self.y_min, self.y_max)
axs0[0].legend()

axs0[1].plot(list(total_duration.keys()), list(total_duration.values()), marker = '^',
color='black', label='Total time Beer')
axs0[1].plot(list(time_beer.keys()), list(time_beer.values()), marker = 'x',
color='blue', label='Total time Beer')
axs0[1].set_title('Total duration [ s ]')
axs0[1].set_ylim(self.yss_min, self.yss_max)

```

```

# Agregar línea horizontal en el valor del límite
axs0[1].axhline(self.limt_max00, color='r', linestyle='--')
axs0[1].axhline(self.limt_min00, color='g', linestyle='--')
axs0[1].axhline(self.limt_max0, color='r', linestyle='-')
axs0[1].axhline(self.limt_min0, color='g', linestyle='-')
axs0[1].legend()

hora_actual = time.strftime("%H_%M_%S")
self.Folder_imag_plot=os.path.join(self.Folder_path_data,
"plot_1_{}.png".format(hora_actual))
plt.savefig(self.Folder_imag_plot)

fig, axs = plt.subplots(4, 1, sharex=True, figsize=(self.fig_width,
self.fig_height*4))
fig.suptitle(" Data analysis of : " + self.collection_name_beer + " , Time : " +
hora_actual )
fig.subplots_adjust(hspace=self.hspace_plot)

axs[0].plot(list(time_limit1.keys()), list(time_limit1.values()), marker = '^')
axs[0].set_title('Time limit 1 [ s ]')
axs[0].set_ylim(self.ys_min, self.ys_max)
# Agregar línea horizontal en el valor del límite
axs[0].axhline(self.limt_max1, color='r', linestyle='--')
axs[0].axhline(self.limt_min1, color='g', linestyle='--')

axs[1].plot(list(time_limit2.keys()), list(time_limit2.values()), marker = '^')
axs[1].set_title('Time limit 2 [ s ]')
axs[1].set_ylim(self.ys_min, self.ys_max)
# Agregar línea horizontal en el valor del límite
axs[1].axhline(self.limt_max2, color='r', linestyle='--')
axs[1].axhline(self.limt_min2, color='g', linestyle='--')

axs[2].plot(list(time_limit3.keys()), list(time_limit3.values()), marker = '^')
axs[2].set_title('Time limit 3 [ s ]')

```

```

    axs[2].set_ylim(self.ys_min, self.ys_max)
    # Agregar línea horizontal en el valor del límite
    axs[2].axhline(self.limt_max3, color='r', linestyle='--')
    axs[2].axhline(self.limt_min3, color='g', linestyle='--')

    axs[3].plot(list(time_limit4.keys()), list(time_limit4.values()), marker = '^')
    axs[3].set_title('Time limit 4 [ s ]')
    axs[3].set_ylim(self.ys_min, self.ys_max)
    # Agregar línea horizontal en el valor del límite
    axs[3].axhline(self.limt_max4, color='r', linestyle='--')
    axs[3].axhline(self.limt_min4, color='g', linestyle='--')

    # Configurar las etiquetas de los ejes
    fig0.text(0.5, 0.04, 'ID Beer', ha='center')
    fig0.text(0.04, 0.5, 'Parameters', va='center', rotation='vertical')

    # Configurar las etiquetas de los ejes
    fig.text(0.5, 0.04, 'ID Beer', ha='center')
    fig.text(0.04, 0.5, 'Parameters', va='center', rotation='vertical')

    self.Folder_imag_plot=os.path.join(self.Folder_path_data,
    "plot_2_{}.png".format(hora_actual))
    plt.savefig(self.Folder_imag_plot)

    # Mostrar la figura
    plt.show()

    def opinions_data_analysis(self):
        db_opinions = self.client[self.database_name_opinions]
        collection_opinions = db_opinions[self.collection_name_opinions]
        cursor_opinions = collection_opinions.find({})

    # Crear diccionarios para almacenar los datos de cada variable
    stars = {}

```

```

failure_type = {}

# Iterar sobre los documentos de la colección y almacenar los datos en los
diccionarios
for document in cursor_opinions:
    id = document["id"]
    stars[id] = document["stars"]
    failure_type[id] = document["failure type"]

# Crear una figura con varios subplots, uno por cada variable
hora_actual = time.strftime("%H:%M:%S")
#fig, axs = plt.subplots(6, sharex=True, sharey=False)
fig0, axs0 = plt.subplots(2, 1, sharex=True, figsize=(self.fig_width,
self.fig_height*2))

fig0.suptitle(" Data analysis of opinions : " + self.collection_name_opinions + "
, Time : " + hora_actual )

fig0.subplots_adjust(hspace=self.hspace_plot)

# Configurar los subplots con los datos de cada variable
axs0[0].plot(list(stars.keys()), list(stars.values()), marker = 'o', color='red',
label='Tap opening')
axs0[0].set_title('Stars /5')
axs0[0].set_ylim(0, 5)
axs0[0].legend()

axs0[1].plot(list(failure_type.keys()), list(failure_type.values()), marker = '^',
color='black', label='Total time Beer')
axs0[1].set_title('type failure')
axs0[1].set_ylim(0, 4)
axs0[1].legend()

# Crear un gráfico de barras que muestre la distribución de las calificaciones

```

```

count_stars = [0]*5
for star in stars.values():
    count_stars[star-1] += 1
fig1, ax1 = plt.subplots()
ax1.bar(['1 star', '2 stars', '3 stars', '4 stars', '5 stars'], count_stars)
ax1.set_title('Distribution of stars')
ax1.set_ylabel('Number of opinions')
ax1.set_xlabel('Stars')

# Crear un gráfico de barras que muestre la distribución de los tipos de fallos
count_fallo = [0]*5
for fallo in failure_type.values():
    count_fallo[fallo-1] += 1
fig2, ax2 = plt.subplots()
ax2.bar(['Perfect Beer', 'Too Much Foam', 'Low Beer', 'Beer overflow', 'Warm Beer'], count_fallo)
ax2.set_title('Distribution of failure types')
ax2.set_ylabel('Number of opinions')
ax2.set_xlabel('Failure types')

plt.show()

def data_analysis_of (self, database_name, collection_name):
    db = self.client[database_name]
    collection = db[collection_name]
    cursor = collection.find({})
    hora_actual = time.strftime("%H:%M:%S")
    # Crear diccionarios para almacenar los datos de cada variable
    # Crear diccionarios para almacenar los datos de cada variable
    tap_opening = {}
    flow_opening = {}
    total_duration = {}
    time_beer = {}
    time_limit1 = {}

```

```

time_limit2 = {}
time_limit3 = {}
time_limit4 = {}

# Iterar sobre los documentos de la colección y almacenar los datos en los
diccionarios
for document in cursor:
    id = document["id"]
    tap_opening[id] = document["Tap Opening"]
    flow_opening[id] = document["Flow Opening"]
    total_duration[id] = document["Total Duration"]

    time_beer[id] = document["Beer Time"]
    time_limit1[id] = document["time limit 1"]
    time_limit2[id] = document["time limit 2"]
    time_limit3[id] = document["time limit 3"]
    #time_limit4[id] = document["time limit 4"]
    time_limit4[id] = 0

# Crear una figura con varios subplots, uno por cada variable

#fig, axs = plt.subplots(6, sharex=True, sharey=False)

fig, axs0 = plt.subplots(2, 1, sharex=True, figsize=(self.fig_width,
self.fig_height*2))
fig.suptitle(" Data analysis of : " + self.collection_name_beer + " , Time : " +
hora_actual )

fig.subplots_adjust(hspace=self.hspace_plot)

fig, axs = plt.subplots(4, 1, sharex=True, figsize=(self.fig_width,
self.fig_height*4))

```

```
fig.suptitle(" Data analysis of : " + collection_name + " , Time : " +
hora_actual )
```

```
# Configurar los subplots con los datos de cada variable
```

```
# Configurar los subplots con los datos de cada variable
```

```
axs0[0].plot(list(tap_opening.keys()), list(tap_opening.values()), marker = 'o',
color='red', label='Tap opening')
```

```
axs0[0].plot(list(flow_opening.keys()), list(flow_opening.values()), marker = 'x',
color='blue', label='Flow opening')
```

```
axs0[0].set_title('Tap opening [ % ] & Flow opening [ % ]')
```

```
axs0[0].set_ylim(self.y_min, self.y_max)
```

```
axs0[0].legend()
```

```
axs0[1].plot(list(total_duration.keys()), list(total_duration.values()), marker = '^',
color='black', label='Total time Beer')
```

```
axs0[1].plot(list(time_beer.keys()), list(time_beer.values()), marker = 'x',
color='blue', label='Total time Beer')
```

```
axs0[1].set_title('Total duration [ s ]')
```

```
axs0[1].set_ylim(self.yss_min, self.yss_max)
```

```
# Agregar línea horizontal en el valor del límite
```

```
axs0[1].axhline(self.limt_max00, color='r', linestyle='--')
```

```
axs0[1].axhline(self.limt_min00, color='g', linestyle='--')
```

```
axs0[1].axhline(self.limt_max0, color='r', linestyle='-.')
```

```
axs0[1].axhline(self.limt_min0, color='g', linestyle='-.')
```

```
axs0[1].legend()
```

```
axs[0].plot(list(time_limit1.keys()), list(time_limit1.values()), marker = '^')
```

```
axs[0].set_title('Time limit 1 [ s ]')
```

```
axs[0].set_ylim(self.ys_min, self.ys_max)
```

```
# Agregar línea horizontal en el valor del límite
```

```
axs[0].axhline(self.limt_max1, color='r', linestyle='--')
```

```
axs[0].axhline(self.limt_min1, color='g', linestyle='--')
```

```
axs[1].plot(list(time_limit2.keys()), list(time_limit2.values()), marker = '^')
```

```

    axs[1].set_title('Time limit 2 [ s ]')
    axs[1].set_ylim(self.ys_min, self.ys_max)
    # Agregar línea horizontal en el valor del límite
    axs[1].axhline(self.limt_max2, color='r', linestyle='--')
    axs[1].axhline(self.limt_min2, color='g', linestyle='--')

    axs[2].plot(list(time_limit3.keys()), list(time_limit3.values()), marker = '^')
    axs[2].set_title('Time limit 3 [ s ]')
    axs[2].set_ylim(self.ys_min, self.ys_max)
    # Agregar línea horizontal en el valor del límite
    axs[2].axhline(self.limt_max3, color='r', linestyle='--')
    axs[2].axhline(self.limt_min3, color='g', linestyle='--')

    axs[3].plot(list(time_limit4.keys()), list(time_limit4.values()), marker = '^')
    axs[3].set_title('Time limit 4 [ s ]')
    axs[3].set_ylim(self.ys_min, self.ys_max)
    # Agregar línea horizontal en el valor del límite
    axs[3].axhline(self.limt_max4, color='r', linestyle='--')
    axs[3].axhline(self.limt_min4, color='g', linestyle='--')

    # Configurar las etiquetas de los ejes
    fig.text(0.5, 0.04, 'ID Beer', ha='center')
    fig.text(0.04, 0.5, 'Parameters', va='center', rotation='vertical')

    # Mostrar la figura

    plt.show()

    hora_actual = time.strftime("%H_%M_%S")
    self.Folder_imag_plot=os.path.join(self.Folder_path_data,
    "plot_{}.png".format(hora_actual))
    plt.savefig(self.Folder_imag_plot)

```

15.1.5. Detección de objetos mediante YOLO v8 (Detector_cup.py)

```
"""

# File name: Detector_cup
# Developer: Maher Bouymejan
# Contact: maherbouymejan@gmail.com
# Creation date: 10/04/2023
# Description: Function to detect if there is a cup using yolo v8.
# Python version: 3.9.0
# Version: 0
"""

from library_imports import *

model = YOLO("YOLOv8_custom/best_cup.pt")

def detectar_objeto(imagen):

    results = model.predict(source=imagen ,save=True,show=True, conf=0.9)
    #results = model(imagen)
    results_plotted = results[0].plot()
    #cv2.imshow("result", results_plotted)

    try:
        #box=[x1, y1, x2, y2, score, label]
        box=results[0].boxes.tolist()[0]
        print(box)
        for r in results:
            for c in r.boxes.cls:
                name_detect = model.names[int(c)]
                #print(name_detect)
                print(" The detected part is : ",name_detect," box = ",box)
        return True, box

    except:
        print("No cup detected")
        return False, 0
```

15.1.6. Comunicación con el Arduino para el control de motores (Control_servo.py)

"""

File name: Control_servo

Developer: Maher Bouymejan

Contact: maherbouymejan@gmail.com

Creation date: 10/04/2023

Description: function to control the arduino which in turn controls a servo motor

and a linear actuator via an H-bridge.

Python version: 3.9.0

Version: 0

"""

from library_imports import *

class Control_Servo():

def __init__(self,Port) -> None:

self.Port=Port

self.Inicializacion()

def Inicializacion(self):

self.ser = serial.Serial(self.Port, 9600, timeout=1)

def mov_servo(self, Ang):

self.ser.write(str(Ang).encode())

def Grifo_cerrado(self):

"""Function for closing the beer tap, corresponds to 20 degrees of the servo"""

self.ser.write(str(20).encode())

def Grifo_Abierto(self):

"""Function for opening the beer tap, corresponds to 70 degrees of the servo"""

```

self.ser.write(str(70).encode())

def Grifo_espuma(self):
    """Function for opening the foam tap, corresponds to 1 degrees of the
servo"""
    self.ser.write(str(1).encode())

def Tap_Porcentaje(self, porcentaje):
    """Function for opening beer tap, Input range [0, 100] %, Servo range [20,
70] DEGREES"""
    if 0 <= porcentaje <= 100:
        grad= 20 + ( porcentaje * 50 )/100

        self.ser.write(str(grad).encode())
    else:
        print("Error Tap range [0, 100] %")

def Lin_open(self):
    """Function for opening the linear drive"""
    self.ser.write(str(900).encode())

def Lin_close(self):
    """Function for closing the linear drive"""
    self.ser.write(str(800).encode())

def Finalizar(self):
    """Function close serial port"""
    self.ser.close()

```

15.1.7. Visión artificial en un hilo (cam_hilo.py)

```
"""
# File name: cam_hilo
# Developer: Maher Bouymejan
# Contact: maherbouymejan@gmail.com
# Creation date: 10/04/2023
# Description: Function that executes two threads,
               one to display the image of the camera and the other to control the
               arduino,
               the program captures images and detects the filling
               of the cup in different sections.
# Python version: 3.9.0
# Version: 0
"""

from library_imports import *
from Control_servo import Control_Servo

class CameraThread:
    def __init__(self, indice_camara, Port, roi1, roi2, roi3,roi4):
        self.Port=Port
        self.roi1=roi1
        self.roi2=roi2
        self.roi3=roi3
        self.roi4=roi4

        self.stopped = False
        self.stopped_indice = False

    try:
        self.cap = cv2.VideoCapture(indice_camara,cv2.CAP_DSHOW)
        if not self.cap.isOpened():
            print(f"Could not open camera with index {indice_camara}")
            self.stopped_indice = True
```

```

        return

except cv2.error as e:
    print(f"Error opening camera: {e}")
    return

self.act_cam=0
self.act_cam0=0
self.act_cam1=0
self.act_cam2=0
self.act_cam3=0
self.m=0

self.select_roi1= 0
self.select_roi2= 0
self.select_roi3= 0
self.select_roi4= 0
self.roi_show=0

self.arg_arduino_control = None
self.hilo_arduino_control = None
#self.servo = None

self.model_cup = YOLO("YOLOv8_custom/best_Cup.pt")
self.model_beer = YOLO("YOLOv8_custom/best_beer.pt")

def start(self):
    if self.stopped_indice == True:
        return
    else:
        self.create_folders()
        print("Ini thread cam")
        self.thread = threading.Thread(target=self.show_camera)

```

```

self.thread.start()

print("Ini thread servo")
self.change_arduino_control_args(0, 0)

return self

def Arduio_control(self,accion,apertura):
    if not self.stopped:
        if accion==0:
            print("Ini Arduino")
            self.servo=Control_Servo(self.Port)
            time.sleep(1) #6

        elif accion==1:
            print("Open disp cup")
            self.servo.Lin_open()
            time.sleep(1) #6

        elif accion==2:
            print("close disp cup")
            self.servo.Lin_close()
            time.sleep(1) #6

        elif accion==3:
            print("close tap")
            self.servo.Grifo_cerrado()
            time.sleep(1) #6

        elif accion==4:
            print("Open tap ",apertura," % ")

            self.servo.Tap_Porcentaje(apertura)
            time.sleep(1) #6

```

```

        elif accion==5:
            print("Open tap foam")
            self.servo.Grifo_espuma()
            time.sleep(1) #6
        else:
            self.hilo_arduino_control = True
            self.stop_arduino_control()

def set_arduino_control_args(self, accion, apertura):
    self.arg_arduino_control = (accion, apertura)

def run_arduino_control(self):
    accion, apertura = self.arg_arduino_control
    self.hilo_arduino_control = threading.Thread(target=self.Arduino_control,
args=(accion, apertura))
    self.hilo_arduino_control.start()

def change_arduino_control_args(self, accion, apertura):
    self.set_arduino_control_args(accion, apertura)
    self.stop_arduino_control()
    self.run_arduino_control()

def stop_arduino_control(self):
    if self.hilo_arduino_control is not None:
        self.hilo_arduino_control.join()
        print("cerrar hilo arduino")

def actuator_open(self):
    self.change_arduino_control_args(1, 0)

def actuator_close(self):
    self.change_arduino_control_args(2, 0)

```

```

def close_tap(self):
    self.change_arduino_control_args(3, 0)

def open_tap(self,a):
    self.change_arduino_control_args(4, a)

def open_foam(self):
    self.change_arduino_control_args(5, 0)

def create_folders(self):
    #nombre de la carpeta
    fecha_actual = time.strftime("%d-%m-%Y-")
    hora_actual = time.strftime("%H_%M_%S")

    self.Folder_data= "Data_" + str(fecha_actual) + "_" +str(hora_actual)

    self.Folder_REF0="Reference_images_limt_0"
    self.Folder_REF1="Reference_images_limt_1"
    self.Folder_REF2="Reference_images_limt_2"
    self.Folder_REF3="Reference_images_limt_3"

    self.Folder_DETC0="Detect_images_limt_0"
    self.Folder_DETC1="Detect_images_limt_1"
    self.Folder_DETC2="Detect_images_limt_2"
    self.Folder_DETC3="Detect_images_limt_3"

    self.Folder_Beer="Images_Beer"
    self.Folder_Cup="Images_Cup"

