

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA,
CIENCIA Y COMPUTACIÓN (STEM)

GRADO EN INGENIERÍA EN SISTEMAS INDUSTRIALES. MENCIÓN EN ELECTRÓNICA Y AUTOMÁTICA + ROBÓTICA

TRABAJO FIN DE GRADO

Control, montaje y simulación de un sistema de llenado de cremas

Alumno: **Dª. Ester López Hierro**
Director: **Dr. Héctor Eloy Sánchez Sardi**

JUNIO 2026

Índice

Capítulo 1: Introducción **9**
1.1 Justificación . . . 9
1.2 Objetivos . . . 9
Capítulo 2: Estado del arte **10**
2.1 Control de llenado . . . 10
2.2 Bomba peristáltica . . . 10
2.3 Análisis Comparativo: MATLAB/Arduino vs. PLC Industrial . . . 11
2.4 Algoritmos de Llenado y Optimizaciones Recientes . . . 11
2.5 Robótica de final de línea . . . 11
2.6 Sistemas de Sellado y Control de Par de Apriete . . . 11
2.7 Planificación de Trayectorias y Algoritmos de Disposición Espacial . . . 12
2.8 Controladores de Bajo Coste en Robótica Educativa y de Prototipado . 12
Capítulo 3: Marco teórico **13**
3.1 Desarrollo de Controladores en el Ecosistema MATLAB-Arduino . . . 13
3.2 Configuraciones Cinemáticas para Tareas de Pick and Place . . . 13
3.3 Entornos de Simulación y Programación Fuera de Línea . . . 14
3.4 El Autómata Programable . . . 14
3.5 Estándares de Programación . . . 15
3.6 Coordinación de la Línea de Transporte y Sincronismo de Sensores . . . 15
3.7 Gestión de Modos de Marcha: La Guía GEMMA . . . 15
3.8 GRAFCET . . . 16
3.9 Validación Mediante Gemelos Digitales . . . 17
3.10 Puente de Wheatstone y Ley de Ohm . . . 17
3.11 El Transistor MOSFET . . . 17
3.12 Ley de Watt y Factor de Derating (Reducción) . . . 18
3.13 El Proceso Integrador . . . 18
3.14 Control PID (Proporcional-Integral-Derivativo) . . . 18
3.15 Sintonización de Ziegler-Nichols en Procesos Integradores . . . 19
3.16 Protocolo de comunicación MQTT en entornos industriales IoT . . . 19
Capítulo 4: Control de llenado **20**
4.1 Montaje . . . 20
4.2 Componentes eléctricos y circuito impreso . . . 32
4.3 Control programado mediante MATLAB . . . 40
Capítulo 5: Simulación del robot - gemelo digital **52**
5.1 Arquitectura de señales (E/S) en el controlador virtual . . . 52
5.2 Señales de simulación para mostrar la ubicación de los envases . . . 55
5.3 Conclusión . . . 56
Capítulo 6: Programación industrial **57**
6.1 Material para la programación industrial: . . . 57
6.2 Programación del PLC . . . 58
6.3 Programación . . . 62
6.4 Pruebas . . . 64
Capítulo 7: Estudio Económico **65**
7.1 Costes de hardware y materiales del prototipado . . . 65

7.2 Costes de hardware industrial 66
7.3 Costes de software 67
7.4 Costes de ingeniería y mano de obra 68
7.5 Presupuesto total 69

Capítulo 8: Conclusiones **70**

Capítulo 9: Líneas Futuras **72**
9.1 Monitorización remota 72
9.2 Unificación de la fuente de alimentación 72
9.3 Fabricación del circuito impreso (PCB) 72
9.4 Cuadro de mandos 72
9.5 Visión artificial 72

Índice de figuras

1 Dentado de las piezas . . . 20
2 Bomba peristáltica parte posterior . . . 21
3 Bomba peristáltica parte frontal . . . 21
4 Bomba peristáltica parte lateral . . . 21
5 Báscula . . . 21
6 Switch de frente . . . 22
7 Switch de lateral . . . 22
8 MDF . . . 22
9 ESP32 . . . 23
10 Pieza vertical en 3D . . . 24
11 Pieza de soporte en 3D . . . 25
12 Pieza corto de soporte en 3D . . . 25
13 Pieza más corto de soporte en 3D . . . 25
14 Pieza tapa de la báscula en 3D . . . 25
15 Pieza lateral 1 en 3D . . . 26
16 Pieza lateral 2 en 3D . . . 26
17 Pieza lateral grande en 3D . . . 26
18 Primer ensamblaje . . . 26
19 Segundo ensamblaje . . . 26
20 Montaje físico parte frontal . . . 27
21 Montaje físico parte posterior . . . 28
22 Prueba de la báscula de precisión . . . 30
23 NAU7802 24-bit ADC . . . 31
24 Galga extensiométrica de pesaje máximo de 100g . . . 31
25 Motor de bomba peristáltica . . . 32
26 Relé electromecánico de 12V . . . 33
27 Módulo de control de potencia MOSFET . . . 33
28 Curva de Características de Salida del MOSFET . . . 34
29 Curva de Transferencia del MOSFET . . . 34
30 Implementación del diodo Flyback 1N4007 . . . 37
31 Visualización de PCB en 3D . . . 39
32 Paquete de Arduino . . . 40
33 S-Function Builder . . . 40
34 Compilador de C++ . . . 41
35 Sample and Hold - flanco de subida . . . 45
36 Configuración del bloque de retardo . . . 46
37 Configuración la saturación del PID . . . 47
38 Obtención de datos para los cálculos de Ziegler-Nichols . . . 48
39 Valor de Tiempo muerto (L) . . . 48
40 Valor de Ganancia (K) . . . 48
41 Parámetros de configuración de PID . . . 49
42 Parámetros de configuración de PID . . . 50
43 Gráfico de llenado de 3 botes (eje X: tiempo; eje Y: gramos) . . . 51
44 Configuración de entradas y salidas digitales . . . 52
45 Configuración de señal del sensor de la segunda cinta . . . 53
46 Configuración de entradas y salidas digitales . . . 54

47 Configuración de entradas y salidas digitales 54
48 Enrutamiento de señales controlador y señal de origen 55
49 Conexiones entre la controladora y el entorno 3D 56
50 Sistema industrial 57
51 Esquema Guía GEMMA 59
52 Configuración de perfiles, grupos y tiempos de cierre de sesión automático en TIA Portal. 64
53 Ventana de autenticación de seguridad en la interfaz HMI. 64

Índice de tablas

1 Desglose de costes de hardware industrial 65
2 Desglose de costes del prototipo 66
3 Desglose de costes de licencias y software 67
4 Desglose de costes de ingeniería y mano de obra 68
5 Resumen general del presupuesto del proyecto 69

TITULO: Control de un sistema de llenado de cremas

AUTOR: Ester López Hierro

DIRECTOR DEL PROYECTO: Héctor Eloy Sánchez Sardi

FECHA: 7 de junio 2026

RESUMEN

Este Trabajo de Fin de Grado expone el diseño, desarrollo e implementación de una celda de fabricación destinada a la automatización integral de una línea de envasado de productos cosméticos. El principal objetivo es garantizar un proceso de dosificación higiénico y de alta precisión, así como un paletizado eficiente, alineado con los requerimientos técnicos de la Industria 4.0.

Para el control del flujo de fluido, se ha implementado un sistema gravimétrico en lazo cerrado gestionado a través de un microcontrolador ESP32 y el entorno computacional de MATLAB.
El control de la bomba peristáltica se realiza a través de un transistor MOSFET a nivel lógico y los componentes están soldados en una placa de circuito impreso (PCB) diseñada a medida. El control matemático de la dosificación de la crema se asegura usando un algoritmo Proporcional-Integral (PI), prescindiendo de la acción derivativa para evitar ruido mecánico y eléctrico.

Para la etapa de final de línea, se ha elaborado el Gemelo Digital de un brazo manipulador industrial empleando el software RobotStudio, lo que ha permitido comprobar y optimizar la cinemática de las señales así como evitar colisiones antes de implementarlo.

Finalmente todo el conjunto (cintas transportadoras, robot y llenado) se controla mediante un Autómata Programable (PLC) industrial configurado mediante TIA Portal. La arquitectura estructural sigue las normas de GRAFCET y la seguridad operativa se rige por la metodología de la Guía GEMMA.
Los resultados técnicos y el estudio económico avalan la viabilidad de una solución híbrida que fusiona tecnologías de prototipado ágil con robustos estándares de control industrial, logrando una línea determinista y estructuralmente preparada para la adopción de protocolos de telemetría IoT como MQTT.

Palabras clave: Control PID; Dosificación gravimétrica; Gemelo Digital; Autómata Programable (PLC); GRAFCET; Electrónica de potencia; Industria 4.0.

ABSTRACT

This Bachelor's Thesis presents the design, development, and implementation of a manufacturing cell aimed at the comprehensive automation of a cosmetic product packaging line. The main objective is to guarantee a highly precise and hygienic dosing process, alongside efficient palletizing, aligned with the technical requirements of Industry 4.0.

For fluid flow control, a closed-loop gravimetric system has been implemented, managed through an ESP32 microcontroller and the MATLAB computational environment. The control of the peristaltic pump is executed via a logic-level MOSFET transistor, with the components soldered onto a custom-designed printed circuit board (PCB). The mathematical control of the cream dosing is ensured using a

Proportional-Integral (PI) algorithm, dispensing with the derivative action to prevent mechanical and electrical noise.

For the end-of-line stage, a Digital Twin of an industrial manipulator arm has been developed using RobotStudio software. This has made it possible to verify and optimize the kinematics and control signals, as well as to prevent collisions prior to its physical implementation.

Finally, the entire assembly (conveyor belts, robot, and filling station) is controlled by an industrial Programmable Logic Controller (PLC) configured via TIA Portal. The structural architecture adheres to GRAFCET standards, and operational safety is governed by the GEMMA Guide methodology.

The technical results and economic study endorse the viability of a hybrid solution that merges agile prototyping technologies with robust industrial control standards, resulting in a deterministic line structurally prepared for the adoption of IoT telemetry protocols such as MQTT.

Keywords: PID Control; Gravimetric dosing; Digital Twin; Programmable Logic Controller (PLC); GRAFCET; Power electronics; Industry 4.0.

Capítulo 1: Introducción

El objetivo principal de este proyecto es el diseño de un sistema de control para el llenado de cremas y su posterior automatización y robotización dentro de un proceso industrial.
El proceso se realiza mediante una cinta transportadora en la que se depositan los botes de crema. Una vez que llegan a la báscula se para la cinta, y empieza el proceso de llenado. Éste se realiza mediante un control de lazo cerrado.
Una vez que el bote alcanza el peso deseado, se detiene el llenado y la cinta pasa a la siguiente etapa en la que el robot cierra los envases y los empaqueta en cajas.

1.1. Justificación

La automatización industrial actual requiere la implementación de la industria 4.0. Esta transformación digital permite aumentar la producción, dotar de flexibilidad a la línea, tomar decisiones en tiempo real, reducir el error humano y garantizar los estándares de calidad necesarios.
En el sector de la cosmética, mantener los controles de calidad es imprescindible debido a la naturaleza de los productos.
Se deben tener en cuenta factores como:

- La inconsistencia en el llenado de los recipientes debido a ajustes incorrectos de la bomba dosificadora o a la presencia de burbujas de aire.
- Los problemas mecánicos como la obstrucción de boquillas o fugas en la línea debida a la degradación de los sellos.

1.2. Objetivos

El objetivo de este proyecto es realizar la automatización de una línea de llenado de cremas de principio a fin, de forma que sea eficiente y pueda implementarse en una industria real.
El procedimiento para desarrollarlo será el siguiente:

- Desarrollo de la maqueta física del control del llenado y realización de pruebas.
- Uso de MATLAB con ESP32 para gestionar el control del llenado
- Integración de una simulación de brazo robótico que se encarga de cerrar los botes y de colocarlos en la caja para ser transportados.
- Diseño y puesta en práctica en una simulación de la lógica de la automatización englobando el control y el empaquetado de las cremas.
- Estudio económico del sistema.

Capítulo 2: Estado del arte

Este apartado hace referencia al uso de las tecnologías actuales aplicadas en la automatización, y especialmente a las medidas que garantizan la precisión e higiene que deben mantenerse en la manipulación de productos cosméticos.
Durante las últimas décadas, esta industria ha experimentado una transformación, evolucionando desde sistemas de producción rígidos de accionamiento puramente mecánico hacia entornos de fabricación flexible (Industria 4.0).

2.1. Control de llenado

La transición tecnológica en las líneas de envasado de los sectores cosmético y alimentario ha estado marcada por el paso de mecanismos puramente mecánicos a sistemas avanzados de lazo cerrado que priorizan la precisión.
Según se describe en el *Springer Handbook of Automation* (Nof 2009), esta evolución responde a la necesidad de optimizar recursos y garantizar la calidad del producto final.
En este contexto, la dosificación gravimétrica (basada en el peso) se presenta como la alternativa más fiable frente a los métodos volumétricos tradicionales, especialmente cuando se trabaja con sustancias cuyas propiedades de flujo son complejas (Hassan 2022).

2.2. Bomba peristáltica

Para el manejo de cremas y geles, la literatura técnica coincide en que la arquitectura peristáltica es la más adecuada.
Investigadores como (Kunt 2017) destacan que su diseño permite un bombeo no invasivo, en el que el material nunca entra en contacto con los componentes mecánicos de la bomba, garantizando la higiene y evitando la degradación del fluido.
Desde la perspectiva del control, la implementación de señales de modulación por ancho de pulsos (PWM) en estos dispositivos, permite gestionar caudales con un error marginal.
(Kunt & Oguz 2017) demuestran que, mediante el uso de microcontroladores, es posible alcanzar precisiones superiores al 98 %, lo que resulta fundamental en aplicaciones en las que el margen de error en la dosificación es mínimo.
Asimismo, (Wang et al. 2018) subrayan la importancia de controlar la velocidad rotacional para compensar la histéresis del tubo elastómero, un factor determinante para la repetibilidad del proceso.

2.2.1. Metodologías de Medición: El Control Gravimétrico y Filtrado de Datos

El uso de sistemas de medición de peso permite anular las incertidumbres provocadas por las variaciones de densidad térmicas o la aireación del producto.
La integración de células de carga junto con amplificadores como el HX711 ha sido validada como una solución de bajo coste y alto rendimiento para la automatización a pequeña escala. Sin embargo, (García-García et al. 2020) advierten que hay que tener en cuenta la necesidad de atenuar las vibraciones mecánicas inducidas por los motores.

Su investigación sugiere que la aplicación de filtros digitales es imperativa para estabilizar la lectura del peso y permitir una toma de decisiones precisa en el lazo de control.

2.3. Análisis Comparativo: MATLAB/Arduino vs. PLC Industrial

Un punto fundamental en la justificación técnica de este proyecto es la superioridad del entorno MATLAB/Arduino sobre los PLCs tradicionales para fines experimentales. Mientras que el PLC es el estándar de robustez en producción masiva, (Mainali et al. 2022) argumentan que carece de la flexibilidad necesaria para el tratamiento de señales complejas y el modelado matemático avanzado que requiere un fluido no newtoniano. La facilidad para poner en marcha filtros de señal sofisticados y algoritmos de control predictivo en MATLAB ofrece una ventaja competitiva en el diseño de prototipos de precisión que un entorno de lógica de contactos (Ladder) difícilmente puede igualar de forma económica.

2.4. Algoritmos de Llenado y Optimizaciones Recientes

Para minimizar el error causado por la inercia del fluido una vez detenida la bomba, se han desarrollado estrategias de llenado multietapa.
El trabajo de (Ramesh et al. 2019) propone un algoritmo que segmenta el proceso en dos fases: una de caudal máximo para reducir el tiempo de operación y una de caudal reducido (goteo) para ajustar el peso final.
Esta técnica, combinada con el procesamiento de la báscula en lazo cerrado descrito por (Setyawan et al. 2019), permite reducir el error residual a niveles inferiores al 1 %, optimizando así la eficiencia del sistema de envasado.

2.5. Robótica de final de línea

La automatización de las etapas posteriores al llenado, conocidas como “final de línea”, ha experimentado una transformación tendente a la flexibilidad.
Según se expone en el *Springer Handbook of Robotics*, (Siciliano & Khatib 2016), la integración de brazos robóticos permite una transición fluida entre la producción y la logística de embalaje.
A diferencia de los sistemas de automatización rígida, la robótica actual permite adaptar la organización de los envases en cajas mediante cambios en el software, lo que resulta esencial en la industria cosmética donde el formato de los envases puede variar con frecuencia.
En este contexto, la validación mediante entornos de simulación digital previa a la implementación física se ha vuelto indispensable.

2.6. Sistemas de Sellado y Control de Par de Apriete

Una fase importante previa a la organización final es el cierre o taponado de los envases. En la industria del envasado, esta tarea ha evolucionado desde cabezales mecánicos fijos hacia actuadores robóticos con control de fuerza.
Según destaca (Bahr et al. 2020), la implementación de servomotores con control de par (torque) en el actuador final es fundamental para garantizar que cada envase quede

sellado herméticamente sin exceder los límites de resistencia del material.
La precisión en el cierre por giro de las tapas se logra mediante la integración de sensores de fuerza/par que permiten al robot monitorizar la resistencia durante el giro. Esta técnica, analizada exhaustivamente en el *Springer Handbook of Robotics*, (Siciliano & Khatib 2016), permite compensar pequeñas desviaciones en la alineación del envase y asegurar que el cierre se realice siguiendo una trayectoria helicoidal perfecta.

2.7. Planificación de Trayectorias y Algoritmos de Disposición Espacial

El reto de colocar envases de crema en cajas de forma organizada reside en la planificación de trayectorias libres de colisiones.
(Gasparetto et al. 2015) proponen que la eficiencia del ciclo dependa de algoritmos de suavizado de trayectoria que minimicen los esfuerzos en los motores.
La sincronización entre la visión artificial y el brazo robótico permite corregir desviaciones en la posición del envase sobre la cinta, asegurando un encajado milimétrico (Bi & Zhang 2023).

2.8. Controladores de Bajo Coste en Robótica Educativa y de Prototipado

La integración de MATLAB como entorno de simulación robótica permite realizar el cálculo de la cinemática inversa de forma eficiente, mientras que los microcontroladores como Arduino gestionan el control de los actuadores del brazo.
Esta metodología, validada en estudios de robótica colaborativa (Katz 2021), permite emular el comportamiento de una célula industrial a una fracción de su coste, facilitando el estudio de la dinámica de manipulación en entornos académicos.

Capítulo 3: Marco teórico

Este apartado establece los fundamentos teóricos, físicos y normativos indispensables para comprender el comportamiento y la arquitectura lógica de la línea de envasado. Este capítulo define los principios teóricos usados en el sistema.

3.1. Desarrollo de Controladores en el Ecosistema MATLAB-Arduino

La implementación de un sistema de control avanzado requiere una arquitectura que combine la potencia de cálculo con la versatilidad en la adquisición de datos.
En este escenario, se definen dos pilares tecnológicos fundamentales bajo el marco de trabajo propuesto por (Katz 2021):

- **MATLAB (Matrix Laboratory):** Se define como un entorno de computación numérica y lenguaje de programación de alto nivel óptimo para el análisis de datos, el desarrollo de algoritmos y el modelado matemático (MathWorks, Inc. 2023).
  En ingeniería de control su relevancia reside en la capacidad de procesar señales en tiempo real y diseñar controladores mediante herramientas visuales como Simulink.

- **Arduino:** Es una plataforma de hardware de código abierto basada en microcontroladores reprogramables y un entorno de desarrollo (IDE) (Arduino LLC 2023). Su arquitectura permite la interacción directa con sensores (células de carga) y actuadores (bombas), actuando como una interfaz de bajo coste para la adquisición de datos (DAQ).

La elección de este ecosistema responde a la necesidad de una rápida iteración y capacidad analítica.
(Katz 2021) expone que esta combinación facilita la transición entre el modelo matemático y la implementación física, permitiendo a los ingenieros sintonizar parámetros de control PID sin las restricciones de programación de lenguajes de bajo nivel.
En esta línea, (Angani et al. 2021) han demostrado que el uso de *MATLAB Support Packages* permite monitorizar variables con una fidelidad comparable a sistemas profesionales.

3.2. Configuraciones Cinemáticas para Tareas de Pick and Place

La selección de la arquitectura del manipulador es fundamental para el éxito de la organización automatizada.
La literatura técnica distingue principalmente dos configuraciones para estas tareas, por sus capacidades cinemáticas:

- **Robots SCARA (Selective Compliance Assembly Robot Arm):** Según la definición técnica proporcionada por (Siciliano & Khatib 2016), estos sistemas se caracterizan por su rigidez en el eje vertical y alta velocidad en el plano horizontal. (Pires 2007) argumenta que esta configuración es óptima para el traslado de envases cilíndricos desde cintas transportadoras hacia cajas, garantizando una repetibilidad excepcional en ciclos de alta cadencia.

- **Manipuladores Antropomórficos (6 GDL):** Se caracterizan por su estructura de brazo articulado que emula el movimiento humano, ofrecen una versatilidad superior para orientar los envases en orientaciones complejas (Siciliano & Khatib 2016). Según (Bahr et al. 2020), estos brazos son preferibles cuando el embalaje requiere movimientos de rotación adicionales para ajustar etiquetas, sellos de seguridad o inserciones angulares.

3.3. Entornos de Simulación y Programación Fuera de Línea

En el diseño de celdas automatizadas contemporáneas, el uso de herramientas de programación fuera de línea (OLP, siglas en inglés) es fundamental para reducir tiempos de puesta en marcha.

- **RobotStudio:** Es una herramienta de simulación y programación fuera de línea basada en el concepto de *VirtualController*, que permite replicar de forma exacta el comportamiento de los robots industriales en un entorno digital (Duka 2014). Su relevancia en este proyecto reside en la capacidad de verificar colisiones, optimizar trayectorias y calcular tiempos de ciclo sin necesidad de interrumpir el proceso físico o arriesgar el hardware.

Investigadores como (Duka 2014) señalan que RobotStudio facilita la creación de "gemelos digitales" de la estación de trabajo, permitiendo integrar cintas transportadoras, sensores y actuadores en un entorno unificado.

3.4. El Autómata Programable

El núcleo operativo de las líneas de producción contemporáneas reside en la capacidad de procesamiento de los Controladores Lógicos Programables (PLC).
(Bolton 2015) define formalmente al PLC como un sistema electrónico de base digital, diseñado para su aplicación en entornos industriales, que utiliza una memoria programable para el almacenamiento interno de instrucciones orientadas a implementar funciones lógicas, de secuenciación, temporización y aritmética.
A diferencia de los sistemas de control basados en lógica cableada, el PLC ofrece una flexibilidad inherente que permite la reconfiguración del proceso sin alteraciones físicas en el hardware.
Esta arquitectura de estado sólido es la que garantiza la sincronización entre los diversos actuadores de una línea de envasado, como motores de inducción para cintas y electroválvulas de dosificación, (Bolton 2015).

3.5. Estándares de Programación

En la práctica industrial, el desarrollo de la lógica de control se sustenta en plataformas de software propietario que, no obstante, convergen en el estándar internacional IEC 61131-3. Este estándar define los lenguajes de programación (como Ladder o Texto Estructurado) que aseguran la portabilidad y comprensión del código entre diferentes fabricantes.

- **TIA Portal (Siemens):** Es la plataforma de ingeniería totalmente integrada más extendida en el mercado europeo. Permite la configuración unificada de controladores SIMATIC (S7-1200/1500), interfaces HMI y sistemas de accionamiento, facilitando la comunicación mediante protocolos industriales como PROFINET.

- **Otras Plataformas Líderes:** Aunque TIA Portal es el estándar en muchos sectores, soluciones como *Studio 5000* (Rockwell Automation / Allen-Bradley) en el mercado americano, *EcoStruxure* (Schneider Electric) o *Sysmac Studio* (Omron) funcionan con principios de automatización similares. Todas estas marcas comparten la capacidad de gestionar tareas de tiempo real y comunicación distribuida, variando principalmente en la interfaz de usuario y la gestión de librerías propietarias, (Bolton 2015).

3.6. Coordinación de la Línea de Transporte y Sincronismo de Sensores

El movimiento de los envases a través de las diversas estaciones de trabajo requiere una gestión precisa de las cintas transportadoras.
El sistema propuesto, que utiliza dos cintas diferenciadas, se fundamenta en la arquitectura de control por eventos.
Según (Bi & Zhang 2023), la detección mediante sensores fotoeléctricos o de proximidad es el disparador que permite al PLC detener la primera cinta en la posición exacta de llenado. Una vez que el sensor de posición confirma la ubicación del bote y el sistema de control (integrado con MATLAB/Arduino) finaliza la dosificación gravimétrica, el PLC recibe una señal de "fin de proceso".
En este punto, se activa la transición hacia la segunda cinta transportadora, que desplaza el envase hacia la zona de acción del robot.
Esta transferencia de información entre el nivel de campo (sensores) y el nivel de control es lo que garantiza un flujo continuo y libre de colisiones (Bi & Zhang 2023).

3.7. Gestión de Modos de Marcha: La Guía GEMMA

Para que un sistema industrial sea considerado seguro y profesional, no basta con programar el "funcionamiento normal".
Es necesario prever cómo se comportará la máquina ante fallos o paradas de mantenimiento. En este ámbito, la **Guía GEMMA** (*Guide des Études des Modes de Marches et d'Arrêts*)ADEPA (1980), es la metodología de referencia (López et al. 2020).
La Guía GEMMA proporciona un marco gráfico y conceptual que organiza los estados del sistema en tres familias principales:

1. **Procedimientos de funcionamiento (F):** Incluyen la producción normal y las fases de preparación.

2. **Procedimientos de parada y de puesta en marcha (A):** Gestionan la transición segura hacia el estado de reposo y el reinicio tras una pausa.

3. **Procedimientos de fallo (D):** Definen la reacción del sistema ante emergencias, garantizando la protección de los operarios y del hardware.

La implementación de la metodología GEMMA mediante el PLC asegura que el sistema sea determinista; es decir, que ante cualquier circunstancia (como una pulsación de la seta de emergencia o un fallo de suministro), la línea de envasado sepa exactamente a qué estado seguro debe evolucionar (López et al. 2020).

3.8. GRAFCET

El diseño de sistemas de control secuencial requiere herramientas que permitan describir de forma rigurosa y sin ambigüedades el comportamiento de un automatismo. En este contexto, el GRAFCET (Graphe Fonctionnel de Commande Étapes/Transitions) se ha consolidado como el estándar industrial y académico por excelencia.
Derivado matemáticamente de las Redes de Petri, el GRAFCET es un lenguaje gráfico de especificación estandarizado a nivel internacional bajo la norma IEC 60848, adoptada en España como UNE-EN 60848 UNE (2013).
Su estructura se fundamenta en la alternancia estricta de etapas (que representan los estados del sistema y tienen asociadas acciones físicas) y transiciones (que evalúan condiciones lógicas o eventos para permitir la evolución entre estados).
La principal cualidad de aplicación del GRAFCET es su capacidad para actuar como puente de comunicación entre los diferentes departamentos de ingeniería (mecánica, procesos y control). Al proporcionar una representación visual intuitiva pero matemáticamente rigurosa, se utiliza
para la síntesis de controladores lógicos programables (PLC).
Permite traducir directamente las especificaciones de funcionamiento de una máquina a lenguajes de programación normalizados por la IEC 61131-3, como el diagrama de contactos (Ladder) o el texto estructurado (ST) Mandado Pérez et al. (2009).
Esta versatilidad lo hace indispensable para diagnosticar fallos, prevenir bloqueos en las secuencias y garantizar la seguridad operativa antes de la implementación física.
En la literatura reciente y en proyectos de ingeniería contemporáneos, el GRAFCET ha demostrado ser altamente escalable, se aplica desde automatismos discretos simples hasta Sistemas de Fabricación Flexible (FMS) de gran complejidad.
Por ejemplo, diversos autores han documentado su uso en la orquestación de células robóticas de ensamblaje, donde el GRAFCET permite gestionar las tolerancias a fallos y la sincronización entre brazos manipuladores y cintas transportadoras Reyes Cortés (2011).
Se usan también en la industria de procesos continuos y de empaquetado (packaging), metodologías basadas en GRAFCET (a menudo combinadas con la Guía GEMMA) se utilizan sistemáticamente para definir los modos de marcha, parada y emergencias, asegurando una reanudación segura de la producción tras una parada imprevista David & Alla (1992).

3.9. Validación Mediante Gemelos Digitales

La simulación de estas lógicas de control se realiza cada vez con mayor frecuencia mediante el uso de gemelos digitales.
Herramientas de simulación como PLCSIM (para TIA Portal) combinadas con entornos de modelado robótico permiten validar la lógica del PLC y la Guía GEMMA de forma virtual.
Según (Duka 2014), esta etapa de validación reduce los errores de programación en un 80 % antes de que la máquina física sea construida.

3.10. Puente de Wheatstone y Ley de Ohm

En el ámbito de la instrumentación industrial y el pesaje electrónico de precisión, la conversión de magnitudes físicas en señales eléctricas se fundamenta en arquitecturas de circuitos capaces de medir variaciones microscópicas de resistencia.
El puente de Wheatstone es una configuración de cuatro resistencias dispuestas en forma de rombo, excitadas por una fuente de tensión continua. En su aplicación para celdas de carga, una o varias de estas resistencias son galgas extensiométricas, cuyo valor óhmico fluctúa físicamente al sufrir una deformación mecánica Fraile Mora (2012).
En estado de reposo, el puente se encuentra balanceado y la diferencia de potencial entre sus ramas centrales es nula.
Al aplicar una fuerza de compresión o tracción, el desequilibrio resistivo genera un voltaje diferencial milimétrico que es proporcional a la masa del envase.
El correcto funcionamiento de este circuito de adquisición y la aparición de posibles perturbaciones en la lectura están intrínsecamente ligados a las leyes del electromagnetismo.
Por un lado, la resistencia inherente de los conductores se rige por la ley de Pouillet, que dictamina que la resistencia de un cable es directamente proporcional a su longitud e inversamente proporcional a su sección transversal ($R = \rho \frac{L}{A}$).
Alterar las dimensiones de los cables trenzados de la celda de carga introduce variaciones resistivas asimétricas.
Por otro lado, la Ley de Ohm establece que la caída de tensión en cualquier elemento de un circuito es igual al producto de la corriente que lo atraviesa por su resistencia ($V = I \cdot R$) Fraile Mora (2012).
En sistemas de adquisición digital de alta resolución —como los conversores ADC de 24 bits—, estas minúsculas resistencias parásitas provocan caídas de tensión que el circuito integrado amplifica por error, introduciendo un ruido significativo que arruina la precisión del lazo de control Bolton (2013).