    #ruta de la carpeta
    self.Folder_path_data = os.path.join(os.getcwd(), self.Folder_data)

    self.Folder_path_REF0 = os.path.join(self.Folder_path_data,
self.Folder_REF0)

```

```

        self.Folder_path_REF1 = os.path.join(self.Folder_path_data,
self.Folder_REF1)

        self.Folder_path_REF2 = os.path.join(self.Folder_path_data,
self.Folder_REF2)

        self.Folder_path_REF3 = os.path.join(self.Folder_path_data,
self.Folder_REF3)


        self.Folder_path_DETC0 = os.path.join(self.Folder_path_data,
self.Folder_DETC0)

        self.Folder_path_DETC1 = os.path.join(self.Folder_path_data,
self.Folder_DETC1)

        self.Folder_path_DETC2 = os.path.join(self.Folder_path_data,
self.Folder_DETC2)

        self.Folder_path_DETC3 = os.path.join(self.Folder_path_data,
self.Folder_DETC3)


        self.Folder_path_Beer = os.path.join(self.Folder_path_data,
self.Folder_Beer)

        self.Folder_path_Cup = os.path.join(self.Folder_path_data,
self.Folder_Cup)


# crear la carpeta si no existe
if not os.path.exists(self.Folder_path_data):
    os.makedirs(self.Folder_path_data)


if not os.path.exists(self.Folder_path_REF0):
    os.makedirs(self.Folder_path_REF0)


if not os.path.exists(self.Folder_path_REF1):
    os.makedirs(self.Folder_path_REF1)


if not os.path.exists(self.Folder_path_REF2):
    os.makedirs(self.Folder_path_REF2)

```

```

if not os.path.exists(self.Folder_path_REF3):
    os.makedirs(self.Folder_path_REF3)

if not os.path.exists(self.Folder_path_DETC0):
    os.makedirs(self.Folder_path_DETC0)

if not os.path.exists(self.Folder_path_DETC1):
    os.makedirs(self.Folder_path_DETC1)

if not os.path.exists(self.Folder_path_DETC2):
    os.makedirs(self.Folder_path_DETC2)

if not os.path.exists(self.Folder_path_DETC3):
    os.makedirs(self.Folder_path_DETC3)

if not os.path.exists(self.Folder_path_Beer):
    os.makedirs(self.Folder_path_Beer)

if not os.path.exists(self.Folder_path_Cup):
    os.makedirs(self.Folder_path_Cup)

def show_camera(self):
    global x0,y0,w0,h0,x1,y1,w1,h1,x2,y2,w2,h2,x3,y3,w3,h3,font

    font= cv2.FONT_HERSHEY_DUPLEX

    x0,y0,w0,h0=self.roi1
    x1,y1,w1,h1=self.roi2
    x2,y2,w2,h2=self.roi3
    x3,y3,w3,h3=self.roi4

    while not self.stopped:
        #-----

```

```

ret, self.frame = self.cap.read()

#self.height, self.width, channels = self.frame.shape

self.frame_C = copy.deepcopy(self.frame)
self.frame_rgb = copy.deepcopy(self.frame)
self.frame_roi = copy.deepcopy(self.frame)

self.frame_gray = cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY)

if cv2.waitKey(1) & 0xFF == ord('q'):
    self.stopped = True
    break

if self.select_roi1== 1 :
    self.roi1= cv2.selectROI('Select the region of interest 1 with the
mouse.',self.frame, False,False)
    x0,y0,w0,h0 = self.roi1
    print(self.roi1)
    self.select_roi1=0
    # Abrir el archivo y leer su contenido
    with open('txt_param\cv_conf.txt', 'r') as f:
        lines = f.readlines()

    # Modificar la línea 2
    lines[0] = f'roi1={self.roi1}\n'

    # Guardar las líneas actualizadas en el archivo
    with open('txt_param\cv_conf.txt', 'w') as f:
        f.writelines(lines)
    cv2.destroyWindow('Select the region of interest 1 with the mouse.')

if self.select_roi2== 1 :

```

```

        self.roi2 = cv2.selectROI('Select the region of interest 2 with the
mouse.',self.frame, False,False)
        x1,y1,w1,h1 = self.roi2
        print(self.roi2)
        self.select_roi2=0

    # Abrir el archivo y leer su contenido
    with open('txt_param\cv_conf.txt', 'r') as f:
        lines = f.readlines()

    # Modificar la línea 2
    lines[1] = f'roi2={self.roi2}\n'

    # Guardar las líneas actualizadas en el archivo
    with open('txt_param\cv_conf.txt', 'w') as f:
        f.writelines(lines)

    cv2.destroyWindow('Select the region of interest 2 with the mouse.')

    if self.select_roi3== 1 :
        self.roi3 = cv2.selectROI('Select the region of interest 3 with the
mouse.',self.frame, False,False)
        x2,y2,w2,h2 = self.roi3
        print(self.roi3)
        self.select_roi3=0

    # Abrir el archivo y leer su contenido
    with open('txt_param\cv_conf.txt', 'r') as f:
        lines = f.readlines()

    # Modificar la línea 2
    lines[2] = f'roi3={self.roi3}\n'

    # Guardar las líneas actualizadas en el archivo

```

```

with open('txt_param\cv_conf.txt', 'w') as f:
    f.writelines(lines)

cv2.destroyAllWindows('Select the region of interest 3 with the mouse.')

if self.select_roi4== 1 :
    self.roi4= cv2.selectROI('Select the region of interest 4 with the
mouse.',self.frame, False,False)
    x3,y3,w3,h3 = self.roi4
    print(self.roi4)
    self.select_roi4=0

# Abrir el archivo y leer su contenido
with open('txt_param\cv_conf.txt', 'r') as f:
    lines = f.readlines()

# Modificar la línea 2
lines[3] = f'roi4={self.roi4}\n'

# Guardar las líneas actualizadas en el archivo
with open('txt_param\cv_conf.txt', 'w') as f:
    f.writelines(lines)

cv2.destroyAllWindows('Select the region of interest 4 with the mouse.')

#-----
if self.act_cam0== 1 :
    cv2.rectangle(self.frame_rgb, (x0, y0), (x0+w0, y0+h0), (0, 0, 255), 2)

if self.act_cam1== 1 :
    cv2.rectangle(self.frame_rgb, (x1, y1), (x1+w1, y1+h1), (0, 255, 0), 2)

if self.act_cam2== 1 :
    cv2.rectangle(self.frame_rgb, (x2, y2), (x2+w2, y2+h2), (255, 0, 0), 2)

```

```

if self.act_cam3== 1 :
    cv2.rectangle(self.frame_rgb, (x3, y3), (x3+w3, y3+h3), (255, 250, 0),
2)

#-----
if self.roi_show==1:
    cv2.rectangle(self.frame_roi, (x0, y0), (x0+w0, y0+h0), (0, 0, 255), 2)
    cv2.rectangle(self.frame_roi, (x1, y1), (x1+w1, y1+h1), (0, 255, 0), 2)
    cv2.rectangle(self.frame_roi, (x2, y2), (x2+w2, y2+h2), (255, 0, 0), 2)
    cv2.rectangle(self.frame_roi, (x3, y3), (x3+w3, y3+h3), (255, 255,0),
2)

    cv2.imshow("ROI show", self.frame_roi)

if self.roi_show== -1:
    if cv2.getWindowProperty("ROI show", cv2.WND_PROP_VISIBLE) !=
-1:

        cv2.destroyWindow("ROI show")
        self.roi_show=0

#-----
if self.act_cam== 1 :
    #cv2.namedWindow("Cam Bot", cv2.WND_PROP_FULLSCREEN)
    #cv2.setWindowProperty("Cam Bot",
cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)

    cv2.namedWindow("Cam Bot") # Crear ventana para mostrar el
frame original
    # Mostrar el frame original y el mejorado en sus respectivas ventanas
    cv2.imshow("Cam Bot", self.frame_rgb)

    #cv2.namedWindow("Cam Bot", cv2.WINDOW_NORMAL)
    #cv2.imshow("Cam Bot", cv2.resize(self.frame_rgb, (self.height,
self.width)))

```

```

        elif self.act_cam== -1 :
            cv2.destroyAllWindows("Cam Bot")
            self.act_cam=0

def Visual_cam(self,Vcam,id_beer):
    self.act_cam=Vcam
    self.id_beer=id_beer

def roi_select(self,a):
    if a==1:
        self.select_roi1= 1
    if a==2:
        self.select_roi2= 1
    if a==3:
        self.select_roi3= 1
    if a==4:
        self.select_roi4= 1
    if a==5:
        self.roi_show=1
    if a==6:
        self.roi_show=-1

def llenar_cerveza(self,apertura,threshold,threshold_np,limt_max1,seguro0):

    if 0 <= apertura <= 100:
        t1 = time.time()
        self.imag_reff0=self.capt0()
        self.open_tap(apertura)
        print("Buscando cambios")
        self.act_cam0=1
        while True:
            tcont1 = time.time()
            roi0=self.frame_gray[y0 : y0 + h0, x0 : x0 + w0]

```

```

# Calcular la diferencia absoluta entre los dos cuadros
diff = cv2.absdiff(self.imag_reff0,roi0)
# Establecer un umbral para detectar cambios significativos
thresh = cv2.threshold(diff,threshold, 255, cv2.THRESH_BINARY)
Diff_Mat = np.uint8(thresh[1])
det=np.count_nonzero(Diff_Mat)
duracion_directo = int(tcont1 - t1)

if duracion_directo > (limt_max1*seguro0):
    self.close_tap()
    #print(" Error cerveza no detectada ")
    self.duracion1=1000
    self.act_cam0=-1
    print("No beer")
    break

if det > threshold_np:
    t2 = time.time()
    self.duracion1=int(t2 - t1)

    img_det0=self.frame_C
    print("cerveza detectada 1",self.duracion1)

    self.Folder_imagD0=os.path.join(self.Folder_path_DETC0,
"Beer_0_Det_{}.png".format(self.id_beer))
    cv2.imwrite(self.Folder_imagD0, img_det0)
    self.act_cam0=-1
    break

else:
    #print("No se detectó cambio")
    pass

else:

```

```

    print("Error porcentaje de apertura")
    self.duracion1=100

return self.duracion1

def llenar_cerveza2(self,threshold,threshold_np,limt_max2,seguro1):
    t3 = time.time()
    self.imag_reff1=self.capt1()

    print("Buscando cambios")
    self.act_cam1=1
    while True:
        tcont2 = time.time()
        roi1=self.frame_gray[y1 : y1 + h1, x1 : x1 + w1]
        # Calcular la diferencia absoluta entre los dos cuadros
        diff = cv2.absdiff(self.imag_reff1,roi1)
        # Establecer un umbral para detectar cambios significativos
        thresh = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)
        Diff_Mat = np.uint8(thresh[1])
        det=np.count_nonzero(Diff_Mat)
        duracion_directo = int(tcont2 - t3)

        if duracion_directo > (limt_max2*seguro1):
            self.close_tap()
            print(" Error cerveza no detectada ")
            duracion2=1000
            self.act_cam1=-1
            break

        if det > threshold_np:
            t4 = time.time()
            duracion2=int(t4 - t3)
            img_det1=self.frame_C
            print("cerveza detectada2",duracion2)

```

```

        self.Folder_imagD1=os.path.join(self.Folder_path_DETC1,
"Beer_1_Det_{}.png".format(self.id_beer))
        cv2.imwrite(self.Folder_imagD1, img_det1)

        self.act_cam1=-1
        break

    else:
        #print("No se detectó cambio")
        pass

    return duracion2

def llenar_cerveza3(self,threshold,threshold_np,limt_max3,seguro2):
    t3 = time.time()
    self.imag_reff1=self.capt2()
    print("Buscando cambios")
    self.act_cam2=1
    while True:

        tcont2 = time.time()
        roi3=self.frame_gray[y2 : y2 + h2, x2 : x2 + w2]
        # Calcular la diferencia absoluta entre los dos cuadros
        diff = cv2.absdiff(self.imag_reff1,roi3)

        # Establecer un umbral para detectar cambios significativos
        thresh = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)
        Diff_Mat = np.uint8(thresh[1])
        det=np.count_nonzero(Diff_Mat)
        duracion_directo = int(tcont2 - t3)

        if duracion_directo > (limt_max3*seguro2):
            self.close_tap()

```

```

        print(" Error cerveza no detectada ")
        duracion3=1000

        self.act_cam2=-1
        break

    # La función numpy.any() devuelve True si al menos un elemento de la
    matriz es True, y False en caso contrario
    if det > threshold_np:
        self.close_tap()
        t4 = time.time()
        duracion3=int(t4 - t3)

        img_det1=self.frame_C
        print("cerveza detectada 3",duracion3)

        self.Folder_imagD2=os.path.join(self.Folder_path_DETC2,
        "Beer_2_Det_{}.png".format(self.id_beer))
        cv2.imwrite(self.Folder_imagD2, img_det1)
        self.act_cam2=-1
        break
    else:
        #print("No se detectó cambio")
        pass

    return duracion3

def llenar_espuma(self,threshold,threshold_np,limt_max4,seguro3):
    t4 = time.time()
    self.imag_reff4=self.capt_foam()
    print("Buscando cambios")
    self.act_cam3=1
    self.open_foam()

```

while True:

```
tcont4 = time.time()
roi4=self.frame_gray[y3 : y3 + h3, x3 : x3 + w3]
# Calcular la diferencia absoluta entre los dos cuadros
diff = cv2.absdiff(self.imag_reff4,roi4)

# Establecer un umbral para detectar cambios significativos
thresh = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)
Diff_Mat = np.uint8(thresh[1])
det=np.count_nonzero(Diff_Mat)
duracion_directo = int(tcont4 - t4)
```

```
if duracion_directo > (limt_max4*seguro3):
```

```
    self.close_tap()
    print(" Error cerveza no detectada llenado 4 ")
    duracion4=1000
    self.act_cam3=-1
    break
```

La función numpy.any() devuelve True si al menos un elemento de la matriz es True, y False en caso contrario

```
if det > threshold_np:
    self.close_tap()
    t44 = time.time()
    duracion4=int(t44 - t4)
```

```
img_det1=self.frame_C
print("cerveza detectada 4-->",duracion4)
```

```
self.Folder_imagD3=os.path.join(self.Folder_path_DETC3,
"Beer_3_Det_{}.png".format(self.id_beer))
cv2.imwrite(self.Folder_imagD3, img_det1)
self.act_cam3=-1
break
```

```

        else:
            #print("No se detectó cambio")
            pass

    return duracion4

def capt0(self):
    img_ref0=self.frame_C

    self.Folder_imagR0=os.path.join(self.Folder_path_REF0,
    "Beer_0_REF_{}.png".format(self.id_beer))
    cv2.imwrite(self.Folder_imagR0, img_ref0)

    img_ref0 = self.frame_gray[y0 : y0 + h0, x0 : x0 + w0]
    print("cap Beer_Ref_0")
    return img_ref0

def capt1(self):
    img_ref11=self.frame_C

    self.Folder_imagR1=os.path.join(self.Folder_path_REF1,
    "Beer_1_REF_{}.png".format(self.id_beer))
    cv2.imwrite(self.Folder_imagR1, img_ref11)

    img_ref1 = self.frame_gray[y1 : y1 + h1, x1 : x1 + w1]
    print("cap Beer_Ref_1")
    return img_ref1

def capt2(self):

    img_ref22=self.frame_C

    self.Folder_imagR2=os.path.join(self.Folder_path_REF2,
    "Beer_2_REF_{}.png".format(self.id_beer))

```

```

cv2.imwrite(self.Folder_imagR2, img_ref22)

img_ref2 = self.frame_gray[y2 : y2 + h2, x2 : x2 + w2]
print("cap beer2_Ref")
return img_ref2

def capt_foam(self):

    img_ref22=self.frame_C

    self.Folder_imagR3=os.path.join(self.Folder_path_REF3,
    "Beer_3_REF_{}.png".format(self.id_beer))
    cv2.imwrite(self.Folder_imagR3, img_ref22)

    img_ref2 = self.frame_gray[y3 : y3 + h3, x3 : x3 + w3]
    print("cap beer3_Ref")
    return img_ref2

def capt3(self):

    self.img_ref33=self.frame_C

    image = np.asarray(self.img_ref33)

    results = self.model_beer.predict(source=image ,save=True,show=False,
conf=0.5)
    try:
        #box=[x1, y1, x2, y2, score, label]
        box=results[0].boxes.tolist()[0]
        print(box)
        for r in results:
            for c in r.boxes.cls:
                name_detect = self.model_beer.names[int(c)]
                print(" The detected part is : ",name_detect," box = ",box)

```

```

except:
    print("No cup detected")
    cup=False
    box=0

    self.Folder_imag_Beer=os.path.join(self.Folder_path_Beer,
"Beer_{}.png".format(self.id_beer))
    cv2.imwrite(self.Folder_imag_Beer, self.img_ref33)

    print("cap Image beer")
    return self.img_ref33

def capt4(self):

    self.img_ref44=self.frame_C

    image = np.asarray(self.img_ref44)
    results = self.model_cup.predict(source=image ,save=True,show=False,
conf=0.5)
    try:
        #box=[x1, y1, x2, y2, score, label]
        box=results[0].boxes.tolist()[0]
        print(box)
        for r in results:
            for c in r.boxes.cls:
                name_detect = self.model_cup.names[int(c)]
                print(" The detected part is : ",name_detect," box = ",box)
            cup=True
    except:
        print("No cup detected")
        cup=False
        box=0

```

```
        self.Folder_imag_Cup=os.path.join(self.Folder_path_Cup,
"Cup_{}.png".format(self.id_beer))
        cv2.imwrite(self.Folder_imag_Cup, self.img_ref44)

        print("cap Image cup")

        return cup, box, self.img_ref44

def stop(self):
    self.stopped = True
    self.cap.release()
    cv2.destroyAllWindows()
```

15.1.8. Interfaz de usuario y Demo (Gui_beer1.py)

```
"""
# File name: Gui_beer1
# Developer: Maher Bouymejan
# Contact: maherbouymejan@gmail.com
# Creation date: 10/04/2023
# Description: Beer.
# Python version: 3.9.0
# Version: 1
"""

from library_imports import *
from bot_hilo import HiloCobot
from cam_hilo import CameraThread
from Database_beer import MongoDBManager

class Window1(QMainWindow):
    def __init__(self):
        super(Window1, self).__init__()