3.11. El Transistor MOSFET

El gobierno de actuadores mecánicos de corriente continua exige dispositivos de estado sólido que eliminen el arco eléctrico, el desgaste y las limitaciones de frecuencia de conmutación asociadas a los relés electromecánicos.
El MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) es un semiconductor de potencia controlado íntegramente por voltaje Sedra & Smith (2014).
Los MOSFET de canal N diseñados para niveles lógicos permiten regular el paso de altas corrientes entre sus terminales de Drenador y Surtidor mediante la inyección de una tensión ínfima en su terminal de Puerta.

Al aplicar la señal de control proveniente del microcontrolador, se induce un canal de conducción electromagnético que presenta una resistencia interna sumamente baja en estado de saturación ($R_{DS(on)}$).
Esta característica crítica minimiza las caídas de tensión internas y facilita el uso de Modulación por Ancho de Pulsos (PWM) a altas frecuencias, posibilitando una regulación fluida y gradual del motor de la bomba dosificadora Bolton (2013).

3.12. Ley de Watt y Factor de Derating (Reducción)

El dimensionamiento de los subsistemas de potencia exige una evaluación térmica y eléctrica rigurosa.
La Ley de Watt rige el análisis de la potencia eléctrica, dictando que la potencia disipada o consumida por un dispositivo es equivalente al producto de la diferencia de potencial por la intensidad de corriente que circula a través de él ($P = V \cdot I$) Sedra & Smith (2014).
Este principio permite estimar la temperatura de unión que alcanzará el MOSFET y prever la necesidad de disipadores térmicos pasivos.
De manera complementaria a los cálculos de potencia, la ingeniería de diseño aplica el Derating o factor de reducción.
Este criterio preventivo consiste en seleccionar componentes electrónicos cuyas capacidades máximas de operación sobrepasen ampliamente las condiciones nominales de la placa Sedra & Smith (2014). Por ejemplo, al implementar un diodo de rueda libre (flyback) para absorber las corrientes inductivas de los motores, el “derating” exige que el componente sea capaz de bloquear una tensión inversa entre dos y tres veces superior al voltaje de trabajo.
Operar con el semiconductor utilizando este margen de seguridad inhibe la probabilidad de fatiga prematura y protege el microcontrolador de picos transitorios y de las fuerzas contraelectromotrices generadas durante la parada abrupta de la bomba.

3.13. El Proceso Integrador

Para establecer un lazo de control funcional, resulta imperativo modelar la dinámica de la planta analizando su respuesta temporal.
Un proceso integrador (o sistema sin autoregulación) es aquel en la que la variable de salida representa la acumulación progresiva de la variable de entrada Ogata (2010).
Frente a un estímulo en escalón continuo, el sistema integrador no tiende a un estado de equilibrio estacionario, sino que su salida aumenta linealmente generando una respuesta en rampa. En consecuencia, estos procesos carecen de la constante de tiempo tradicional τ.
El sistema de dosificación gravimétrico es un ejemplo paradigmático de esta dinámica: si la bomba peristáltica mantiene un caudal constante, la masa acumulada sobre la báscula crecerá indefinidamente hasta el desbordamiento, requiere de un agente externo que interrumpa el flujo para estabilizarse.

3.14. Control PID (Proporcional-Integral-Derivativo)

La consecución de un llenado libre de fluctuaciones o sobreimpulsos se orquesta a través de un controlador PID.

Este algoritmo de retroalimentación calcula el margen de error entre la consigna (peso objetivo del envase) y la variable de proceso (lectura actual de la celda de carga) basándose en tres comportamientos analíticos Bolton (2013). El término Proporcional reacciona a la magnitud del error instantáneo, disminuyendo el caudal de la bomba a medida que el recipiente se llena.
El término Integral evalúa el histórico del error acumulado en el tiempo, asegurando que se alcance exactamente la consigna deseada y compensando variaciones físicas como la viscosidad de la crema Ogata (2010).
Por su parte, el término Derivativo examina la velocidad a la que cambia el error, actuando como un freno predictivo que atenúa el empuje final y evita que el sistema sobrepase el peso establecido a causa de la inercia del líquido en suspensión.

3.15. Sintonización de Ziegler-Nichols en Procesos Integradores

El hallazgo analítico de las ganancias ideales del PID exige modelos matemáticos complejos y sujetos a incertidumbres por la reología de los fluidos. La alternativa industrial radica en la sintonización empírica de Ziegler-Nichols Ogata (2010).
En sistemas autoregulados, esta técnica evalúa la curva de reacción en lazo abierto. No obstante, dado que la planta es un proceso integrador (incapaz de estabilizarse autónomamente), el método clásico debe modificarse empleando el modelo integrador con retardo.
Esta variante se fundamenta en la obtención de la pendiente máxima de la respuesta (la ganancia intrínseca de integración, K) y del tiempo muerto (L), que representa la latencia mecánica entre el accionamiento del actuador y el primer impacto de la crema sobre la báscula.
Al cruzar estos datos físicos con las ecuaciones adaptadas de Ziegler-Nichols para procesos integradores (tales como $K_p = \frac{0{,}9}{K \cdot L}$ o un tiempo de integración $T_i = 3{,}33 \cdot L$), se configuran las ganancias P, I, y D Ogata (2010).
Este ajuste empírico garantiza un compromiso estructural entre una dosificación agresiva y un asentamiento sin sobreoscilaciones.

3.16. Protocolo de comunicación MQTT en entornos industriales IoT

MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación en tiempo real alineado con las especificaciones del IoT (Internet de las cosas) y la industria 4.0, Banks et al. (2019).
A diferencia de la arquitectura tradicional cliente-servidor, donde el cliente debe solicitar la información, MQTT utiliza un modelo de publicación/suscripción. Este modelo desacopla temporal y espacialmente a los dispositivos que generan la información de aquellos los que la consumen Stanford-Clark & Nipper (1999).
El uso de estos sistemas están en auge debido a la ligereza de los paquetes (2 bytes) Stanford-Clark & Nipper (1999). Al requerir una capacidad de procesamiento mínima para la serialización de los datos, el microcontrolador puede dedicar la mayor parte de sus ciclos de reloj a las tareas críticas de control en tiempo real, delegando la transmisión asíncrona de la telemetría al broker sin comprometer la estabilidad del lazo de control Serpanos & Wolf (2017).

Capítulo 4: Control de llenado

4.1. Montaje

Para realizar el control del llenado de los botes con cremas se ha desarrollado un soporte para los componentes eléctricos.
Los componentes son:

- Interruptor para encender y apagar el sistema.
- Placa de ESP32 para el control en Matlab
- Báscula con su PCB.
- PCB de PWM para controlar el encendido/apagado
- Bomba peristáltica
- Cable de alimentación

El material del soporte es MDF (Medium Density Fierboard) de 3mm de grosor.
Cada pieza tiene una forma dentada en los laterales para encajar las piezas y asegurarse de que quedan bien sujetas entre sí, como se muestra en la figura 1. Las piezas se desarrollan en el software de SolidWorks por separado y luego se unen todas mediante la opción de ensamblaje para comprobar que encajan correctamente.

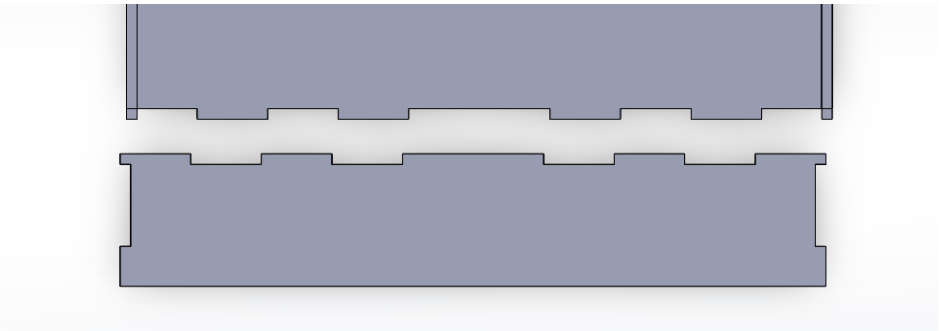


Figura 1: Dentado de las piezas

4.1.1. Material

Elementos necesarios para el montaje práctico:

- **Bomba peristáltica:**

La bomba peristáltica que se muestra en las imágenes 2,3,4 consiste en un motor giratorio y un tubo flexible por el que viaja el fluido.
Al girar el motor, se genera un vacío en distintas partes del tubo, lo que hace que el fluido se desplace. Este tipo de bomba es muy utilizada en la industria cuando se requieren condiciones de limpieza estrictas, ya que el producto no entra en contacto directo en ningún momento con la bomba, evitando así su contaminación por el entorno externo.



Figura 2: Bomba peristáltica parte posterior



Figura 3: Bomba peristáltica parte frontal



Figura 4: Bomba peristáltica parte lateral

- **Báscula:**

El componente HX711 que se muestra en la figura 5 es un conversor analógico-digital que transforma la fuerza, en este caso el peso, en una señal eléctrica mediante galgas extensiométricas cuya resistencia eléctrica varía en función de la deformación producida por el peso.
La báscula se comunica con el microcontrolador, ESP32, mediante dos pines: un pin de reloj SCK y otro de datos DT.

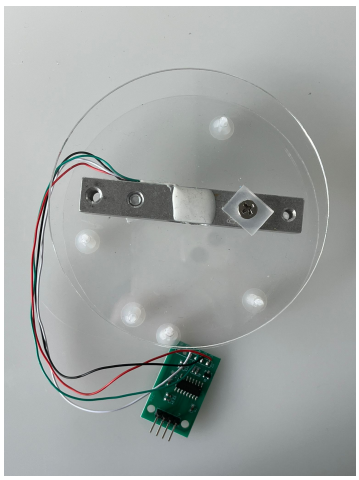


Figura 5: Báscula

- **Switch:**

Se conecta un interruptor switch, figuras 6 y 7 , para poder encender y apagar el sistema de manera sencilla y poder realizar las pruebas.



Figura 6: Switch de frente



Figura 7: Switch de lateral

- **Panel:**

Se crea un soporte de MDF (Medium Density Fierboard), como se aprecia en la figura 8, utilizando la cortadora láser. Este panel permite el montaje de los componentes eléctricos y mejora la visualización del funcionamiento del proyecto.



Figura 8: MDF

- **Placa de desarrollo:**

En un primer momento se escogió un Arduino para hacer el control con el software MATLAB, finalmente se prefirió la placa ESP32, figura 9, por su mayor potencia de

procesamiento y conectividad, ya que contiene un módulo de conexión de WiFi y Bluetooth para enviar y recibir datos sin necesidad de conectarlos a la placa mediante cables.

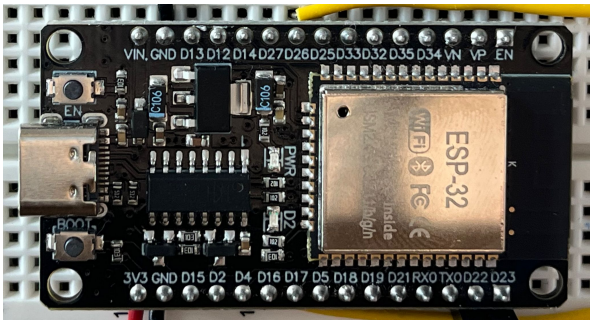


Figura 9: ESP32

4.1.2. Soporte para los componentes electrónicos

- **Pieza Vertical:** Las placas electrónicas se pegan a la parte posterior de esta pieza, previamente se comprueba la ubicación de cada una de ellas y el espacio necesario que requieren. Se realizan los cortes necesarios para colocar la bomba y el interruptor que sobresalen de la placa. También, se realiza una pequeña apertura en la parte interior de la pieza para pasar la placa de la báscula que está previamente soldada al sensor galga extensiométrica, como se aprecia en la imagen 10.
  La crema se desplaza desde el tanque en la parte posterior hasta el centro de la superficie de la báscula pasando por la bomba a través de un tubo de goma. Este tubo necesita orificios para realizar este recorrido.
  La crema se dirige desde el tanque situado en la parte posterior hasta el centro de la superficie de la báscula, pasando por la bomba a través de un tubo de goma. Para ello, el tubo se conduce a través de la estructura mediante orificios de paso. Los detalles técnicos se encuentran en el apartado de anexo A.1

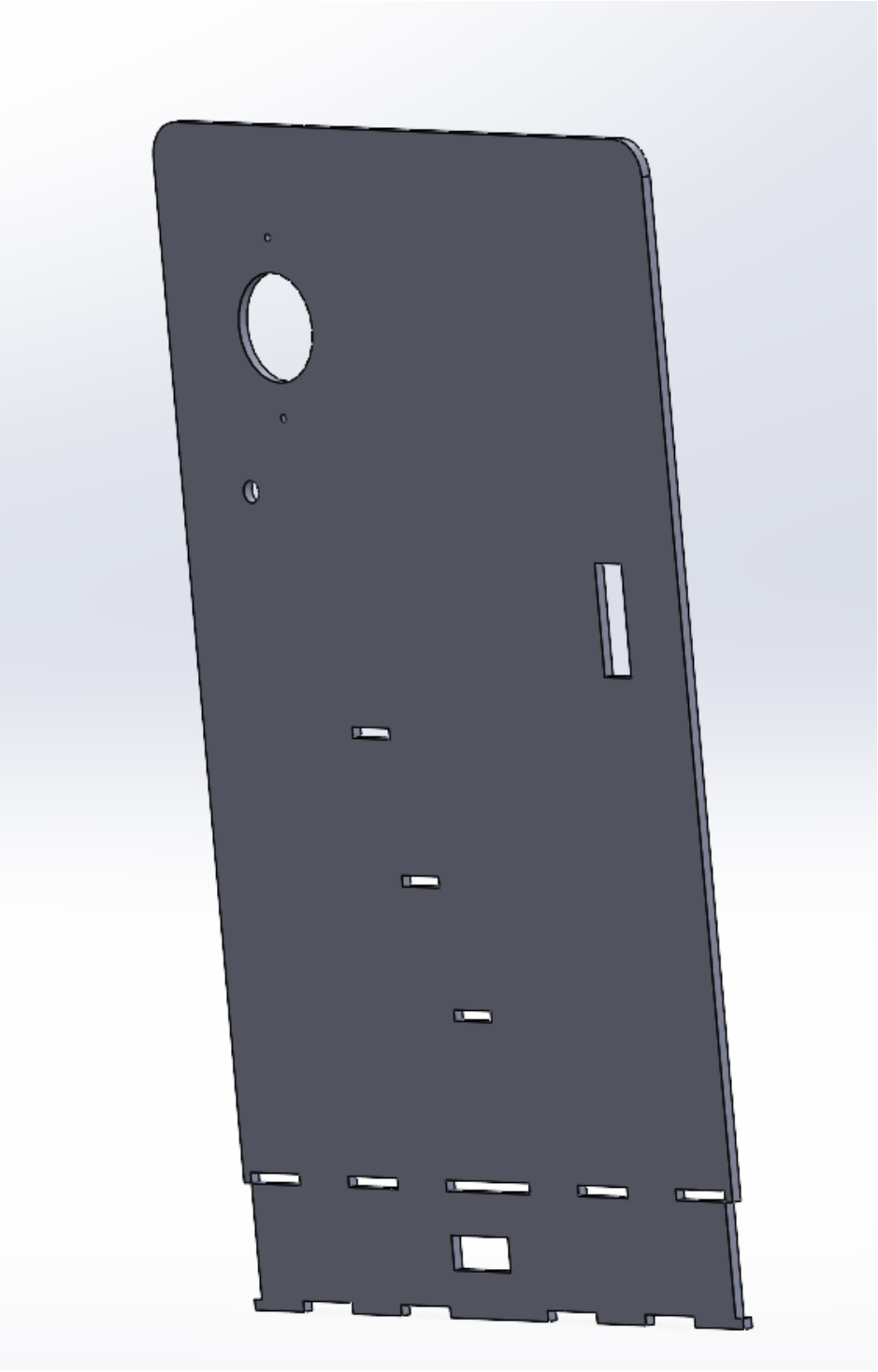


Figura 10: Pieza vertical en 3D

- **Soportes para el tubo:** El tubo de goma es conducido al centro de la báscula mediante unos soportes que sobresalen de la pieza vertical.
  Estos soportes tienen tres longitudes distintas para evitar que el tubo se pince e impida la circulación de la crema.
  Los detalles técnicos se encuentran en los apartados de los anexos A.8, A.7, A.6

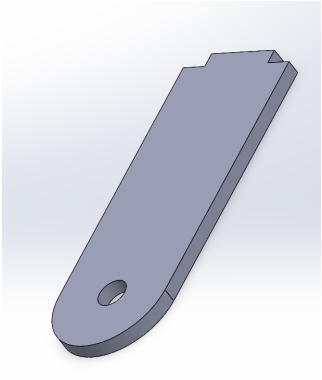


Figura 11: Pieza de soporte en 3D

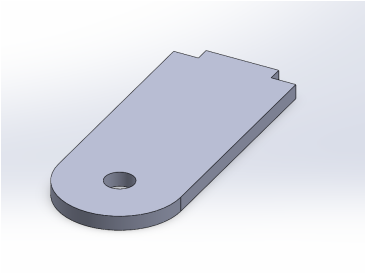


Figura 12: Pieza corto de soporte en 3D

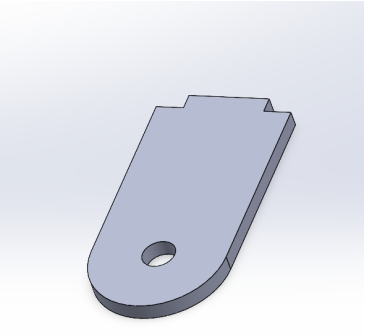


Figura 13: Pieza más corto de soporte en 3D

- **Tapa:** Esta pieza cierra en forma de prisma la mitad anterior de todo el ensamblaje. Tiene un círculo en la zona central para poder dejar libre la zona de pesado. Esta estructura permite futuras pruebas con una cinta transportadora y protege la báscula.
  Los detalles técnicos se encuentran en el apartado de anexo A.9

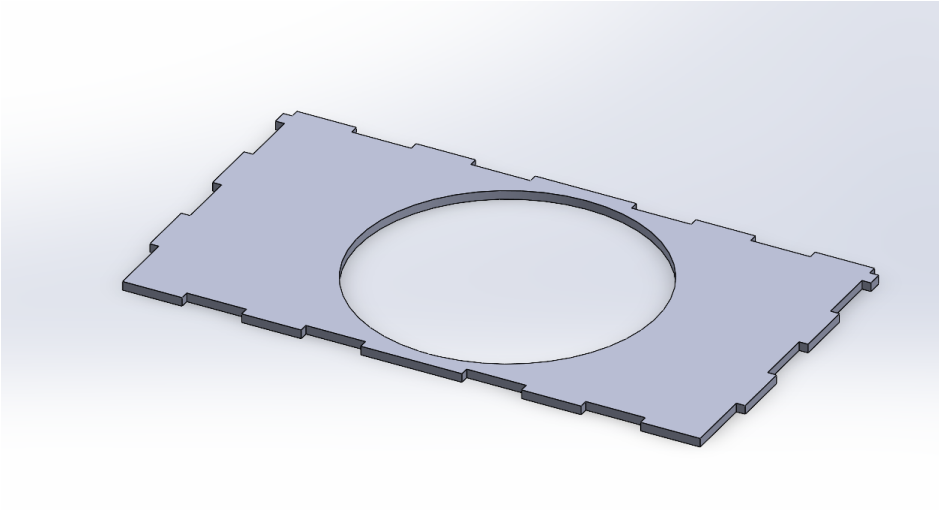


Figura 14: Pieza tapa de la báscula en 3D

- **Base y paredes laterales:** Las paredes laterales y la base forman una caja que asegura el soporte de todas las piezas, de forma que queden bien fijadas y permite la colocación del tanque de crema de manera estable.
  Los detalles técnicos de la base se encuentran en el apartado de anexo A.2 Los detalles técnicos de las paredes laterales se encuentran en los apartados de los anexos A.4, A.5, A.3

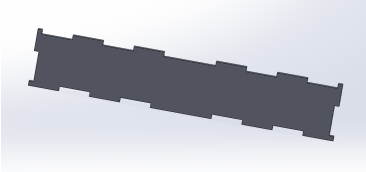


Figura 15: Pieza lateral 1 en 3D

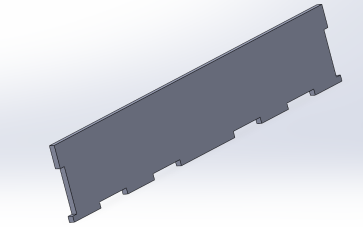


Figura 16: Pieza lateral 2 en 3D



Figura 17: Pieza lateral grande en 3D

- **Cortadora láser:** En la primera prueba (figura 18), se detectaron errores que impedían encajar la pieza vertical y las laterales más largas. También se observó que situar el interruptor debajo del motor dificultaba el acceso debido a la cercanía de los tubos de goma por los que se desplazaba la crema, por lo que se optó por cambiar el interruptor al lado opuesto.
  Asimismo, se cambió la posición de la goma de la parte posterior para evitar giros que pudiesen obstruir el cable.
  Para el segundo ensamblaje, figura 19, se verificó que las piezas no chocaban entre sí mediante la herramienta de detección de interferencias dentro de la pestaña de cálculo en el ensamblaje.

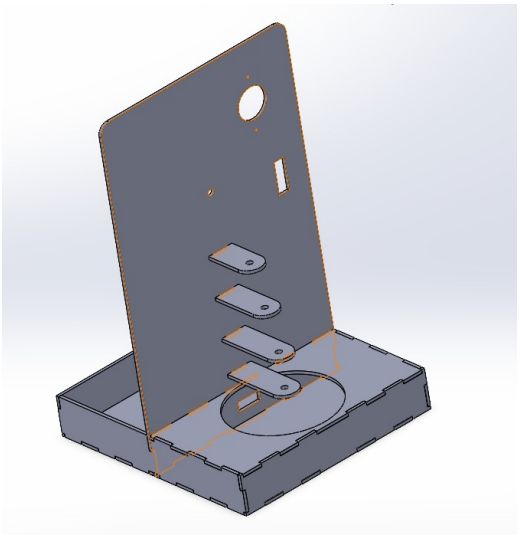


Figura 18: Primer ensamblaje

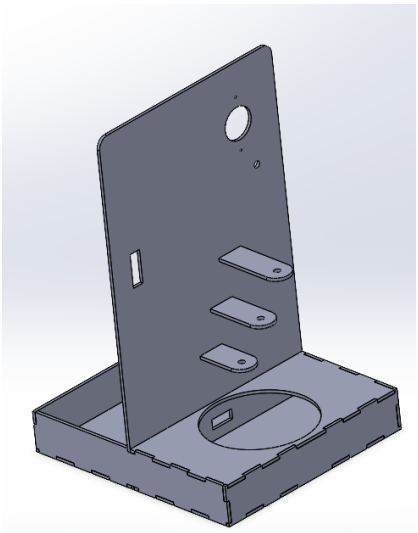


Figura 19: Segundo ensamblaje

4.1.3. Montaje físico

Los cortes del segundo modelaje fueron correctos, pero durante la colocación de los componentes electrónicos, los cables de la placa de la báscula se soltaban, por lo que hubo que soldarlos de nuevo.



Figura 20: Montaje físico parte frontal

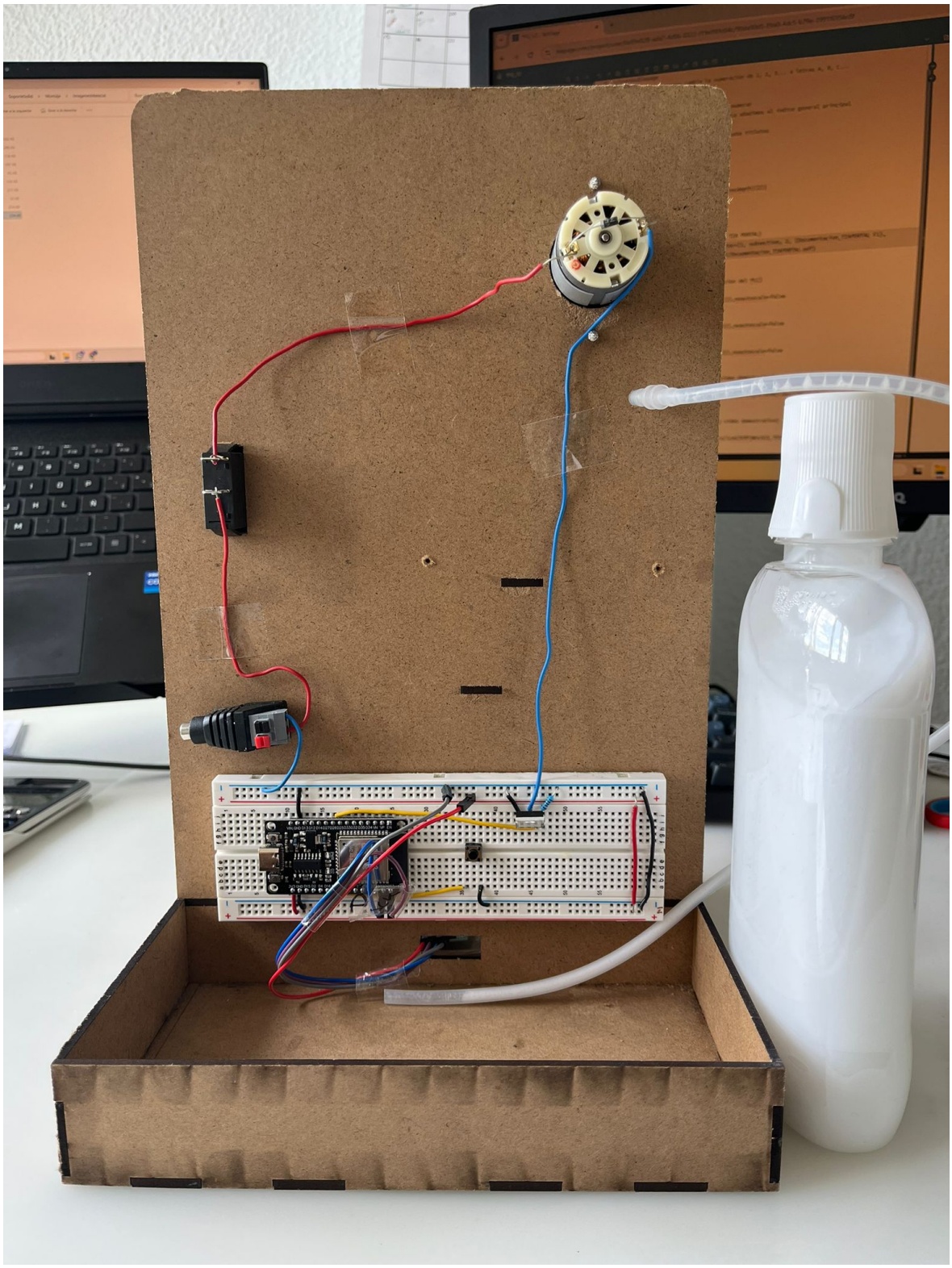


Figura 21: Montaje físico parte posterior

4.1.4. Pruebas en el montaje

Al realizar el montaje hay que tener en cuenta algunos factores relacionados con la báscula:
Al pasar los cables por el hueco de la estructura, estas uniones tienden a soltarse, lo que obliga a volver a soldarlas manualmente, una tarea compleja debido a la fragilidad de los conductores. Como consecuencia de estas reparaciones, es necesario recortar el cable en cada intervención, lo que provoca que los conductores queden con longitudes desiguales. Esta variación en las longitudes es la que origina las fluctuaciones en las medidas de la bomba impidiendo realizar el control. Este fenómeno ocurre en las células de carga porque son un conjunto de resistencias que varían cuando se deforman físicamente (galgas extensiométricas), las resistencias están en configuración de puente de Wheatstone, por lo que al acortar los cables con longitudes desiguales se añade una resistencia parásita distinta en cada cable. El módulo HX711 es un amplificador analógico-digital de 24 bits que es capaz de diferenciar voltajes en rangos de microvoltios por lo que estas resistencias parásitas afectan a la lectura.
Este fenómeno físico se evidencia a través de la ley de Pouillet (ley de resistencia eléctrica):

$$R = \rho \frac{L}{A}$$

Donde:

- R**:** es la resistencia
- ρ**:** es la resistividad del material (cable de cobre)
- L**:** es la longitud del cable
- A**:** es el área de la sección transversal

Por tanto, si se disminuye la longitud del cable (L) la resistencia de este también disminuye.

Otro factor que debe considerarse es la sensibilidad a las interferencias electromagnéticas del módulo HX711, si se cortan los cables originales de fábrica se pierde el apantallamiento debido al trenzado interno de los cables. Un cable trenzado de manera asimétrica puede actuar como una antena de radio, lo que afecta a la amplificación del módulo.

Una opción sería hacer ingeniería inversa, incorporando una báscula de precisión que está funcionando. Sin embargo, al abrirla se ve un gran punto negro, es el microcontrolador que está implementado en la propia placa COB (Chip-on-board), por lo que es imposible saber que tecnología se está usando.
Algunas placas de este estilo tienen unos terminales E+, E-, S+, S- igual que en la placa HX711, por lo que se podría conectar al módulo de conversión analógico-digital para comprobar el funcionamiento del sensor. Pero al soldar ambos convertidores de analógico y digital al mismo sensor no se inhabilita uno de ellos, sino que ambos reciben alimentación a través de los pines E+ y E- por lo que la señal que llega al módulo es

casi imperceptible para el HX711.