        # Cargar el archivo .ui
        loadUi('GUI_ui\GUI_BEER.ui', self)

        #Iniciamos la lectura de los parametros iniciales
        self.ini_cv()
        self.ini_AI()
        self.ini_tab_param()
        self.ini_tab()

        self.cambiar_encuesta()

        #valores iniciales
        self.cont_beer=0
```

```

self.error_tipo=0

self.mov=0
self.show_cam = False
self.motor_cup = False

#indice de las paginas de la interfaz
self.pag_Home=0
self.pag_demo=1
self.pag_wait=2
self.pag_serv=3
self.pag_error=4
self.pag_poss=5
self.pag_param=6
self.pag_ai=7
self.pag_cv=8
self.pag_anal=9
self.pag_test=10
self.pag_opinion=11

print("INI APP")
self.progressBar_ini.setValue(0)
self.progressBar_beer.setValue(0)

#conectar todos los botones
self.connect_Button(0)
#pagina espera
self.stackedWidget.setCurrentIndex(self.pag_wait)

#esperar un poco antes de mostrar la siguiente página
QTimer.singleShot(1000, self.ini_class)

def ver_estado_wait(self,valor,text):
    self.valor=valor

```

```

self.progressBar_ini.setValue(self.valor)

self.mensaje_estado=text
self.label_estado.setText(self.mensaje_estado)
QApplication.processEvents()

def ver_estado_beer(self,valor,text):
    self.valor_beer=valor
    self.progressBar_beer.setValue(self.valor_beer)

    self.mensaje_estado_beer=text
    self.label_estado_beer.setText(self.mensaje_estado_beer)
    QApplication.processEvents()

def ini_class(self ):

    self.ver_estado_wait(30,"Initiation cam")

    try:
        pass
        self.cam_hilo =
CameraThread(self.indice_camara,self.Port,self.roi1,self.roi2,self.roi3,self.roi4).
start()
    except:
        print("Error INI CAM !")
        self.ver_estado_wait(50,"Error INI CAM ! Is the usb of the webcam and
the arduino connected?")
        time.sleep(5)
        self.error_tipo = 4

    self.ver_estado_wait(70,"Initiation Bot ")

    try:
        self.hilo_bot = HiloCobot(self.lp)

```

```

        self.hilo_bot.start()
    except:
        print("Error INI BOT !!!!")
        self.ver_estado_wait(80,"Error INI BOT ! Is the ethernet port
connected?")
        if self.error_tipo == 4:
            self.error_tipo =7
        else:
            self.error_tipo =5
        time.sleep(5)

    self.ver_estado_wait(90,"Initiation database")

    try:
        self.mongo_manager =
MongoDBManager("mongodb+srv://maher:1234@beer.uoimnt2.mongodb.net/b
eer_bot?retryWrites=true&w=majority")

    except:
        print("Error INI DATABASE")
        self.ver_estado_wait(95,"Error INI DATABASE ! Do i have internet
access?")
        if self.error_tipo == 0:
            self.error_tipo =6
        time.sleep(5)

    self.ver_estado_wait(99,"Initiation")

    if self.error_tipo == 4:
        self.sincam()

    elif self.error_tipo == 5:
        self.sinbot()

```

```

elif self.error_tipo == 6:
    self.sinDB()

elif self.error_tipo == 7:
    self.sincam_bot()

else:
    self.stackedWidget.setCurrentIndex(self.pag_Home)

def connect_Button(self,indx):

    if indx==0:
        self.pushButton_exit.clicked.connect(self.on_push_exit_clicked)
        self.pushButton_help.clicked.connect(self.on_push_help_clicked)

        #self.pushButton_analytics.clicked.connect(lambda:
self.stackedWidget.setCurrentIndex(4))

self.pushButton_analytics.clicked.connect(self.on_pushButton_analytics_clicked
)

self.pushButton_settings.clicked.connect(self.on_pushButton_settings_clicked)
    self.pushButton_test.clicked.connect(self.on_pushButton_test_clicked)

    self.pushButton_AI.clicked.connect(self.on_pushButton_AI_clicked)
    self.pushButton_CV.clicked.connect(self.on_pushButton_CV_clicked)

self.pushButton_param.clicked.connect(self.on_pushButton_param_clicked)

self.pushButton_poss.clicked.connect(self.on_pushButton_poss_clicked)

self.pushButton_demo0.clicked.connect(self.on_pushButton_demo0_clicked)

```

```
self.pushButton_demo1.clicked.connect(self.on_pushButton_demo1_clicked)
```

```
self.pushButton_demo2.clicked.connect(self.on_pushButton_demo2_clicked)
```

```
self.pushButton_reset.clicked.connect(self.on_pushButton_reset_clicked)
```

```
self.pushButton_save.clicked.connect(self.on_pushButton_save_clicked)
```

```
self.pushButton_default.clicked.connect(self.on_pushButton_default_clicked)
```

```
self.pushButton_save_Param.clicked.connect(self.on_pushButton_save_Param_clicked)
```

```
self.pushButton_default_param.clicked.connect(self.on_pushButton_default_param_clicked)
```

```
self.pushButton_save_ai.clicked.connect(self.on_pushButton_save_ai_clicked)
```

```
self.pushButton_default_ai.clicked.connect(self.on_pushButton_default_ai_clicked)
```

```
self.pushButton_save_cv.clicked.connect(self.on_pushButton_save_cv_clicked)
```

```
self.pushButton_default_cv.clicked.connect(self.on_pushButton_default_cv_clicked)
```

```
self.pushButton_roi1.clicked.connect(self.on_pushButton_roi1_clicked)
```

```
self.pushButton_roi2.clicked.connect(self.on_pushButton_roi2_clicked)
```

```
self.pushButton_roi3.clicked.connect(self.on_pushButton_roi3_clicked)
self.pushButton_roi4.clicked.connect(self.on_pushButton_roi4_clicked)
```

```
self.pushButton_threshold.clicked.connect(self.on_pushButton_threshold_clicked)
d)
```

```
self.pushButton_next.clicked.connect(self.on_pushButton_next_clicked)
self.pushButton_last.clicked.connect(self.on_pushButton_last_clicked)
```

```
self.pushButton_taps.clicked.connect(self.on_pushButton_taps_clicked)
```

```
self.pushButton_tap_tocar.clicked.connect(self.on_pushButton_tap_tocar_clicked)
d)
```

```
self.pushButton_reset_bot.clicked.connect(self.on_pushButton_reset_bot_clicked)
d)
```

```
self.pushButton_reset_cam.clicked.connect(self.on_pushButton_reset_cam_clicked)
ked)
```

```
self.pushButton_reset_db.clicked.connect(self.on_pushButton_reset_db_clicked)
)
```

```
self.pushButton_cam.clicked.connect(self.on_pushButton_cam_clicked)
self.pushButton_cup.clicked.connect(self.on_pushButton_cup_clicked)
```

```
self.pushButton_enviar.clicked.connect(self.on_pushButton_enviar_clicked)
```

```
self.pushButton_1.clicked.connect(self.on_pushButton_1_clicked)
self.pushButton_2.clicked.connect(self.on_pushButton_2_clicked)
self.pushButton_3.clicked.connect(self.on_pushButton_3_clicked)
```

```

self.pushButton_4.clicked.connect(self.on_pushButton_4_clicked)
self.pushButton_5.clicked.connect(self.on_pushButton_5_clicked)

#
self.pushButton_studio.clicked.connect(self.on_pushButton_studio_clicked)

self.pushButton_mov_home.clicked.connect(self.on_pushButton_mov_home_clicked)

# Establecer el estado inicial del QFrame
self.widget_conf2.setVisible(False)

elif indx==1:

self.pushButton_analytics.clicked.connect(self.on_pushButton_analytics_clicked
)

elif indx==2:

self.pushButton_settings.clicked.connect(self.on_pushButton_settings_clicked)

elif indx==3:
    self.pushButton_test.clicked.connect(self.on_pushButton_test_clicked)

elif indx==4:
    self.pushButton_AI.clicked.connect(self.on_pushButton_AI_clicked)

elif indx==5:
    self.pushButton_CV.clicked.connect(self.on_pushButton_CV_clicked)

elif indx==6:

self.pushButton_param.clicked.connect(self.on_pushButton_param_clicked)

```

elif indx==7:

self.pushButton_poss.clicked.connect(self.on_pushButton_poss_clicked)

elif indx==8:

self.pushButton_demo0.clicked.connect(self.on_pushButton_demo0_clicked)

elif indx==9:

self.pushButton_demo1.clicked.connect(self.on_pushButton_demo1_clicked)

elif indx==10:

self.pushButton_demo2.clicked.connect(self.on_pushButton_demo2_clicked)

elif indx==11:

self.pushButton_reset.clicked.connect(self.on_pushButton_reset_clicked)

elif indx==12:

self.pushButton_save.clicked.connect(self.on_pushButton_save_clicked)

elif indx==13:

self.pushButton_default.clicked.connect(self.on_pushButton_default_clicked)

elif indx==14:

self.pushButton_save_Param.clicked.connect(self.on_pushButton_save_Param_clicked)

elif indx==15:

self.pushButton_default_param.clicked.connect(self.on_pushButton_default_param_clicked)

elif indx==16:

self.pushButton_save_ai.clicked.connect(self.on_pushButton_save_ai_clicked)

elif indx==17:

self.pushButton_default_ai.clicked.connect(self.on_pushButton_default_ai_clicked)

elif indx==18:

self.pushButton_save_cv.clicked.connect(self.on_pushButton_save_cv_clicked)

elif indx==19:

self.pushButton_default_cv.clicked.connect(self.on_pushButton_default_cv_clicked)

elif indx==20:

self.pushButton_roi1.clicked.connect(self.on_pushButton_roi1_clicked)

elif indx==21:

self.pushButton_roi2.clicked.connect(self.on_pushButton_roi2_clicked)

elif indx==22:

self.pushButton_roi3.clicked.connect(self.on_pushButton_roi3_clicked)

elif indx==23:

self.pushButton_roi4.clicked.connect(self.on_pushButton_roi4_clicked)

elif indx==24:

self.pushButton_threshold.clicked.connect(self.on_pushButton_threshold_clicked)

elif indx==25:

self.pushButton_next.clicked.connect(self.on_pushButton_next_clicked)

elif indx==26:

self.pushButton_last.clicked.connect(self.on_pushButton_last_clicked)

elif indx==27:

self.pushButton_taps.clicked.connect(self.on_pushButton_taps_clicked)

elif indx==28:

self.pushButton_tap_tocar.clicked.connect(self.on_pushButton_tap_tocar_clicked)

elif indx==29:

self.pushButton_enviar.clicked.connect(self.on_pushButton_enviar_clicked)

elif indx==30:

self.pushButton_1.clicked.connect(self.on_pushButton_1_clicked)

elif indx==31:

self.pushButton_2.clicked.connect(self.on_pushButton_2_clicked)

elif indx==32:

self.pushButton_3.clicked.connect(self.on_pushButton_3_clicked)

elif indx==33:

```

        self.pushButton_4.clicked.connect(self.on_pushButton_4_clicked)

elif indx==34:
    self.pushButton_5.clicked.connect(self.on_pushButton_5_clicked)

elif indx==35:

self.pushButton_reset_bot.clicked.connect(self.on_pushButton_reset_bot_clicked)

elif indx==36:
    pass

self.pushButton_reset_cam.clicked.connect(self.on_pushButton_reset_cam_clicked)

elif indx==37:

self.pushButton_reset_db.clicked.connect(self.on_pushButton_reset_db_clicked
)

elif indx==38:
    self.pushButton_cam.clicked.connect(self.on_pushButton_cam_clicked)

elif indx==39:
    self.pushButton_cup.clicked.connect(self.on_pushButton_cup_clicked)

elif indx==40:
    #
self.pushButton_studio.clicked.connect(self.on_pushButton_studio_clicked)
    pass

elif indx==41:

```

```
self.pushButton_mov_home.clicked.connect(self.on_pushButton_mov_home_clicked)
```

```
def on_pushButton_mov_home_clicked(self):  
    self.pushButton_mov_home.clicked.disconnect()  
    print("home")  
    self.hilo_bot.go_home()
```

```
self.connect_Button(41)
```

```
# def on_pushButton_studio_clicked(self):  
#     self.pushButton_studio.clicked.disconnect()  
#     ip="192.168.1.203:18333"  
#     url=QUrl("http://" + ip)  
#     QDesktopServices.openUrl(url)
```

```
#     self.connect_Button(40)
```

```
def on_pushButton_cam_clicked(self):  
    self.pushButton_cam.clicked.disconnect()  
    if self.show_cam == False:  
        self.cam_hilo.Visual_cam(1,self.cont_beer)  
        self.show_cam = True  
    else:  
        self.cam_hilo.Visual_cam(-1,self.cont_beer)  
        self.show_cam = False  
    self.connect_Button(38)
```

```
def on_pushButton_cup_clicked(self):  
    self.pushButton_cup.clicked.disconnect()
```

```

if self.motor_cup == False:
    self.cam_hilo.actuator_open()
    self.motor_cup = True
else:
    self.cam_hilo.actuator_close()
    self.motor_cup = False

self.connect_Button(39)

def on_pushButton_next_clicked(self):
    self.pushButton_next.clicked.disconnect()

    if self.mov <0:
        self.mov=0

    if 0 <= self.mov <= 32:

        self.hilo_bot.mov_poss(self.mov)
        print("poss : ", self.mov)
        if self.mov==32:
            self.mov=0
        else:
            self.mov=self.mov+1
        self.pushButton_last.setEnabled(True)

    else:
        self.pushButton_next.setEnabled(False)

self.connect_Button(25)

def on_pushButton_last_clicked(self):
    self.pushButton_last.clicked.disconnect()

```

```

self.mov=self.mov-1

if self.mov >= 32:
    self.mov=31

if 0 <= self.mov <= 31:
    self.hilo_bot.mov_poss(self.mov)
    print("poss : ", self.mov)
    self.pushButton_next.setEnabled(True)

else:
    self.pushButton_last.setEnabled(False)

self.connect_Button(26)

def on_pushButton_reset_bot_clicked(self):
    self.pushButton_reset_bot.clicked.disconnect()
    self.ini_tab_param()
    self.ini_tab()
    print("reset bot")
    self.hilo_bot.reset()
    self.connect_Button(35)

def on_pushButton_reset_cam_clicked(self):
    self.pushButton_reset_cam.clicked.disconnect()
    self.ini_cv()
    self.ini_tab_param()

try:
    self.cam_hilo.stop()
except:
    print("camera not initialised before")

time.sleep(2)

```

```

try:
    self.cam_hilo =
CameraThread(self.indice_camara,self.Port,self.roi1,self.roi2,self.roi3,self.roi4).
start()
except:
    print("Error reset cam")
    self.connect_Button(36)

def on_pushButton_reset_db_clicked(self):
    self.pushButton_reset_db.clicked.disconnect()
    try:
        self.mongo_manager =
MongoDBManager("mongodb+srv://maher:1234@beer.uoimnt2.mongodb.net/b
eer_bot?retryWrites=true&w=majority")
    except:
        print("Error reset DATABASE")

    self.connect_Button(37)

def on_pushButton_taps_clicked(self):
    self.pushButton_taps.clicked.disconnect()
    self.cam_hilo.close_tap()
    self.connect_Button(27)

def on_pushButton_tap_tocar_clicked(self):
    self.pushButton_tap_tocar.clicked.disconnect()
    self.t=1
    self.pushButton_tap_tocar.pressed.connect(self.print_button_pressed)

    self.pushButton_tap_tocar.released.connect(self.stop_printing)

    self.connect_Button(28)

def print_button_pressed(self):

```

```

if self.t==1:
    self.pushButton_tap_tocar.clicked.disconnect()
    print("El botón está presionado")
    self.cam_hilo.open_tap(90)
    self.connect_Button(28)
    self.t=2

def stop_printing(self):
    if self.t==2:
        self.pushButton_tap_tocar.clicked.disconnect()
        self.cam_hilo.close_tap()
        print("El botón fue liberado")
        self.connect_Button(28)
        self.t=1

def on_pushButton_enviar_clicked(self):
    self.pushButton_enviar.clicked.disconnect()

    #self.stackedWidget.setCurrentIndex(self.pag_Home)
    self.stackedWidget.setCurrentIndex(self.pag_demo)

    self.texto = self.textEdit.toPlainText()
    self.select_error()

    self.hora_ini=time.strftime("%H_%M_%S")

    try:
        self.mongo_manager.insert_opinions(
            id=self.cont_beer,
            date= self.hora_ini,
            stars = self.stars,
            failure_type=self.fallo_tipo,
            failure=self.fallo,
            opinion= self.texto,

```

```

        others= 0,
    )

except:
    pass

self.cambiar_encuesta()

self.connect_Button(29)

def cambiar_encuesta(self):

    self.pushButton_1.setEnabled(True)
    self.pushButton_2.setEnabled(True)
    self.pushButton_3.setEnabled(True)
    self.pushButton_4.setEnabled(True)
    self.pushButton_5.setEnabled(True)

    self.radioButton_0.setChecked(True)
    self.radioButton_1.setChecked(False)
    self.radioButton_2.setChecked(False)
    self.radioButton_3.setChecked(False)
    self.radioButton_4.setChecked(False)

    self.textEdit.clear()

def on_pushButton_1_clicked(self):
    self.pushButton_1.clicked.disconnect()

    self.pushButton_1.setEnabled(False)
    self.pushButton_2.setEnabled(True)
    self.pushButton_3.setEnabled(True)
    self.pushButton_4.setEnabled(True)

```

```

self.pushButton_5.setEnabled(True)

self.stars=1

self.connect_Button(30)

def on_pushButton_2_clicked(self):
    self.pushButton_2.clicked.disconnect()

    self.pushButton_2.setEnabled(False)
    self.pushButton_1.setEnabled(True)
    self.pushButton_3.setEnabled(True)
    self.pushButton_4.setEnabled(True)
    self.pushButton_5.setEnabled(True)

    self.stars=2

    self.connect_Button(31)

def on_pushButton_3_clicked(self):
    self.pushButton_3.clicked.disconnect()

    self.pushButton_3.setEnabled(False)
    self.pushButton_2.setEnabled(True)
    self.pushButton_1.setEnabled(True)
    self.pushButton_4.setEnabled(True)
    self.pushButton_5.setEnabled(True)

    self.stars=3

    self.connect_Button(32)

def on_pushButton_4_clicked(self):
    self.pushButton_4.clicked.disconnect()

```

```

self.pushButton_4.setEnabled(False)
self.pushButton_2.setEnabled(True)
self.pushButton_1.setEnabled(True)
self.pushButton_3.setEnabled(True)
self.pushButton_5.setEnabled(True)

self.stars=4

self.connect_Button(33)

def on_pushButton_5_clicked(self):
    self.pushButton_5.clicked.disconnect()

    self.pushButton_5.setEnabled(False)
    self.pushButton_1.setEnabled(True)
    self.pushButton_3.setEnabled(True)
    self.pushButton_4.setEnabled(True)
    self.pushButton_2.setEnabled(True)

    self.stars=5

    self.connect_Button(34)

def select_error(self):

    if self.radioButton_0.isChecked():
        #print(self.radioButton_1.text())
        self.fallo=self.radioButton_0.text()
        self.fallo_tipo=1

    if self.radioButton_1.isChecked():
        #print(self.radioButton_1.text())
        self.fallo=self.radioButton_1.text()

```

```

self.fallo_tipo=2

elif self.radioButton_2.isChecked():
    #print(self.radioButton_2.text())
    self.fallo=self.radioButton_2.text()
    self.fallo_tipo=3

elif self.radioButton_3.isChecked():
    #print(self.radioButton_3.text())
    self.fallo=self.radioButton_3.text()
    self.fallo_tipo=4

elif self.radioButton_4.isChecked():
    #print(self.radioButton_4.text())
    self.fallo=self.radioButton_4.text()
    self.fallo_tipo=5

#else:
    #print("Ningún botón de radio está seleccionado")
    #self.fallo_tipo=0
    #self.fallo="no failures"
    #self.fallo_tipo=0

def ini_cv(self):
    # Abrir el archivo de configuración
    with open('txt_param\cv_conf.txt', 'r') as f:
        config_lines = f.readlines()

    # Leer los valores de la tabla desde el archivo
    self.roi1 = ast.literal_eval(config_lines[0].split('=')[1].strip())
    self.roi2 = ast.literal_eval(config_lines[1].split('=')[1].strip())
    self.roi3 = ast.literal_eval(config_lines[2].split('=')[1].strip())
    self.roi4 = ast.literal_eval(config_lines[3].split('=')[1].strip())
    self.threshold = int(config_lines[4].split('=')[1].strip())

```

```
self.threshold_np = int(config_lines[5].split('=')[1].strip())
```

```
table13 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_13')
```

```
table13.setItem(0, 0, QTableWidgetItem(str(self.roi1)))
```

```
table13.setItem(1, 0, QTableWidgetItem(str(self.roi2)))
```

```
table13.setItem(2, 0, QTableWidgetItem(str(self.roi3)))
```

```
table13.setItem(3, 0, QTableWidgetItem(str(self.roi4)))
```

```
table13.setItem(4, 0, QTableWidgetItem(str(self.threshold)))
```

```
table13.setItem(5, 0, QTableWidgetItem(str(self.threshold_np)))
```

```
def on_pushButton_save_cv_clicked(self):
```

```
    self.pushButton_save_cv.clicked.disconnect()
```

```
    table13 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_13')
```

```
    self.roi1 = (table13.item(0, 0).text())
```

```
    self.roi2 = (table13.item(1, 0).text())
```

```
    self.roi3 = (table13.item(2, 0).text())
```

```
    self.roi4 = (table13.item(3, 0).text())
```

```
    self.threshold= (table13.item(4, 0).text())
```

```
    self.threshold_np = (table13.item(5, 0).text())
```

```
# Escribir los valores en el archivo de configuración
```

```
with open('txt_param\cv_conf.txt', 'w') as f:
```

```
    f.write(f'roi1={self.roi1}\n')
```

```
    f.write(f'roi2={self.roi2}\n')
```

```
    f.write(f'roi3={self.roi3}\n')
```

```
    f.write(f'roi4={self.roi4}\n')
```

```
    f.write(f'threshold={self.threshold}\n')
```

```
    f.write(f'threshold_np={self.threshold_np}\n')
```

```
QtWidgets.QMessageBox.information(self, 'Saved', 'The data has been  
saved successfully')
```

```
self.connect_Button(18)
```

```
def on_pushButton_default_cv_clicked(self):  
    self.pushButton_default_cv.clicked.disconnect()
```

```
# Abrir el archivo de configuración
```

```
with open('txt_param\cv_conf0.txt', 'r') as f:
```

```
    config_lines = f.readlines()
```

```
# Leer los valores de la tabla desde el archivo
```

```
self.roi1 = ast.literal_eval(config_lines[0].split('=')[1].strip())
```

```
self.roi2 = ast.literal_eval(config_lines[1].split('=')[1].strip())
```

```
self.roi3 = ast.literal_eval(config_lines[2].split('=')[1].strip())
```

```
self.roi4 = ast.literal_eval(config_lines[3].split('=')[1].strip())
```

```
self.threshold = int(config_lines[4].split('=')[1].strip())
```

```
self.threshold_np = int(config_lines[5].split('=')[1].strip())
```

```
# Escribir los valores en el archivo de configuración
```

```
with open('txt_param\cv_conf.txt', 'w') as f:
```

```
    f.write(f'roi1={self.roi1}\n')
```

```
    f.write(f'roi2={self.roi2}\n')
```

```
    f.write(f'roi3={self.roi3}\n')
```

```
    f.write(f'roi4={self.roi4}\n')
```

```
    f.write(f'threshold={self.threshold}\n')
```

```
    f.write(f'threshold_np={self.threshold_np}\n')
```

```
self.ini_cv()
```

```
QtWidgets.QMessageBox.information(self, 'Default', 'The Data has been  
successfully restored')
```

```
self.connect_Button(19)
```

```
def on_pushButton_roi1_clicked(self):  
    self.pushButton_roi1.clicked.disconnect()  
    #self.hilo_bot.roi1()  
    self.cam_hilo.roi_select(1)  
    time.sleep(2)  
    self.ini_cv()  
    #self.hilo_bot.roi1_end()  
    self.connect_Button(20)
```

```
def on_pushButton_roi2_clicked(self):  
    self.pushButton_roi2.clicked.disconnect()  
    #self.hilo_bot.roi2()  
    self.cam_hilo.roi_select(2)  
    time.sleep(2)  
    #self.hilo_bot.roi2_end()  
    self.ini_cv()  
    self.connect_Button(21)
```

```
def on_pushButton_roi3_clicked(self):  
    self.pushButton_roi3.clicked.disconnect()  
    #self.hilo_bot.roi3()  
    self.cam_hilo.roi_select(3)  
    time.sleep(2)  
    #self.hilo_bot.roi3_end()  
    self.ini_cv()  
    self.connect_Button(22)
```

```
def on_pushButton_roi4_clicked(self):  
    self.pushButton_roi4.clicked.disconnect()  
    #self.hilo_bot.roi4()  
    self.cam_hilo.roi_select(4)  
    time.sleep(2)
```

```

        #self.hilo_bot.roi4_end()
        self.ini_cv()
        self.connect_Button(23)

def on_pushButton_threshold_clicked(self):
    self.pushButton_threshold.clicked.disconnect()
    self.ini_cv()
    self.cam_hilo.roi_select(5)
    time.sleep(5)
    self.cam_hilo.roi_select(6)
    #self.hilo_bot.roi4_end()
    self.connect_Button(24)

def ini_AI(self):
    # Abrir el archivo de configuración
    with open('txt_param\Ai_conf.txt', 'r') as f:
        config_lines = f.readlines()

    # Leer los valores de la tabla desde el archivo
    self.limt_min00 = float(config_lines[0].split('=')[1].strip())
    self.limt_max00 = float(config_lines[1].split('=')[1].strip())
    self.limt_min0 = float(config_lines[2].split('=')[1].strip())
    self.limt_max0 = float(config_lines[3].split('=')[1].strip())
    self.limt_min1 = float(config_lines[4].split('=')[1].strip())
    self.limt_max1 = float(config_lines[5].split('=')[1].strip())
    self.limt_min2 = float(config_lines[6].split('=')[1].strip())
    self.limt_max2 = float(config_lines[7].split('=')[1].strip())
    self.limt_min3 = float(config_lines[8].split('=')[1].strip())
    self.limt_max3 = float(config_lines[9].split('=')[1].strip())
    self.limt_min4 = float(config_lines[10].split('=')[1].strip())
    self.limt_max4 = float(config_lines[11].split('=')[1].strip())
    self.seguro0 = float(config_lines[12].split('=')[1].strip())
    self.seguro1 = float(config_lines[13].split('=')[1].strip())
    self.seguro2 = float(config_lines[14].split('=')[1].strip())

```

```
self.seguro3 = float(config_lines[15].split('=')[1].strip())
```

```
table12 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_12')
```

```
table12.setItem(0, 0, QTableWidgetItem(str(self.limt_min0)))
table12.setItem(1, 0, QTableWidgetItem(str(self.limt_max0)))
table12.setItem(2, 0, QTableWidgetItem(str(self.limt_min0)))
table12.setItem(3, 0, QTableWidgetItem(str(self.limt_max0)))
table12.setItem(4, 0, QTableWidgetItem(str(self.limt_min1)))
table12.setItem(5, 0, QTableWidgetItem(str(self.limt_max1)))
table12.setItem(6, 0, QTableWidgetItem(str(self.limt_min2)))
table12.setItem(7, 0, QTableWidgetItem(str(self.limt_max2)))
table12.setItem(8, 0, QTableWidgetItem(str(self.limt_min3)))
table12.setItem(9, 0, QTableWidgetItem(str(self.limt_max3)))
table12.setItem(10, 0, QTableWidgetItem(str(self.limt_min4)))
table12.setItem(11, 0, QTableWidgetItem(str(self.limt_max4 )))
table12.setItem(12, 0, QTableWidgetItem(str(self.seguro0 )))
table12.setItem(13, 0, QTableWidgetItem(str(self.seguro1 )))
table12.setItem(14, 0, QTableWidgetItem(str(self.seguro2 )))
table12.setItem(15, 0, QTableWidgetItem(str(self.seguro3 )))
```

```
def on_pushButton_save_ai_clicked(self):
```

```
self.pushButton_save_ai.clicked.disconnect()
```

```
table12 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_12')
```

```
self.limt_min0 = float((table12.item(0, 0).text()))
self.limt_max0 = float((table12.item(1, 0).text()))
self.limt_min0 = float((table12.item(2, 0).text()))
self.limt_max0 = float((table12.item(3, 0).text()))
self.limt_min1 = float((table12.item(4, 0).text()))
self.limt_max1 = float((table12.item(5, 0).text()))
self.limt_min2 = float((table12.item(6, 0).text()))
```

```

self.limt_max2 = float((table12.item(7, 0).text()))
self.limt_min3 = float((table12.item(8, 0).text()))
self.limt_max3 = float((table12.item(9, 0).text()))
self.limt_min4 = float((table12.item(10, 0).text()))
self.limt_max4 = float((table12.item(11, 0).text()))
self.seguro0= float((table12.item(12, 0).text()))
self.seguro1= float((table12.item(13, 0).text()))
self.seguro2= float((table12.item(14, 0).text()))
self.seguro3= float((table12.item(15, 0).text()))

```

Escribir los valores en el archivo de configuración

```
with open('txt_param\Ai_conf.txt', 'w') as f:
```

```

    f.write(f'limt_min00={self.limt_min00}\n')
    f.write(f'limt_max00={self.limt_max00}\n')
    f.write(f'limt_min0={self.limt_min0}\n')
    f.write(f'limt_max0={self.limt_max0}\n')
    f.write(f'limt_min1={self.limt_min1}\n')
    f.write(f'limt_max1={self.limt_max1}\n')
    f.write(f'limt_min2={self.limt_min2}\n')
    f.write(f'limt_max2={self.limt_max2}\n')
    f.write(f'limt_min3={self.limt_min3}\n')
    f.write(f'limt_max3={self.limt_max3}\n')
    f.write(f'limt_min4={self.limt_min4}\n')
    f.write(f'limt_max4={self.limt_max4}\n')
    f.write(f'seguro0={self.seguro0}\n')
    f.write(f'seguro1={self.seguro1}\n')
    f.write(f'seguro2={self.seguro2}\n')
    f.write(f'seguro3={self.seguro3}\n')

```

```

QtWidgets.QMessageBox.information(self, 'Saved', 'The data has been
saved successfully')

```

```
self.connect_Button(16)
```

```

def on_pushButton_default_ai_clicked(self):
    self.pushButton_default_ai.clicked.disconnect()

    # Abrir el archivo de configuración
    with open('txt_param\Ai_conf0.txt', 'r') as f:
        config_lines = f.readlines()
    # Leer los valores de la tabla desde el archivo
    self.limt_min00 = float(config_lines[0].split('=')[1].strip())
    self.limt_max00 = float(config_lines[1].split('=')[1].strip())
    self.limt_min0 = float(config_lines[2].split('=')[1].strip())
    self.limt_max0 = float(config_lines[3].split('=')[1].strip())
    self.limt_min1 = float(config_lines[4].split('=')[1].strip())
    self.limt_max1 = float(config_lines[5].split('=')[1].strip())
    self.limt_min2 = float(config_lines[6].split('=')[1].strip())
    self.limt_max2 = float(config_lines[7].split('=')[1].strip())
    self.limt_min3 = float(config_lines[8].split('=')[1].strip())
    self.limt_max3 = float(config_lines[9].split('=')[1].strip())
    self.limt_min4 = float(config_lines[10].split('=')[1].strip())
    self.limt_max4 = float(config_lines[11].split('=')[1].strip())
    self.seguro0 = float(config_lines[12].split('=')[1].strip())
    self.seguro1 = float(config_lines[13].split('=')[1].strip())
    self.seguro2 = float(config_lines[14].split('=')[1].strip())
    self.seguro3 = float(config_lines[15].split('=')[1].strip())

    # Escribir los valores en el archivo de configuración
    with open('txt_param\Ai_conf.txt', 'w') as f:

        f.write(f'limt_min00={self.limt_min00}\n')
        f.write(f'limt_max00={self.limt_max00}\n')
        f.write(f'limt_min0={self.limt_min0}\n')
        f.write(f'limt_max0={self.limt_max0}\n')
        f.write(f'limt_min1={self.limt_min1}\n')
        f.write(f'limt_max1={self.limt_max1}\n')
        f.write(f'limt_min2={self.limt_min2}\n')

```

```

f.write(f'limt_max2={self.limt_max2}\n')
f.write(f'limt_min3={self.limt_min3}\n')
f.write(f'limt_max3={self.limt_max3}\n')
f.write(f'limt_min4={self.limt_min4}\n')
f.write(f'limt_max4={self.limt_max4}\n')
f.write(f'seguro0={self.seguro0}\n')
f.write(f'seguro1={self.seguro1}\n')
f.write(f'seguro2={self.seguro2}\n')
f.write(f'seguro3={self.seguro3}\n')

self.ini_AI()

QtWidgets.QMessageBox.information(self, 'Default', 'The Data has been
successfully restored')

self.connect_Button(17)

def on_pushButton_save_Param_clicked(self):

    self.pushButton_save_Param.clicked.disconnect()

    table11 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_11')

    self.indice_camara =(table11.item(0, 0).text())
    self.Port =(table11.item(1, 0).text())
    self.lp =(table11.item(2, 0).text())
    self.Flowmeter =(table11.item(3, 0).text())
    self.Apertura =(table11.item(4, 0).text())

    # Escribir los valores en el archivo de configuración
    with open('txt_param\Param_conf.txt', 'w') as f:
        f.write(f'indice_camara={self.indice_camara}\n')
        f.write(f'Port={self.Port}\n')
        f.write(f'lp={self.lp}\n')
        f.write(f'Flowmeter={self.Flowmeter}\n')

```

```

f.write(f'Apertura={self.Apertura}\n')

QtWidgets.QMessageBox.information(self, 'Saved', 'The data has been
saved successfully')

self.connect_Button(14)

def on_pushButton_default_param_clicked(self):
    self.pushButton_default_param.clicked.disconnect()

    # Abrir el archivo de configuración
    with open('txt_param\Param_conf0.txt', 'r') as f:
        config_lines = f.readlines()
    # Leer los valores de la tabla desde el archivo
    self.indice_camara = int(config_lines[0].split('=')[1].strip())
    self.Port = str(config_lines[1].split('=')[1].strip())
    self.Ip = str(config_lines[2].split('=')[1].strip())
    self.Flowmeter = float(config_lines[3].split('=')[1].strip())
    self.Apertura = float(config_lines[4].split('=')[1].strip())

    # Escribir los valores en el archivo de configuración
    with open('txt_param\Param_conf.txt', 'w') as f:
        f.write(f'indice_camara={self.indice_camara}\n')
        f.write(f'Port={self.Port}\n')
        f.write(f'Ip={self.Ip}\n')
        f.write(f'Flowmeter={self.Flowmeter}\n')
        f.write(f'Apertura={self.Apertura}\n')

    self.ini_tab_param()

    QtWidgets.QMessageBox.information(self, 'Default', 'The Data has been
successfully restored')

```

```

self.connect_Button(15)

def ini_tab_param(self):
    # Abrir el archivo de configuración
    with open('txt_param\Param_conf.txt', 'r') as f:
        config_lines = f.readlines()

    # Leer los valores de la tabla desde el archivo
    self.indice_camara = int(config_lines[0].split('=')[1].strip())
    self.Port = str(config_lines[1].split('=')[1].strip())
    self.Ip = str(config_lines[2].split('=')[1].strip())
    self.Flowmeter = float(config_lines[3].split('=')[1].strip())
    self.Apertura = float(config_lines[4].split('=')[1].strip())

    table11 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_11')

    # Establecer los datos en la tabla
    table11.setItem(0, 0, QTableWidgetItem(str(self.indice_camara)))
    table11.setItem(1, 0, QTableWidgetItem(str(self.Port)))
    table11.setItem(2, 0, QTableWidgetItem(str(self.Ip)))
    table11.setItem(3, 0, QTableWidgetItem(str(self.Flowmeter)))
    table11.setItem(4, 0, QTableWidgetItem(str(self.Apertura)))

def on_pushButton_default_clicked(self):

    self.pushButton_default.clicked.disconnect()

    # Abrir el archivo de configuración
    with open('txt_param\Poss_conf0.txt', 'r') as f:
        config_lines = f.readlines()

    # Leer los valores de la tabla desde el archivo
    pos_0 = ast.literal_eval(config_lines[0].split('=')[1].strip())
    pos_1 = ast.literal_eval(config_lines[1].split('=')[1].strip())

```