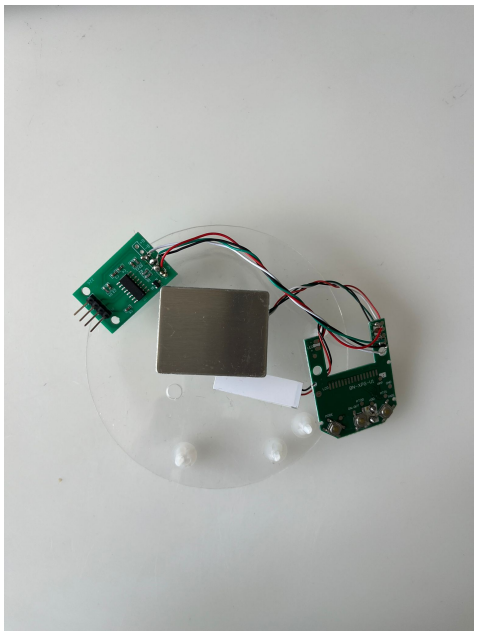


Figura 22: Prueba de la báscula de precisión

Finalmente, tras investigar los distintos componentes para las células de carga se utilizaron varios materiales para probarlos:

- Se seleccionó una báscula de un kilogramo porque al acortar el rango de medición y tener los cables soldados de fábrica con longitudes iguales se podría evitar los errores.
- Se analizó un sensor con capacidad de medición máxima de cien gramos y una precisión del orden de los centígramos, (imagen 24). Este sensor incorpora el conversor analógico-digital NAU7802 (figura 23), que podría sustituir al HX711. El conversor ofrece alta precisión y bajo consumo, y está diseñado específicamente para la detección de variaciones de peso muy pequeñas.

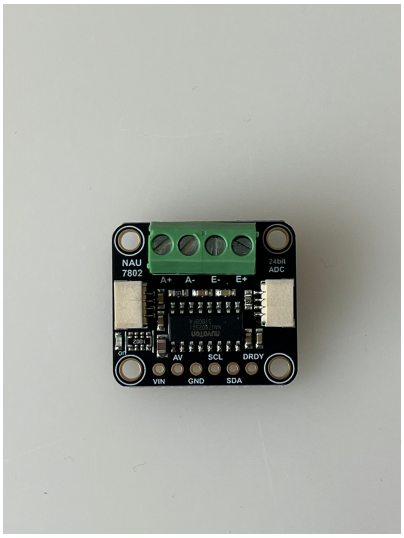


Figura 23: NAU7802 24-bit ADC

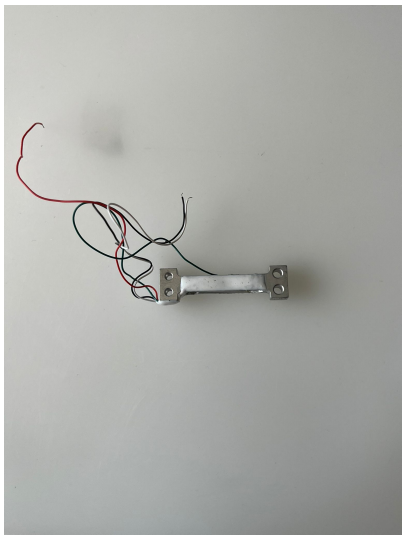


Figura 24: Galga extensiométrica de pesaje máximo de 100g

Implementación de la báscula

En la primera prueba, la más sencilla, se utilizó la báscula ya montada y soldada, con un rango de medición de un kilogramo. La medida se mantuvo estable en el rango de los gramos, por lo que fue posible realizar el control del llenado.
Si no hubiese funcionado correctamente, habría sido necesario montar y soldar el módulo NAU7802 junto con la galga de precisión.

4.2. Componentes eléctricos y circuito impreso

Para el sistema, es necesario controlar una báscula y una bomba mediante un microcontrolador ESP32.
Una vez comprobado el funcionamiento del sistema con una placa de protoboard se procede al desarrollo de la PCB.

4.2.1. Circuito de potencia

El circuito de potencia se utiliza en la etapa de suministro y gestión de la alimentación de la bomba peristáltica.
En primer lugar, se calcula la corriente nominal de la bomba en régimen permanente aplicando la Ley de Watt en corriente continua:

$$I = \frac{P}{V}$$

Para realizar este cálculo se toman los valores nominales de voltaje (12 V) y potencia (5 W) indicados en la serigrafía del motor (Figura 25). Este valor es fundamental para calibrar el resto de los componentes del sistema de actuación.

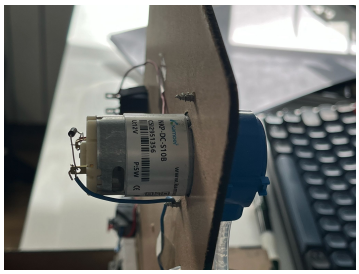


Figura 25: Motor de bomba peristáltica

$$I = \frac{P}{V} = \frac{5\ \text{W}}{12\ \text{V}} = 0,416\ \text{A} = 416\ \text{mA}$$

La bomba se alimenta a 12 V y el ESP32 funciona a 3,3 V, por lo que no pueden conectarse amobos directamente entre sí, ya que se dañaría el microcontrolador. Por esta razón se exploran distintas opciones:

La primera opción consiste en el uso de un relé electromagnético (figura 26), pero presenta varios inconvenientes:

- Solo permite un control On/Off (todo o nada) debido a su baja frecuencia máxima de conmutación (típicamente inferior a 20 Hz). Esto impide la modulación por ancho de pulsos (PWM) necesaria para variar la velocidad de la bomba e implementar un control PID.
- Contiene partes mecánicas móviles, lo que genera desgaste por arco eléctrico y reduce drásticamente su vida útil ante conmutaciones repetitivas.



Figura 26: Relé electromecánico de 12V

La segunda opción considerada fue un módulo preensamblado de doble MOSFET (D4181/LR7843), figura 27. Este módulo ofrece una solución de estado sólido que elimina las partes móviles y basa el control en la acción de los semiconductores.
Aunque es una buena solución, dado que el objetivo final es desarrollar un circuito impreso, implementar una placa preensamblada no sería la solución óptima.

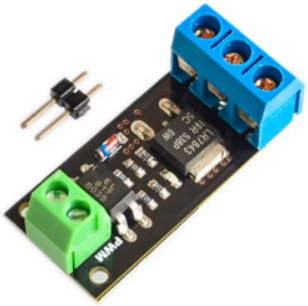


Figura 27: Módulo de control de potencia MOSFET

La última opción, y la que finalmente fue utilizada, es un MOSFET de canal N, el modelo IRLZ44N (Anexo F). Este componente pertenece a la serie IRL, diseñada para ser activada mediante niveles lógicos de baja tensión, como es el caso del microcontrolador. Mientras que un MOSFET estándar requiere habitualmente una tensión de puerta $V_{GS} \geq 10$ V para alcanzar su mínima resistencia interna ($R_{DS(on)}$), el IRLZ44N garantiza una resistencia de conducción muy baja (aprox. $0,025\ \Omega$) con una excitación de apenas $3,3$ V. Esto permite que el ESP32 lo conmute directamente sin necesidad de integrar un circuito *gate driver* adicional, optimizando el diseño de la PCB, reduciendo la disipación térmica y permitiendo un control suave de la bomba, que es factor fundamental para la precisión del pesaje.

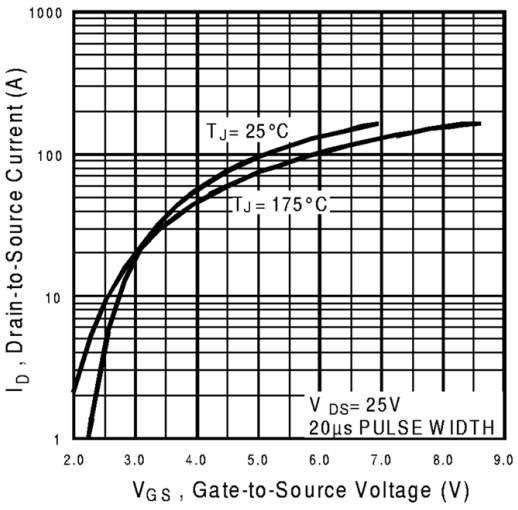


Figura 28: Curva de Características de Salida del MOSFET

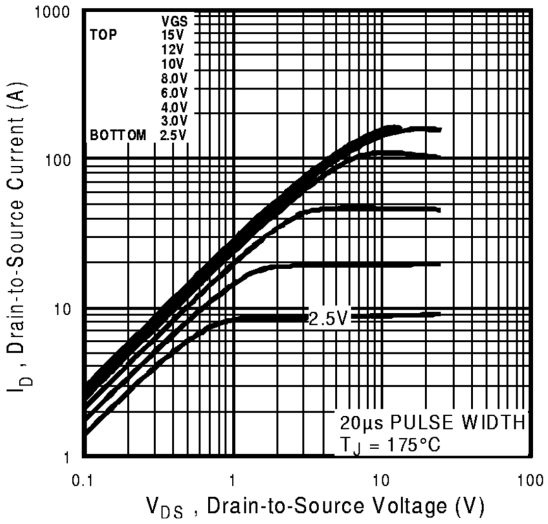


Figura 29: Curva de Transferencia del MOSFET

Para garantizar el correcto funcionamiento de este semiconductor en el diseño, se procede a su justificación matemática:
Primero, se verifica la condición de saturación. Es necesario comprobar que el voltaje proporcionado por el microcontrolador supera la tensión umbral mínima de la puerta ($V_{GS(th)}$) requerida para que el transistor permita el flujo de electrones entre Surtidor (S) y Drenador (D). Consultando el Anexo F (Página 2), el valor máximo de fabricación para $V_{GS(th)}$ es de $2,0$ V. Como los pines del ESP32 operan a $3,3$ V, se cumple la inecuación:

$$V_{salida_ESP32} > V_{GS(th)_max} \implies 3,3 \text{ V} > 2,0 \text{ V}$$

Esto garantiza que el MOSFET alcance la región de saturación y conduzca la corriente, evitando que opere en la región lineal, donde actuaría como una resistencia, y disiparía el calor excesivo hasta su destrucción.

A continuación, se evalúa la necesidad de incorporar refrigeración adicional (disipador térmico) calculando la potencia disipada en estado de conducción mediante la Ley de Joule:

$$P_D = I^2 \cdot R_{DS(on)}$$

Donde:

- I (Corriente nominal de la bomba) $= 0,416$ A
- $R_{DS(on)}$ (Resistencia interna del canal) $\approx 0,050\ \Omega$

El valor de $R_{DS(on)}$ se ha estimado de forma conservadora a partir de las especificaciones del fabricante (Anexo F, Página 2), en las que se indican las siguientes lecturas garantizadas:

- $V_{GS} = 10$ V $\rightarrow 0,022\ \Omega$
- $V_{GS} = 5,0$ V $\rightarrow 0,025\ \Omega$
- $V_{GS} = 4,0$ V $\rightarrow 0,035\ \Omega$

Se observa que ante una caída de 1 V (de 5V a 4V), la resistencia se incrementa en $0,010\ \Omega$. Dado que el microcontrolador suministra $3,3$ V (situándose $0,7$ V por debajo del último dato tabulado), y sabiendo que la curva de resistencia no es lineal cerca de la región de corte, se ha adoptado una extrapolación pesimista añadiendo $0,015\ \Omega$ adicionales. El valor de cálculo asumido es de $0,050\ \Omega$, situando el diseño en el peor caso operativo posible.

Resolviendo la ecuación de potencia:

$$P_D = (0,416\ \text{A})^2 \cdot 0,050\ \Omega = 0,0086\ \text{W}$$

En estado de conducción continuo, el MOSFET disipa únicamente $8,6$ mW de potencia térmica.

Consultando las características térmicas del encapsulado TO-220, la resistencia térmica unión-ambiente es de $R_{\theta JA} = 62\ ^\circ\text{C/W}$. Esto indica que la temperatura interna de la unión semiconductora (T_J) se elevará 62 grados por cada vatio disipado, respecto a la temperatura ambiente.
Sabiendo que el límite máximo destructivo del silicio es de 175 °C, se calcula la temperatura final esperada operando al aire libre (sin radiador) a una temperatura ambiente estimada de 25 °C:

$$T_{final} = T_{ambiente} + (P_D \cdot R_{\theta JA})$$

$$T_{final} = 25\ ^\circ\text{C} + (0,0086\ \text{W} \cdot 62\ ^\circ\text{C/W}) = 25 + 0,53 = 25,53\ ^\circ\text{C}$$

Este cálculo demuestra matemáticamente que el componente operará a tan solo medio grado por encima de la temperatura ambiente. Al mantenerse frío, se justifica técnica y económicamente la omisión de un disipador térmico externo en la placa.

4.2.2. Resistencia de polarización (Pull-Down)

El circuito del MOSFET incorpora una resistencia de "pull-down" conectada entre el terminal de Puerta (Gate) y Tierra (GND) para prevenir estados lógicos indeterminados. Si esta línea de control quedase sin conectar (estado de alta impedancia o pin flotante durante un reinicio del sistema), la capacitancia parásita de la puerta acumularía cargas electrostáticas e interferencias electromagnéticas del entorno. Esto produciría una elevación del voltaje de forma incontrolada, provocando puestas en marcha aleatorias e indeseadas del actuador.

Para dimensionar este componente, se debe establecer un límite superior que garantice el apagado del MOSFET. La función de la resistencia es evacuar eficazmente a tierra cualquier carga parásita. Si la resistencia es demasiado alta, el ruido eléctrico y las corrientes de fuga podrían elevar la tensión en la puerta por encima del umbral de encendido.

Límite superior: Garantía de apagado del MOSFET
Datos operativos:

- Corriente de fuga máxima en la puerta: $I_{GSS} = 100$ nA $(0,0000001$ A), según especificaciones (Anexo F, Pág. 2).

- Tensión umbral mínima de encendido en el peor caso de fabricación: $V_{GS(th)_min} = 1,0$ V (Anexo F, Pág. 2). Si la caída de tensión en la resistencia originada por las fugas supera $1,0$ V, el transistor abandonará la región de corte.

$$R_{max} = \frac{V_{GS(th)_min}}{I_{GSS}} = \frac{1,0 \text{ V}}{0,0000001 \text{ A}} = 10{,}000{,}000\ \Omega = 10\ \text{M}\Omega$$

La resistencia no debe superar los 10 MΩ.

Selección final

Por debajo de este límite máximo de 10 MΩ, se requiere adoptar un valor de compromiso. Resistencias excesivamente bajas generarían un consumo energético ineficiente durante los estados de activación, ya que derivarían corriente del pin del microcontrolador directamente a masa. Por otro lado, valores muy cercanos al límite superior reducirían la inmunidad del circuito ante ruidos transitorios de alta frecuencia y harían la descarga de la puerta más lenta.

Se ha seleccionado el valor estandarizado de 10 kΩ (10,000 Ω). Este valor drena una corriente insignificante de $0,33$ mA ($3,3$ V/10,000 Ω) durante la emisión de la señal de control, garantizando a su vez un anclaje eléctrico robusto y rápido a 0 V durante los estados de reposo o alta impedancia del microcontrolador.

4.2.3. Protección inductiva (Diodo Flyback)

Al conmutar cargas inductivas como el bobinado de un motor, la interrupción abrupta de la corriente provoca que el campo magnético colapse de forma repentina. Este fenómeno induce un pico de alta tensión inversa (fuerza contraelectromotriz) que puede superar el límite de ruptura del transistor MOSFET y destruirlo.

Para suprimir este pico destructivo, se implementa un diodo de rueda libre o *flyback* (Anexo L) conectado en antiparalelo con los terminales del motor (Figura 30). Su función es proporcionar un camino cerrado de baja impedancia para que la corriente inductiva remanente recircule y se disipe térmicamente de forma segura.

Se ha seleccionado el modelo 1N4007, ya que su capacidad de bloqueo en inversa es de 1000 V. Esta magnitud excede ampliamente los 12 V nominales del bus de alimentación, superando las reglas de "derating" (factor de reducción) en diseño electrónico, que exigen que se dimensionen los semiconductores de bloqueo entre 2 y 3 veces por encima de la tensión de trabajo. Aunque un diodo 1N4001 (50 V) sería funcionalmente suficiente, el uso del 1N4007 no incrementa el coste del prototipo y maximiza el margen de seguridad de la PCB ante picos transitorios severos en la red de alimentación.

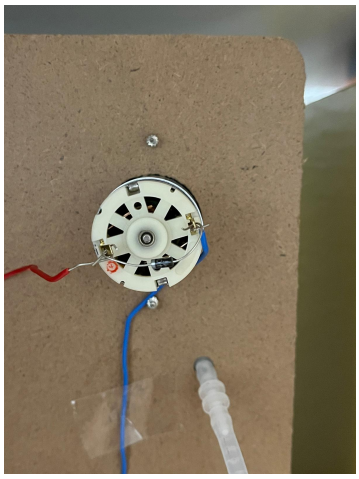


Figura 30: Implementación del diodo Flyback 1N4007

4.2.4. Sistema de adquisición de peso (Báscula)

La celda de carga proporciona la medición del peso a través de micro variaciones de tensión diferencial, las cuales deben ser digitalizadas para su procesamiento por el microcontrolador.
El elemento sensor presenta una capacidad nominal de 1 kg y una sensibilidad característica de $S = 1$ mV/V (Anexo H). Sabiendo que el voltaje de excitación suministrado por la circuitería analógica es de $V_{exc} = 3,3$ V, el voltaje máximo diferencial a fondo de escala se define como:

$$V_{max_out} = V_{exc} \cdot S = 3,3 \text{ V} \cdot 1 \text{ mV/V} = 3,3 \text{ mV}$$

Esto implica que, bajo una carga física de 1000 g, el puente de Wheatstone genera un diferencial de apenas $0,0033$ V. Extrapolando la resolución teórica por gramo:

$$Sensibilidad_unitaria = \frac{3,3 \text{ mV}}{1000 \text{ g}} = 3,3 \ \mu\text{V/g}$$

Como las variaciones tensionales mecánicas son del orden de los microvoltios, el convertidor analógico-digital (ADC) integrado nativamente en el ESP32 carece de la resolución y de la relación señal/ruido necesarias para leer estos datos de forma directa. Por consiguiente, se justifica el uso de la interfaz HX711, la que incorpora un Amplificador de Ganancia Programable (PGA) y un ADC de 24 bits específico para básculas.

A nivel de hardware, el módulo HX711 se comunica con el microcontrolador a través de dos hilos lógicos. Aunque la presencia de un pin de reloj y uno de los datos pueda sugerir el uso de un bus estandarizado como I2C o SPI, el convertidor emplea un protocolo síncrono propietario del fabricante. La transferencia se gestiona por *bit-banging* (control de software a nivel de pin) siguiendo la siguiente secuencia (Anexo K):

1. El pin de salida de datos (DT) es forzado a nivel bajo por el HX711 para indicar que una nueva conversión analógico-digital está completa y lista para ser extraída.

2. El ESP32 actúa como maestro generando exactamente 24 pulsos en la línea de reloj (SCK). En cada flanco, el módulo desplaza hacia fuera un bit correspondiente a la lectura ponderada del peso.

3. Una vez extraídos los datos, el ESP32 envía entre 1 y 3 pulsos de reloj adicionales. Esta secuencia final configura el canal de entrada y el factor de ganancia interna (128, 64 o 32) que el PGA va a utilizar en la siguiente conversión.

4.2.5. Alimentación

La bomba se alimenta a 12V directamente con un enchufe y el ESP32 se conecta al ordenador que le proporciona 5V.
Es importante a la hora de hacer el circuito que todas las tierras estén juntas, esto es fundamental para que todo el sistema tenga la misma referencia y funcione a la perfección.
Con este sistema de arquitectura de alimentación dual se evita que el ruido eléctrico que genera el motor viaje por los cables de la alimentación hasta el microcontrolador, haciendo innecesario añadir módulos de protección de corrientes ni reguladores.

4.2.6. PCB

Los circuitos impresos tienen caminos ("paths") de cobre por donde viajan los electrones, antes de realizar la placa, debemos calcular el grosor mínimo de la pista.
Estos cálculos se rigen por la norma internacional IPC-2221, que utiliza fórmulas para asegurar que el cobre no se caliente demasiado y se termine despegando de la placa.

1. Cálculo del área de la sección transversal

Se calcula el área necesaria del cobre (en milésimas cuadradas o sq mils) en función de la corriente y el calor permitido:

$$A = \left(\frac{I}{k \cdot \Delta T^b}\right)^{\frac{1}{c}}$$

- I: Corriente máxima del motor. $(1A)$
  Aunque este valor es de $416mA$ se harán los cálculos con $1A$ para dejar margen de seguridad.
- ΔT: Aumento máximo de temperatura permitido (10 °C)
- k, b, c = Constantes de la norma para pistas externas ($k = 0,048$, $b = 0,44$, $c = 0,725$).

$$A = \left(\frac{1}{0,048 \cdot 10^{0,44}}\right)^{\frac{1}{0,725}} \approx 15,22 \text{ sq mils}$$

2. Calculo del ancho de la pista (W)

Una vez tenemos el área, la dividimos por el grosor del cobre para obtener el ancho:

$$W = \frac{A}{t \cdot 1,378}$$

- t: Grosor del cobre $(1oz/ft^2)$ que es el estándar universal de los fabricantes de placas

- $1,378$: Factor de conversión para pasar de onzas a milis

$$W = \frac{15,22}{1 \cdot 1,378} \approx 11,04 \text{ mils}$$

3. Justificación del dimensionamiento final

El cálculo normativo establece que cualquier pista con un ancho inferior a 11,04 mils sufrirá daños estructurales por estrés térmico. Sin embargo, en el diseño de las etapas de potencia, es una buena práctica de ingeniería sobredimensionar el trazado para optimizar el rendimiento eléctrico.

Por este motivo, en el software de diseño CAD, las pistas son de 50 mils. Este sobredimensionamiento reduce drásticamente la resistencia parásita de la línea, minimizando las caídas de tensión y mejorando la disipación térmica pasiva del circuito.

Inicialmente se diseñó la PCB en el software de Proteus, sin embargo, si se quiere imprimir las placas con la empresa JLCPCB, es más cómodo usar el software de diseño de placas que han desarrollado ellos porque tiene todos los componentes de los que se puede solicitar la soldadura. Esto facilita la generación de las huellas como asegurarse el stock de los componentes.

El software permite la visualización en 3D del ensamblaje y PCB para comprobar que el montaje es correcto (31). Los esquemas eléctricos y los paths diseñados para la placa se encuentran en los anexos B

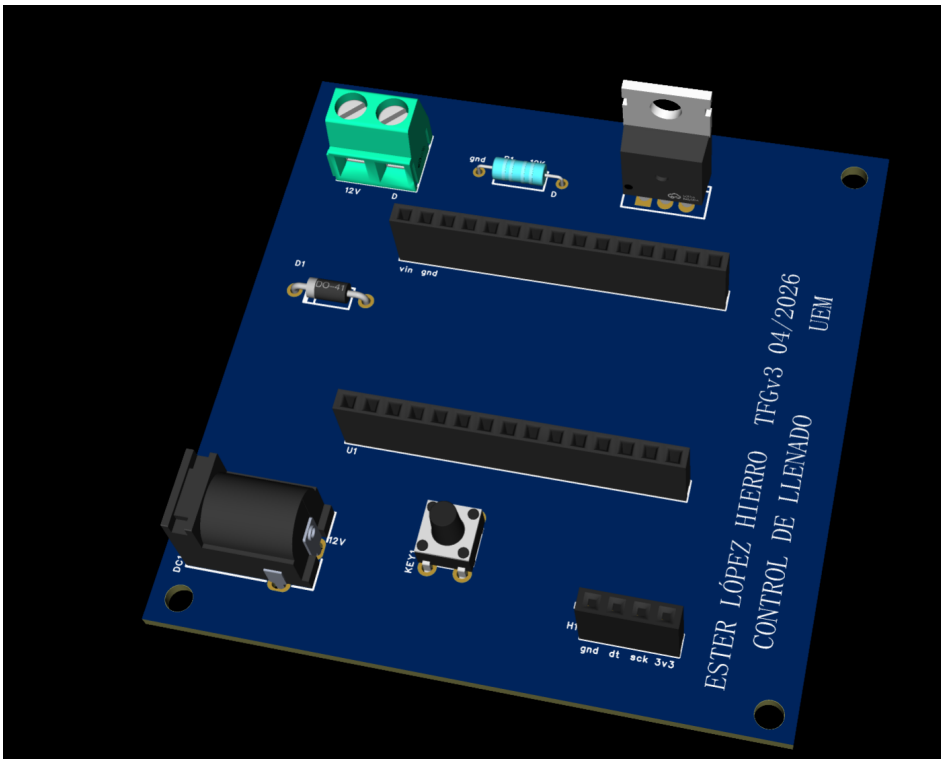


Figura 31: Visualización de PCB en 3D

4.3. Control programado mediante MATLAB

Se debe programar el funcionamiento del sistema. En este caso se ha realizado mediante el software de MATLAB para visualizar de manera clara las ondas que se producen durante el proceso de llenado y posteriormente analizar los resultados.
En un entorno real, a lo largo del tiempo los materiales se desgastan, este desgaste se puede predecir antes de que se manifieste mediante los gráficos y abastecerse de las piezas para reemplazarlas cuando sea necesario, ahorrando costes y tiempo de producción.

4.3.1. Comunicación ESP32 con MATLAB

Para la comunicación del microcontrolador con MATLAB se ha utilizado un paquete de *Add-Ons-Explorer >Simulink Support Package for Arduino Hardware* (32).

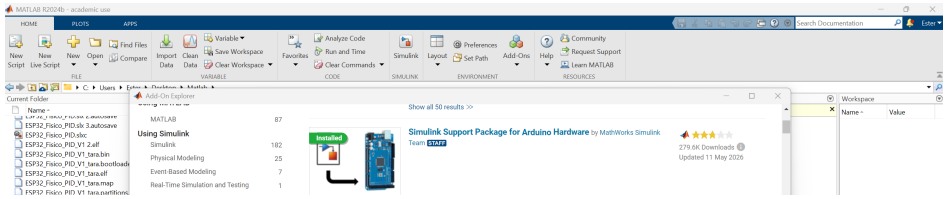


Figura 32: Paquete de Arduino

El paquete tiene compatibilidad con Arduino y muchos otros controladores como Raspberry Pi o en este caso ESP32.

4.3.2. Conexión con HX711

No se ha encontrado ningún paquete que permita la lectura de HX711 en la programación de Simulink. Por lo que hay que generar esta librería manualmente ya que simulink no es capaz de leer los datos a la velocidad que exige el protocolo interno de HX711.
Para generar el bloque y que permita leer el sensor de la báscula se debe generar un S-Function builder.

1. Crear un módulo independiente arrastrando el bloque *S-Function Builder* desde la librería de Simulink (33).

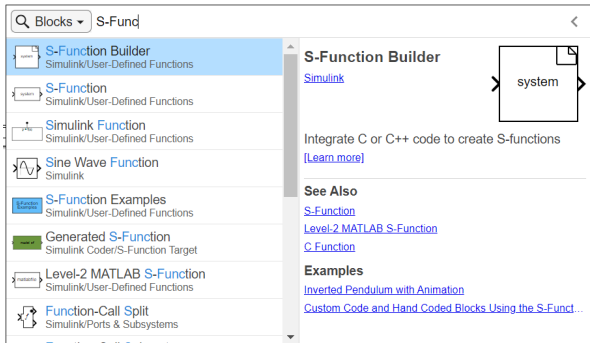


Figura 33: S-Function Builder

2. A continuación, es necesario cambiar el lenguaje de compilación de C a C++, porque el programa que se va a generar usa clases y C no lo compila correctamente, imagen 34.

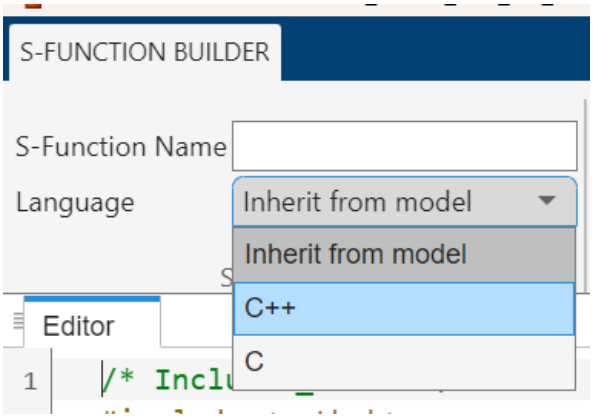


Figura 34: Compilador de C++

3. Generación del código, explicado en 4.3.2

4. Muestreo (Sample Time): El bloque se debe configurar con un muestreo de 0,1 segundos. Esto limita a 10 lecturas por segundo, que es suficiente para un llenado de precisión y evita saturar la CPU del ESP32, permitiendo que mantenga la comunicación fluida con el PC.

Explicación del código

```
1   /*Includes_BEGIN*/
2 #ifndef MATLAB_MEX_FILE
3 #include <Arduino.h>
```

Listing 1: Configuración inicial y lectura de la librería

Como se muestra en el bloque de código de listing 1:
MATLAB_MEX_FILE es una macro que genera automáticamente el propio software cuando está compilando código para ejecutarlo en el ordenador.

La línea `ifndef MATLAB_MEX_FILE` es una directiva de preprocesador fundamental para la estabilidad del sistema. *MATLAB_MEX_FILE* es una macro que el software define de forma automática solo cuando compila el código para ser ejecutado en el procesador del ordenador (generando un archivo MEX) lo que permite que el código del bloque S-Function Builder gestione dos escenarios de funcionamiento distintos:

- **Escenario 1: Simulación y validación en el PC (Modo Host):** Cuando se pulsa el botón *Run* en Simulink para probar la lógica en el ordenador, MATLAB activa la macro. En este momento, el compilador del PC lee el código, pero gracias a la directiva `ifndef`, ignora las instrucciones de hardware (como `<Arduino.h>` o comandos de pines).
  Este paso es imprescindible porque Simulink realiza una validación estática y

una revisión interna de seguridad del bloque antes de cualquier despliegue. Si el ordenador intentara compilar el código de electrónica real sin este escudo, el sistema colapsaría al no encontrar las librerías físicas del microcontrolador en el sistema operativo del PC.