```

pos_2 = ast.literal_eval(config_lines[2].split('=')[1].strip())
pos_3 = ast.literal_eval(config_lines[3].split('=')[1].strip())
pos_4 = ast.literal_eval(config_lines[4].split('=')[1].strip())
pos_5 = ast.literal_eval(config_lines[5].split('=')[1].strip())
pos_6 = ast.literal_eval(config_lines[6].split('=')[1].strip())
pos_7 = ast.literal_eval(config_lines[7].split('=')[1].strip())
pos_8 = ast.literal_eval(config_lines[8].split('=')[1].strip())
pos_9 = ast.literal_eval(config_lines[9].split('=')[1].strip())
pos_10 = ast.literal_eval(config_lines[10].split('=')[1].strip())
pos_11 = ast.literal_eval(config_lines[11].split('=')[1].strip())
pos_12 = ast.literal_eval(config_lines[12].split('=')[1].strip())
pos_13 = ast.literal_eval(config_lines[13].split('=')[1].strip())
pos_14 = ast.literal_eval(config_lines[14].split('=')[1].strip())
pos_15 = ast.literal_eval(config_lines[15].split('=')[1].strip())
pos_16 = ast.literal_eval(config_lines[16].split('=')[1].strip())
pos_flow_0 = ast.literal_eval(config_lines[17].split('=')[1].strip())
pos_flow_1 = ast.literal_eval(config_lines[18].split('=')[1].strip())
pos_flow_2 = ast.literal_eval(config_lines[19].split('=')[1].strip())
pos_flow_3 = ast.literal_eval(config_lines[20].split('=')[1].strip())
pos_flow_4 = ast.literal_eval(config_lines[21].split('=')[1].strip())

```

```

gri_vaso1=float(config_lines[22].split('=')[1].strip())
gri_vaso2=float(config_lines[23].split('=')[1].strip())
gri_vaso3=float(config_lines[24].split('=')[1].strip())
gri_vaso4=float(config_lines[25].split('=')[1].strip())
gri_flow=float(config_lines[26].split('=')[1].strip())

```

```

speed_mov = float(config_lines[27].split('=')[1].strip())
acc_mov = float(config_lines[28].split('=')[1].strip())
speed_mov1=float(config_lines[29].split('=')[1].strip())
acc_mov1=float(config_lines[30].split('=')[1].strip())
speed_mov2=float(config_lines[31].split('=')[1].strip())
acc_mov2=float(config_lines[32].split('=')[1].strip())
speed_flow=float(config_lines[33].split('=')[1].strip())

```

```
acc_flow=float(config_lines[34].split('=')[1].strip())
```

```
# Escribir los valores en el archivo de configuración
```

```
with open('txt_param\Poss_conf.txt', 'w') as f:
```

```
    f.write(f'pos_0={pos_0}\n')
    f.write(f'pos_1={pos_1}\n')
    f.write(f'pos_2={pos_2}\n')
    f.write(f'pos_3={pos_3}\n')
    f.write(f'pos_4={pos_4}\n')
    f.write(f'pos_5={pos_5}\n')
    f.write(f'pos_6={pos_6}\n')
    f.write(f'pos_7={pos_7}\n')
    f.write(f'pos_8={pos_8}\n')
    f.write(f'pos_9={pos_9}\n')
    f.write(f'pos_10={pos_10}\n')
    f.write(f'pos_11={pos_11}\n')
    f.write(f'pos_12={pos_12}\n')
    f.write(f'pos_13={pos_13}\n')
    f.write(f'pos_14={pos_14}\n')
    f.write(f'pos_15={pos_15}\n')
    f.write(f'pos_16={pos_16}\n')
    f.write(f'pos_flow_0={pos_flow_0}\n')
    f.write(f'pos_flow_1={pos_flow_1}\n')
    f.write(f'pos_flow_2={pos_flow_2}\n')
    f.write(f'pos_flow_3={pos_flow_3}\n')
    f.write(f'pos_flow_4={pos_flow_4}\n')
    f.write(f'gri_vaso1={gri_vaso1}\n')
    f.write(f'gri_vaso2={gri_vaso2}\n')
    f.write(f'gri_vaso3={gri_vaso3}\n')
    f.write(f'gri_vaso4={gri_vaso4}\n')
    f.write(f'gri_flow={gri_flow}\n')
    f.write(f'speed_mov={speed_mov}\n')
    f.write(f'acc_mov={acc_mov}\n')
    f.write(f'speed_mov1={speed_mov1}\n')
```

```

f.write(f'acc_mov1={acc_mov1}\\n')
f.write(f'speed_mov2={speed_mov2}\\n')
f.write(f'acc_mov2={acc_mov2}\\n')
f.write(f'speed_flow={speed_flow}\\n')
f.write(f'acc_flow={acc_flow}\\n')

self.ini_tab()

QtWidgets.QMessageBox.information(self, 'Default', 'The Data has been
successfully restored')

self.connect_Button(13)

def on_pushButton_save_clicked(self):

    self.pushButton_save.clicked.disconnect()

    # Obtener los valores de la tabla
    table = self.findChild(QtWidgets.QTableWidget, 'tableWidget_1')
    table2 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_2')
    table3 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_3')
    table4 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_4')
    table5 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_5')
    table6 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_6')
    table7 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_7')
    table8 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_8')
    table9 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_9')
    table10 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_10')

    pos_0 = (table.item(0, 0).text())

    pos_1 =(table2.item(0, 0).text())
    pos_2 = (table2.item(1, 0).text())
    pos_3 = (table2.item(2, 0).text())

```

pos_4 = (table3.item(1, 0).text())

pos_5 = (table3.item(2, 0).text())

pos_6 =(table4.item(0, 0).text())

pos_7 = (table4.item(1, 0).text())

pos_8 = (table4.item(2, 0).text())

pos_9 = (table5.item(0, 0).text())

pos_10 = (table6.item(0, 0).text())

pos_11 =(table7.item(0, 0).text())

pos_12 =(table7.item(1, 0).text())

pos_13 =(table7.item(2, 0).text())

pos_14 = (table8.item(0, 0).text())

pos_15 = (table8.item(1, 0).text())

pos_16 = (table9.item(0, 0).text())

pos_flow_0 = (table10.item(0, 0).text())

pos_flow_1 =(table10.item(1, 0).text())

pos_flow_2 = (table10.item(3, 0).text())

pos_flow_3 = (table10.item(4, 0).text())

pos_flow_4 = (table10.item(5, 0).text())

gri_vaso1=float(table2.item(3, 0).text())

gri_vaso2=float(table3.item(0, 0).text())

gri_vaso3=float(table3.item(3, 0).text())

gri_vaso4=float(table3.item(4, 0).text())

gri_flow=float(table10.item(2, 0).text())

speed_mov = float(table.item(0, 1).text())

```
acc_mov = float(table.item(0, 2).text())
```

```
speed_mov1=float(table2.item(0, 1).text())
```

```
acc_mov1=float(table2.item(0, 2).text())
```

```
speed_mov2=float(table.item(0, 1).text())
```

```
acc_mov2=float(table.item(0, 2).text())
```

```
speed_flow=float(table10.item(0, 1).text())
```

```
acc_flow=float(table10.item(0, 2).text())
```

```
# Escribir los valores en el archivo de configuración
```

```
with open('txt_param\Poss_conf.txt', 'w') as f:
```

```
    f.write(f'pos_0={pos_0}\n')
```

```
    f.write(f'pos_1={pos_1}\n')
```

```
    f.write(f'pos_2={pos_2}\n')
```

```
    f.write(f'pos_3={pos_3}\n')
```

```
    f.write(f'pos_4={pos_4}\n')
```

```
    f.write(f'pos_5={pos_5}\n')
```

```
    f.write(f'pos_6={pos_6}\n')
```

```
    f.write(f'pos_7={pos_7}\n')
```

```
    f.write(f'pos_8={pos_8}\n')
```

```
    f.write(f'pos_9={pos_9}\n')
```

```
    f.write(f'pos_10={pos_10}\n')
```

```
    f.write(f'pos_11={pos_11}\n')
```

```
    f.write(f'pos_12={pos_12}\n')
```

```
    f.write(f'pos_13={pos_13}\n')
```

```
    f.write(f'pos_14={pos_14}\n')
```

```
    f.write(f'pos_15={pos_15}\n')
```

```
    f.write(f'pos_16={pos_16}\n')
```

```
    f.write(f'pos_flow_0={pos_flow_0}\n')
```

```
    f.write(f'pos_flow_1={pos_flow_1}\n')
```

```

f.write(f'pos_flow_2={pos_flow_2}\n')
f.write(f'pos_flow_3={pos_flow_3}\n')
f.write(f'pos_flow_4={pos_flow_4}\n')
f.write(f'gri_vaso1={gri_vaso1}\n')
f.write(f'gri_vaso2={gri_vaso2}\n')
f.write(f'gri_vaso3={gri_vaso3}\n')
f.write(f'gri_vaso4={gri_vaso4}\n')
f.write(f'gri_flow={gri_flow}\n')
f.write(f'speed_mov={speed_mov}\n')
f.write(f'acc_mov={acc_mov}\n')
f.write(f'speed_mov1={speed_mov1}\n')
f.write(f'acc_mov1={acc_mov1}\n')
f.write(f'speed_mov2={speed_mov2}\n')
f.write(f'acc_mov2={acc_mov2}\n')
f.write(f'speed_flow={speed_flow}\n')
f.write(f'acc_flow={acc_flow}\n')

```

```

QtWidgets.QMessageBox.information(self, 'Saved', 'The data has been
saved successfully')

```

```

self.connect_Button(12)

```

```

def ini_tab(self):

```

```

# Abrir el archivo de configuración
with open('txt_param\Poss_conf.txt', 'r') as f:
    config_lines = f.readlines()

```

```

# Leer los valores de la tabla desde el archivo
pos_0 = ast.literal_eval(config_lines[0].split('=')[1].strip())
pos_1 = ast.literal_eval(config_lines[1].split('=')[1].strip())
pos_2 = ast.literal_eval(config_lines[2].split('=')[1].strip())
pos_3 = ast.literal_eval(config_lines[3].split('=')[1].strip())
pos_4 = ast.literal_eval(config_lines[4].split('=')[1].strip())

```

```

pos_5 = ast.literal_eval(config_lines[5].split('=')[1].strip())
pos_6 = ast.literal_eval(config_lines[6].split('=')[1].strip())
pos_7 = ast.literal_eval(config_lines[7].split('=')[1].strip())
pos_8 = ast.literal_eval(config_lines[8].split('=')[1].strip())
pos_9 = ast.literal_eval(config_lines[9].split('=')[1].strip())
pos_10 = ast.literal_eval(config_lines[10].split('=')[1].strip())
pos_11 = ast.literal_eval(config_lines[11].split('=')[1].strip())
pos_12 = ast.literal_eval(config_lines[12].split('=')[1].strip())
pos_13 = ast.literal_eval(config_lines[13].split('=')[1].strip())
pos_14 = ast.literal_eval(config_lines[14].split('=')[1].strip())
pos_15 = ast.literal_eval(config_lines[15].split('=')[1].strip())
pos_16 = ast.literal_eval(config_lines[16].split('=')[1].strip())
pos_flow_0 = ast.literal_eval(config_lines[17].split('=')[1].strip())
pos_flow_1 = ast.literal_eval(config_lines[18].split('=')[1].strip())
pos_flow_2 = ast.literal_eval(config_lines[19].split('=')[1].strip())
pos_flow_3 = ast.literal_eval(config_lines[20].split('=')[1].strip())
pos_flow_4 = ast.literal_eval(config_lines[21].split('=')[1].strip())

```

```

gri_vaso1=float(config_lines[22].split('=')[1].strip())
gri_vaso2=float(config_lines[23].split('=')[1].strip())
gri_vaso3=float(config_lines[24].split('=')[1].strip())
gri_vaso4=float(config_lines[25].split('=')[1].strip())
gri_flow=float(config_lines[26].split('=')[1].strip())

```

```

speed_mov = float(config_lines[27].split('=')[1].strip())
acc_mov = float(config_lines[28].split('=')[1].strip())
speed_mov1=float(config_lines[29].split('=')[1].strip())
acc_mov1=float(config_lines[30].split('=')[1].strip())
speed_mov2=float(config_lines[31].split('=')[1].strip())
acc_mov2=float(config_lines[32].split('=')[1].strip())
speed_flow=float(config_lines[33].split('=')[1].strip())
acc_flow=float(config_lines[34].split('=')[1].strip())

```

```

table = self.findChild(QtWidgets.QTableWidget, 'tableWidget_1')

```

```

# Establecer los datos en la tabla
table.setItem(0, 0, QTableWidgetItem(str(pos_0)))
table.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
table.setItem(0, 2, QTableWidgetItem(str(acc_mov)))

table.setItem(1, 0, QTableWidgetItem('OPEN'))

table2 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_2')

# Establecer los datos en la tabla
table2.setItem(0, 0, QTableWidgetItem(str(pos_1)))
table2.setItem(0, 1, QTableWidgetItem(str(speed_mov1)))
table2.setItem(0, 2, QTableWidgetItem(str(acc_mov1)))

table2.setItem(1, 0, QTableWidgetItem(str(pos_2)))
table2.setItem(1, 1, QTableWidgetItem(str(speed_mov1)))
table2.setItem(1, 2, QTableWidgetItem(str(acc_mov1)))

table2.setItem(2, 0, QTableWidgetItem(str(pos_3)))
table2.setItem(2, 1, QTableWidgetItem(str(speed_mov1)))
table2.setItem(2, 2, QTableWidgetItem(str(acc_mov1)))

table2.setItem(3, 0, QTableWidgetItem(str(gri_vaso1)))

table3 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_3')

# Establecer los datos en la tabla
table3.setItem(0, 0, QTableWidgetItem(str(gri_vaso2)))

table3.setItem(1, 0, QTableWidgetItem(str(pos_4)))
table3.setItem(1, 1, QTableWidgetItem(str(speed_mov)))
table3.setItem(1, 2, QTableWidgetItem(str(acc_mov)))

```

```

table3.setItem(2, 0, QTableWidgetItem(str(pos_5)))
table3.setItem(2, 1, QTableWidgetItem(str(speed_mov)))
table3.setItem(2, 2, QTableWidgetItem(str(acc_mov)))

table3.setItem(3, 0, QTableWidgetItem(str(gri_vaso3)))

table3.setItem(4, 0, QTableWidgetItem(str(gri_vaso4)))

table4 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_4')

table4.setItem(0, 0, QTableWidgetItem(str(pos_6)))
table4.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
table4.setItem(0, 2, QTableWidgetItem(str(acc_mov)))

table4.setItem(1, 0, QTableWidgetItem(str(pos_7)))
table4.setItem(1, 1, QTableWidgetItem(str(speed_mov)))
table4.setItem(1, 2, QTableWidgetItem(str(acc_mov)))

table4.setItem(2, 0, QTableWidgetItem(str(pos_8)))
table4.setItem(2, 1, QTableWidgetItem(str(speed_mov)))
table4.setItem(2, 2, QTableWidgetItem(str(acc_mov)))

table5 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_5')

table5.setItem(0, 0, QTableWidgetItem(str(pos_9)))
table5.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
table5.setItem(0, 2, QTableWidgetItem(str(acc_mov)))

table6 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_6')

table6.setItem(0, 0, QTableWidgetItem(str(pos_10)))
table6.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
table6.setItem(0, 2, QTableWidgetItem(str(acc_mov)))

```

```
table7 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_7')
```

```
table7.setItem(0, 0, QTableWidgetItem(str(pos_11)))
```

```
table7.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
```

```
table7.setItem(0, 2, QTableWidgetItem(str(acc_mov)))
```

```
table7.setItem(1, 0, QTableWidgetItem(str(pos_12)))
```

```
table7.setItem(1, 1, QTableWidgetItem(str(speed_mov)))
```

```
table7.setItem(1, 2, QTableWidgetItem(str(acc_mov)))
```

```
table7.setItem(2, 0, QTableWidgetItem(str(pos_13)))
```

```
table7.setItem(2, 1, QTableWidgetItem(str(speed_mov)))
```

```
table7.setItem(2, 2, QTableWidgetItem(str(acc_mov)))
```

```
table7.setItem(3, 0, QTableWidgetItem("OPEN"))
```

```
table8 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_8')
```

```
table8.setItem(0, 0, QTableWidgetItem(str(pos_14)))
```

```
table8.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
```

```
table8.setItem(0, 2, QTableWidgetItem(str(acc_mov)))
```

```
table8.setItem(1, 0, QTableWidgetItem(str(pos_15)))
```

```
table8.setItem(1, 1, QTableWidgetItem(str(speed_mov)))
```

```
table8.setItem(1, 2, QTableWidgetItem(str(acc_mov)))
```

```
table9 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_9')
```

```
table9.setItem(0, 0, QTableWidgetItem(str(pos_16)))
```

```
table9.setItem(0, 1, QTableWidgetItem(str(speed_mov)))
```

```
table9.setItem(0, 2, QTableWidgetItem(str(acc_mov)))
```

```
table10 = self.findChild(QtWidgets.QTableWidget, 'tableWidget_10')
```

```
table10.setItem(0, 0, QTableWidgetItem(str(pos_flow_0)))
table10.setItem(0, 1, QTableWidgetItem(str(speed_flow)))
table10.setItem(0, 2, QTableWidgetItem(str(acc_flow)))
```

```
table10.setItem(1, 0, QTableWidgetItem(str(pos_flow_1)))
table10.setItem(1, 1, QTableWidgetItem(str(speed_flow)))
table10.setItem(1, 2, QTableWidgetItem(str(acc_flow)))
```

```
table10.setItem(2, 0, QTableWidgetItem(str(gri_flow)))
```

```
table10.setItem(3, 0, QTableWidgetItem(str(pos_flow_2)))
table10.setItem(3, 1, QTableWidgetItem(str(speed_flow)))
table10.setItem(3, 2, QTableWidgetItem(str(acc_flow)))
```

```
table10.setItem(4, 0, QTableWidgetItem(str(pos_flow_3)))
table10.setItem(4, 1, QTableWidgetItem(str(speed_flow)))
table10.setItem(4, 2, QTableWidgetItem(str(acc_flow)))
```

```
table10.setItem(5, 0, QTableWidgetItem(str(pos_flow_4)))
table10.setItem(5, 1, QTableWidgetItem(str(speed_flow)))
table10.setItem(5, 2, QTableWidgetItem(str(acc_flow)))
```