- **Escenario 2: Despliegue en el microcontrolador (Modo Target):** Al ejecutar la acción *Build, Deploy & Start*, el código se envía al ESP32 para una compilación cruzada (*cross-compilation*). En este entorno, la macro *MATLAB_MEX_FILE* no existe. Por lo tanto, el compilador del ESP32 sí procesa el código protegido, carga la librería `Arduino.h` y habilita la comunicación real con el sensor HX711 a través de los pines digitales.

```
4  class MiniHX711 {
5    int dt_pin;
6    int sck_pin;
```

Listing 2: Definición de la clase MiniHX711 y sus atributos

Como se muestra en el bloque de código de listing 2:
Se crea una clase llamada *miniHX711* para declarar los espacios de memoria para guardar los pines del ESP32 que se conectan con la báscula (dt_pin y sck_pin).
Aunque se podría crear las variables cada una de forma independiente se ha optado por crear una clase para empaquetar las variables con las acciones de la báscula y que el programa sepa a cuál de ellas pertence.
Además, si el nombre de estas variables ya está presente en alguna parte del código de configuración de MATLAB o de ESP32 el programa no lo confunde porque está incluido en la clase creada anteriormente.

```
7    public:
8      void begin(int dt, int sck) {
9        dt_pin = dt;
10       sck_pin = sck;
11       pinMode(sck_pin, OUTPUT);
12       pinMode(dt_pin, INPUT);
13       digitalWrite(sck_pin, LOW);
14     }
```

Listing 3: Método de inicialización del sensor y configuración de pines

Como se muestra en el bloque de código de listing 3:
Es la rutina de inicialización, el programa recibe los números de los pines, los guarda en la memoria, configura el pin sck_pin como salida (ESP32 envía los pulsos a través de este pin) y se inicializa a cero para que no empiece la lectura en cuanto se inicialice, sino que se active cuando se va a hacer la lectura.

```
16 long read() {
17   unsigned long tiempo = millis();
18   while (digitalRead(dt_pin) == HIGH) {
19     if (millis() - tiempo > 150) return 0; // Seguridad anti-cuelgues
20   }
```

Listing 4: Sincronización y sistema de seguridad (*timeout*)

Como se muestra en el bloque de código de listing 4:
Este bloque extrae los datos.
El sensor activa el pin de datos (DT=HIGH), en este tiempo el sensor convierte la deformación física en un número binario de 24 bits, el proceso lleva unos milisegundos y es importante que en ese tiempo no se envíen comandos nuevos.
Una vez el pin pasa a 'low' el dato está listo para ser enviado.
Para evitar fallos como un cable suelto de la báscula y que el programa entre en bucle, se pone un límite de tiempo, un "watchdog" que a los 150 milisegundos el ESP32 asume que algo no está funcionando de forma correcta y puede significar que el sensor se ha roto o se ha desoldado un cable y en ese caso hace un return 0 abortando la ejecución de la lectura del peso.

```
22  long valor = 0;
23  for (int i = 0; i < 24; i++) {
24    digitalWrite(sck_pin, HIGH);
25    delayMicroseconds(1);
26    valor = valor << 1;
27    digitalWrite(sck_pin, LOW);
28    delayMicroseconds(1);
29    if (digitalRead(dt_pin)) valor++;
30  }
31
32  digitalWrite(sck_pin, HIGH);
33  delayMicroseconds(1);
34  digitalWrite(sck_pin, LOW);
35  delayMicroseconds(1);
```

Listing 5: Lectura de bits y configuración de ganancia/canal

Como se muestra en el bloque de código de listing 5 se produce el Bit-Banging:

1. SCK_pin envía un pulso alto de 1 μs y se baja a LOW.
2. En ese momento el sensor devuelve un valor de bit por el cable de datos (dt_pin)
3. Se mueven todos los bits ya guardados a la izquierda para poder escribir el nuevo bit.
4. Si el pin de datos (DT) se lee como alto(1), se suma uno al valor acumulado
5. Al finalizar los 24 ciclos, se envía un último ciclo de pulso de reloj para configurar la ganancia de la siguiente lectura de forma automática.
   Esto se realiza dando un solo pulso al final, que escoge el canal A con una ganancia de 128 (preestablecido en HX711 como se puede ver en el anexo K en la tabla 3.)

Este bucle se repite los 24 bits que dan el valor del pesaje como se explica en K en la figura 2.

El sensor devuelve el número del peso en 24 bits, pero el ESP32 guarda los datos en variables enteras de 32 bits, por lo que para guardar este número correctamente se debe aplicar la máscara.
El número de 24 bits está codificado en complemento a dos, lo que significa que si empieza con un 1 el pesaje es negativo, por ello se aplica la máscara 0x800000 para

detectar si ese bit está a uno.
Si el valor es negativo se debe rellenar los 8 bits hasta llegar a 32 con unos para hallar el valor del peso en negativo $valor| = 0xFF000000$ La barra vertical es la operación OR a nivel de bits, es una manera simplificada de escribir la operación de OR con el valor y el numero especificado y escribirla en valor otra vez.

```
52 void leer_bascula_Start_wrapper(void)
53 {
54 /* Start_BEGIN */
55 #ifndef MATLAB_MEX_FILE
56 bascula.begin(22, 23);
57 #endif
```

Listing 6: Función de inicio del bloque S-Function en Simulink

Como se muestra en el bloque de código de listing 6:
Para que Simulink lea el programa correctamente debemos unirle las variables de las clases que hemos creado en el propio programa.
En este apartado se especifica el cable de datos (DT) está en el pin 22 y el reloj (SCK) conectado al pin 23 del ESP32.

```
61 void leer_bascula_Outputs_wrapper(int32_T *peso)
62 {
63 /* Output_BEGIN */
64 #ifndef MATLAB_MEX_FILE
65 peso[0] = bascula.read();
66 #else
67 peso[0] = 0;
68 #endif
```

Listing 7: Función de salida para la actualización de datos en tiempo real

Como puede apreciarse en el bloque de código de listing 7:
En bucle simulink llama a la función de 'bascula.read()' constantemente y el resultado lo guarda en el "array" de peso, si se está simulando en el ordenador se envía un cero para que no produzca errores el programa.

4.3.3. Calibración de la báscula

Los datos en crudo que se obtienen de la báscula no son legibles, éste envía las variaciones de voltaje que se deben traducir a gramos.
Para ello se realiza una tara básica siguiendo la ecuación de la resta

$$Y = mX + C$$

- Offset (C): Se resta el valor que marca la báscula cuando el sensor está vacío (-217400)
- Ganancia (m): Usando un peso conocido, se halla el factor multiplicador

$$m = \frac{\delta pesos}{\delta valoresdelabáscula} = \frac{208 - 0}{-238800 - (-217400)} = -0,0097196$$

Este valor es negativo por cómo está orientada la báscula internamente.

En este paso se ha realizado una tara muy general que sirve para traducir el valor de la báscula a gramos, sin embargo el valor del cero cambia constantemente debido al calentamiento del sensor, al ruido eléctrico o a la diferencia de envase ya que los envases no tienen exactamente el mismo peso.
por lo que es necesario implementar una tara dinámica. Cada vez que el PLC envíe la señal de empezar el llenado, antes de encender la báscula se realizará una tara dinámica para que la cantidad en gramos de la crema que vierte sea la establecida y no dependa ni de fluctuaciones externas ni de la diferencia de densidades de los botes.
La tara dinámica usa el bloque de Smaple Hold, este bloque tiene dos entradas

- Data (u): La señal del peso traducida a gramos
- Trigger: Señal física del pin 16 de ESP32 que indica que se debe empezar el llenado

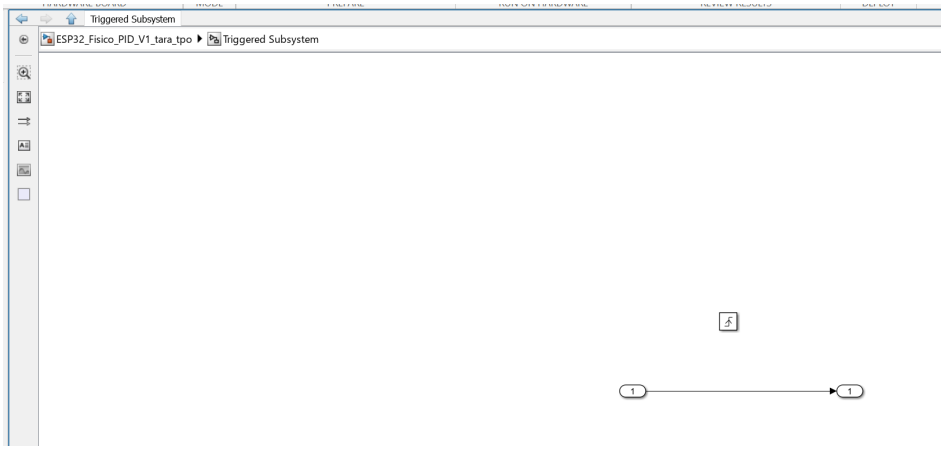


Figura 35: Sample and Hold - flanco de subida

La tara se activa (HIGH) con el pin 16 del microcontrolador, cuando el bloque Sample Hold detecta un flanco positivo se guarda el valor que le la báscula en ese instante.
Se espera unos milisegundos antes de empezar el llenado en el que el programa realiza la resta del peso variable con el valor guardado, en este momento el valor es cero por lo que el peso neto se pondrá a este valor. Este proceso lo realizará durante todo el ciclo mostrando en cada momento el peso real de la crema sin tener en cuenta el peso del recipiente.
Este dato será el utilizado para realizar el control de PID que garantiza que todos los botes tengan la misma cantidad de crema.

Bloque de retardo

Este bloque permite esperar un tiempo antes de comenzar el llenado, de esta manera se asegurar que al sistema le da tiempo a realizar la puesta en cero.
El bloque se llama *Unit Delay* (z^{-n}) en Simulink, donde n representa el número de periodos de muestreo que la señal será retenida.
La simulación opera con un tiempo de muestreo de 0,1 segundos, para retener un segundo la señal se debe calcular T_s y configurarlo como se muestra en la imagen 36.

$$T_s = \frac{t_{retardo}}{n} = \frac{1}{0,1} = 10 muestreos$$

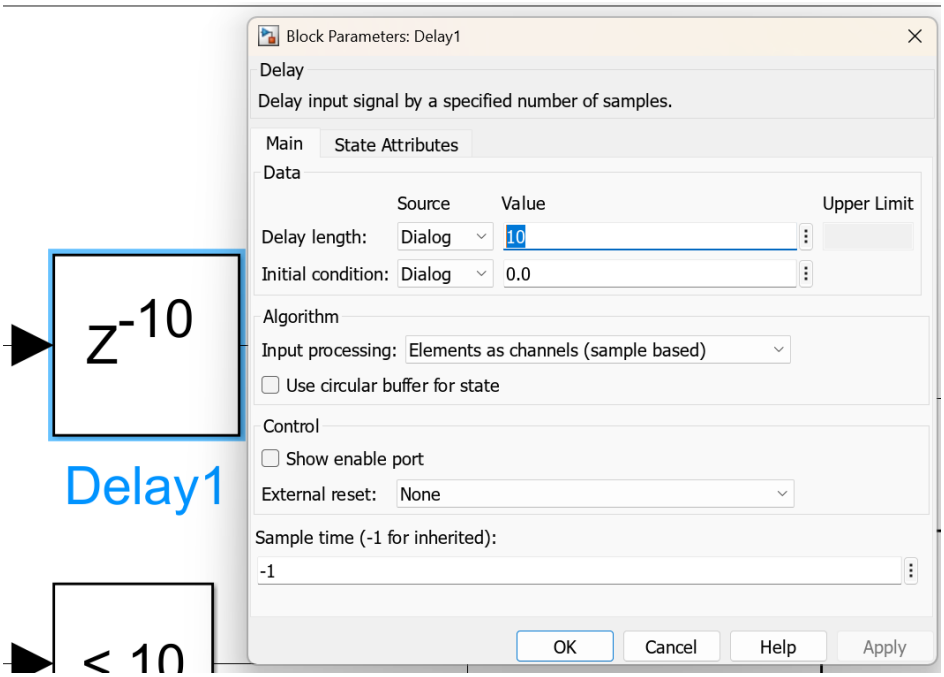


Figura 36: Configuración del bloque de retardo

La señal del botón llega sin retardo al bloque de *Sample and Hold* para ejecutar la tara dinámica. Simultáneamente, la señal retardada entra a la puerta AND junto con la señal del comparador de peso (<10). Este comparador permite el llenado mientras la crema del bote pese menos de 10 gramos (la consigna establecida). Al alcanzar dicha consigna, el nivel lógico del comparador pasa a bajo, lo que desactiva la salida de la puerta AND y evita que se accione la bomba.

4.3.4. Controlador PID

El controlador PID es el encargado de realizar el control del llenado de manera controlada, es decir, sin salpicar ni hacer paros bruscos ajustándose al peso asignado. Para ello, el PID, realiza los cálculos con el error:

$$Error = Consigna - pesoneto$$

En el software el PID se conecta con el pin 26, físicamente es el MOSFET que actúa como interruptor para activar la bomba.
Para que el microcontrolador entienda los valores del PID se debe configurar los valores de saturación del bloque como se muestra en la figura 37 para que la salida del bloque sea entre 0 y 255 que se el mínimo y máximo de potencia para la bomba.
El rango de 0 a 255 es un estándar en la electrónica digital, proviene de la resolución del bloque de Simulink que lo mapea por defecto a 8 bits que permite tener $2^8 = 256$ estados diferentes, lo que hace un rango de 0 a 255 tomando en cuenta el cero.

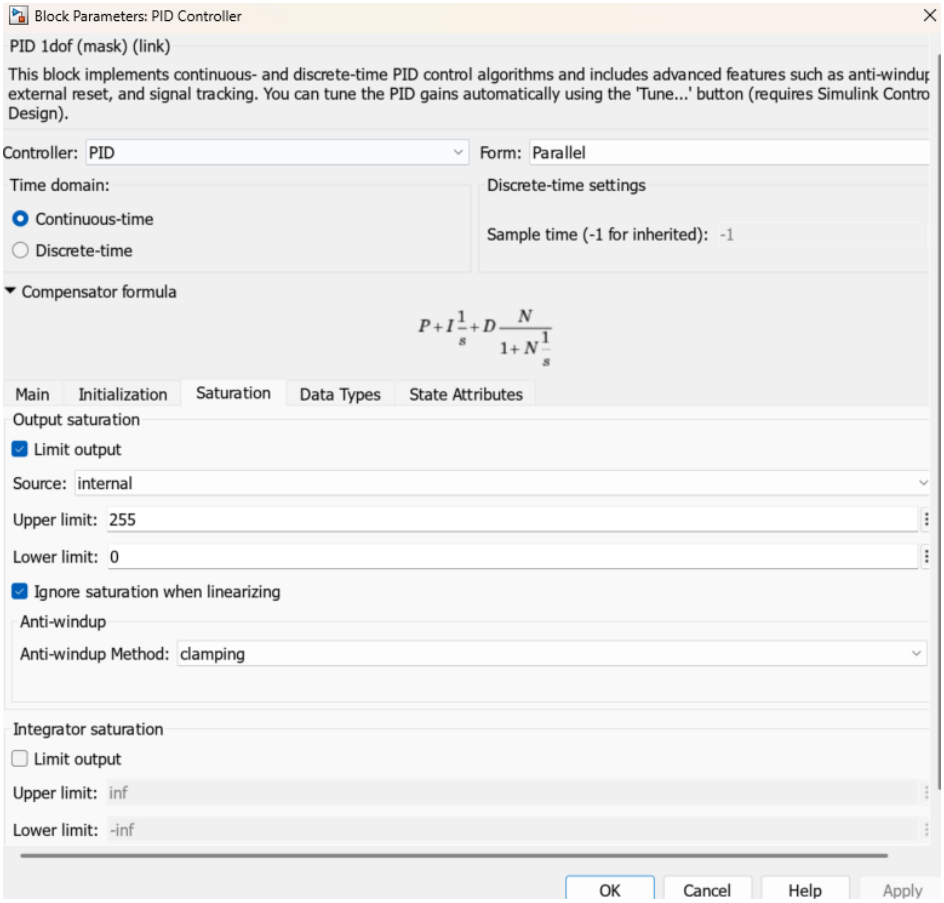


Figura 37: Configuración la saturación del PID

Para calcular los parámetros del PID se debería de conocer la función de transferencia, lo que implica realizar un modelo matemático de todos los componentes del sistema lo cual es muy complejo. Por ello estos valores se calculan de manera empírica mediante ensayo y error, sin embargo, se va a utilizar el sistema de Ziegler-Nichols para acercarnos a unos valores adecuados de PID. Para ello hay que realizar un bucle en lazo abierto ante una entrada escalón, como se muestra en la figura 38, para sacar los siguientes datos:

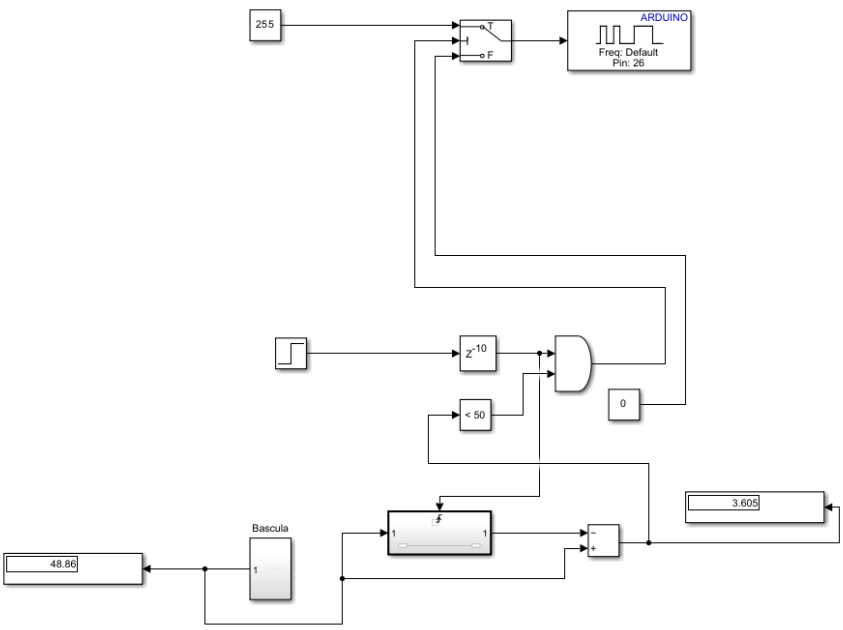


Figura 38: Obtención de datos para los cálculos de Ziegler-Nichols

1. Tiempo muerto (L): Es el tiempo entre la entrada escalón y el inicio del movimiento del motor. Representa la inercia del líquido, ver gráfico 39

$$\Delta t = L = 0,2s$$

2. Ganancia del proceso (K): Indica el cambio de la salida ante la entrada escalón, mostrado en el gráfico 40.

$$K = \frac{\Delta y}{\Delta t} = \frac{2,0 - 0,9}{14,0 - 8,0} = \frac{1,1}{6,0} \approx \mathbf{0,183}\ \text{g/s}$$

3. Constante de tiempo (τ): Es el tiempo que tarda la curva de la salida en alcanzar el $63,2\,\%$ del valor final máximo.
   En este caso el sistema no tiene un valor de tau porque no se estabiliza nunca ya que es un proceso integrador.
   Al ser este tipo de proceso, Ziegler y Nichols diseñaron fórmulas alternativas para este tipo de procesos.

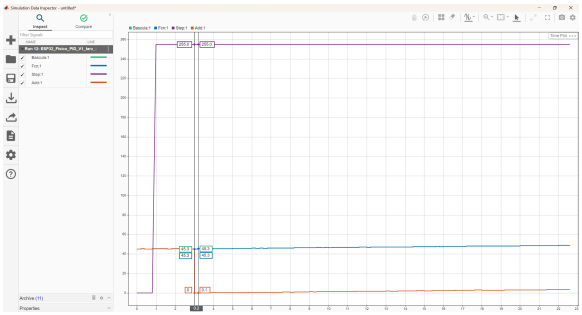


Figura 39: Valor de Tiempo muerto (L)



Figura 40: Valor de Ganancia (K)

Con todos los datos se aplica las fórmulas de Zieger-Nichols para procesos integradores:

$$K_p = \frac{0,9}{K \cdot L} = \frac{0,9}{0,183 \cdot 0,2} = 24,59$$

$$T_i = 3,3 \cdot L = 3,3 \cdot 0,2 = 0,66s$$

$$K_i = \frac{K_p}{T_i} = \frac{24,59}{0,66} = 37,26$$

Con estos valores se configura el bloque de PID del sistema, como se muestra en la imagen 41.
Al probarlo se verifica que realiza el control de manera correcta y precisa.

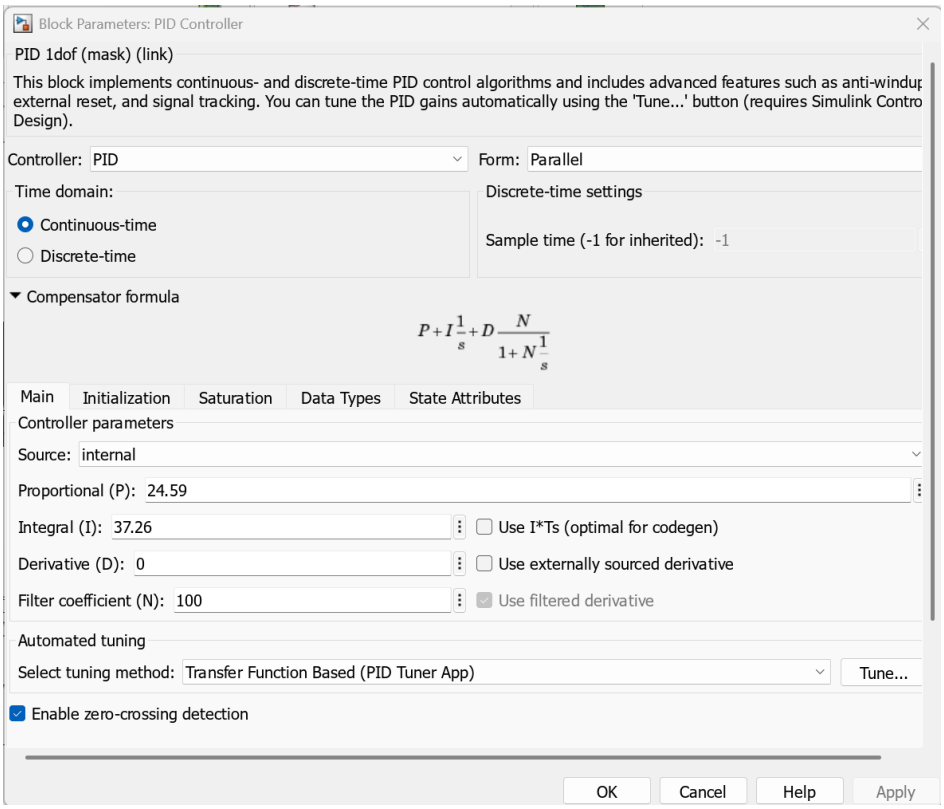


Figura 41: Parámetros de configuración de PID

Aunque se ha utilizado el bloque de PID, se ha optado por eliminar la acción derivativa ($K_d = 0$)y operar exclusivamente con un controlador Proporcional-Integral (PI).
El sistema implementa una báscula en la realimentación, muy sensible a los ruidos eléctricos y mecánicos como las vibraciones del motor o el impacto de las gotas de crema al caer.
Matemáticamente, el término derivativo calcula la tasa de variación del error respecto al tiempo ($\frac{de}{dt}$). Al aplicar una derivada sobre una señal ruidosa, las fluctuaciones de alta frecuencia se amplifican drásticamente, lo que provoca picos de voltaje en la señal de control (PWM) enviada al actuador.
Se sometería a los componentes de potencia a estrés mecánico y eléctrico innecesario que acorta la vida útil de los elementos además de generar un comportamiento inestable en el bombeo que no supondría una mejora real en la precisión del peso.

Por otro lado, la dinámica de llenado de fluidos es un proceso predecible y lento. La acción proporcional es suficiente para reducir el caudal a medida que el peso se acerca a la consigna y la acción integral se encarga de que el error en régimen permanente sea nulo. Al prescindir del término derivativo, se sacrifica una mínima velocidad teórica de asentamiento a cambio de obtener una señal de control suave, que prolonga la vida útil del hardware y garantiza una dosificación estable y libre de oscilaciones bruscas.

4.3.5. Bloque switch

Este bloque es el encargado de escoger la señal que pasa a la bomba.
Tiene tres entradas:

- Arriba la salida del PID
- Salida de la puerta AND
- Constante cero

La entrada del medio, que decide la posición que toma el interruptor, como se muestra en la imagen 42. Comprueba si la salida del AND es 1, y permite el paso de la señal de la salida del PID; si su valor es 0, se activa la constante cero y la bomba se mantiene apagada.
Esto lo consigue mediante el comprador en el mismo bloque del switch que compara con el valor de 0.5.

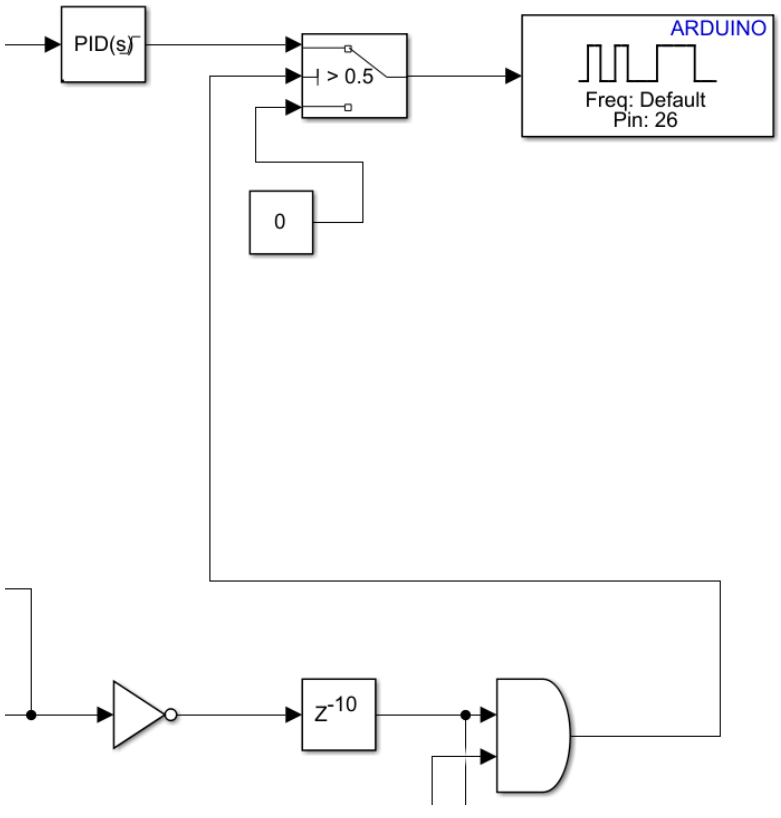


Figura 42: Parámetros de configuración de PID

4.3.6. Pruebas físicas

Para evaluar el correcto comportamiento del sistema se ha llevado a cabo varios ensayos físicos consistentes en el llenado consecutivo de envases, durante este tiempo se analizó la evolución temporal del peso registrado en la báscula.

En la siguiente gráfica 43, en la que el eje de abscisas representa el tiempo y el eje de ordenadas el peso en gramos, se pueden distinguir tres ciclos de trabajo claramente diferenciados correspondientes a cada uno de los botes del proceso. Se observa como la señal en marrón (*Peso neto*) va subiendo cuando esta la señal morada activa (*Digital Imput1*, señal de llenado) realizando una tara de un segundo antes de iniciar el llenado.

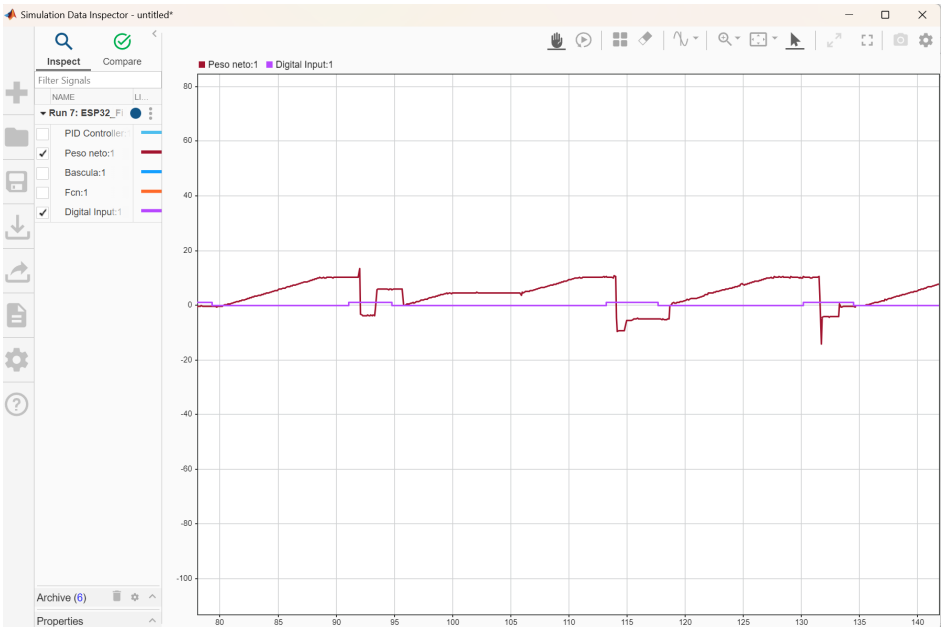


Figura 43: Gráfico de llenado de 3 botes (eje X: tiempo; eje Y: gramos)

Capítulo 5: Simulación del robot - gemelo digital

Para simular el desempeño del robot y comprobar su correcto funcionamiento se debe realizar la simulación previa a las pruebas físicas. De esta forma se evitan errores futuros y se puede dar soporte a un robot en una empresa sin necesidad de realizar desplazamientos, de ahí la importancia de realizar gemelos digitales.
En esta aplicación se ha escogido un robot de ABB para realizar la simulación ya que el software es muy preciso al simular la aplicación y asegura que el envase se desplazará del mismo modo que en el sistema.

5.1. Arquitectura de señales (E/S) en el controlador virtual

El primer paso para establecer la comunicación con la célula es definir las señales en la memoria del controlador virtual. El software distingue entre el entorno de las señales de la estación (entono 3D) y el controlador del robot. Para que el código RAPID pueda interactuar, las variables deben ser creadas con los datos de configuración del sistema.

5.1.1. Entradas digitales (DI)

La simulación de los sensores de posición (los sensores de final de cinta) se debe configurar como entrada en la tabla *Controlador >Configuración >I/O System >Signal* como se muestra en la figura 44, aquí se genera la nueva señal denominada `DI_S2` (fig 45), esta variable se configura como "digital input".
Se realiza con un cable conectado a la tarjeta de entradas del armario de control del robot.

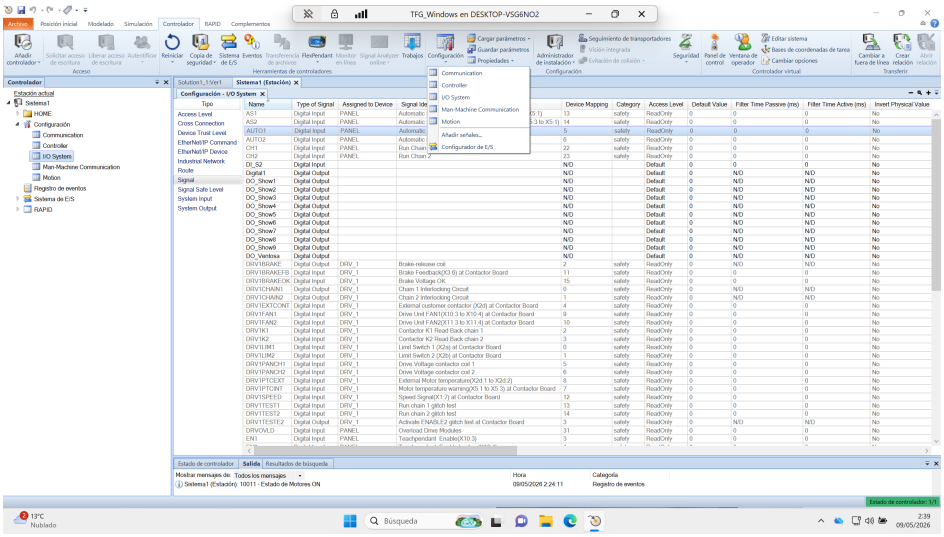


Figura 44: Configuración de entradas y salidas digitales

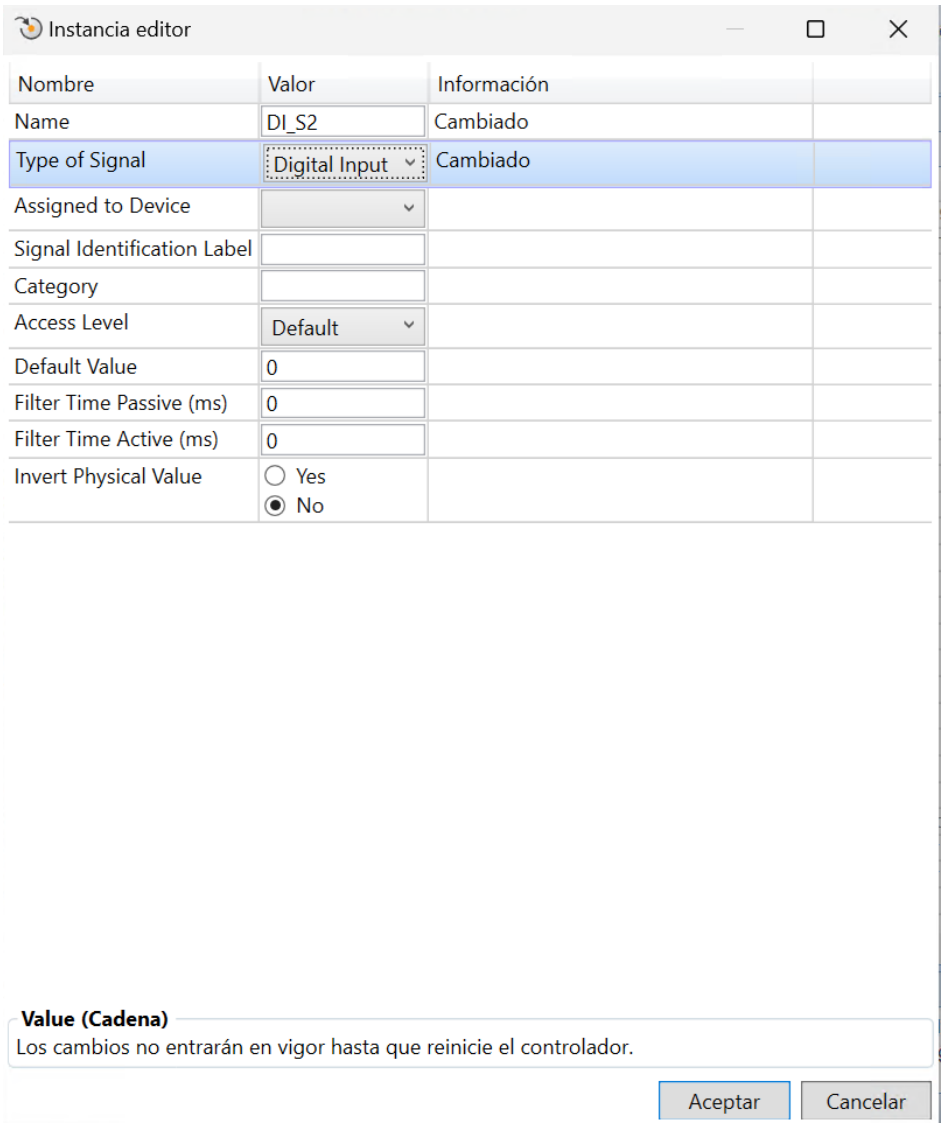


Figura 45: Configuración de señal del sensor de la segunda cinta

5.1.2. Salidas digitales (DO)

De manera inversa, el robot necesita enviar comandos al entorno para activar o desactivar la ventosa que tiene como herramienta. Para ello se configura la salida `DO_Ventosa` como se muestra en la figura 46.

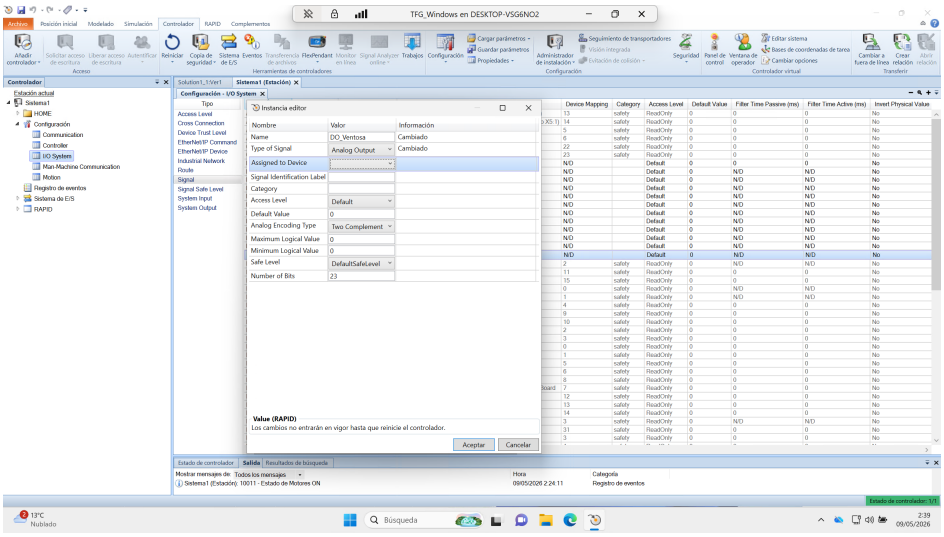


Figura 46: Configuración de entradas y salidas digitales

5.1.3. Guardar entradas y salidas en la memoria RAM

Una vez que se han configurado las salidas y entradas para el controlador hay que reiniciar en caliente, como se muestra en la imagen 47 , en caso contrario el código RAPID no encontraría estas señales porque no habrían sido guardadas en la memoria RAM del programa.

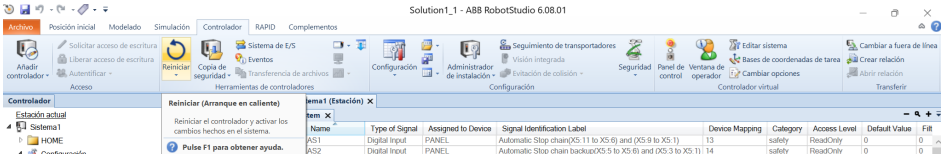


Figura 47: Configuración de entradas y salidas digitales

5.1.4. Integración de lógica de estación (Station logic)

Una vez que el robot es capaz de cambiar los valores de sus salidas, el entorno 3D no reacciona por sí solo, hay que enlazar estas variables con el entorno.
Para conseguirlo desde el menú de *Simulación >Lógica de estación >Conexiones E/S* se realiza el enrutamiento entre la señal del objeto (controlador del robot) y la señal de origen (salida correspondiente) como se muestra en la figura 48.
Este proceso se debe realizar con todas las salidas digitales que tienen efecto en la simulación 3D que son `Digital1`, `DO_Ventosa`, y las nueve señales de salida para mostrar el envase al que corresponde.
Esta arquitectura de eventos (Event-Driven) permite desacoplar el código RAPID de la física de la simulación. El robot simplemente activa un interruptor, y es la plataforma RobotStudio la que, a través de la Lógica de Estación, procesa ese cambio y ejecuta la acción 3D pertinente, manteniendo así una simulación más realista con un robot industrial físico.

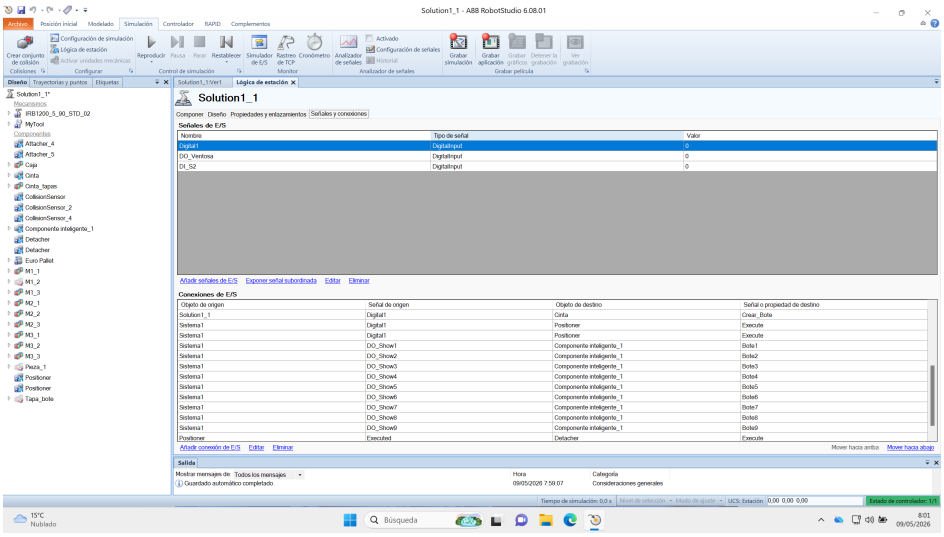


Figura 48: Enrutamiento de señales controlador y señal de origen

5.2. Señales de simulación para mostrar la ubicación de los envases

El mayor reto es la simulación visual del paletizado, ya que se debe conocer la posición exacta de cada envase para poder mostrar el que corresponde en la simulación, lo que no sería necesario en una implementación real.

Uno de los retos que se encontraron al realizar el programa era la simulación de varios envases iguales que fuesen manipulados en varios ciclos, para ello, en vez de realizar 9 botes y 9 tapas distintas en cada ciclo, se colocan el bote y la tapa a su posición inicial como si fuese un envase nuevo, mientras que se muestra en la posición del palet un nuevo envase ya colocado. Para ello se necesitaba poder cambiar el valor de la señal `Digital1` que hace que el envase y la tapa vuelvan a su posición inicial.

En el código de RAPID, ver anexo E.1, se implementaron nueve salidas digitales en el controlador (`DO_Show1` a `DO_Show9`). Dentro del bucle de movimiento del robot se realizó la operación $n_{bote} := 10 - ((i * 3) + j + 1)$;para conocer la posición en la que estaba dejando el robot el bote, luego con una estructura CASE se activa la señal correspondiente a la posición.

Los botes se colocan desde el más lejano al más cercano al robot, para evitar posibles choques entre ellos, ya que el robot no pasa por encima de ninguno de los frascos colocados, además, de esta manera no es necesario levantar tanto el brazo del robot y disminuye el tiempo que tarda este en colocar los recipientes.

Para que estas señales tuviesen acciones visuales se utilizó el componente inteligente, en esta parte del programa en él se crearon nueve bloques de la clase show (mostrar) que permite que cada vez que el RAPID envíe la señal de show se activen estos bloques para que se pueda visualizar la posición de los envases como se muestra en la figura 49. El paso final fue la conexión de E/S de la lógica de la estación con el controlador.

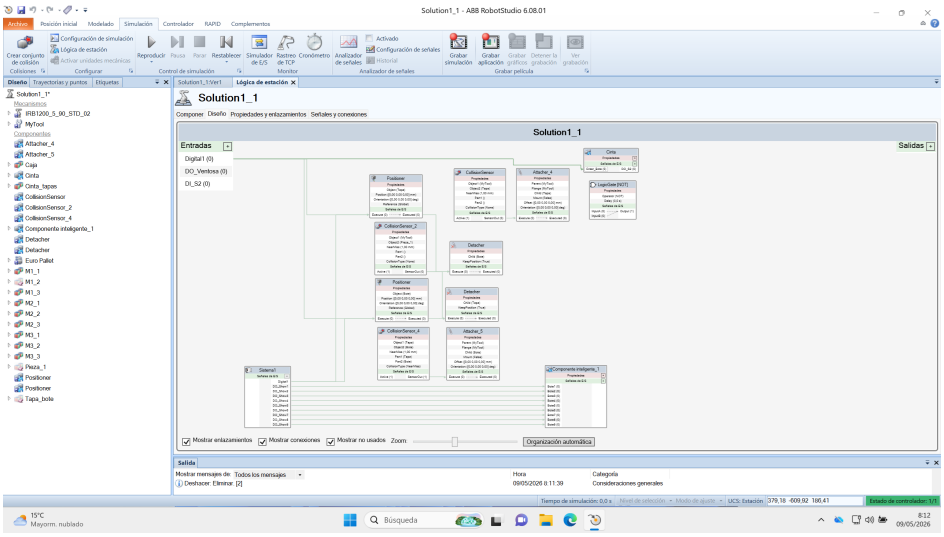


Figura 49: Conexiones entre la controladora y el entorno 3D

5.3. Conclusión

La configuración de las señales y lógica en RobotStudio muestra que una celda robótica es una máquina de estados compleja. Mediante la asignación correcta de Entradas/Salidas, el uso de Componentes Inteligentes para encapsular lógicas repetitivas y la resolución de conflictos cinemáticos, se ha logrado un Digital Twin (Gemelo Digital) preciso.En este entorno simulado se valida que el robot no colisione y se garantiza que la lógica del programa RAPID sea idéntica a la que se requiere en un entorno industrial real con PLC y sensores físicos. El programa completo y un vídeo demostrativo del funcionamiento de esta simulación se pueden consultar en el drive del anexo Ñ .

Capítulo 6: Programación industrial

En este apartado se explica el desarrollo e implementación del sistema industrial de la programación en un PLC.

6.1. Material para la programación industrial:

Para facilitar la comprensión del sistema se van a enumerar los componentes que se usan en el proyecto de la automatización.



Figura 50: Sistema industrial

Actuadores: Estos elementos son los que se mueven o realizan algún movimiento físico

- **Primera cinta:** Por esta cinta entran los envases vacíos y se procede a su llenado.
- **Control de llenado:** Se realiza sobre de la primera cinta. Consiste en el control mediante la bomba peristáltica y la báscula desarrollados en el capítulo 4.
- **Segunda cinta:** Una vez que el recipiente se ha llenado se realiza el movimiento de la primera cinta a la segunda.
- **Paletizado:** Se realiza en la segunda cinta. Un robot de ABB coloca la tapa y cierra los botes, luego los deposita sobre un palet listos para ser transportados. Se ha desarrollado en el capítulo 5.

Sensores: En el proceso se cuenta con dos sensores.

- **S1:** Controla el detenimiento de la primera cinta para que se pare justo encima de la báscula y se pueda llenar el bote correctamente.

- **S2:** Controla la parada de la segunda cinta para que se detenga justo donde el robot tiene calculado el punto en el que debe recibir el bote lleno de crema y lo coloca para ser transportado.

Comunicaciones: Las comunicaciones entre los distintos elementos se gobiernan a través del PLC.
El PLC envía una señal booleana de llenado que permite que el control empiece a dispensar crema y a calcular el PID. Además, el PLC le determina los requisitos de los botes del lote seleccionado. De esta manera la automatización tiene más flexibilidad, ya que se puede escoger para cada ciclo de funcionamiento el peso de cada lote.

Cuando el control de llenado acaba, éste envía otra señal booleana al PLC para que el sistema pueda continuar con el movimiento de la primera cinta. El robot funciona de manera similar, el PLC manda la señal booleana de flanco positivo del sensor, el robot realiza el paletizado y por último éste emite una señal de que ha acabado y está esperando para realizar la misma operación con el siguiente envase.

El sistema se ha dividido en dos cintas independientes para reducir los tiempos de procesamiento. Esta configuración permite realizar el llenado de las cremas y el paletizado con el robot de forma simultánea, esto supone un aumento de la productividad de la empresa.

6.2. Programación del PLC

Para la gestión de la línea de producción de llenado de cremas que integra el control de llenado y el robot paletizador se implementa la programación de un PLC mediante el software de TIA PORTAL. Con el fin de garantizar que el programa funciona antes de su montaje en una planta real y que se comporta de manera fiable y segura se realizará la simulación de la planta en la pantalla HMI del software.

6.2.1. Diseño de la programación mediante Guía GEMMA

Se he utilizado una versión simplificada de la Guía GEMMA para realizar una programación limpia y de fácil mantenimiento.

- **A1 (Parada en el estado inicial):** El sistema esta alimentado a la espera de una orden de marcha para inicial cualquier ciclo.
- **A2(Parada pedida final ciclo):** Permite que los envases que ya están en la cinta finalicen su proceso de llenado y paletizado antes de detenerse totalmente.
- **A5 (Preparación posterior al defecto):** Tras liberar la emergencia, este estado permite el rearme del sistema y la limpieza o despeje de la zona antes de pasar a la puesta en origen (A6).
- **A6(Puesta en estado inicial):** En esta etapa el operario debe retirar envases de la cinta y el robot automáticamente se desplaza hasta su posición de inicio para poder reiniciar el ciclo en A1.

- **F1(Producción normal):** Esta etapa ejecuta el ciclo completo de manera autónoma.
- **F4(Marcha de verificación sin orden):** Permite al operario o técnico accionar actuadores de forma individual para comprobar su estado físico.
- **F5(Marcha de verificación en orden):** Se utiliza para comprobar que la secuencia lógica y las señales de los sensores llegan correctamente al PLC.
- **D1(Parada de emergencia):** Es activada por la seta de emergencia. Deshabilita inmediatamente las cintas, el robot y el llenado para garantizar la seguridad de la máquina y el personal.

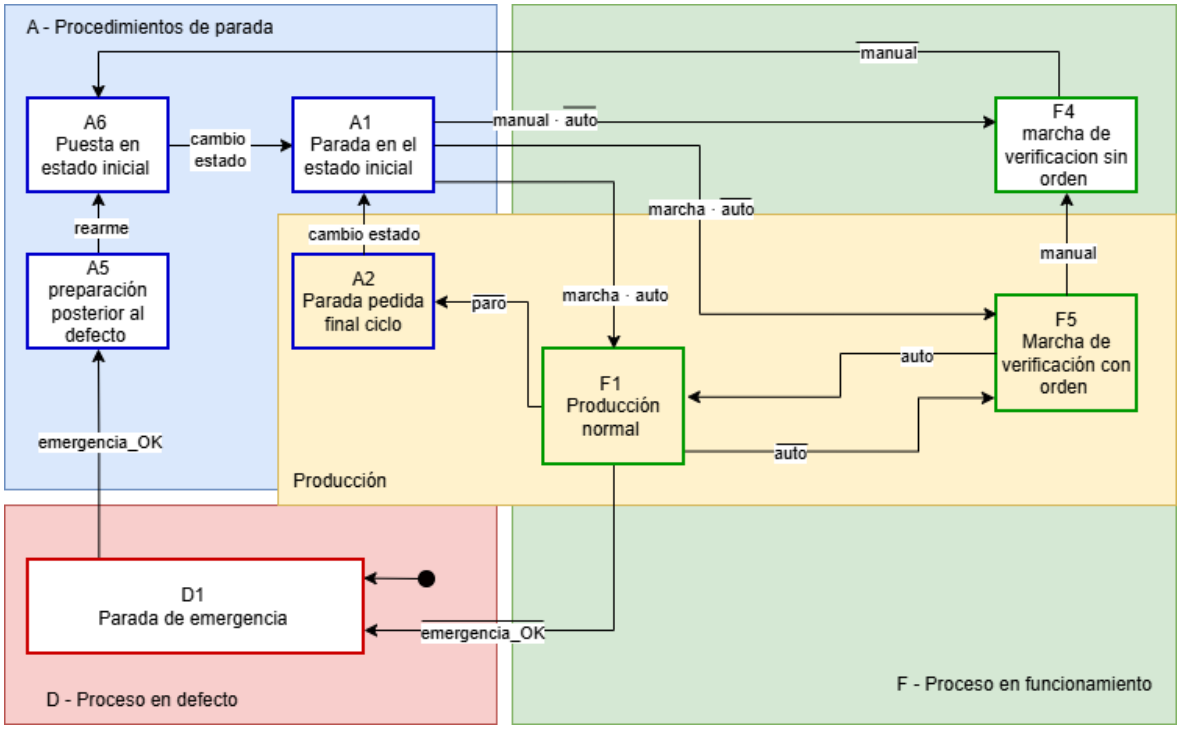


Figura 51: Esquema Guía GEMMA

Para el desarrollo de la lógica interna de cada modo de marcha definido en la Guía GEMMA, se ha empleado el lenguaje GRAFCET. Esta herramienta de modelado gráfico se ha escogido porque permite estructurar el comportamiento secuencial de la planta de forma jerárquica y clara, la elección de este método responde a la necesidad de crear un código modular, escalable y de fácil mantenimiento. Al dividir el proceso en etapas y transiciones, se garantiza que el PLC ejecute las acciones de llenado, pesaje y paletizado de forma estrictamente ordenada, evitando conflictos en la ejecución de los actuadores.

6.2.2. F1 Producción normal

El funcionamiento normal de la planta se gestiona mediante el GRAFCET F1, detallado en el anexo D) que garantiza la sincronización entre el sistema de dosificación y la célula de paletizado haciendo así que el sistema sea más rápido produciendo.

- **Etapa 0:** El sistema permanece a la espera de que el operario defina los gramos de crema que se deben depositar en cada envase a través del terminal HMI. La transición a la siguiente etapa solo se habilita tras validar la configuración mediante un pulsador de confirmación.
- **Etapa 1:** En este punto, el PLC realiza una puesta a cero de los registros internos. Se reinicia el contador de cajas del lote actual y se aseguran las condiciones iniciales de las ramas secundarias, garantizando que no existan órdenes residuales de ciclos anteriores.
- **Etapa 2:** Tras recibir la orden de marcha, el programa vuelca el valor de la receta desde el HMI a una variable de memoria interna para su procesamiento en tiempo real. Simultáneamente, se activa la Cinta 1, iniciando el desplazamiento del primer envase hacia la estación de dosificación.
- **Etapa 3:** La llegada del envase es detectada por el sensor S1. El programa utiliza el flanco de subida de esta señal para detener el avance y enviar una señal booleana al bloque de control de la bomba peristáltica y la báscula, iniciando el llenado preciso. El paso a la siguiente etapa se produce con la señal booleana de salida del control que entra al PLC indicando que ya ha acabado.
- **Etapa 4:** Una vez que el sistema de pesaje confirma que se ha alcanzado la consigna, se habilita el paso a la macro etapa 100.
- **Etapa 100:** El sistema se bifurca en dos ramas para optimizar el tiempo. De esta manera permite que la planta realice la acción de llenado mientras que el robot puede ir colocando las cremas.
    1. **Rama de producción (Etapas 10 - 12):** Replica la lógica de transporte y llenado para el siguiente envase de la cadena.
    2. **Rama de paletizado (Etapas 20 - 22):** De forma paralela, se activa la Cinta 2 y se envía una orden lógica al robot paletizador. El robot inicia su ciclo de recogida y ubicación del envase ya procesado en la caja correspondiente.
    3. **Etapas 12 - 22:** Permite una sincronización de las dos acciones y cuando ambas hayan acabado se sale de la macro etapa.
- **Etapa 5:** Se actualiza el contador de unidades completadas del lote y, de forma independiente, el contador histórico de la máquina, que registra el número total de ciclos acumulados desde su puesta en marcha para realizar tareas de mantenimiento preventivo.

6.2.3. F5 Marcha de verificación con orden

El bloque F5 (ver anexo D) es una herramienta fundamental para la puesta en marcha inicial y el mantenimiento preventivo. Comparte la estructura lógica de la producción normal (F1) pero su funcionamiento es estrictamente secuencial y por demanda del operario.

En este modo, el ciclo de trabajo no es automático; cada transición entre etapas requiere la pulsación del botón de marcha. Esto permite al técnico validar paso a paso

el comportamiento de los actuadores y la respuesta de los sensores. Para garantizar la seguridad en este modo manual, se han integrado etapas de bloqueo adicionales. Estas aseguran que, aun cuando el sensor detecte la presencia de un envase, el sistema no inicie el llenado o el movimiento del robot de forma autónoma, permanezca en estado de espera hasta recibir la confirmación física del operario.

6.2.4. A2 parada perdida final de ciclo

La implementación del estado A2 (ver anexo D) tiene como objetivo realizar una detención controlada de la planta sin dejar el producto a medio procesar. A diferencia de una parada de emergencia, la orden de parada a final de ciclo permite que el sistema complete las tareas pendientes hasta que todos los envases en tránsito hayan sido llenados y paletizados.

La lógica de este bloque se adapta a la carga de la cinta en el momento de la pulsación:

- **Caso de un solo envase:** Si la parada se solicita cuando solo hay un bote entre las etapas 1 y 3, el programa asegura su llenado y posterior colocación por el robot antes de detenerse.
- **Caso de dos envases simultáneos:** Si existen dos unidades en proceso, el sistema gestiona la prioridad de ambas ramas. El último envase debe ser colocado, mientras que el penúltimo debe completar primero su fase de llenado para después pasar a la zona de paletizado. Para conseguirlo, se han incorporado las etapas 9, 10 y 8 en la primera rama del GRAFCET, garantizando que ninguna unidad quede desatendida antes de que el PLC conmute al estado de reposo.

6.2.5. A6 Puesta en estado inicial

Una vez finalizada la intervención manual en la etapa anterior (A5), en la que el operario ha procedido a despejar las cintas de cualquier envase o residuo, el estado A6 se centra de forma exclusiva en la recuperación de la configuración mecánica del sistema.

En esta fase, el PLC ejecuta de manera automática la secuencia de retorno a la posición de origen del robot. Esta maniobra es necesaria para garantizar que el brazo robótico se encuentre en un punto de partida libre de colisiones antes de reiniciar la producción. Solo cuando el sistema recibe la confirmación de posición por parte del controlador del robot, se valida la transición definitiva al estado A1, quedando la planta en reposo y totalmente disponible para un nuevo ciclo de trabajo.

6.3. Programación

La programación del sistema en TIA Portal se ha diseñado siguiendo criterios de modularidad y eficiencia, que facilitan tanto la depuración inicial como futuras actualizaciones. La organización del código se divide en diferentes tipos de bloques según su función:

6.3.1. Tipos de bloques usados para las entradas, salidas y lógica

Funciones (FC): Se ha llevado a cabo un bloque de función dedicado a la gestión de la periferia (entradas y salidas). Las FC son bloques de código sin memoria propia que ejecutan una tarea específica cada vez que se les da la orden. En este caso, actúan como una capa de abstracción que mapea las señales físicas de los sensores y actuadores hacia variables internas del programa, y mantiene el código principal limpio y desacoplado del hardware.

Bloques de Datos (DB): Se ha definido un bloque de datos global para centralizar la lógica y el estado del sistema. A diferencia de las FC, los DB tienen memoria estática, lo que permite almacenar y conservar valores (como el estado de las etapas, las recetas o los contadores) a lo largo de los ciclos de sacan del PLC. Este bloque de lógica gestiona la Guía GEMMA, se encarga de controlar los distintos bloques de funcionamiento que están en bloques de tipo DB.

6.3.2. Implementación de GRAFCETS

La traslación del diseño del GRAFCET al lenguaje de programación SCL se ha realizado mediante una estructura de control de flujo tipo CASE. Esta elección técnica aporta ventajas decisivas:

La traslación del diseño del GRAFCET al lenguaje de programación SCL se ha articulado mediante una estructura de control de flujo tipo CASE, lo que permite una transición directa desde el esquema hacia el código fuente en el PLC. Esta elección técnica facilita la legibilidad del programa, ya que establece una analogía visual con el esquema original. Al segmentar la lógica en pasos numerados, se puede identificar de forma inmediata la etapa que se encuentra activa y las condiciones de transición asociadas.

La seguridad intrínseca del software se ve reforzada por el empleo de una variable de estado única en la instrucción CASE. Al gestionar el flujo, se garantiza por diseño la exclusión mutua entre las distintas fases del proceso, impidiendo que el autómata pueda ejecutar simultáneamente dos etapas que resulten incompatibles o contradictorias. Este control de la secuencia elimina el riesgo de colisiones lógicas y asegura que el sistema respete el orden de operaciones definido, aportando robustez a la fiabilidad del entorno de producción.

6.3.3. Control de actuadores y salidas

Para el accionamiento de los dispositivos físicos, se ha priorizado el uso de bobinas de asignación directa en lugar de instrucciones set/reset.

Esta metodología simplifica la programación ya que cuando una etapa deja de estar activa, su salida asociada se desactiva automáticamente. Esto evita que el sistema mantenga órdenes activas de forma involuntaria por un fallo de lógica en los reajustes (resets).