```
def on_push_exit_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    current_page = self.stackedWidget.currentIndex()
    if current_page == 0:
        # Llamar a la función correspondiente para la página 0
        self.cam_hilo.stop()
        self.hilo_bot.End_bot()

        self.close()
        sys.exit()
    else:
        self.stackedWidget.setCurrentIndex(0)
```

```

def on_push_help_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    print('Botón Help presionado')
    QDesktopServices.openUrl(QUrl.fromLocalFile("help.pdf"))

def on_pushButton_analytics_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_analytics.clicked.disconnect()

    self.mongo_manager.beer_data_analysis()
    self.mongo_manager.opinions_data_analysis()

    self.connect_Button(1)

def on_pushButton_settings_clicked(self):

    if self.widget_conf2.isVisible():
        self.widget_conf2.setVisible(False)
    else:
        self.widget_conf2.setVisible(True)

def on_pushButton_test_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_test.clicked.disconnect()
    #ventana error
    # self.stackedWidget.setCurrentIndex(self.pag_error)
    # self.mensaje_error1="No implementado"
    # self.mensaje_error2="boton test"
    # self.label_Error_sup.setText(self.mensaje_error1)
    # self.label_Error_inf.setText(self.mensaje_error2)
    self.stackedWidget.setCurrentIndex(self.pag_test)

    self.connect_Button(3)

```

```

def on_pushButton_AI_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_AI.clicked.disconnect()
    print('Botón AI presionado')
    self.ini_AI()
    self.stackedWidget.setCurrentIndex(self.pag_ai)
    self.connect_Button(4)

def on_pushButton_CV_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_CV.clicked.disconnect()
    print('Botón CV presionado')
    self.ini_cv()
    self.stackedWidget.setCurrentIndex(self.pag_cv)
    self.connect_Button(5)

def on_pushButton_param_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_param.clicked.disconnect()
    print('Botón param presionado')
    self.ini_tab_param()
    self.stackedWidget.setCurrentIndex(self.pag_param)
    self.connect_Button(6)

def on_pushButton_poss_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_poss.clicked.disconnect()
    self.ini_tab()
    print('Botón Poss presionado')
    self.stackedWidget.setCurrentIndex(self.pag_poss)

    self.connect_Button(7)

```

```

def on_pushButton_demo0_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_demo0.clicked.disconnect()
    print('Botón Demo 0 presionado')
    #self.cont_beer=self.cont_beer+1

    #self.stackedWidget.setCurrentIndex(self.pag_opinion)

    self.stackedWidget.setCurrentIndex(self.pag_wait)
    #esperar un poco antes de mostrar la siguiente página
    QTimer.singleShot(1000, self.ini_estado0)
    #QTimer.singleShot(1000, self.prueba00)

    self.connect_Button(8)

def on_pushButton_demo1_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_demo1.clicked.disconnect()
    print('Botón demo 1 presionado')
    #self.stackedWidget.setCurrentIndex(3)

    self.connect_Button(9)

def on_pushButton_demo2_clicked(self):
    # Función a ejecutar al hacer clic en el botón
    self.pushButton_demo2.clicked.disconnect()
    print('Botón demo 2 presionado')
    self.stackedWidget.setCurrentIndex(self.pag_serv)

    QTimer.singleShot(1000, self.serv_beer)

    self.connect_Button(10)

def on_pushButton_reset_clicked(self):

```

```

# Función a ejecutar al hacer clic en el botón
self.pushButton_reset.clicked.disconnect()
print('Botón Reset presionado')

if self.error_tipo == 1:
    #error no cup
    self.hilo_bot.go_home()

    self.cam_hilo.actuator_close()
    self.motor_cup = False

    self.error_tipo=0
    self.stackedWidget.setCurrentIndex(self.pag_demo)

elif self.error_tipo == 2:
    #error no colision bot parametrizar
    if 0 <= self.code_error <= 9:
        self.hilo_bot.reset_mov(self.code_error)

    elif self.code_error > 9:

        self.hilo_bot.reset()
        time.sleep(1)
        completed5, estado = self.hilo_bot.mov_0g()
        completed6, estado = self.hilo_bot.exit_beer()
        completed7, estado = self.hilo_bot.exist_table()
        completed7, estado = self.hilo_bot.exist_table2()
        completed10 , estado = self.hilo_bot.go_home()

        disp1=self.cam_hilo.actuator_close()
        self.motor_cup = False

    self.error_tipo=0
    self.stackedWidget.setCurrentIndex(self.pag_demo)

```

```

elif self.error_tipo == 3:
    #error no beer
    completed5, estado = self.hilo_bot.mov_0g()
    completed6, estado = self.hilo_bot.exit_beer()
    completed7, estado = self.hilo_bot.exist_table()
    completed7, estado = self.hilo_bot.exist_table2()
    if self.Flowmeter <=90:
        self.Flowmeter = self.Flowmeter + 10
        completed8 , estado = self.hilo_bot.Regulate_flow(self.Flowmeter)
        text_bot="NEW PRESSURE REGULATOR VALUE : ", self.Flowmeter
, " % , (+10%)"
        completed10 , estado = self.hilo_bot.go_home()
    if self.Apertura <=90:
        self.Apertura = self.Apertura + 10
        text_tap="NEW OPENING TAP ", self.Apertura , " ---> ", " % ,
(+10%)"
        print("NEW OPENING TAP ", self.Apertura , " ---> ", " %")

    if self.Flowmeter ==100 and self.Apertura==100:
        text_others="Error: Empty beer barrel"
        print("Error: Empty beer barrel")
    else:
        self.error_tipo=0
        self.stackedWidget.setCurrentIndex(self.pag_demo)

    try:
        self.mongo_manager.insert_info(
            id=self.cont_beer,
            date= self.hora_ini,
            info_full = "interruption for lack of beer",
            info1=0,
            info2=0,
            info3=0,

```

```

        info4=0,
        Bot_Action=text_bot,
        Tap_Action=text_tap,
        others= text_others,
    )
except:
    pass

elif self.error_tipo == 4:

    try:
        self.cam_hilo.stop()

    except:
        print("camera not initialised before")

    time.sleep(2)
    try:
        self.cam_hilo =
CameraThread(self.indice_camara,self.Port,self.roi1,self.roi2,self.roi3,self.roi4).
start()

        self.error_tipo=0
        self.stackedWidget.setCurrentIndex(self.pag_Home)
    except:
        print("Error reset cam")
        self.error_tipo=4
        self.stackedWidget.setCurrentIndex(self.pag_error)

elif self.error_tipo == 5:
    print("ini bot")
    try:
        self.hilo_bot = HiloCobot(self.lp)
        self.hilo_bot.start()
        self.error_tipo=0

```

```

        self.stackedWidget.setCurrentIndex(self.pag_Home)
    except:
        print("Error ini bot")
        self.error_tipo=5
        self.stackedWidget.setCurrentIndex(self.pag_error)

elif self.error_tipo == 6:
    try:
        self.mongo_manager =
MongoDBManager("mongodb+srv://maher:1234@beer.uoimnt2.mongodb.net/b
eer_bot?retryWrites=true&w=majority")
        self.error_tipo=0
        self.stackedWidget.setCurrentIndex(self.pag_Home)
    except:
        print("Error reset DATABASE")
        print("Error ini bot")
        self.error_tipo=5
        self.stackedWidget.setCurrentIndex(self.pag_error)

elif self.error_tipo == 7:
    try:
        self.cam_hilo.stop()
    except:
        print("camera not initialised before")

    time.sleep(2)
    try:
        self.cam_hilo =
CameraThread(self.indice_camara,self.Port,self.roi1,self.roi2,self.roi3,self.roi4).
start()

        self.error_tipo=0
        self.stackedWidget.setCurrentIndex(self.pag_Home)
    except:
        print("Error reset cam")

```

```

        self.error_tipo=4
        self.stackedWidget.setCurrentIndex(self.pag_error)

    try:
        self.hilo_bot = HiloCobot(self.lp)
        self.hilo_bot.start()
        self.error_tipo=0
        self.stackedWidget.setCurrentIndex(self.pag_Home)
    except:
        print("Error ini bot")
        self.error_tipo=5
        self.stackedWidget.setCurrentIndex(self.pag_error)

    self.connect_Button(11)

def prueba00(self):
    self.Apertura0=self.Apertura
    self.cam_hilo.Visual_cam(1,self.cont_beer)
    self.show_cam = True

    duracion1=self.cam_hilo.llenar_cerveza(self.Apertura0,self.threshold,self.thresh
old_np,self.limt_max1,self.seguro0)
    time.sleep(2)
    if duracion1 == 1000:

        self.cam_hilo.close_tap()
        time.sleep(2)
        self.sinbeer()
        return

    self.ver_estado_beer(45,"The robot moves to put the cup at 30 degrees in
the beer dispenser.")

    self.ver_estado_beer(50,"Filling of the second section of the beer.")

```

```
duracion2=self.cam_hilo.llenar_cerveza2(self.threshold,self.threshold_np,self.lim
mt_max2,self.seguro1)
```

```
    time.sleep(2)
```

```
    self.ver_estado_beer(55,"The robot moves to put the cup at 0 degrees in
the beer dispenser.")
```

```
    self.ver_estado_beer(60,"Filling of the third section of the beer.")
```

```
duracion3=self.cam_hilo.llenar_cerveza3(self.threshold,self.threshold_np,self.lim
mt_max3,self.seguro2)
```

```
    time.sleep(2)
```

```
    self.ver_estado_beer(63,"Filling of the third section of the beer.")
```

```
duracion4=self.cam_hilo.llenar_espuma(self.threshold,self.threshold_np,self.lim
t_max4,self.seguro3)
```

```
def ini_estado0(self):
```

```
    #completed10 , estado = self.hilo_bot.hello()
```

```
    self.ver_estado_wait(20,"Initialisation of the initial parameters of the beer
dispenser.")
```

```
    self.hilo_bot.Regulate_flow(self.Flowmeter)
```

```
    self.ver_estado_wait(90,"Finished, go to home position.")
```

```
    self.hilo_bot.go_home()
```

```
    self.ver_estado_wait(0,".....")
```

```
    self.stackedWidget.setCurrentIndex(self.pag_demo)
```

```
def serv_beer(self):
```

```

self.Apertura0=self.Apertura
self.Flowmeter0=self.Flowmeter

self.ver_estado_beer(10,"Initiation")

t_ini= time.time()
self.fecha_ini = time.strftime("%d-%m-%Y-")
self.hora_ini = time.strftime("%H:%M:%S")

self.cont_beer = self.cont_beer + 1

#visualizar camara
self.cam_hilo.Visual_cam(1,self.cont_beer)
self.show_cam = True

#completed10 , estado = self.hilo_bot.hello()
# self.detect_error(completed10 ,estado)

# if completed10 == False:
#     self.cam_hilo.close_tap()
#     return

self.ver_estado_beer(12,"Open cup dispenser")

if self.motor_cup == False:
    self.cam_hilo.actuator_open()
    self.motor_cup = True
else:
    self.cam_hilo.actuator_close()
    self.motor_cup = False
    time.sleep(6)
    self.cam_hilo.actuator_open()
    self.motor_cup = True

```

```

time.sleep(1)

self.ver_estado_beer(15,"The bot moves towards the cup dispenser")

completed1,estado = self.hilo_bot.go_glass()

self.detect_error(completed1 ,estado)
if completed1 == False:
    self.cam_hilo.close_tap()
    return

self.ver_estado_beer(17,"Fully open cup dispenser")
#disp=self.cam_hilo.dispensador1()

self.ver_estado_beer(20,"The bot picks up the cup.")
completed2, estado = self.hilo_bot.exit_glass()

self.detect_error(completed2 ,estado)
if completed2 == False:
    self.cam_hilo.close_tap()
    return

self.ver_estado_beer(25,"AI Check if there is a cup.")
Detect,box,img_cup=self.cam_hilo.capt4()
# img_cup = np.asarray(img_cup)
# Detect,box =detectar_objeto(img_cup)

if Detect == False:
    print("Error no cup")
    self.cam_hilo.close_tap()
    self.sincup()
    return

self.ver_estado_beer(30,"Cup detected")

```

```
disp1=self.cam_hilo.actuator_close()  
self.motor_cup = False
```

```
self.ver_estado_beer(35,"The robot moves to put the cup at 45 degrees in  
the beer dispenser.")
```

```
completed3, estado = self.hilo_bot.go_beer()  
self.detect_error(completed3 ,estado)  
if completed3 == False:  
    self.cam_hilo.close_tap()  
    time.sleep(1)  
    return
```

```
self.cam_hilo.open_tap(self.Apertura0)  
time.sleep(1)
```

```
completed33, estado = self.hilo_bot.mov_45g()  
self.detect_error(completed3 ,estado)  
if completed33 == False:  
    self.cam_hilo.close_tap()  
    time.sleep(1)  
    return
```

```
self.ver_estado_beer(40,"Filling of the first section of the beer.")
```

```
duracion1=self.cam_hilo.llenar_cerveza(self.Apertura0,self.threshold,self.thresh  
old_np,self.limt_max1,self.seguro0)
```

```
if duracion1 == 1000:
```

```
    self.cam_hilo.close_tap()  
    time.sleep(1)  
    self.sinbeer()
```

```

return

self.ver_estado_beer(45,"The robot moves to put the cup at 30 degrees in
the beer dispenser.")
completed4 , estado = self.hilo_bot.mov_30g()

self.detect_error(completed4 ,estado)
if completed4 == False:
    self.cam_hilo.close_tap()
    return

self.ver_estado_beer(50,"Filling of the second section of the beer.")

duracion2=self.cam_hilo.llenar_cerveza2(self.threshold,self.threshold_np,self.li
mt_max2,self.seguro1)

self.ver_estado_beer(55,"The robot moves to put the cup at 0 degrees in
the beer dispenser.")
completed5, estado = self.hilo_bot.mov_0g()

self.detect_error(completed5 ,estado)
if completed5 == False:
    self.cam_hilo.Visual_cam(-1,self.cont_beer)
    self.cam_hilo.close_tap()
    return

self.ver_estado_beer(60,"Filling of the third section of the beer.")

duracion3=self.cam_hilo.llenar_cerveza3(self.threshold,self.threshold_np,self.li
mt_max3,self.seguro2)

self.ver_estado_beer(63,"Filling of the third section of the beer.")

```

```
duracion4=self.cam_hilo.llenar_espuma(self.threshold,self.threshold_np,self.lim  
t_max4,self.seguro3)
```

```
time.sleep(1)
```

```
self.ver_estado_beer(65,"Beer completed, move to table.")
```

```
completed6, estado = self.hilo_bot.exit_beer()
```

```
self.detect_error(completed6 ,estado)
```

```
if completed6 == False:
```

```
    self.cam_hilo.close_tap()
```

```
    return
```

```
self.ver_estado_beer(70,"Beer completed.")
```

```
completed7, estado = self.hilo_bot.exist_table()
```

```
self.detect_error(completed7 ,estado)
```

```
if completed7 == False:
```

```
    self.cam_hilo.close_tap()
```

```
    return
```

```
self.cam_hilo.capt3()
```

```
self.ver_estado_beer(75,"Beer ready.")
```

```
completed71, estado = self.hilo_bot.exist_table2()
```

```
self.detect_error(completed71 ,estado)
```

```
if completed71 == False:
```

```
    self.cam_hilo.close_tap()
```

```
    return
```

```
self.ver_estado_beer(77,"Thank you very much.")
```

```
t_end= time.time()
hora_end = time.strftime("%H:%M:%S")
duracion_full=t_end-t_ini
```

```
try:
```

```
    self.ver_estado_beer(80,"Analysing service data.")
```

```
    self.mongo_manager.insert_beer_parameters(
```

```
        id=self.cont_beer,
        date= self.fecha_ini,
        tap_opening = self.Apertura0,
        flow_opening=self.Flowmeter0,
        start_time= self.hora_ini,
        end_time= hora_end,
        total_duration= duracion_full,
        beer_time=duracion1+duracion2+duracion3,
        time_limit1=duracion1,
        time_limit2=duracion2,
        time_limit3=duracion3,
        time_limit4=duracion4,
        others=0
    )
```

```
except:
```

```
    pass
```

```
self.conf_param_beer(duracion_full,duracion1,duracion2,duracion3,duracion4)
```

```

self.ver_estado_beer(90,"Bot come home.")
completed10 , estado = self.hilo_bot.go_home()

self.detect_error(completed10 ,estado)
if completed10 == False:
    self.cam_hilo.close_tap()
    return

self.cam_hilo.Visual_cam(-1,self.cont_beer)
self.show_cam = False

self.stackedWidget.setCurrentIndex(self.pag_opinion)
#self.stackedWidget.setCurrentIndex(self.pag_demo)
self.ver_estado_beer(0,".....")

def detect_error(self, completed ,estado):

    if completed == True:
        print("Bot movement successfully completed, mov = ", estado)
    else:
        self.code_error=estado
        self.error_tipo=2
        self.cam_hilo.Visual_cam(-1,self.cont_beer)
        print("Interrupted Bot movement, mov = ", estado)
        self.mensaje_error1="BOT collision"
        self.mensaje_error2="state : {} error".format(estado)
        self.label_Error_sup.setText(self.mensaje_error1)
        self.label_Error_inf.setText(self.mensaje_error2)
        QApplication.processEvents()
        self.stackedWidget.setCurrentIndex(self.pag_error)
        try:

            self.mongo_manager.insert_beer_parameters(

```

```

        id=self.cont_beer,
        date= self.fecha_ini,
        tap_opening = self.Apertura0,
        flow_opening=self.Flowmeter0,
        start_time= self.hora_ini,
        end_time= 0,
        total_duration= 0,
        beer_time=0,
        time_limit1=0,
        time_limit2=0,
        time_limit3=0,
        time_limit4=0,
        others="BOT collision"
    )

except:
    pass

return " error "

def sinbeer(self):
    self.error_tipo=3
    self.mensaje_error1="ERROR"
    self.mensaje_error2="NO BEER"
    self.cam_hilo.Visual_cam(-1,self.cont_beer)
    self.label_Error_sup.setText(self.mensaje_error1)
    self.label_Error_inf.setText(self.mensaje_error2)
    QApplication.processEvents()
    self.stackedWidget.setCurrentIndex(self.pag_error)

try:
    self.mongo_manager.insert_beer_parameters(
        id=self.cont_beer,
        date= self.fecha_ini,

```

```

        tap_opening = self.Apertura0,
        flow_opening=self.Flowmeter0,
        start_time= self.hora_ini,
        end_time= 0,
        total_duration= 0,
        beer_time=0,
        time_limit1=0,
        time_limit2=0,
        time_limit3=0,
        time_limit4=0,
        others="no beer"
    )

except:
    pass

def sincup(self):
    self.error_tipo=1
    self.mensaje_error1="ERROR"
    self.mensaje_error2="NO cup"
    self.cam_hilo.Visual_cam(-1,self.cont_beer)
    self.label_Error_sup.setText(self.mensaje_error1)
    self.label_Error_inf.setText(self.mensaje_error2)
    QApplication.processEvents()
    self.stackedWidget.setCurrentIndex(self.pag_error)

try:
    self.mongo_manager.insert_beer_parameters(
        id=self.cont_beer,
        date= self.fecha_ini,
        tap_opening = self.Apertura0,
        flow_opening=self.Flowmeter0,
        start_time= self.hora_ini,
        end_time= 0,

```

```

        total_duration= 0,
        beer_time=0,
        time_limit1=0,
        time_limit2=0,
        time_limit3=0,
        time_limit4=0,
        others="no cup"
    )
except:
    pass

try:
    self.mongo_manager.insert_info(
        id=self.cont_beer,
        date= self.hora_ini,
        info_full = "interruption for NO cup",
        info1=0,
        info2=0,
        info3=0,
        info4=0,
        Bot_Action=0,
        Tap_Action=0,
        others= "NO cup",
    )
except:
    pass

return

def sincam(self):
    self.error_tipo=4

    self.mensaje_error1="Error INI CAM !"

```

```
self.mensaje_error2="Is the usb of the webcam and the Arduino  
connected?"
```

```
self.label_Error_sup.setText(self.mensaje_error1)  
self.label_Error_inf.setText(self.mensaje_error2)  
QApplication.processEvents()
```

```
self.stackedWidget.setCurrentIndex(self.pag_error)  
return
```

```
def sinbot(self):
```

```
    self.error_tipo=5
```

```
    self.mensaje_error1="Error INI BOT !"  
    self.mensaje_error2="Is the ethernet port connected?"
```

```
    self.label_Error_sup.setText(self.mensaje_error1)  
    self.label_Error_inf.setText(self.mensaje_error2)  
    QApplication.processEvents()
```

```
    self.stackedWidget.setCurrentIndex(self.pag_error)  
    return
```

```
def sinDB(self):
```

```
    self.error_tipo=6
```

```
    self.mensaje_error1="Error INI DATABASE !"  
    self.mensaje_error2="Do i have internet access?"
```

```
    self.label_Error_sup.setText(self.mensaje_error1)  
    self.label_Error_inf.setText(self.mensaje_error2)  
    QApplication.processEvents()
```

```
    self.stackedWidget.setCurrentIndex(self.pag_error)
```

```
return
```

```
def sincam_bot(self):
```

```
    self.error_tipo=7
```

```
    self.mensaje_error1="Error INI Cam and bot !"
```

```
    self.mensaje_error2="Please, connect the usb-c hub."
```

```
    self.label_Error_sup.setText(self.mensaje_error1)
```

```
    self.label_Error_inf.setText(self.mensaje_error2)
```

```
    QApplication.processEvents()
```

```
    self.stackedWidget.setCurrentIndex(self.pag_error)
```

```
    return
```

```
def
```

```
conf_param_beer(self,duracion_full,duracion1,duracion2,duracion3,duracion4):
```

```
    cambio=False
```

```
    if duracion_full >= self.limt_max00:
```

```
        sob=int(duracion_full-self.limt_max00)
```

```
        info_full_text="The beer with ID = {} , has exceeded the maximum
```

```
duration of the system by: {} s".format(self.cont_beer,sob)
```

```
        print("The beer with ID = {} , has exceeded the maximum duration of the  
system by: {} s".format(self.cont_beer,sob))
```

```
        print("takes too long")
```

```
    elif duracion_full <= self.limt_min00:
```

```
        sob=int(self.limt_min00-duracion_full)
```

```
        info_full_text="Beer with ID = {} , has not exceeded the minimum system
```

```
duration by: {} s".format(self.cont_beer,sob)
```

```
        print("Beer with ID = {} , has not exceeded the minimum system duration  
by: {} s".format(self.cont_beer,sob))
```

```
        print("too fast")
```

```
    else:
```

```

info_full_text="Beer with ID = {} , duration of system: {} s, is within the
limits".format(self.cont_beer,duracion_full)

if duracion1 >= self.limt_max1:
    sob=int(duracion1-self.limt_max1)
    text1="The beer with ID = {} , has exceeded the maximum duration of
limit 1 by: {} s. ".format(self.cont_beer,sob)
    print("takes too long")

    self.ver_estado_beer(85,text1)

if self.Flowmeter0 <=90:

    self.Flowmeter = self.Flowmeter0 + 10
    completed8 , estado = self.hilo_bot.Regulate_flow(self.Flowmeter)
    text_bot="NEW PRESSURE REGULATOR VALUE : ", self.Flowmeter
, " % , (+10%)"

    self.detect_error(completed8 ,estado)
    if completed8 == False:
        self.cam_hilo.close_tap()
        return
    else:
        if self.Apertura0 <=90:
            self.Apertura = self.Apertura0 + 10
            text_tap="NEW OPENING TAP ", self.Apertura , " ---> ", " % ,
(+10%)"

            cambio=True
            print("NEW OPENING TAP ", self.Apertura , " ---> ", " %")

elif duracion1 <= self.limt_min1:
    sob=int(self.limt_min1-duracion1)
    text1="Beer with ID = {} , has not exceeded the minimum of limit 1 by: {}
s".format(self.cont_beer,sob)

```

```

print("too fast")
self.ver_estado_beer(85,text1)

if self.Flowmeter0 >= 20:

    self.Flowmeter=self.Flowmeter0-10
    completed9 , estado = self.hilo_bot.Regulate_flow(self.Flowmeter)
    text_bot="NEW PRESSURE REGULATOR VALUE : ", self.Flowmeter
, " % , (-10%)"

    self.detect_error(completed9 ,estado)
    if completed9 == False:
        self.cam_hilo.close_tap()
        return
    else:
        if self.Apertura0 >=30:

            self.Apertura = self.Apertura0 - 10
            text_tap="NEW OPENING TAP ", self.Apertura , " ---> ", " % , (-
10%)"

            cambio=True
            print("NEW OPENING TAP ", self.Apertura , " ---> ", " %")

        else:
            text1="Beer with ID = {} , duration of limit 1: {} s, is within the
limits".format(self.cont_beer,duracion1)

            if duracion2 >= self.limt_max2:
                sob=int(duracion2-self.limt_max2)
                text2="The beer with ID = {} , has exceeded the maximum duration of
limit 2 by: {} s".format(self.cont_beer,sob)
                print("The beer with ID = {} , has exceeded the maximum duration of
limit 2 by: {} s".format(self.cont_beer,sob))

```

```

print("takes too long")

if self.Apertura0 <=90:
    if cambio==False:
        self.Apertura = self.Apertura0 + 10
        text_tap="NEW OPENING TAP ", self.Apertura ," ---> ", " % ,
(+10%)"
        print("NEW OPENING TAP ", self.Apertura ," ---> ", " %")

elif duracion2 <= self.limt_min2:
    sob=int(self.limt_min2-duracion2)
    text2="Beer with ID = {} , has not exceeded the minimum of limit 2 by: {}
s".format(self.cont_beer,sob)
    print("Beer with ID = {} , has not exceeded the minimum of limit 2 by: {}
s".format(self.cont_beer,sob))
    print("too fast")

if self.Apertura0 >=30:
    if cambio==False:
        self.Apertura = self.Apertura0 - 10
        text_tap="NEW OPENING TAP ", self.Apertura ," ---> ", " % , (-
10%)"
        print("NEW OPENING TAP ", self.Apertura ," ---> ", " %")

else:
    text2="Beer with ID = {} , duration of limit 2: {} s, is within the
limits".format(self.cont_beer,duracion2)

if duracion3 >= self.limt_max3:
    sob=int(duracion3-self.limt_max3)
    text3="The beer with ID = {} , has exceeded the maximum duration of
limit 3 by: {} s".format(self.cont_beer,sob)
    print("The beer with ID = {} , has exceeded the maximum duration of
limit 3 by: {} s".format(self.cont_beer,sob))

```

```

        print("takes too long")

    elif duracion3 <= self.limt_min3:
        sob=int(self.limt_min3-duracion3)
        text3="Beer with ID = {} , has not exceeded the minimum of limit 3 by: {}
s".format(self.cont_beer,sob)
        print("Beer with ID = {} , has not exceeded the minimum of limit 3 by: {}
s".format(self.cont_beer,sob))
        print("too fast")

    else:
        text3="Beer with ID = {} , duration of limit 3: {} s, is within the
limits".format(self.cont_beer,duracion3)

    if duracion4 >= self.limt_max4:
        sob=int(duracion4-self.limt_max4)
        text4="The beer with ID = {} , has exceeded the maximum duration of
limit 4 by: {} s".format(self.cont_beer,sob)
        print("The beer with ID = {} , has exceeded the maximum duration of
limit 4 by: {} s".format(self.cont_beer,sob))
        print("takes too long")

    elif duracion4 <= self.limt_min3:
        sob=int(self.limt_min4-duracion4)
        text4="Beer with ID = {} , has not exceeded the minimum of limit 4 by: {}
s".format(self.cont_beer,sob)
        print("Beer with ID = {} , has not exceeded the minimum of limit 4 by: {}
s".format(self.cont_beer,sob))
        print("too fast")

    else:
        text4="Beer with ID = {} , duration of limit 4: {} s, is within the
limits".format(self.cont_beer,duracion4)

```

```

try:
    self.mongo_manager.insert_info(
        id=self.cont_beer,
        date= self.hora_ini,
        info_full = info_full_text,
        info1=text1,
        info2=text2,
        info3=text3,
        info4=text4,
        Bot_Action=text_bot,
        Tap_Action=text_tap,
        others= 0,
    )

except:
    pass

if __name__ == '__main__':

    app = QApplication(sys.argv)
    # # Crear una instancia de la ventana principal
    window = Window1()
    # # Mostrar la ventana
    window.show()
    # # Configurar la ventana para que sea a pantalla completa
    window.showFullScreen()
    # # Ejecutar el ciclo de eventos de la aplicación
    sys.exit(app.exec_())

```

15.1.9. Control del Arduino (Control_arduino.ino)

```
// Author: Maher Bouymejan
// Date Created: 30/03/2023
// Contact: maherbouymejan@gmail.com

#include <Servo.h>
//Servo motor tap
Servo servo1;

//Puente H
int enaPin = 9; // pin ENA conectado al pin 9
int in1Pin = 8; // pin IN1 conectado al pin 8
int in2Pin = 7; // pin IN2 conectado al pin 7

//Linear angle
int angle_lineal;

void setup() {
  //Pines puente H
  pinMode(enaPin, OUTPUT);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);

  servo1.attach(3); //servo tap --> pin 3

  Serial.begin(9600);

  //Initial position
  servo1.write(20); //Close Tap

  //Cerrar actuador lineal
  digitalWrite(in1Pin, LOW);
```

```

digitalWrite(in2Pin, HIGH);
analogWrite(enaPin, 255); // velocidad máxima
delay(5000); // Esperar 5 segundos

}

void loop() {

  //Reads input from serial port
  if (Serial.available() > 0) {
    int angle = Serial.parseInt();

    if(angle > 0 && angle <= 70){

      //angle Tap servo rango [0-70]
      servo1.write(angle);

    }
    if(angle == 800){

      digitalWrite(in1Pin, LOW);
      digitalWrite(in2Pin, HIGH);
      analogWrite(enaPin, 255); // velocidad máxima
      delay(5000); // Esperar 5 segundos

      // Hacer que el actuador se detenga
      digitalWrite(in1Pin, LOW);
      digitalWrite(in2Pin, LOW);
      analogWrite(enaPin, 0); // velocidad cero
    }

    if(angle == 900){

      digitalWrite(in1Pin, HIGH);

```

```
digitalWrite(in2Pin, LOW);  
analogWrite(enaPin, 255); // velocidad máxima  
delay(4000); // Esperar 5 segundos  
  
// Hacer que el actuador se detenga  
digitalWrite(in1Pin, LOW);  
digitalWrite(in2Pin, LOW);  
analogWrite(enaPin, 0); // velocidad cero;  
}  
  
}  
}
```

15.2. Lista de parámetros de configuración

15.2.1. Parámetros de control del sistema (AI_conf.txt)

limt_min00=40.0

limt_max00=60.0

limt_min0=8.0

limt_max0=12.0

limt_min1=5.0

limt_max1=7.0

limt_min2=0.0

limt_max2=3.0

limt_min3=1.0

limt_max3=3.0

limt_min4=0.0

limt_max4=3.0

seguro0=5.0

seguro1=4.0

seguro2=3.0

seguro3=3.0

15.2.2. Parámetros de visión computacional (cv_conf.txt)

roi1=(251, 200, 17, 12)

roi2=(271, 139, 18, 16)

roi3=(336, 419, 22, 16)

roi4=(350, 389, 22, 21)

threshold=20

threshold_np=6

15.2.3. Parámetros de configuración inicial (Param_conf.txt)

indice_camara=0

Port=COM3

Ip=192.168.1.203

Flowmeter=20

Apertura=30.0

15.2.4. Posiciones del brazo robótico (Poss_conf.txt)