6.3.4. Interfaz Hombre-Máquina (HMI) y Simulación

El diseño de la pantalla HMI se ha programado como un entorno de supervisión y control intuitivo. La interfaz se divide visualmente en tres áreas clave: En la sección izquierda de la pantalla se ha integrado un panel de control que emula la botonera física, facilitando la interacción directa con los mandos de marcha, paro, rearme y parámetros de configuración del sistema. En el área central de la interfaz se despliega una representación gráfica dinámica de la célula de producción que permite monitorizar en tiempo real el desplazamiento de los botes a través de las cintas transportadoras además de las estaciones de llenado y de robot paletizador.

6.3.5. Gestión de usuarios y control de accesos

Para garantizar la seguridad y el correcto funcionamiento de la línea de envasado, se ha implementado un sistema robusto de gestión de usuarios mediante las herramientas de seguridad del entorno TIA Portal, configurado específicamente para el panel HMI. Esta arquitectura de seguridad restringe el acceso a las funciones críticas de la máquina, asegurando que solo el personal cualificado pueda intervenir en el sistema.

El diseño de seguridad se ha estructurado creando grupos de autorización específicos y asignándoles usuarios nominales. Se han definido tres perfiles jerárquicos:

- **Grupo Operario (Usuario creado: `Operario`):** Dispone de los permisos básicos requeridos para la producción diaria. Su nivel de autorización permite la navegación por la interfaz principal, arrancar el ciclo de producción normal (modo F1) y detener el sistema de forma controlada.
- **Grupo Mantenimiento (Usuario creado: `Manteni`):** Nivel de autorización diseñado expresamente para el personal técnico. Habilita el acceso a las pantallas de diagnóstico y la ejecución de los modos de marcha manual y de verificación (F4 y F5), permitiendo accionar los actuadores de forma individual. Dado que las labores de calibración y diagnóstico pueden requerir pausas para la observación prolongada del comportamiento físico de la línea, el tiempo de inactividad permitido antes del cierre de sesión automático se ha extendido a 4 minutos.
- **Grupo Administracion (Usuario creado: `Admin`):** Representa el nivel con privilegios totales sobre el sistema.

La configuración detallada se ha centralizado en el árbol del proyecto bajo el menú "User administration". Tal y como se refleja en la imagen52, todos los perfiles cuentan con protección mediante contraseñas encriptadas. La operabilidad de los distintos botones y campos de entrada está programado en las pantallas HMI. Si un operador intenta accionar una función restringida sin la autorización pertinente, el sistema interrumpe la acción y despliega automáticamente una ventana de autenticación.

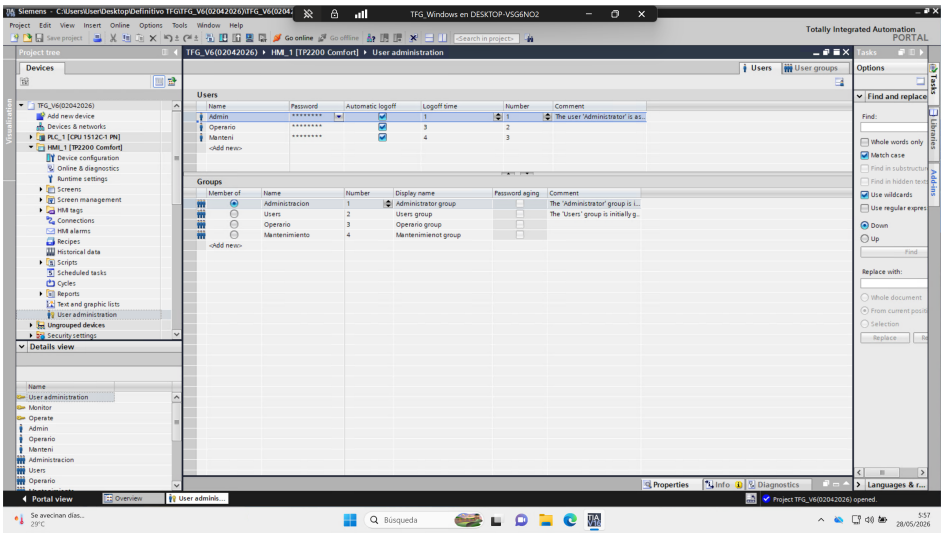


Figura 52: Configuración de perfiles, grupos y tiempos de cierre de sesión automático en TIA Portal.

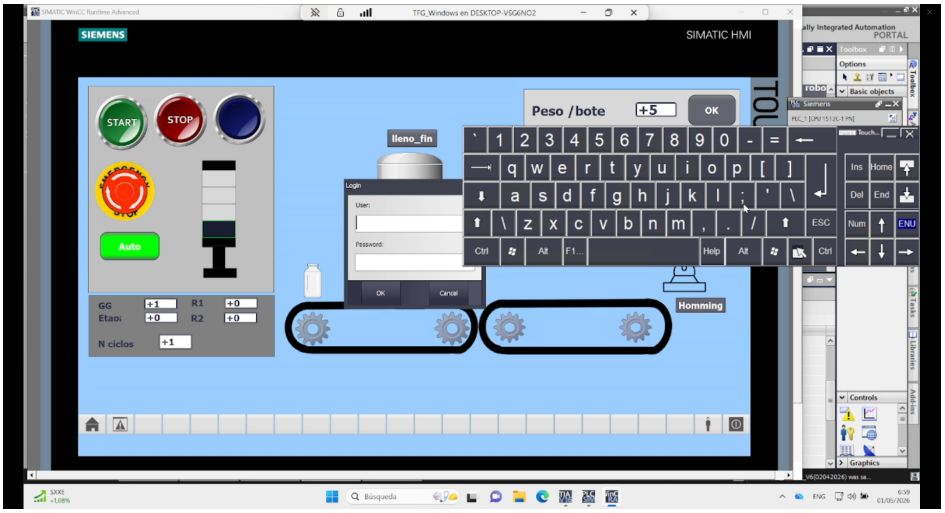


Figura 53: Ventana de autenticación de seguridad en la interfaz HMI.

6.4. Pruebas

Una vez terminada la programación y la pantalla se simuló en esta para comprobar el correcto funcionamiento. Fue un desafío mostrar los envases correctamente ya que en la vida real no desaparecen sin más y en la simulación hay que controlar cada etapa para poder visualizarlas de una manera correcta y que parezca natural. Como complemento a la información expuesta en este capítulo, en el Anexo N se facilita el acceso a un repositorio en la nube. En dicho enlace se incluye un vídeo demostrativo del funcionamiento real de la estación, el archivo del proyecto original desarrollado en TIA Portal y su correspondiente documentación técnica en formato PDF.

Capítulo 7: Estudio Económico

Este estudio económico detalla la cuantificación financiera requerida para la viabilidad, desarrollo y puesta en marcha de la línea de llenado de cremas. Para facilitar el análisis el presupuesto se ha desglosado en tres partes: costes de hardware y materiales, costes de licencias de software industrial y costes de ingeniera. Anexo M

7.1. Costes de hardware y materiales del prototipado

Esta sección comprende los componentes físicos necesarios para el montaje de la estación.

Descripción	**Cantidad**	**Precio unitario**	**Total**
Bomba peristáltica	1	5,19 €	5,19 €
MOSFFET	1	0,78 €	0,78 €
ESP32	1	3,33 €	3,33 €
Switch	1	1,33 €	1,33 €
Manguera	1	1,81 €	1,81 €
Báscula	1	4,57 €	4,57 €
Protoboard	1	1,75 €	1,75 €
Cables de protoboard	1	2,39 €	2,39 €
Cables de 12V	1	2,44 €	2,44 €
Convertor de DC universal a cables positivo y negativo	1	0,67 €	0,67 €
MDF	1	0,72 €	0,72 €
Pulsador	1	0,06 €	0,06 €
Diodo	1	0,20 €	0,20 €
TOTAL:			**25,24 €**

Tabla 1: Desglose de costes de hardware industrial

7.2. Costes de hardware industrial

Para la viabilidad técnica y la escalabilidad hacia un entorno industrial, la arquitectura física se simplifica al centralizar todo el control en un único PLC. Este autómata es capaz de asumir la gestión íntegra del sistema y de ejecutar el control PI validado en el prototipo de manera robusta, rápida y eficiente a gran escala. Dicha centralización optimiza significativamente los recursos, ya que un solo dispositivo posee la capacidad de gobernar de forma simultánea múltiples líneas de producción.

Descripción	**Cantidad**	**Precio unitario**	**Total**
Bomba peristáltica	1	342,20 €	342,20 €
Manguera	5	2,80 €	14 €
Báscula	1	80,18 €	80,18 €
CPU	1	764,36 €	764,36 €
Robot	1	11.000,00 €	11.000,00 €
Cintas	2	256,90 €	513,80 €
Fotocélula	2	277,75 €	555,50 €
HMI	1	1.799,00 €	1.799,00 €
Envolvente (Caja Rose	1	60,88 €	60,88 €
Pulsador de Eme	1	45,75 €	45,75 €
Botones de Paro	1	6,12 €	6,12 €
Botones de Marc	1	6,12 €	6,12 €
Contactor (Mani	1	71,78 €	71,78 €
Guardamotor (Se	1	180,28 €	180,28 €
Baliza de Señalización	1	224,31 €	224,31 €
TOTAL:			**15.664,28 €**

Tabla 2: Desglose de costes del prototipo

7.3. Costes de software

El desarrollo del control, la programación del PLC y la simulación del gemelo digital exigen herramientas de software de alto nivel. Dado que estas licencias industriales suelen adquirirse con carácter anual en una empresa, se imputa al proyecto una cuota de amortización estimada correspondiente a tres meses de uso por un único puesto de ingeniería.

Concepto	**Descripción**	**Coste(€)**
MATLAB & Simulink	Programación algorítmica, S-Functions y diseño del control	964,00
Siemens TIA Portal	Desarrollo de secuencias GRAFCET y Guía GEMMA	760,26
ABB RobotStudio	Modelado 3D, cinemática y validación del gemelo digital	1068,00
Dassault Systèmes SolidWorks	Diseño CAD de las piezas de MDF y ensamblajes	3.480,00
Subtotal Software		**6.272,26 €**

Tabla 3: Desglose de costes de licencias y software

7.4. Costes de ingeniería y mano de obra

El desarrollo íntegro del sistema requiere perfiles cualificados en diseño mecánico, electrónica, teoría de control y programación de autómatas. Para este presupuesto, se asume un coste estándar de 40 €/hora que corresponde a la remuneración de a un ingeniero de automatización junior, que abarque todas las tareas de las distintas fases del ciclo de vida del proyecto.

Fase del Proyecto	**Horas Estimadas**	**Tarifa (€/h)**	**Precio Total (€)**
Diseño CAD 3D y modelado del soporte	15	40,00	600,00
Diseño eléctrico, dimensionamiento térmico y ruteo de la PCB	20	40,00	800,00
Programación y sintonización de control PID (MATLAB/ESP32)	40	40,00	1.600,00
Programación del PLC (Lógica de estados y modos de marcha)	45	40,00	1.800,00
Simulación y programación de trayectorias (RobotStudio)	40	40,00	1.600,00
Montaje físico, soldadura y pruebas de puesta en marcha	25	40,00	1.000,00
Subtotal Ingeniería	**185**		**7.400,00 €**

Tabla 4: Desglose de costes de ingeniería y mano de obra

7.5. Presupuesto total

La tabla final expone los gastos directos del proyecto. A la suma de los subtotales se le ha añadido un margen de contingencia del 10 % para absorber posibles imprevistos operativos propios de las fases de prototipado, como la rotura de componentes sensibles (por ejemplo, la fragilidad de los cables de la celda de carga experimentada durante el montaje) o desviaciones en las horas de integración del software.

Partida	**Importe (€)**
1. Hardware y Materiales	15.664,28
2. Licencias de Software	6.272,26
3. Ingeniería y Mano de Obra	7.400,00
Base Imponible	**29.336,54**
Margen de Contingencia (10 %)	2.933,65
TOTAL PROYECTO (Sin impuestos)	**32.270,19 €**

Tabla 5: Resumen general del presupuesto del proyecto

Capítulo 8: Conclusiones

El desarrollo e implementación de este proyecto ha demostrado la viabilidad técnica y operativa de automatizar integralmente una línea de envasado de cremas, logrando unificar hardware de prototipado accesible con estándares exigentes de la industria actual. Tras superar las fases de diseño, montaje, programación y validación, se extraen las siguientes conclusiones fundamentales del proyecto.

La integración del microcontrolador ESP32 en el entorno de MATLAB/Simulink se ha consolidado como una solución óptima para el control dinámico del pesaje de la crema, superando la rigidez de los PLC tradicionales en el manejo de señales.
La implementación de un controlador PI (Proporcional-Integral) ha garantizado una dosificación precisa, prescindiendo de la acción derivativa ya que evita la amplificación del ruido a la celda de carga y aumenta la vida útil de los actuadores mecánicos al evitar las oscilaciones.

La aplicación del transistor MOSFET (IRLZ44N) ha permitido modular el caudal a través de señales PWM de la bomba peristáltica sin experimentar fatiga mecánica ni sobrecalentamiento.
La aplicación de factores de reducción y la protección inductiva mediante el diodo freewheel dan lugar a un circuito robusto, que aísla de perturbaciones transitorias al microprocesador.

La combinación de la estructura lógica secuencial a través de GRAFCET y la jerarquía de estados de la Guía GEMMA, ha permitido que la programación mediante el PLC de TIA Portal sea completamente fiable y determinista para sincronizar las cintas transportadoras en paralelo.
De esta manera se ha conseguido maximizar la producción en el mínimo tiempo, solapando los tiempos de dosificación, paletizado y asegurando una respuesta segura y predecible.

La creación del gemelo digital del brazo manipulador ABB permitió validar las trayectorias libres de colisiones, depurar el código RAPID y afinar el enrutamiento de las entradas y salidas lógicas sin poner en riesgo el hardware físico.

En el ámbito económico, el proyecto presenta como principal fortaleza el bajo coste de los componentes empleados en la fase de prototipado del control de llenado.
El modelo desarrollado es escalable a nivel industrial mediante la integración de un único PLC, el cual dispone de la capacidad para gestionar de forma centralizada las múltiples entradas y salidas de toda la línea de producción.
En contraste, la principal debilidad del sistema radica en en el coste inicial del brazo robótico.

En definitiva, este trabajo no solo resuelve el reto específico del llenado de precisión en el sector cosmético garantizando la higiene del producto, sino que materializa un flujo de ingeniería completo desde el tratamiento de señales a bajo nivel y el diseño de la placa de circuito impreso, hasta la programación lógica de alto nivel y la integración robótica.

Capítulo 9: Líneas Futuras

El desarrollo de este trabajo ha permitido validad la eficacia del control de llenado mediante el microcontrolador ESP32, el robot y el PLC. Con el objetivo de escalar el prototipo de producción industrial y que se adapte a la Industria 4.0 se proponen las siguientes líneas de trabajo:

9.1. Monitorización remota

Para dotar al sistema de mayor flexibilidad, se plantea utilizar la Wi-Fi 2,4GHz nativa del microcontrolador y enlazarla a la red local de la empresa para poder acceder a los datos de pesaje, las curvas de llenado y las alertas del sistema de producción y de esta manera poder ser supervisados en tiempo real desde cualquier equipo autorizado, eliminando la necesidad de acceder presencialmente a la línea de envasado.
Para asegurar la ligereza y fiabilidad de esta comunicación, se integrará el protocolo MQTT, el estándar principal para telemetría en el Internet de las Cosas (IoT) industrial.

9.2. Unificación de la fuente de alimentación

La maqueta actualmente cuenta con una arquitectura dual que alimenta a la bomba y al microcontrolador de manera separada, y aunque es un buen sistema para evitar ruidos producidos por la electricidad se puede desarrollar un módulo que sea capaz de alimentarlos con ambos voltajes eliminando la necesidad de uno de los cables.
Para conseguirlo se debe integrar un conversor DC-DC de topología reductora para convertir 12V a 5V.

9.3. Fabricación del circuito impreso (PCB)

El diseño de las pistas eléctricas y las huellas ya ha sido modelado teóricamente en el software, el siguiente paso es ordenar la fabricación física de la placa. La impresión industrial eliminará las resistencias parásitas generadas por el cableado manual que actualmente desestabilizan las lecturas del puente de Wheatstone de la báscula.
Adicionalmente, se diseñará un encapsulado protector para la PCB que aísle los componentes críticos de potencia, como el transistor MOSFET IRLZ44N, frente a las partículas y la humedad habituales en una planta cosmética.

9.4. Cuadro de mandos

Aprovechando el flujo de datos inalámbrico vía MQTT, se propone el desarrollo de un panel de control interactivo utilizando herramientas de código abierto como Node-RED o Grafana. Esta interfaz proporcionará a los operarios gráficas históricas de rendimiento (OEE) y permitirá programar de forma predictiva las labores de mantenimiento mecánico en el tubo de la bomba peristáltica.

9.5. Visión artificial

Para mejorar los resultados del sistema se propone implementar una cámara de visión en el brazo robótico que permita recoger los botes correctamente independientemente

de que se estén moviendo por la cinta y verificar el correcto cierre de las tapas.

Referencias

ADEPA (1980), *GEMMA: Guide d'étude des modes de marches et d'arrêts*, Agence Nationale pour le Développement de la Productique Appliquée à l'Industrie, Montrouge, Francia.

Angani, A. et al. (2021), Design and implementation of pid controller using matlab and arduino for automation applications, *in* 'Smart Innovation, Systems and Technologies', Springer.

Arduino LLC (2023), 'Arduino documentation'.
URL: *https://docs.arduino.cc/*

Bahr, B. et al. (2020), *Automation and Robotics in the Food and Beverage Industry*, Springer Nature.

Banks, A., Gupta, R., Larson, E. & Bober, C. (2019), Mqtt version 5.0, Technical report, OASIS Standard.

Bi, Z. & Zhang, W. J. (2023), *Flexible Automation and Intelligent Manufacturing: The Role of Robotics*, Springer.

Bolton, W. (2013), *Mecatrónica: Sistemas de control electrónico en la ingeniería mecánica y eléctrica*, 5 edn, Alfaomega.

Bolton, W. (2015), *Programmable Logic Controllers*, Elsevier / Springer Engineering Reference.

David, R. & Alla, H. (1992), 'Grafcet: A powerful tool for specification of logic controllers', *IEEE Transactions on Control Systems Technology* **3**(3), 253–265.

Duka, B. (2014), Simulation of industrial robots using robotstudio, *in* 'New Technologies, Development and Application', Springer.

Fraile Mora, J. (2012), *Electromagnetismo y circuitos eléctricos*, 6 edn, McGraw-Hill.

García-García, J. et al. (2020), 'Evaluation of low-cost weighing systems for industrial automation', *Journal of Sensors* .

Gasparetto, A. et al. (2015), Trajectory planning in robotics, *in* 'Mathematics in Computer Science', Springer.

Hassan, M. (2022), 'Low-cost automated filling machine using arduino and load cell', *International Journal of Engineering Research & Technology (IJERT)* .

Katz, J. S. (2021), *Control Engineering with MATLAB and Arduino*, Springer Nature.

Kunt, M. A. (2017), *Control Systems for Peristaltic Pumps in Biotechnology Applications*, Springer.

Kunt, M. A. & Oguz, A. E. (2017), Design and control of a peristaltic pump for precise liquid dispensing, *in* 'Advances in Engineering Design', Springer.

López, J. et al. (2020), *Industrial Automation and Control: GEMMA Methodology*, Springer.

Mainali, K. et al. (2022), A comparative study of plc and microcontroller based systems for lab-scale automation, *in* 'Advances in Engineering Design', Springer.

Mandado Pérez, E., Marcos Acevedo, J., Fernández Silva, C. & Pérez López, J. I. (2009), *Autómatas programables y sistemas de automatización*, Marcombo, Barcelona, España.

MathWorks, Inc. (2023), *MATLAB Documentation*, Natick, Massachusetts, United States.

Nof, S. Y., ed. (2009), *Springer Handbook of Automation*, Springer Science & Business Media.

Ogata, K. (2010), *Ingeniería de control moderna*, 5 edn, Pearson Educación.

Pires, J. N. (2007), *Industrial Robotics: Programming, Simulation and Applications*, Springer.

Ramesh, T. et al. (2019), Implementation of dual-stage filling algorithm for industrial applications, *in* 'Innovative Product Design and Intelligent Manufacturing Systems', Springer.

Reyes Cortés, F. (2011), *Robótica: control de robots manipuladores*, Alfaomega, México D.F., México.

Sedra, A. S. & Smith, K. C. (2014), *Microelectronic Circuits*, 7 edn, Oxford University Press.

Serpanos, D. & Wolf, M. (2017), *Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies*, Springer International Publishing.

Setyawan, R. et al. (2019), 'Weight-based liquid filling system using matlab and arduino', *Journal of Physics: Conference Series* .

Siciliano, B. & Khatib, O., eds (2016), *Springer Handbook of Robotics*, Springer.

Stanford-Clark, A. & Nipper, A. (1999), Mqtt v3.1 protocol specification, Technical report, IBM Corporation.

UNE (2013), Lenguaje de especificación grafcet para diagramas funcionales de secuencia, Norma UNE-EN 60848:2013, Asociación Española de Normalización y Certificación (AENOR), Madrid, España.

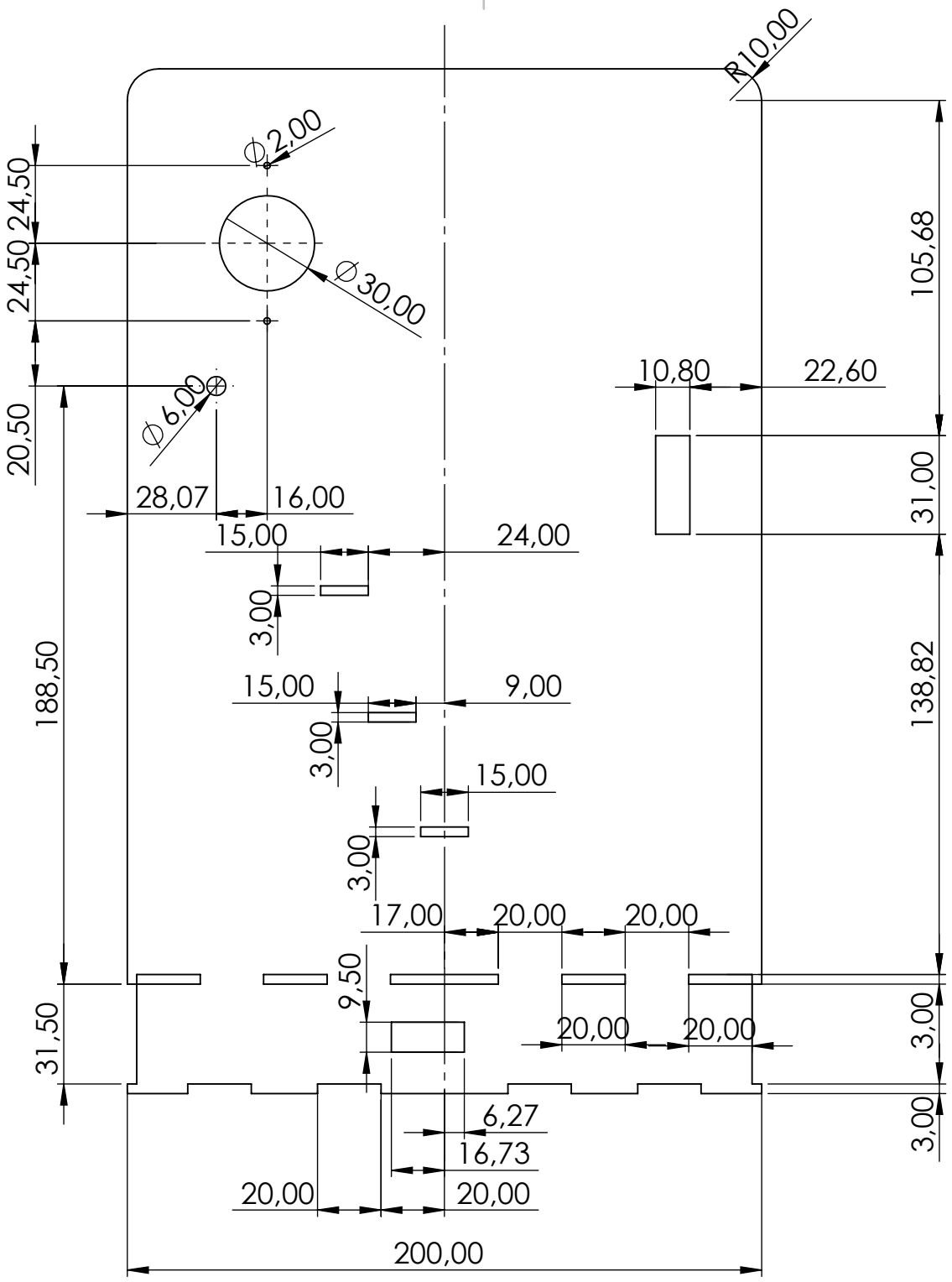
Wang, Y. et al. (2018), 'Analysis of the influence of peristaltic pump speed control on filling repeatability', *Journal of Dynamic Systems, Measurement, and Control* .

Índice de Anexos

Capítulo A:Planos del montaje **77**
A.1 Pieza Vertical . . . 78
A.2 Pieza Base . . . 79
A.3 Pieza LatGrande . . . 80
A.4 Pieza Lat1 . . . 81
A.5 Pieza Lat2 . . . 82
A.6 Pieza Soporte . . . 83
A.7 Pieza SoporteCorto . . . 84
A.8 Pieza SoporteMasCorto . . . 85
A.9 Pieza Tapa . . . 86
A.10 Ensamblaje . . . 87
Capítulo B:Diseño de PCB en EasyEDA **88**
B.1 Esquema eléctrico . . . 89
B.2 Diseño de PCB . . . 90
Capítulo C:Documentación de programación de TIA PORTAL **91**
C.1 Documentacion TIAPORTAL . . . 92
Capítulo D:Esquemas GRAFCETS de la programación del PLC **118**
Capítulo E:Código RAPID del robot **122**
E.1 Código RAPID Robot . . . 123
Capítulo F:Datasheet de MOSFET Canal N IRLZ44N **125**
Capítulo G:Datasheet de diodo **125**
Capítulo H:Sensor de la célula de carga de 1kg **125**
Capítulo I: Bomba peristáltica **125**
Capítulo J: Datasheet de ESP32 **125**
Capítulo K:Datasheet de HX711 **125**
Capítulo L:Diodo Flyback **125**
Capítulo M:Estudio económico **125**
Capítulo N:Archivos del proyecto del PLC **125**
Capítulo Ñ:Archivos del proyecto del robot **125**

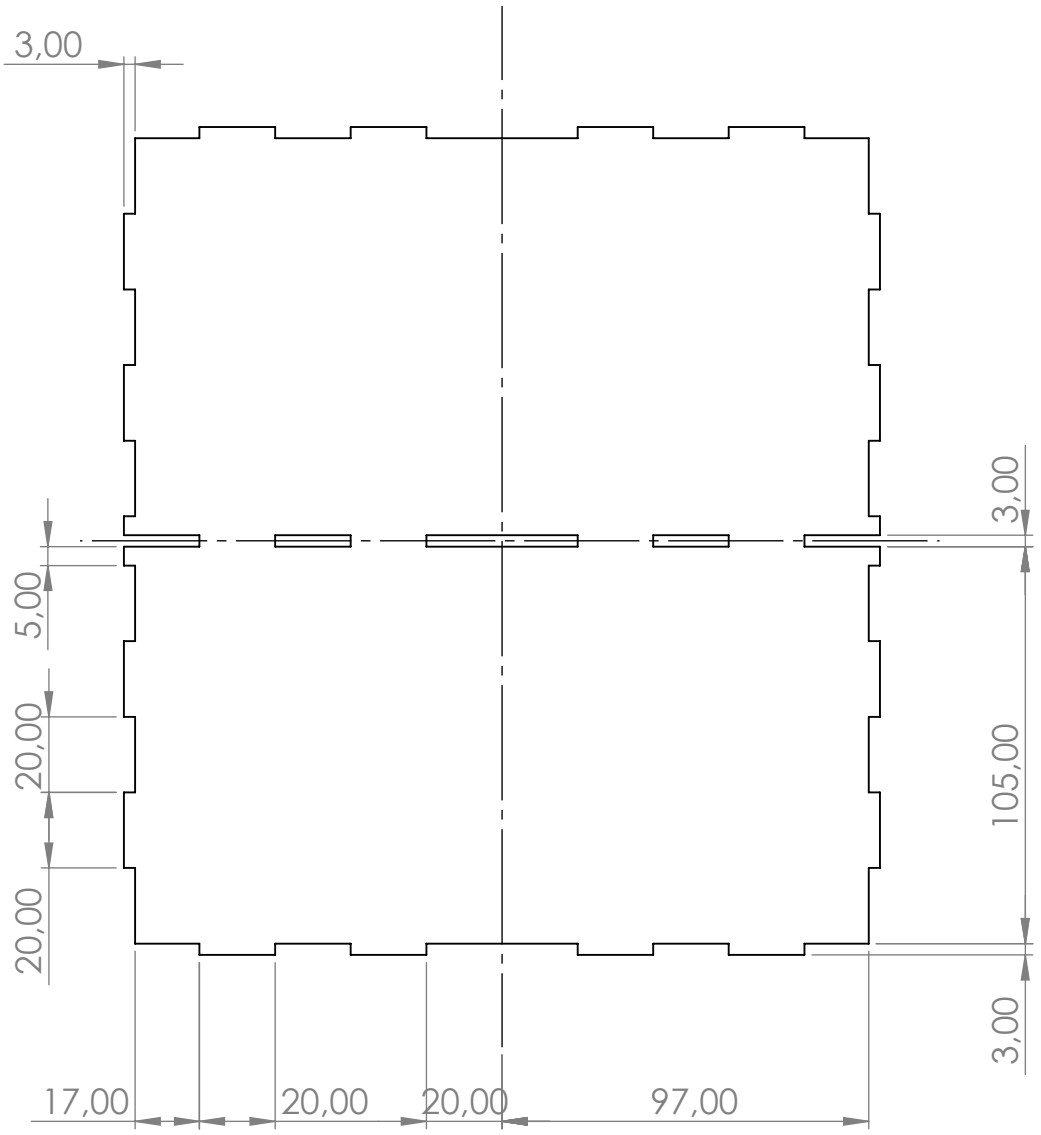
Capítulo A: Planos del montaje

A continuación se adjuntan los planos mecánicos correspondientes a las piezas y al ensamblaje final.



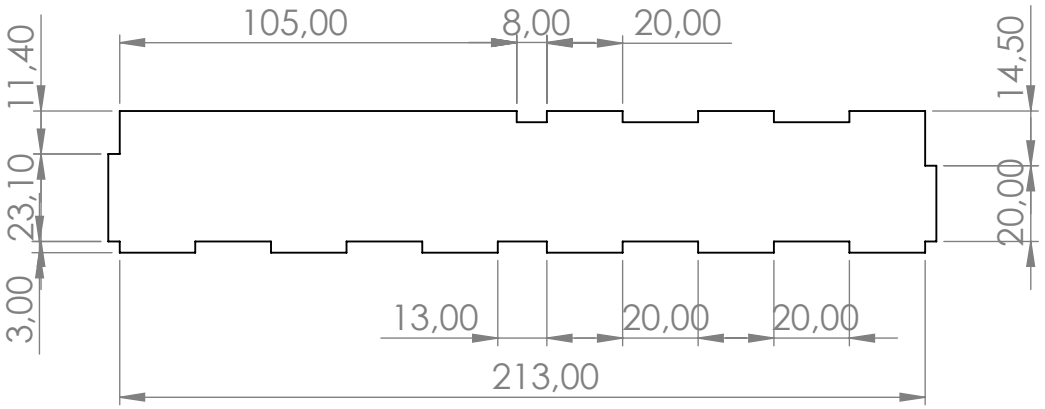
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA		REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.	Ester López		6/2/26				Soporte
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Vertical	
				PESO:		ESCALA:1:2	HOJA 1 DE 9

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	Ester López		6/2/26			Soporte	
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Base	
				PESO:		ESCALA:1:2	HOJA 2 DE 9

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CALID.			

TÍTULO:

MATERIAL:

N.º DE DIBUJO

Lat_Grande

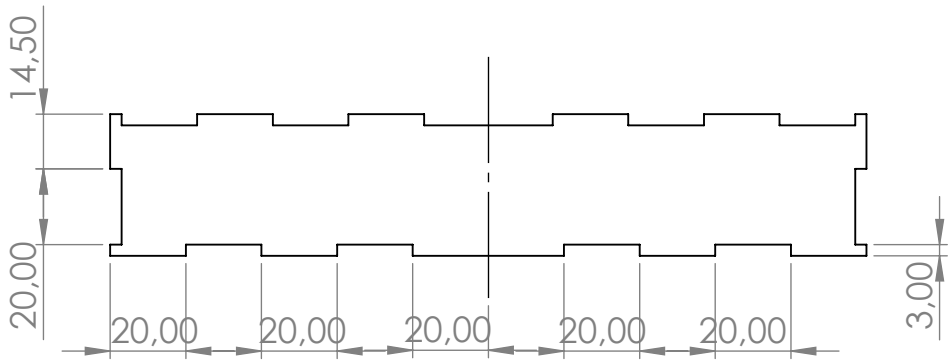
A4

PESO:

ESCALA:1:2

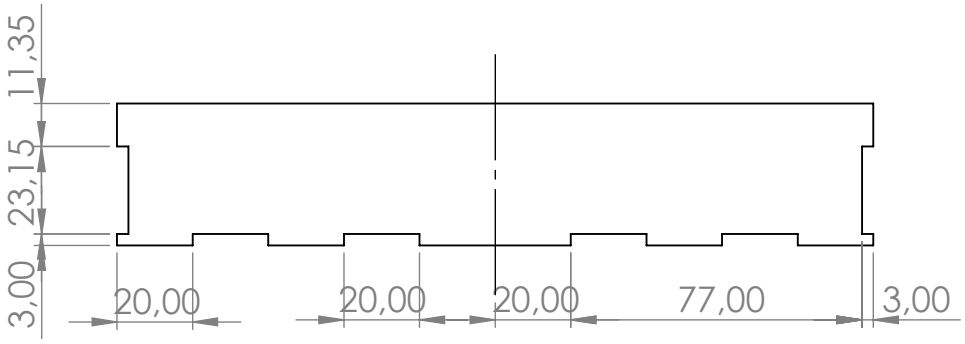
HOJA 1 DE 1

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



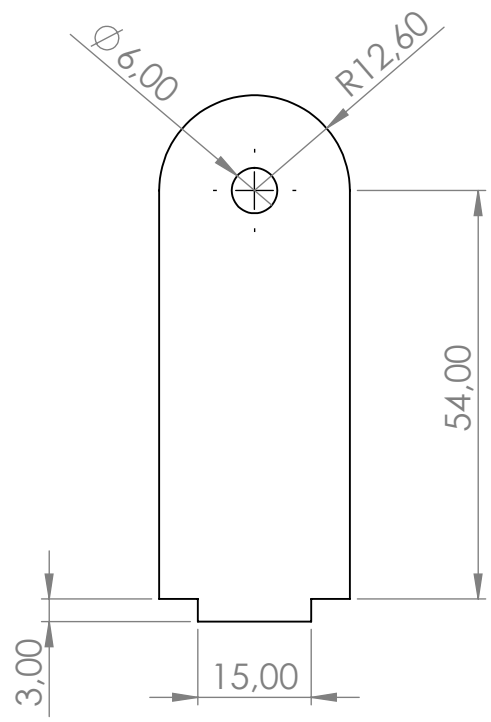
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	Ester López		6/2/26			Soporte	
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Lat1	
				PESO:		ESCALA:1:2	HOJA 4 DE 9

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



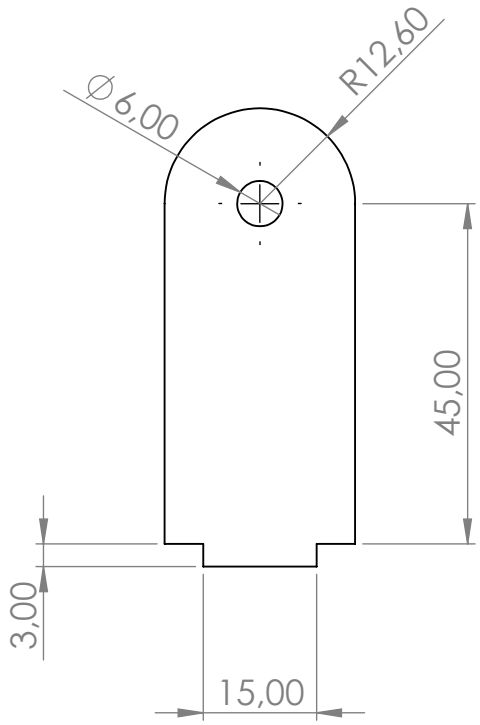
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO:	
DIBUJ.	Ester López		6/2/26				Soporte	
VERIF.								
APROB.								
FABR.								
CALID.				MATERIAL:			N.º DE DIBUJO	A4
							Lat2	
				PESO:			ESCALA:1:2	HOJA 5 DE 9

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



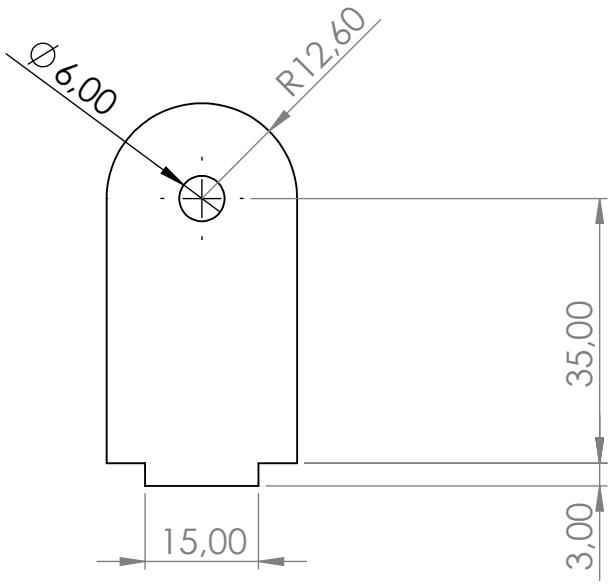
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA		REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO: Soporte
DIBUJ.	Ester López		6/2/26				
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:			N.º DE DIBUJO Soporte
							A4
				PESO:		ESCALA:1:1	HOJA 6 DE 9

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



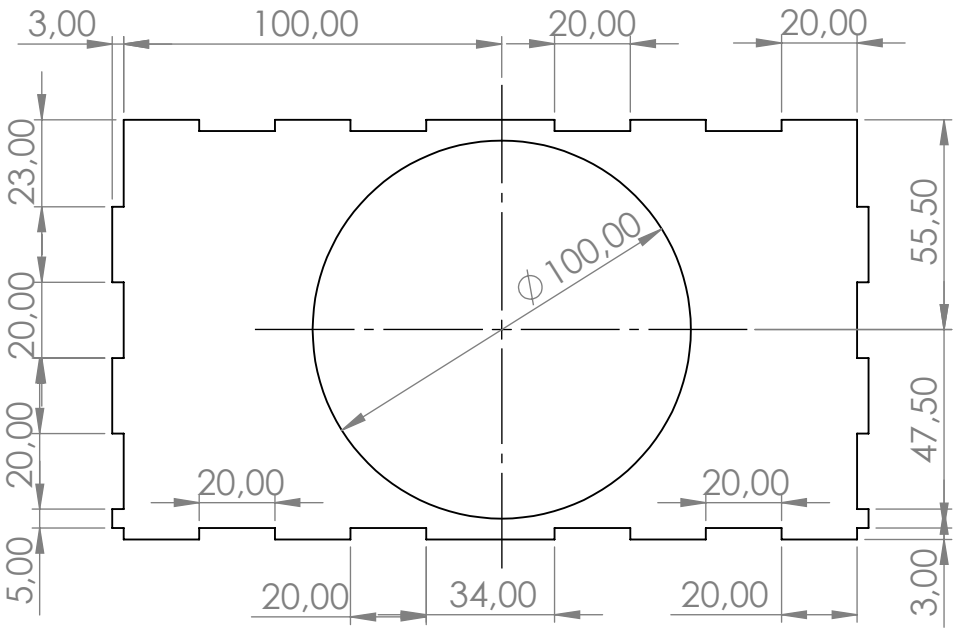
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA				TÍTULO: Soporte
DIBUJ.	Ester López		6/2/26				
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO SoporteCorto	A4
				PESO:		ESCALA:1:1	HOJA 7 DE 9

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



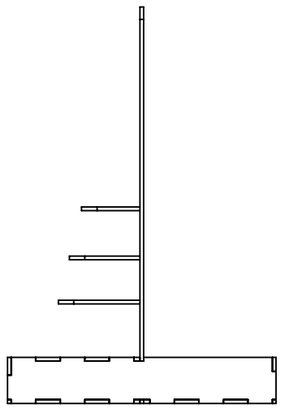
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA				TÍTULO: Soporte
DIBUJ.	Ester López		6/2/26				
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:			N.º DE DIBUJO SoporteMasCorto A4
				PESO:		ESCALA:1:1	HOJA 8 DE 9

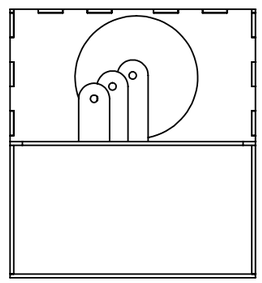
Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

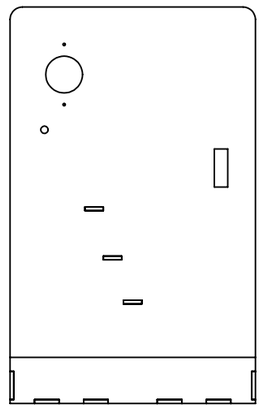


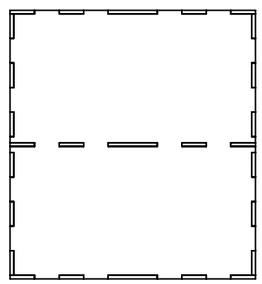
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA		REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.	Ester López		6/2/26				Soporte
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Tapa	
				PESO:		ESCALA:1:2	HOJA 9 DE 9

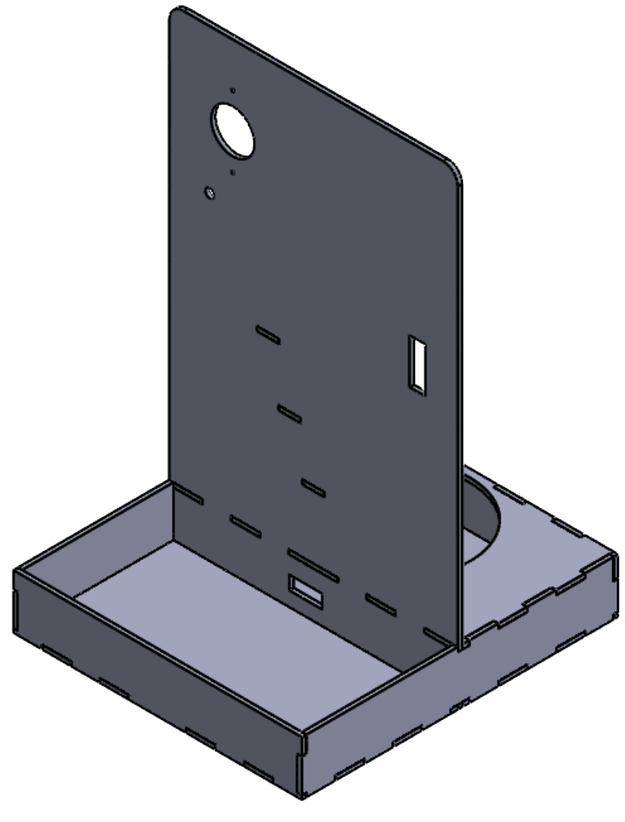
Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.











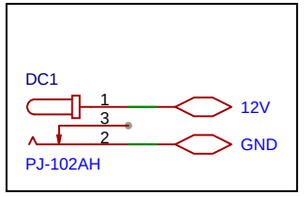
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:				REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO: Soporte Completo
DIBUJ.	Ester López		6/2/26				
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:			N.º DE DIBUJO Ensamblaje / A3
				PESO:		ESCALA:1:5	HOJA 1 DE 1

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

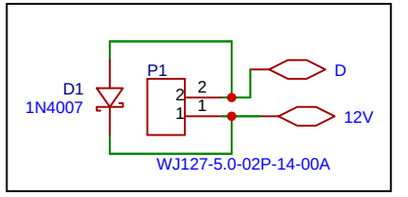
Capítulo B: Diseño de PCB en EasyEDA

En este apartado se incluyen los esquemas eléctricos y el diseño del circuito impreso desarrollados mediante el software de diseño electrónico EasyEDA.

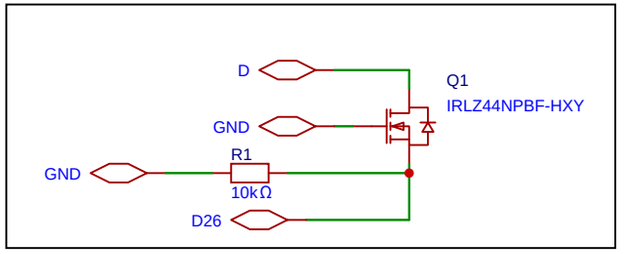
ALIMENTACIÓN



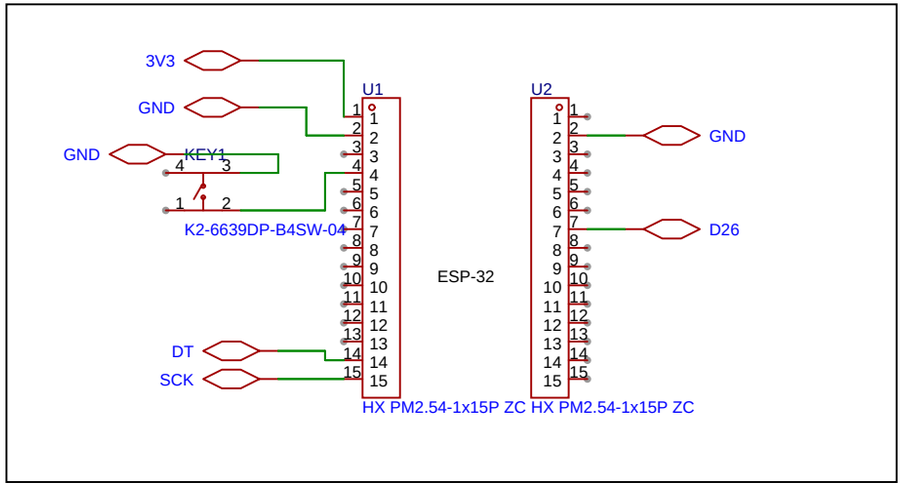
MOTOR



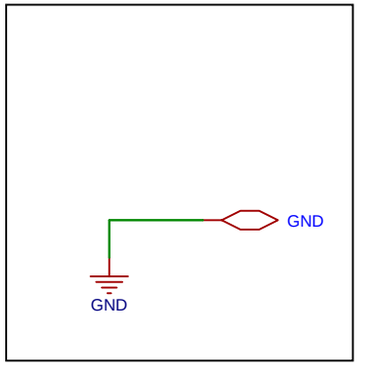
MOSFET



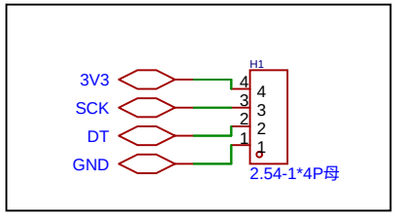
MICROCONTROLADOR - ESP32



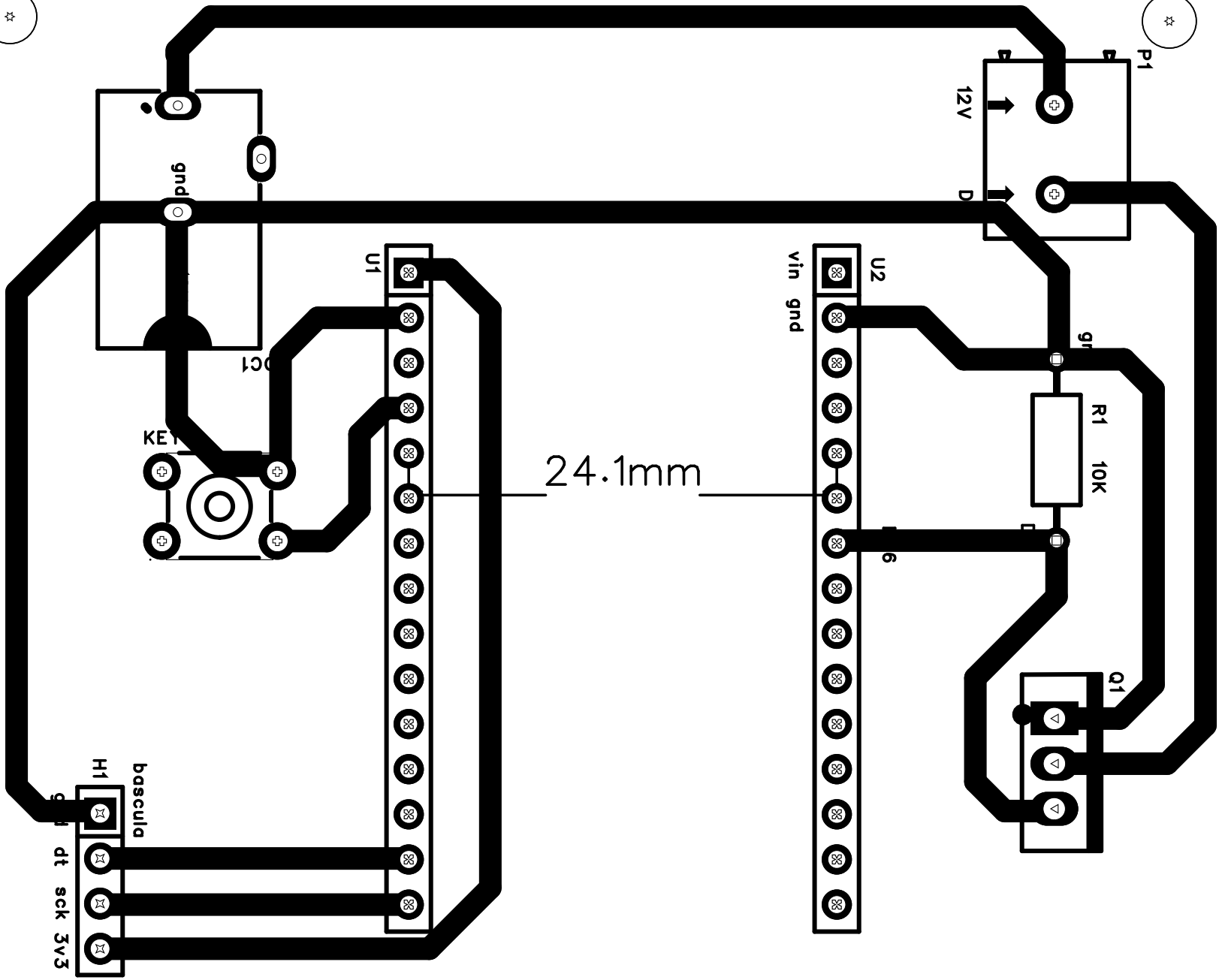
CONEXIONES



BÁSCULA



Schematic	Schematic1	Create at	2026-04-15
		Update at	2026-05-10
Board	Board1	Page	P1
Drawn		PCB_TFG	
Reviewed			
AUTHOR	ESTER LÓPEZ HIERRO		
		Version / Size	Page 1 Total 1
EasyEDA		V1.0 / A4	EasyEDA.com



ESTER LÓPEZ HIERRO TFGv3 04/2026

CONTROL DE LLENADO UEM

Calculadora de ancho de pistas según IPC-2221 (DigiKey)

Capítulo C: Documentación de programación de TIA PORTAL

A continuación se adjunta una versión simplificada del informe técnico generado por el entorno TIA Portal. Se han seleccionado los bloques de las funciones, las variables críticas y la configuración de hardware principal del PLC.

TFG_V6(02042026)

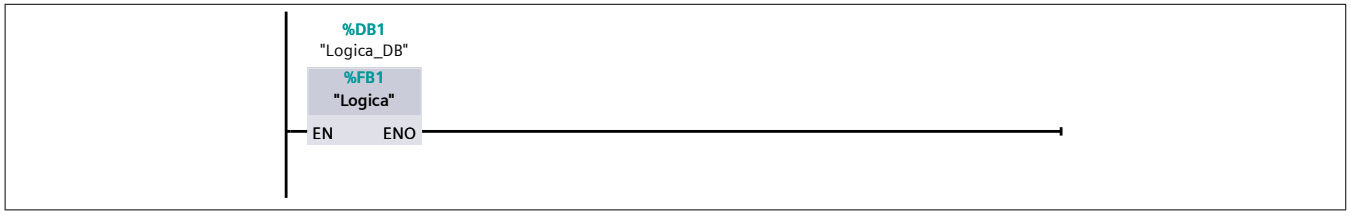
TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

Main [OB1]

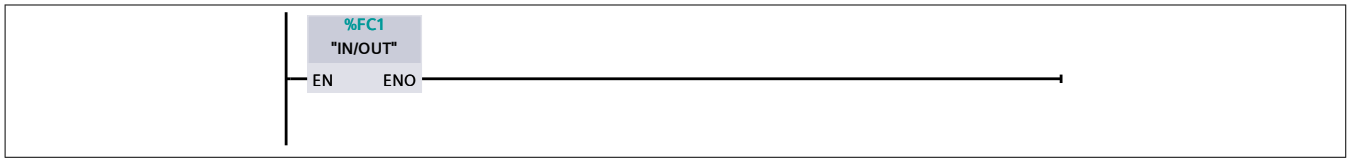
Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	"Main Program Sweep (Cycle)"	**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Remanence	Bool	
Temp		
Constant		

Network 1:



Network 2:



TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

Logica [FB1]

Logica Properties					
General					
Name	Logica	**Number**	1	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Pos_Botella_1	Int	0	Non-retain
Pos_Botella_2	Int	0	Non-retain
F1_Automatico_Instance	"F1_Automatico"		
F5_Semiautomatico_Instance	"F5_Semiautomatico"		
F4_Manual_Instance	"F4_Manual"		
A6_Homing_Instance	"A6_Homing"		
D1_Emergencia_Instance	"D1_Emergencia"		
A2_Paro_Instance	"A2_Paro"		
Temp			
Constant			

```
0001 REGION Datos
0002   //Desarrollo: Ester López Hierro
0003   //Fecha: 04/04/2026
0004   ////Módulo: Lógica
0005
0006 END_REGION
0007
0008 REGION Etapas GG
0009   CASE "Etapa_GG" OF
0010     0: //Estado A1: Parada en el estado inicial
0011       //A1 -> marcha + automatico -> Estado F1 automatico
0012       IF "M_Boton_Marcha" AND "M_Auto" THEN
0013         "Etapa_GG" := 1;
0014         //A1 -> marcha + no automatico -> F5 semiautomatico
0015       ELSIF "M_Boton_Marcha" AND NOT "M_Auto" THEN
0016         "Etapa_GG" := 2;
0017         //A1 -> no automatico + manualF4 -> F5 semiautomatico
0018       ELSIF "M_ManualF4" AND NOT "M_Auto" THEN
0019         "Etapa_GG" := 4;
0020         //A1 ->emergencia -> D1 emergencia
0021       ELSIF NOT "M_Emergenica_OK" THEN
0022         "Etapa_GG" := 3;
0023       END_IF;
0024
0025     1: //Estado F1: Produccion normal
0026       //F1 -> not automatico -> F5 semiautomatico
0027       IF NOT "M_Auto" THEN
```

```
0028          "Etapa_GG" := 2;
0029          //F1 -> manualF4 + no automatico -> F5 semiautomatico
0030        ELSIF "M_ManualF4" AND NOT "M_Auto" THEN
0031          "Etapa_GG" := 4;
0032          //F1 -> paro (logica negativa) -> F5 semiautomatico
0033        ELSIF NOT "M_Boton_Paro" THEN
0034          "Etapa_GG" := 7;
0035          //F1 ->emergencia -> D1 emergencia
0036        ELSIF NOT "M_Emergenica_OK" THEN
0037          "Etapa_GG" := 3;
0038        END_IF;
0039
0040
0041      2: //Estado F1: Semiautomatico
0042        //F5 -> automatico -> F1 automatico
0043        IF "M_Auto" THEN
0044          "Etapa_GG" := 1;
0045          //F5 -> manualF4 -> F4 manual
0046        ELSIF "M_ManualF4" THEN
0047          "Etapa_GG" := 4;
0048          //F5 -> emergencia -> D1 emergencia
0049        ELSIF NOT "M_Emergenica_OK" THEN
0050          "Etapa_GG" := 3;
0051        END_IF;
0052      3: //D1 Emergencia
0053        //D1 -> la seta noesta pulsada -> A5 preparacion posterior al defecto
0054        IF "M_Emergenica_OK" THEN
0055          "Etapa_GG" := 5;
0056        END_IF;
0057      4: //F4 Marcha de verificaicon con orden
0058        //F4 -> manualF4 -> A6 homing
0059        IF NOT "M_ManualF4" THEN
0060          "Etapa_GG" := 6;
0061          //F4 -> emergencia -> D1 Emergencia
0062        ELSIF NOT "M_Emergenica_OK" THEN
0063            "Etapa_GG" := 3;
0064          END_IF;
0065
0066      5: //A5 Preparacion posterior al defecto
0067        //A5 -> rearme -> A6 homing
0068        IF "M_Rearme" THEN
0069          "Etapa_GG" := 6;
0070        //A5 -> emergencia -> D1 Emergencia
0071        ELSIF NOT "M_Emergenica_OK" THEN
0072          "Etapa_GG" := 3;
0073        END_IF;
0074      6: //A5 Preparacion posterior al defecto
0075        //A5 -> cambio de estado -> A0 inicio
0076        IF "M_CambioDeEstado" THEN
0077          "Etapa_GG" := 0;
0078        //A5 -> emergencia -> D1 Emergencia
0079        ELSIF NOT "M_Emergenica_OK" THEN
0080          "Etapa_GG" := 3;
0081        END_IF;
0082      7: //A2 Parada fin de ciclo
0083        //A2 -> cambio de estado -> A0 inicio
0084        IF "M_CambioDeEstado" THEN
0085          "Etapa_GG" := 0;
0086        //A2 -> emergencia -> D1 Emergencia
```

```
0087        ELSIF NOT "M_Emergenica_OK" THEN
0088          "Etapa_GG" := 3;
0089        END_IF;
0090      END_CASE;
0091  END_REGION
0092
0093  REGION Acciones GG
0094  //A1 Inicializacion
0095    IF "Etapa_GG" =0 THEN
0096      "M_CambioDeEstado" := FALSE;
0097      "Etapa_GG" := 0;
0098    END_IF;
0099  //F1 Automatico
0100    IF "Etapa_GG" = 1 THEN
0101      #F1_Automatico_Instance(Marcha:="M_Boton_Marcha",
0102                  Receta_Actual:="HMI_Receta",
0103                  Configuracion:="HMI_Configuracion",
0104                  S1:="M_S1",
0105                  S2:="M_S2",
0106                  lleno:="M_IN_lleno",
0107                  fin_robot:="M_IN_fin_robot",
0108                  cinta1=>"M_Cinta1",
0109                  cinta2=>"M_Cinta2",
0110                  llenado=>"M_Out_llenado",
0111                  robot=>"M_Out_robot",
0112                  Out_Receta_Llenado=>"M_Out_Receta_Llenado",
0113                  Etapa:="M_Etapa_Grafcet",
0114                  Etapa_Rama1:="M_Etapa_Grafcet_1",
0115                  Etapa_Rama2:="M_Etapa_Grafcet_2",
0116                  cont:="M_N_Cilcos");
0117
0118    END_IF;
0119  //F5 Semiautomatico
0120    IF "Etapa_GG" = 2 THEN
0121      #F5_Semiautomatico_Instance(Marcha:="M_Boton_Marcha",
0122                    Receta_Actual:="HMI_Receta",
0123                    Configuracion:="HMI_Configuracion",
0124                    S1:="M_S1",
0125                    S2:="M_S2",
0126                    lleno:="M_IN_lleno",
0127                    fin_robot:="M_IN_fin_robot",
0128                    cinta1=>"M_Cinta1",
0129                    cinta2=>"M_Cinta2",
0130                    llenado=>"M_Out_llenado",
0131                    robot=>"M_Out_robot",
0132                    Out_Receta_Llenado=>"M_Out_Receta_Llenado",
0133                    Etapa:="M_Etapa_Grafcet",
0134                    Etapa_Rama1:="M_Etapa_Grafcet_1",
0135                    cont:="M_N_Cilcos",
0136                    Etapa_Rama2:="M_Etapa_Grafcet_2");
0137
0138
0139    END_IF;
0140    //F4 Manual
0141    IF "Etapa_GG" = 4 THEN
0142      #F4_Manual_Instance(Btn_Man_Cinta1:="HMI_Man_Cinta1",
0143                Btn_Man_Cinta2:="HMI_Man_Cinta2",
0144                Btn_Man_Llenado:="M_IN_lleno",
0145                Btn_Man_Robot:="M_IN_fin_robot",
```

```
0146                cinta1=>"M_Cinta1",
0147                cinta2=>"M_Cinta2",
0148                llenado=>"M_Out_llenado",
0149                robot=>"M_Out_robot");
0150
0151
0152
0153
0154    END_IF;
0155  //A2 Paro
0156    IF "Etapa_GG" = 7 THEN
0157      #A2_Paro_Instance(Marcha:="M_Boton_Marcha",
0158                Receta_Actual:="HMI_Receta",
0159                Configuracion:="HMI_Configuracion",
0160                S1:="M_S1",
0161                S2:="M_S2",
0162                lleno:="M_IN_lleno",
0163                fin_robot:="M_IN_fin_robot",
0164                cinta1=>"M_Cinta1",
0165                cinta2=>"M_Cinta2",
0166                llenado=>"M_Out_llenado",
0167                robot=>"M_Out_robot",
0168                Out_Receta_Llenado=>"M_Out_Receta_Llenado",
0169                Cambio_Estado=>"M_CambioDeEstado",
0170                Etapa:="M_Etapa_Grafcet",
0171                Etapa_Rama1:="M_Etapa_Grafcet_1",
0172                Etapa_Rama2:="M_Etapa_Grafcet_2",
0173                cont:="M_N_Cilcos");
0174
0175
0176
0177
0178    END_IF;
0179  //A5 Preparacion posterior al defecto
0180    IF "Etapa_GG" = 5 THEN
0181      "M_Etapa_Grafcet" := 100;
0182      "M_Etapa_Grafcet_1" := 0;
0183      "M_Etapa_Grafcet_2" := 0;
0184
0185    END_IF;
0186  //A6 Homing
0187    IF "Etapa_GG" = 6 THEN
0188      #A6_Homing_Instance(Robot_Posicion_Inicial:="M_IN_fin_robot",
0189                 Mandar_Robot_Home=>"M_Robot_PosInical",
0190                 Etapa:="M_Etapa_Grafcet",
0191                 Etapa_Rama1:="M_Etapa_Grafcet_1",
0192                 Etapa_Rama2:="M_Etapa_Grafcet_2",
0193                 CambioEstado:="M_CambioDeEstado");
0194       "M_Auto" := 1;
0195
0196
0197    END_IF;
0198  //D1 Emergencia
0199    IF "Etapa_GG" = 3 THEN
0200      #D1_Emergencia_Instance();
0201
0202    END_IF;
0203  END_REGION
0204
```

```
0205 REGION Señalizacion
0206   // Statement section REGION
0207
0208 END_REGION
0209
0210 REGION HMI_Animaciones
0211
0212   // Cuando se genera la señal de salida, memorizamos que debe ser verde (SET)
0213   IF "M_Out_llenado" THEN
0214     "HMIColor_Tanque_Verde" := TRUE;
0215   END_IF;
0216
0217   // Cuando se genera la señal de entrada, borramos la memoria para que vuel-
      va a gris (RESET)
0218   IF "M_IN_lleno" THEN
0219     "HMIColor_Tanque_Verde" := FALSE;
0220   END_IF;
0221
0222 END_REGION
0223
0224
0225
0226
0227
0228
0229
0230
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

F1_Automatico [FB2]

F1_Automatico Properties					
General					
Name	F1_Automatico	**Number**	2	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Marcha	Bool	false	Non-retain
Receta_Actual	Int	0	Non-retain
Configuracion	Bool	false	Non-retain
S1	Bool	false	Non-retain
S2	Bool	false	Non-retain
lleno	Bool	false	Non-retain
fin_robot	Bool	false	Non-retain
▼ Output			
cinta1	Bool	false	Non-retain
cinta2	Bool	false	Non-retain
llenado	Bool	false	Non-retain
robot	Bool	false	Non-retain
Out_Receta_Llenado	Int	0	Non-retain
▼ InOut			
Etapa	Int	0	Non-retain
Etapa_Rama1	Int	0	Non-retain
Etapa_Rama2	Int	0	Non-retain
cont	Int	0	Non-retain
▼ Static			
cont_global	Int	0	Non-retain
M_receta	Int	0	Non-retain
Pulso_Etapa5	Bool	false	Non-retain
Pulso_S1	Bool	false	Non-retain
Pulso_S2	Bool	false	Non-retain
Memoria_Etapa5	Bool	false	Non-retain
Memoria_S1	Bool	false	Non-retain
Memoria_S2	Bool	false	Non-retain
Pulso_Clock	Bool	false	Non-retain
Memoria_Clock	Bool	false	Non-retain
Temp			
Constant			