```
pos_0=[7.9, 229.7, 134.5, 38.6, -86.6, -130.8]
pos_1=[95.6, 391.0, 143.0, -117.0, -89.2, 94.2]
pos_2=[80.0, 490.4, 148.2, -91.5, -89.2, 94.2]
pos_3=[78.7, 490.4, 206.7, -91.5, -89.2, 94.2]
pos_4=[78.7, 501.7, 188.3, -89.5, -79.1, 94.2]
pos_5=[78.7, 490.4, 143.8, -91.5, -89.2, 94.2]
pos_6=[72.4, 390.9, 149.8, -136.2, -87.6, 82.1]
pos_7=[108.9, 466.6, 399.1, -178.5, -47.6, 120.0]
pos_8=[108.9, 497.6, 399.1, -178.5, -47.6, 120.0]
pos_9=[129.4, 468.0, 385.7, -175.7, -57.2, 117.1]
pos_10=[156.8, 436.7, 251.5, -126.9, -89.2, 71.6]
pos_11=[154.0, 444.6, 213.0, 37.4, -90.0, -92.1]
pos_12=[78.9, 259.4, 175.6, -52.2, -90.0, -119.5]
pos_13=[78.9, 259.4, 141.8, -47.1, -90.0, -124.6]
pos_14=[17.9, 259.4, 93.0, 137.8, -90.0, 50.5]
pos_15=[17.9, 259.4, 172.6, -50.4, -90.0, -121.3]
pos_16=[17.9, 259.4, 188.3, 89.2, -90.0, 156.4]
pos_flow_0=[190.3, 335.1, 296.3, -88.9, 2.1, 29.3]
pos_flow_1=[179.0, 486.6, 332.6, -91.1, 0.4, 31.6]
pos_flow_2=[179.0, 486.6, 379.8, -91.1, 0.4, 31.6]
pos_flow_3=[177.7, 491.0, 325.0, -91.0, 0.3, 31.9]
pos_flow_4=[177.7, 491.0, 293, -91.0, 0.3, 31.9]
gri_vaso1=130.0
gri_vaso2=120.0
gri_vaso3=370.0
gri_vaso4=302.0
gri_flow=500.0
speed_mov=100.0
acc_mov=2000.0
speed_mov1=500.0
acc_mov1=10000.0
speed_mov2=100.0
```

acc_mov2=2000.0

speed_flow=200.0

acc_flow=4000.0

15.3. Índice de planos

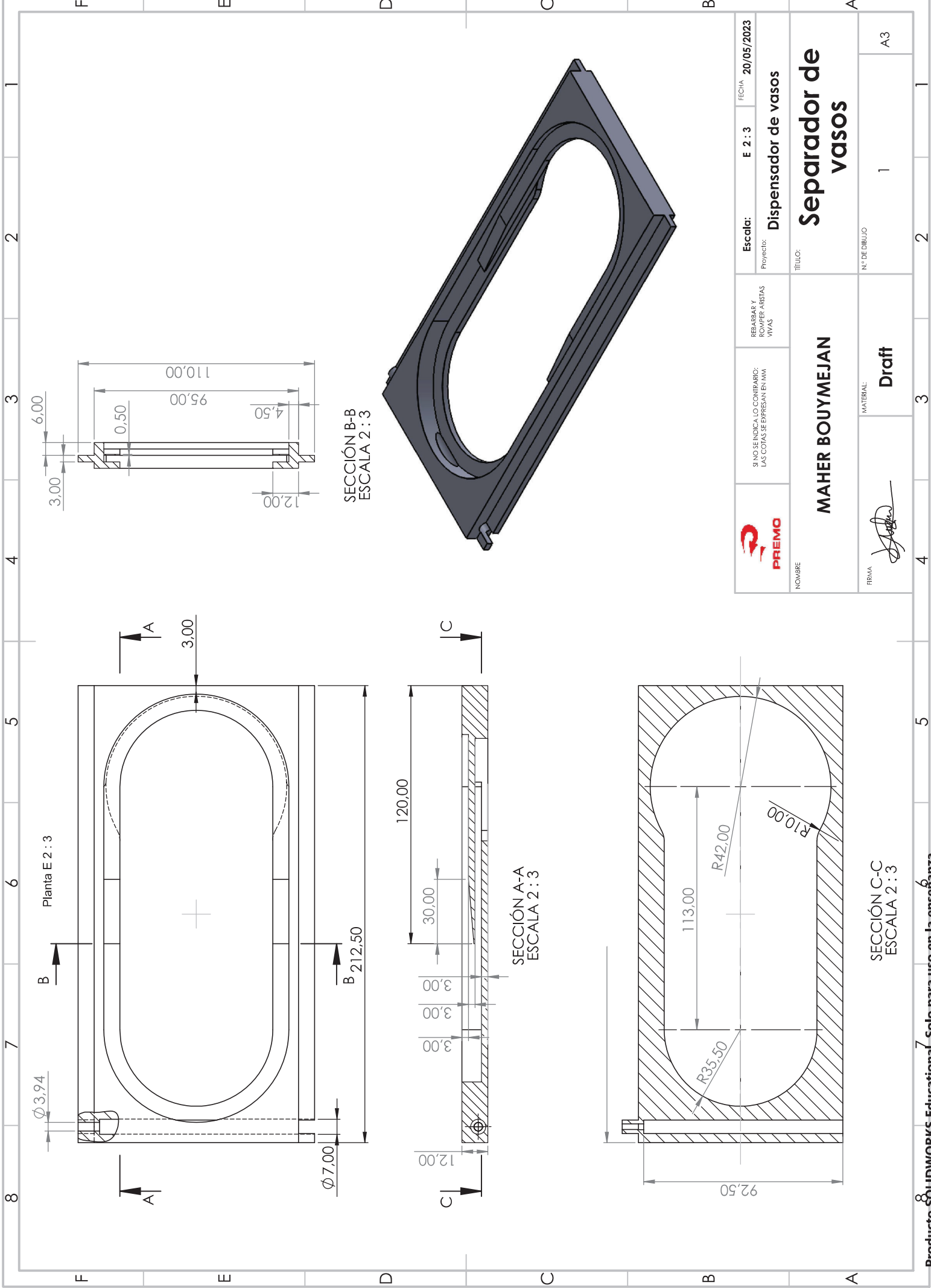
Plano 1: Separador de vasos.

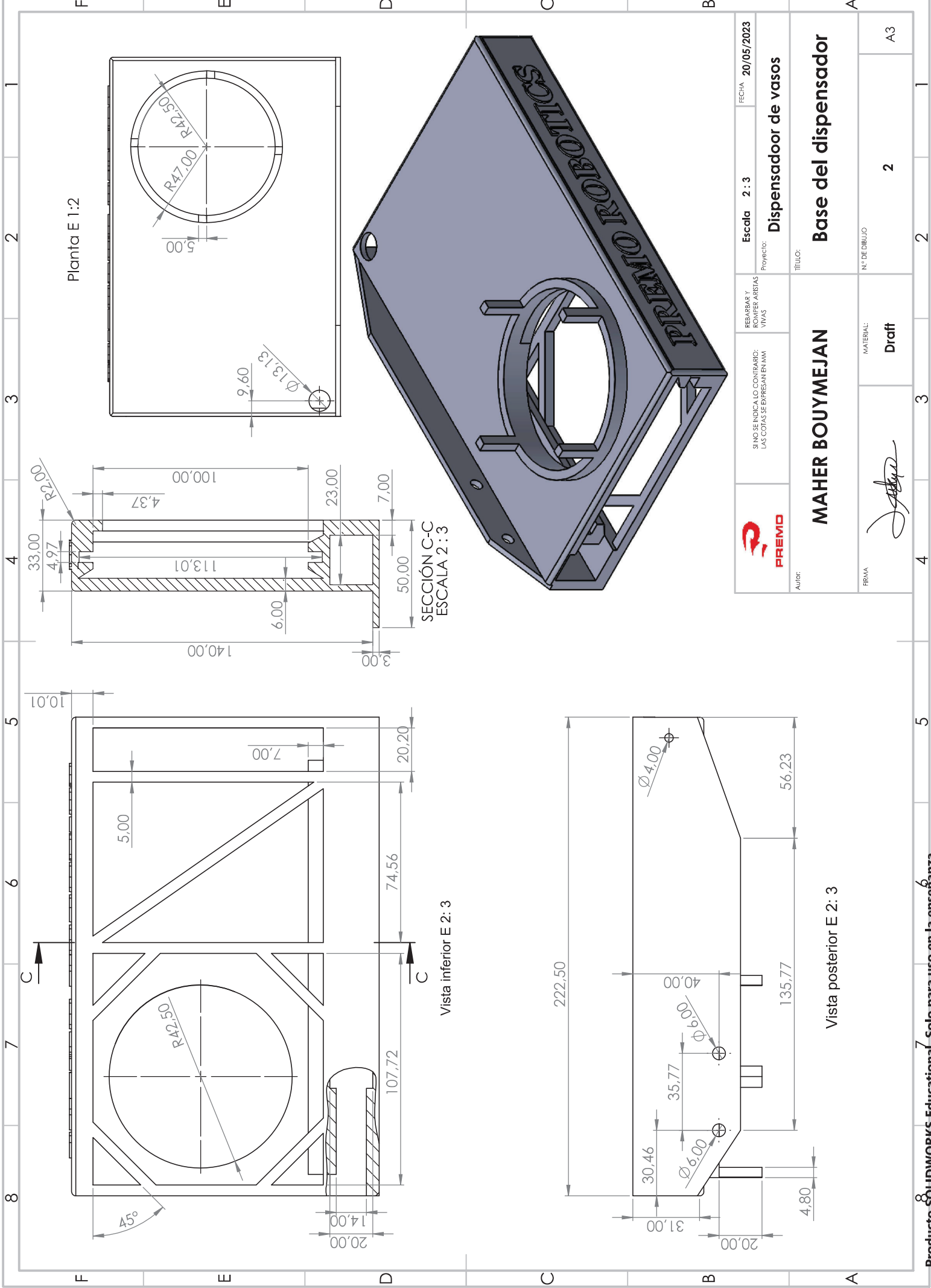
Plano 2: Base del dispensador de vasos.

Plano 3: Ensamblaje del dispensador de vasos.

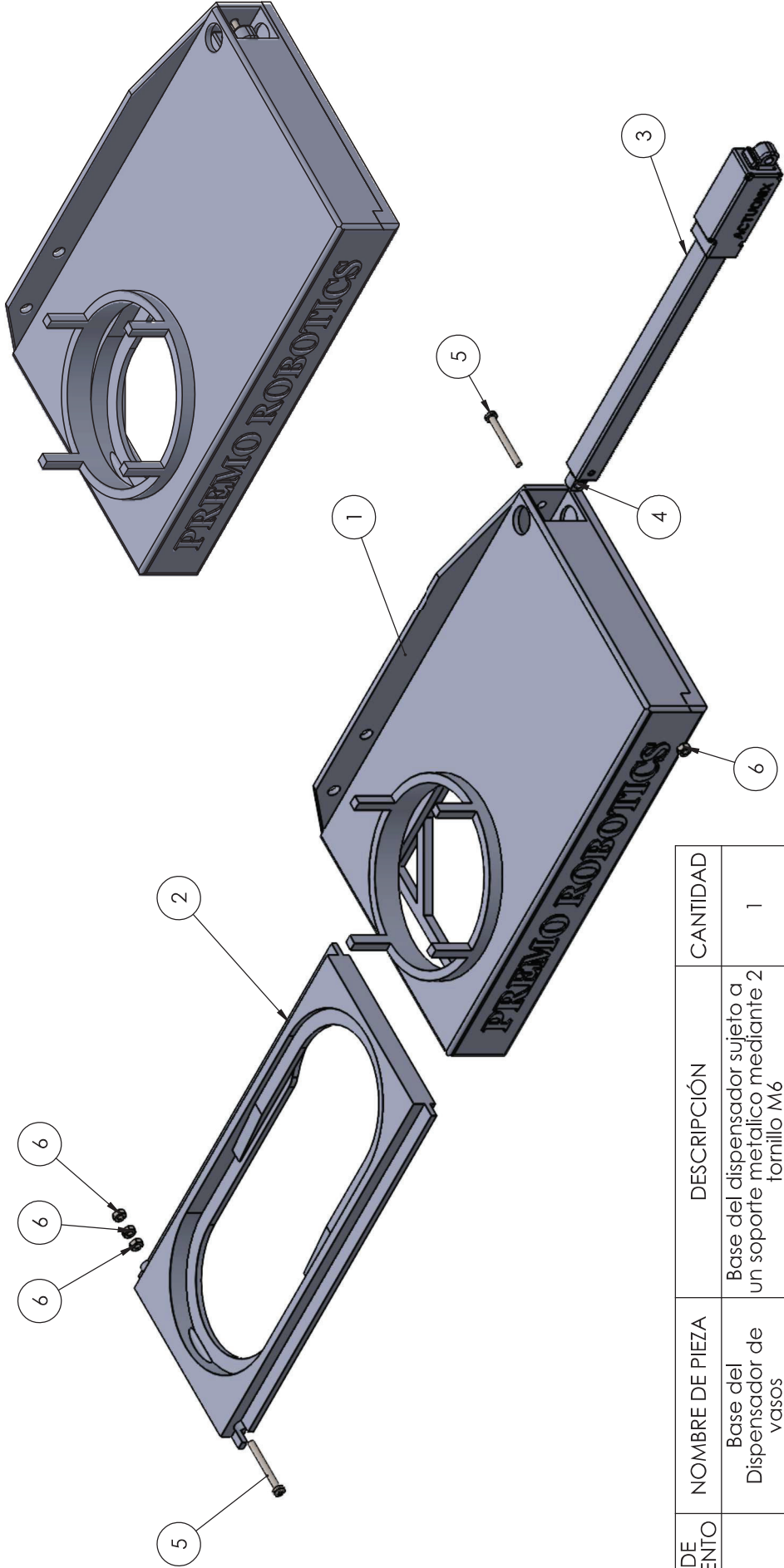
Plano 4: Pinzas del brazo robótico.

Plano 5: Adaptador biela manivela.



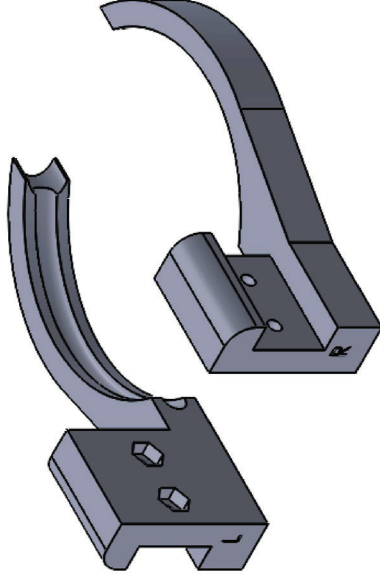
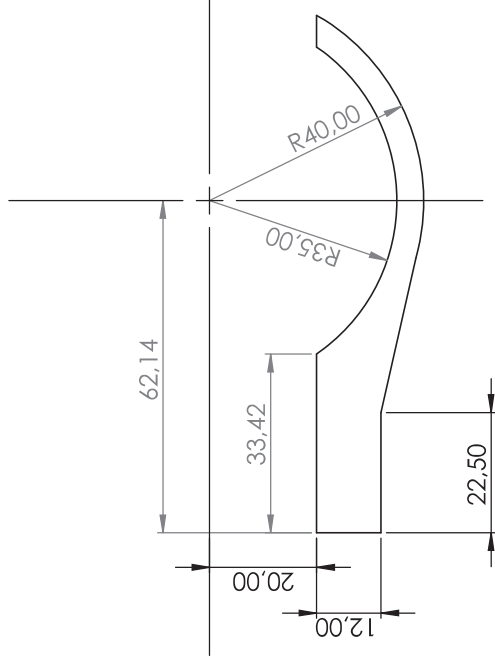
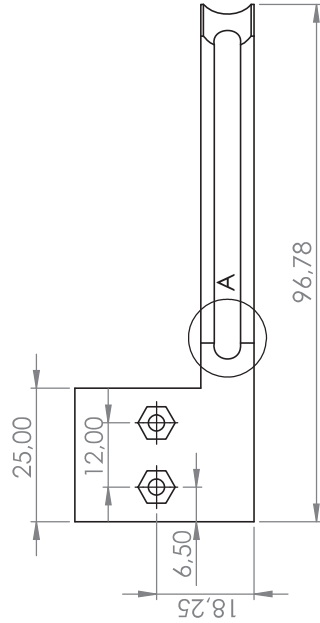
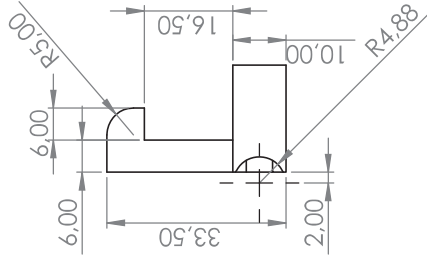
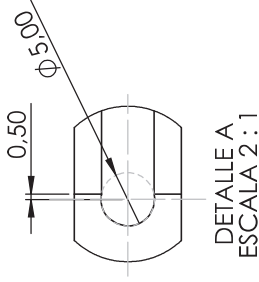
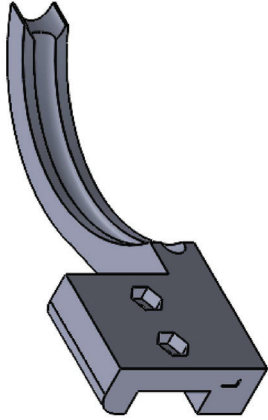


	REBARBAR Y ROMPER ARISTAS VIVAS		Escala 2 : 3		FECHA 20/05/2023				
	SINO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM		Proyecto:		Dispensador de vasos				
	Autor:		TÍTULO:		Base del dispensador				
MAHER BOUYMEJAN		MATERIAL:		N° DE DIBUJO		2		A3	
FIRMA				Draft					

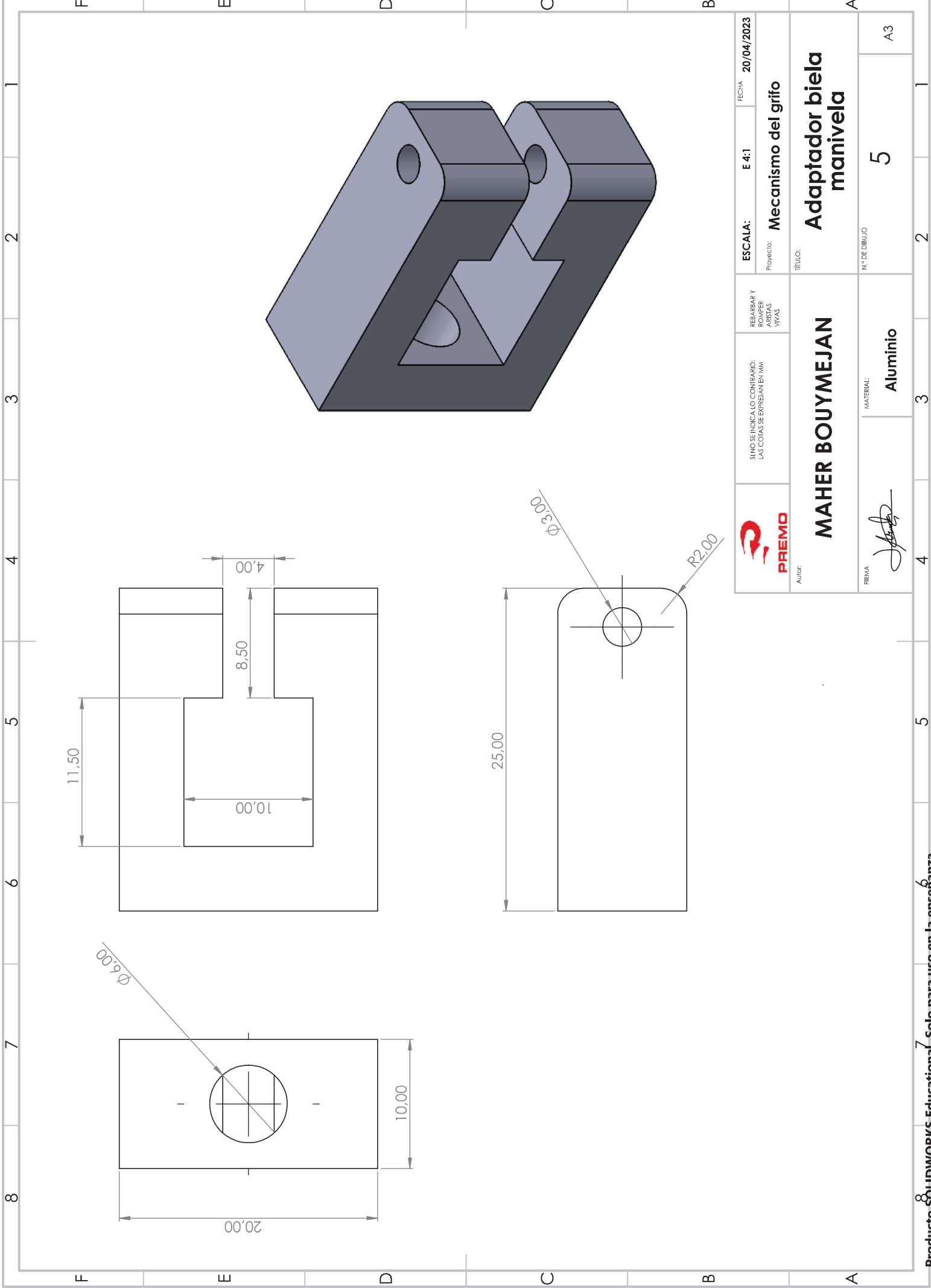


N.º DE ELEMENTO	NOMBRE DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	Base del Dispensador de vasos	Base del dispensador sujeto a un soporte metálico mediante 2 tornillo M6	1
2	Separador de vasos	Parte móvil accionado por el actuador lineal	1
3	Actuador lineal L16_140mm	Fijado a la base mediante un tornillo y una tuerca M3	1
4	Vástago del Actuador lineal	Unido al separador de vasos mediante un tornillo M3 y ajustado mediante 3 Tuercas M3	1
5	Tornillo M3 x 0.5 x 30		2
6	Tuerca - Hex M3 x 0.5		4

	SINO SE INDICA LO CONTRARIO, LAS COTAS SE EXPRESAN EN MM	REBARBAR Y ROMPER ARISTAS VIVAS	Escala	E 1 : 2	FECHA	20/04/2023
			Proyecto: Dispensador de vasos			
			TÍTULO: Ensamblaje del dispensador de vasos			
Autor: MAHER BOUYMEJAN		MATERIAL: Drafft		N.º DE DIBUJO		A3
FIRMA: 						



	SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM	REBARBAR Y ROMPER ARISTAS VIVAS	ESCALA: E 1:1	FECHA: 20/04/2023
			Proyecto: Pinza del robot	
			TÍTULO: Pinzas del brazo robotico	
Autor: MAHER BOUYMEJAN			N° DE DIBUJO: 4	
MATERIAL: Draft			A3	
Firma: 				



	SINO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM	REBARBAR Y ROMPER ARISTAS VIVAS	ESCALA:	E 4:1	FECHA	20/04/2023	
			Proyecto:				Mecanismo del grifo
			TÍTULO:				Adaptador biela manivela
			N° DE DIBUJO				5
MATERIAL:			Aluminio				
Firma							
Autor:			MAHER BOUYMEJAN				