```
0001 REGION Datos
0002   //Desarrollo: Ester López Hierro
0003   //Fecha: 04/04/2026
0004   ////Módulo: Lógica - F1 Producción normal
0005
0006 END_REGION
0007
```

```
0008
0009 REGION Flancos
0010   #Pulso_Etapa5 := #Etapa = 5 AND NOT #Memoria_Etapa5;
0011   #Memoria_Etapa5 := #Etapa = 5;
0012
0013   #Pulso_S1 := #S1 AND NOT #Memoria_S1;
0014   #Memoria_S1 := #S1;
0015
0016   #Pulso_S2 := #S2 AND NOT #Memoria_S2;
0017   #Memoria_S2 := #S2;
0018
0019 END_REGION
0020
0021 REGION Etapas
0022   CASE #Etapa OF
0023     0:
0024       IF  #Configuracion  THEN
0025         #Etapa := 1;
0026       END_IF;
0027
0028     1:
0029       IF #Marcha THEN
0030         #Etapa := 2;
0031       END_IF;
0032     2:
0033       IF #Pulso_S1 THEN
0034         #Etapa := 3;
0035       END_IF;
0036     3:
0037       IF #lleno THEN
0038         #Etapa := 4;
0039       END_IF;
0040     4: //Divergencia ANDS
0041       IF TRUE THEN
0042         #Etapa_Rama1 := 10;
0043         #Etapa_Rama2 := 20;
0044         #Etapa := 100;
0045       END_IF;
0046
0047     100: //Convergencia en AND
0048       IF (#Etapa_Rama1 = 12) AND (#Etapa_Rama2 = 22) THEN
0049         #Etapa := 5;
0050       END_IF;
0051
0052     5:
0053       #Etapa := 4;
0054
0055   END_CASE;
0056   // --- RAMA 1 ---
0057   CASE #Etapa_Rama1 OF
0058     10:
0059       IF #Pulso_S1 THEN
0060         #Etapa_Rama1 := 11;
0061       END_IF;
0062     11:
0063       IF #lleno THEN
0064         #Etapa_Rama1 := 12;
0065       END_IF;
0066   END_CASE;
```

```
0067
0068    // --- RAMA 2 ---
0069    CASE #Etapa_Rama2 OF
0070      20:
0071        IF #Pulso_S2 THEN
0072          #Etapa_Rama2 := 21;
0073        END_IF;
0074      21:
0075        IF #fin_robot THEN
0076          #Etapa_Rama2 := 22;
0077        END_IF;
0078    END_CASE;
0079
0080  END_REGION
0081  REGION Acciones
0082    IF #Etapa = 1 THEN
0083      #cont := 0;
0084      #Etapa_Rama1 := 0;
0085      #Etapa_Rama2 := 0;
0086
0087    END_IF;
0088
0089    IF #Etapa = 2 THEN
0090      #M_receta := #Receta_Actual;
0091    END_IF;
0092
0093    IF #Pulso_Etapa5 THEN
0094      #cont := #cont + 1;
0095      #cont_global := #cont_global + 1;
0096    END_IF;
0097
0098    IF (#Etapa = 3) OR (#Etapa_Rama1 = 11) OR (#Etapa_Rama2 = 21) THEN
0099      #Out_Receta_Llenado := #M_receta;
0100    END_IF;
0101
0102    #cinta1 := (#Etapa = 2) OR (#Etapa_Rama1 = 10);
0103    #llenado := (#Etapa = 3) OR (#Etapa_Rama1 = 11);
0104    #cinta2 := (#Etapa_Rama2 = 20);
0105    #robot := (#Etapa_Rama2 = 21);
0106
0107  END_REGION
0108
0109  REGION Simulación_Movimiento
0110    #Pulso_Clock := "Clock_10Hz" AND NOT #Memoria_Clock;
0111    #Memoria_Clock := "Clock_10Hz";
0112      // Cinta 1
0113    #cinta1 := (#Etapa = 2) OR (#Etapa_Rama1 = 10);
0114    IF (#Etapa = 2 OR #Etapa_Rama1 = 10) AND #Pulso_Clock THEN
0115      "HMI_Pos_Botella1" := "HMI_Pos_Botella1" + 1;
0116    END_IF;
0117    "M_S1" := ("HMI_Pos_Botella1" >= 100);
0118    IF #Etapa_Rama1 = 12 OR #Etapa = 4 THEN
0119      "HMI_Pos_Botella1" := 0;
0120    END_IF;
0121     // cinta 2
0122    #cinta2 := (#Etapa_Rama2 = 20);
0123    IF (#Etapa_Rama2 = 20) AND #Pulso_Clock THEN
0124      "HMI_Pos_Botella2" := "HMI_Pos_Botella2" + 1;
0125    END_IF;
```

```
0126    "M_S2" := ("HMI_Pos_Botella2" >= 100);
0127    IF (#Etapa_Rama2 = 21) OR (#Etapa = 4) THEN
0128      "HMI_Pos_Botella2" := 0;
0129    END_IF;
0130
0131 END_REGION
0132
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

F5_Semiautomatico [FB3]

F5_Semiautomatico Properties					
General					
Name	F5_Semiautomatico	**Number**	3	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Marcha	Bool	false	Non-retain
Receta_Actual	Int	0	Non-retain
Configuracion	Bool	false	Non-retain
S1	Bool	false	Non-retain
S2	Bool	false	Non-retain
lleno	Bool	false	Non-retain
fin_robot	Bool	false	Non-retain
▼ Output			
cinta1	Bool	false	Non-retain
cinta2	Bool	false	Non-retain
llenado	Bool	false	Non-retain
robot	Bool	false	Non-retain
Out_Receta_Llenado	Int	0	Non-retain
▼ InOut			
Etapa	Int	0	Non-retain
Etapa_Rama1	Int	0	Non-retain
cont	Int	0	Non-retain
Etapa_Rama2	Int	0	Non-retain
▼ Static			
cont_global	Int	0	Non-retain
M_receta	Int	0	Non-retain
Memoria_Marcha	Bool	false	Non-retain
Pulso_Marcha	Bool	false	Non-retain
Pulso_Etapa7	Bool	false	Non-retain
Memoria_Etapa7	Bool	false	Non-retain
Pulso_Clock	Bool	false	Non-retain
Memoria_Clock	Bool	false	Non-retain
Pulso_S1	Bool	false	Non-retain
Pulso_S2	Bool	false	Non-retain
Memoria_S1	Bool	false	Non-retain
Memoria_S2	Bool	false	Non-retain
Temp			
Constant			

```
0001 REGION Datos
0002    //Desarrollo: Ester López Hierro
0003    //Fecha: 04/04/2026
0004    ////Módulo: Lógica - F5 Marcha de verificacion con orden
```

```
0005
0006 END_REGION
0007 REGION Flancos
0008   #Pulso_Etapa7 := #Etapa = 7 AND NOT #Memoria_Etapa7;
0009   #Memoria_Etapa7 := #Etapa = 7;
0010
0011   #Pulso_Marcha := #Marcha AND NOT #Memoria_Marcha;
0012   #Memoria_Marcha := #Marcha;
0013
0014   #Pulso_S1 := #S1 AND NOT #Memoria_S1;
0015   #Memoria_S1 := #S1;
0016
0017   #Pulso_S2 := #S2 AND NOT #Memoria_S2;
0018   #Memoria_S2 := #S2;
0019 END_REGION
0020
0021 REGION Etapas
0022   CASE #Etapa OF
0023     0:
0024       IF #Configuracion THEN
0025         #Etapa := 1;
0026       END_IF;
0027
0028     1:
0029       IF #Pulso_Marcha THEN
0030         #Etapa := 2;
0031       END_IF;
0032     2:
0033       IF #Pulso_S1 THEN
0034         #Etapa := 3;
0035       END_IF;
0036     3:
0037       IF #Pulso_Marcha THEN
0038         #Etapa := 4;
0039       END_IF;
0040     4:
0041       IF #lleno THEN
0042         #Etapa := 5;
0043       END_IF;
0044     5:
0045       IF #Pulso_Marcha THEN
0046         #Etapa := 6;
0047       END_IF;
0048
0049     6:
0050       IF TRUE THEN
0051         #Etapa_Rama1 := 10;
0052         #Etapa_Rama2 := 20;
0053         #Etapa := 100;
0054       END_IF;
0055
0056     100:
0057       IF (#Etapa_Rama1 = 13) AND (#Etapa_Rama2 = 23) AND #Pulso_Marcha THEN
0058         #Etapa := 5;
0059       END_IF;
0060
0061     7:
0062       IF #Pulso_Marcha THEN
0063         #Etapa := 5;
```

```
0064        END_IF;
0065    END_CASE;
0066    // --- RAMA 1 ---
0067    CASE #Etapa_Rama1 OF
0068      10:
0069        IF #Pulso_S1 THEN
0070          #Etapa_Rama1 := 11;
0071        END_IF;
0072      11:
0073        IF #Pulso_Marcha THEN
0074          #Etapa_Rama1 := 12;
0075        END_IF;
0076      12:
0077        IF #lleno THEN
0078          #Etapa_Rama1 := 13;
0079        END_IF;
0080    END_CASE;
0081
0082    // --- RAMA 2 ---
0083    CASE #Etapa_Rama2 OF
0084      20:
0085        IF #Pulso_S2 THEN
0086          #Etapa_Rama2 := 21;
0087        END_IF;
0088      21:
0089        IF #Pulso_Marcha THEN
0090          #Etapa_Rama2 := 22;
0091        END_IF;
0092      22:
0093        IF #fin_robot THEN
0094          #Etapa_Rama2 := 23;
0095        END_IF;
0096    END_CASE;
0097
0098 END_REGION
0099 REGION Acciones
0100
0101    IF #Etapa = 1 THEN
0102      #cont := 0;
0103      #Etapa_Rama1 := 0;
0104      #Etapa_Rama2 := 0;
0105    END_IF;
0106
0107    IF #Etapa = 2 THEN
0108      #M_receta := #Receta_Actual;
0109    END_IF;
0110
0111    IF #Pulso_Etapa7 THEN
0112      #cont := #cont + 1;
0113      #cont_global := #cont_global + 1;
0114    END_IF;
0115
0116
0117    #cinta1 := (#Etapa = 2) OR (#Etapa_Rama1 = 10);
0118    #llenado := (#Etapa = 4) OR (#Etapa_Rama1 = 12);
0119    #cinta2 := (#Etapa_Rama2 = 20);
0120    #robot := (#Etapa_Rama2 = 22);
0121    IF (#Etapa = 3) OR (#Etapa_Rama1 = 11) OR (#Etapa_Rama2 = 21) THEN
0122      #Out_Receta_Llenado := #M_receta;
```

```
0123    END_IF;
0124  END_REGION
0125
0126  REGION Simulación_Movimiento
0127    #Pulso_Clock := "Clock_10Hz" AND NOT #Memoria_Clock;
0128    #Memoria_Clock := "Clock_10Hz";
0129
0130    #cinta1 := (#Etapa = 2) OR (#Etapa_Rama1 = 10);
0131
0132    IF (#Etapa = 2 OR #Etapa_Rama1 = 10) AND #Pulso_Clock THEN
0133      "HMI_Pos_Botella1" := "HMI_Pos_Botella1" + 1;
0134    END_IF;
0135
0136
0137    "M_S1" := ("HMI_Pos_Botella1" >= 100);
0138    IF #Etapa_Rama1 = 13 OR #Etapa = 6 THEN
0139      "HMI_Pos_Botella1" := 0;
0140    END_IF;
0141
0142    #cinta2 := (#Etapa_Rama2 = 20);
0143
0144    IF (#Etapa_Rama2 = 20) AND #Pulso_Clock THEN
0145      "HMI_Pos_Botella2" := "HMI_Pos_Botella2" + 1;
0146    END_IF;
0147
0148    "M_S2" := ("HMI_Pos_Botella2" >= 100);
0149
0150    IF (#Etapa_Rama2 = 23) OR (#Etapa = 6) THEN
0151      "HMI_Pos_Botella2" := 0;
0152    END_IF;
0153
0154  END_REGION
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

F4_Manual [FB4]

F4_Manual Properties					
General					
Name	F4_Manual	**Number**	4	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Btn_Man_Cinta1	Bool	false	Non-retain
Btn_Man_Cinta2	Bool	false	Non-retain
Btn_Man_Llenado	Bool	false	Non-retain
Btn_Man_Robot	Bool	false	Non-retain
▼ Output			
cinta1	Bool	false	Non-retain
cinta2	Bool	false	Non-retain
llenado	Bool	false	Non-retain
robot	Bool	false	Non-retain
InOut			
Static			
Temp			
Constant			

```
0001
0002 REGION Datos
0003   //Desarrollo: Ester López Hierro
0004   //Fecha: 04/04/2026
0005   ////Módulo: Lógica - F4 Marcha de verificacion sin orden
0006
0007 END_REGION
0008 REGION Actuadores
0009   #cinta1 := #Btn_Man_Cinta1;
0010   #llenado := #Btn_Man_Llenado;
0011   #cinta2 := #Btn_Man_Cinta2;
0012   #robot := #Btn_Man_Robot;
0013   "M_Etapa_Grafcet" := 100;
0014
0015 END_REGION
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

A6_Homing [FB6]

A6_Homing Properties					
General					
Name	A6_Homing	**Number**	6	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Robot_Posicion_Inicial	Bool	false	Non-retain
▼ Output			
Mandar_Robot_Home	Bool	false	Non-retain
▼ InOut			
Etapa	Int	0	Non-retain
Etapa_Rama1	Int	0	Non-retain
Etapa_Rama2	Int	0	Non-retain
CambioEstado	Bool	false	Non-retain
Static			
Temp			
Constant			

```
0001 REGION Datos
0002   //Desarrollo: Ester López Hierro
0003   //Fecha: 04/04/2026
0004   ////Módulo: Lógica - A6 Puesta en estado inicial
0005
0006 END_REGION
0007
0008 REGION Etapas
0009   CASE #Etapa OF
0010     100:
0011       IF #Robot_Posicion_Inicial THEN
0012         #Etapa := 101;
0013       END_IF;
0014
0015     101:
0016       IF TRUE THEN
0017         #Etapa := 0;
0018       END_IF;
0019
0020   END_CASE;
0021 END_REGION
0022
0023 REGION Acciones
0024   #Mandar_Robot_Home := (#Etapa = 100);
0025
0026   #CambioEstado := (#Etapa = 101);
0027   IF #Etapa = 101 THEN
0028     #Etapa := 0;
0029
```

```
0030      "HMI_Pos_Botella1" := 0;
0031      "HMI_Pos_Botella2" := 0;
0032
0033    END_IF;
0034    "M_Auto" := (#Etapa = 101);
0035  END_REGION
0036
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

D1_Emergencia [FB7]

D1_Emergencia Properties					
General					
Name	D1_Emergencia	**Number**	7	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
Etapa	Bool	false	Non-retain
Static			
Temp			
Constant			

```
0001
0002 REGION Datos
0003   //Desarrollo: Ester López Hierro
0004   //Fecha: 04/04/2026
0005   ////Módulo: Lógica - D1 Parada de emergencia
0006
0007 END_REGION
0008
0009 "M_Cinta1" := FALSE;
0010 "M_Cinta2" := FALSE;
0011 "M_Out_llenado" := FALSE;
0012 "M_Out_robot" := FALSE;
0013
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

Startup [OB100]

Startup Properties					
General					
Name	Startup	**Number**	100	**Type**	OB
Language	SCL	**Numbering**	Automatic		
Information					
Title	"Complete Restart"	**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
Input		
LostRetentive	Bool	
LostRTC	Bool	
Temp		
Constant		

```
0001
0002 REGION Datos
0003   //Desarrollo: Ester López Hierro
0004   //Fecha: 04/04/2026
0005   ////Módulo: Setup
0006
0007 END_REGION
0008 //Los contactos nomrlamnete cerrados :
0009   //paro
0010 "M_Boton_Paro" := TRUE;
0011   //emergencia
0012 "M_Emergenica_OK" := TRUE;
0013 //selector automatico/manual se ininicializa en automatico
0014 "M_Auto" := TRUE;
0015 //se inicializa el grafcet a la etapa inicial
0016 "Etapa_GG" := 0;
0017
0018 "M_Boton_Paro" := TRUE;
0019
0020 //Variables
0021 "M_Etapa_Grafcet" := 0;
0022 "M_Etapa_Grafcet_1" := 0;
0023 "M_Etapa_Grafcet_2" := 0;
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

A2_Paro [FB5]

A2_Paro Properties					
General					
Name	A2_Paro	**Number**	5	**Type**	FB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Marcha	Bool	false	Non-retain
Receta_Actual	Int	0	Non-retain
Configuracion	Bool	false	Non-retain
S1	Bool	false	Non-retain
S2	Bool	false	Non-retain
lleno	Bool	false	Non-retain
fin_robot	Bool	false	Non-retain
▼ Output			
cinta1	Bool	false	Non-retain
cinta2	Bool	false	Non-retain
llenado	Bool	false	Non-retain
robot	Bool	false	Non-retain
Out_Receta_Llenado	Int	0	Non-retain
Cambio_Estado	Bool	false	Non-retain
▼ InOut			
Etapa	Int	0	Non-retain
Etapa_Rama1	Int	0	Non-retain
Etapa_Rama2	Int	0	Non-retain
cont	Int	0	Non-retain
▼ Static			
M_receta	Int	0	Non-retain
Pulso_Etapa5	Bool	false	Non-retain
Pulso_S1	Bool	false	Non-retain
Pulso_S2	Bool	false	Non-retain
Memoria_Etapa5	Bool	false	Non-retain
Memoria_S1	Bool	false	Non-retain
Memoria_S2	Bool	false	Non-retain
Pulso_Clock	Bool	false	Non-retain
Memoria_Clock	Bool	false	Non-retain
Rama1_Ejecutada	Bool	false	Non-retain
QuedaBotella	Bool	false	Non-retain
Pulso_Marcha	Bool	false	Non-retain
Memoria_Marcha	Bool	false	Non-retain
cont_global	Int	0	Non-retain
Temp			
Constant			

```
0001 REGION Datos
```

```
0002    //Desarrollo: Ester López Hierro
0003    //Fecha: 04/04/2026
0004    ////Módulo: Lógica - A2 Parada perddida final ciclo
0005
0006  END_REGION
0007
0008  REGION Flancos
0009    #Pulso_Etapa5 := #Etapa = 5 AND NOT #Memoria_Etapa5;
0010    #Memoria_Etapa5 := #Etapa = 5;
0011
0012    #Pulso_S1 := #S1 AND NOT #Memoria_S1;
0013    #Memoria_S1 := #S1;
0014
0015    #Pulso_S2 := #S2 AND NOT #Memoria_S2;
0016    #Memoria_S2 := #S2;
0017
0018    #Pulso_Marcha := #Marcha AND NOT #Memoria_Marcha;
0019    #Memoria_Marcha := #Marcha;
0020
0021  END_REGION
0022  REGION Etapas
0023    CASE #Etapa OF
0024      0:
0025        IF #Configuracion THEN
0026          #Etapa := 1;
0027        END_IF;
0028
0029      1:
0030        IF #Pulso_Marcha THEN
0031          #Etapa := 2;
0032        END_IF;
0033
0034      2:
0035        IF #Pulso_S1 THEN
0036          #Etapa := 3;
0037        END_IF;
0038
0039      3:
0040        IF #lleno THEN
0041          #Etapa := 4;
0042        END_IF;
0043
0044      4:
0045        IF "Etapa_GG" = 7 THEN
0046            //solo el robot
0047          #Etapa_Rama1 := 14;
0048          #Etapa_Rama2 := 20;
0049          #Etapa := 100;
0050        ELSE
0051            //ambas ramas
0052          #Etapa_Rama1 := 10;
0053          #Etapa_Rama2 := 20;
0054          #Etapa := 100;
0055        END_IF;
0056
0057      100: // Sincronización
0058        IF (#Etapa_Rama1 = 14) AND (#Etapa_Rama2 = 23) THEN
0059          #Etapa := 5;
0060        END_IF;
```

```
0061
0062      5:
0063        #Etapa := 0;
0064
0065    END_CASE;
0066
0067    // --- RAMA 1 ---
0068    CASE #Etapa_Rama1 OF
0069      10:
0070        IF #Pulso_S1 THEN
0071          #Etapa_Rama1 := 11;
0072        END_IF;
0073      11:
0074        IF #lleno AND #Etapa_Rama2 = 22 THEN
0075          #Etapa_Rama1 := 12;
0076        END_IF;
0077      12:
0078        IF #Pulso_S2 THEN
0079          #Etapa_Rama1 := 13;
0080        END_IF;
0081      13:
0082        IF #fin_robot THEN
0083          #Etapa_Rama1 := 14;
0084        END_IF;
0085
0086    END_CASE;
0087
0088    // --- RAMA 2 ---
0089    CASE #Etapa_Rama2 OF
0090      20:
0091        IF #Pulso_S2 THEN
0092          #Etapa_Rama2 := 21;
0093        END_IF;
0094      21:
0095        IF #fin_robot THEN
0096          #Etapa_Rama2 := 22;
0097        END_IF;
0098      22:
0099        IF #Etapa_Rama1 = 14  THEN
0100          #Etapa_Rama2 := 23;
0101        END_IF;
0102    END_CASE;
0103
0104  END_REGION
0105  REGION Acciones
0106
0107    #Cambio_Estado := FALSE;
0108
0109    IF #Etapa = 0 THEN
0110
0111      #Etapa_Rama1 := 0;
0112      #Etapa_Rama2 := 0;
0113      #Cambio_Estado := TRUE; //pulso para ir a A1
0114    END_IF;
0115
0116    IF #Etapa = 2 THEN
0117      #M_receta := #Receta_Actual;
0118    END_IF;
0119    IF #Etapa = 1 THEN
```

```
0120      #cont := 0;
0121    END_IF;
0122
0123    IF #Pulso_Etapa5 THEN
0124      #cont := #cont + 1;
0125      #cont_global := #cont_global + 1;
0126    END_IF;
0127
0128    #cinta1 := (#Etapa = 2) OR (#Etapa_Rama1 = 10)OR (#Etapa_Rama1 = 12);
0129    #llenado := (#Etapa = 3) OR (#Etapa_Rama1 = 11);
0130    #cinta2 := (#Etapa_Rama2 = 20) OR (#Etapa_Rama1 = 12);
0131    #robot := (#Etapa_Rama2 = 21) OR (#Etapa_Rama1 = 13);
0132
0133    IF (#Etapa = 3) OR (#Etapa_Rama1 = 11) OR (#Etapa_Rama2 = 21) OR (#Eta-
      pa_Rama1 = 13) THEN
0134      #Out_Receta_Llenado := #M_receta;
0135    END_IF;
0136  END_REGION
0137
0138
0139
0140  REGION Simulación_Movimiento
0141    #Pulso_Clock := "Clock_10Hz" AND NOT #Memoria_Clock;
0142    #Memoria_Clock := "Clock_10Hz";
0143
0144    IF (#Etapa = 2 OR #Etapa_Rama1 = 10 OR #Etapa_Rama1 = 14) AND #Pulso_Clock
      THEN
0145      "HMI_Pos_Botella1" := "HMI_Pos_Botella1" + 1;
0146    END_IF;
0147
0148    "M_S1" := ("HMI_Pos_Botella1" >= 100);
0149
0150    IF (#Etapa = 0) OR (#Etapa_Rama1 = 12) OR (#Etapa = 4) THEN
0151      "HMI_Pos_Botella1" := 0;
0152    END_IF;
0153
0154    IF ((#Etapa_Rama2 = 20) OR (#Etapa_Rama1 = 12)) AND #Pulso_Clock THEN
0155      "HMI_Pos_Botella2" := "HMI_Pos_Botella2" + 1;
0156    END_IF;
0157
0158    "M_S2" := ("HMI_Pos_Botella2" >= 100);
0159
0160    IF (#Etapa = 0) OR (#Etapa = 4) OR (#Etapa_Rama2 = 22 AND #Etapa_Rama1 <> 12
       AND #Etapa_Rama1 <> 13) THEN
0161      "HMI_Pos_Botella2" := 0;
0162    END_IF;
0163
0164  END_REGION
```

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

Logica_DB [DB1]

Logica_DB Properties					
General					
Name	Logica_DB	**Number**	1	**Type**	DB
Language	DB	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
Input			
Output			
InOut			
▼ Static			
Pos_Botella_1	Int	0	False
Pos_Botella_2	Int	0	False
F1_Automatico_Instance	"F1_Automatico"		False
F5_Semiautomatico_Instance	"F5_Semiautomatico"		False
F4_Manual_Instance	"F4_Manual"		False
A6_Homing_Instance	"A6_Homing"		False
D1_Emergencia_Instance	"D1_Emergencia"		False
A2_Paro_Instance	"A2_Paro"		False

TFG_V6(02042026) / PLC_1 [CPU 1512C-1 PN] / Program blocks

IN/OUT [FC1]

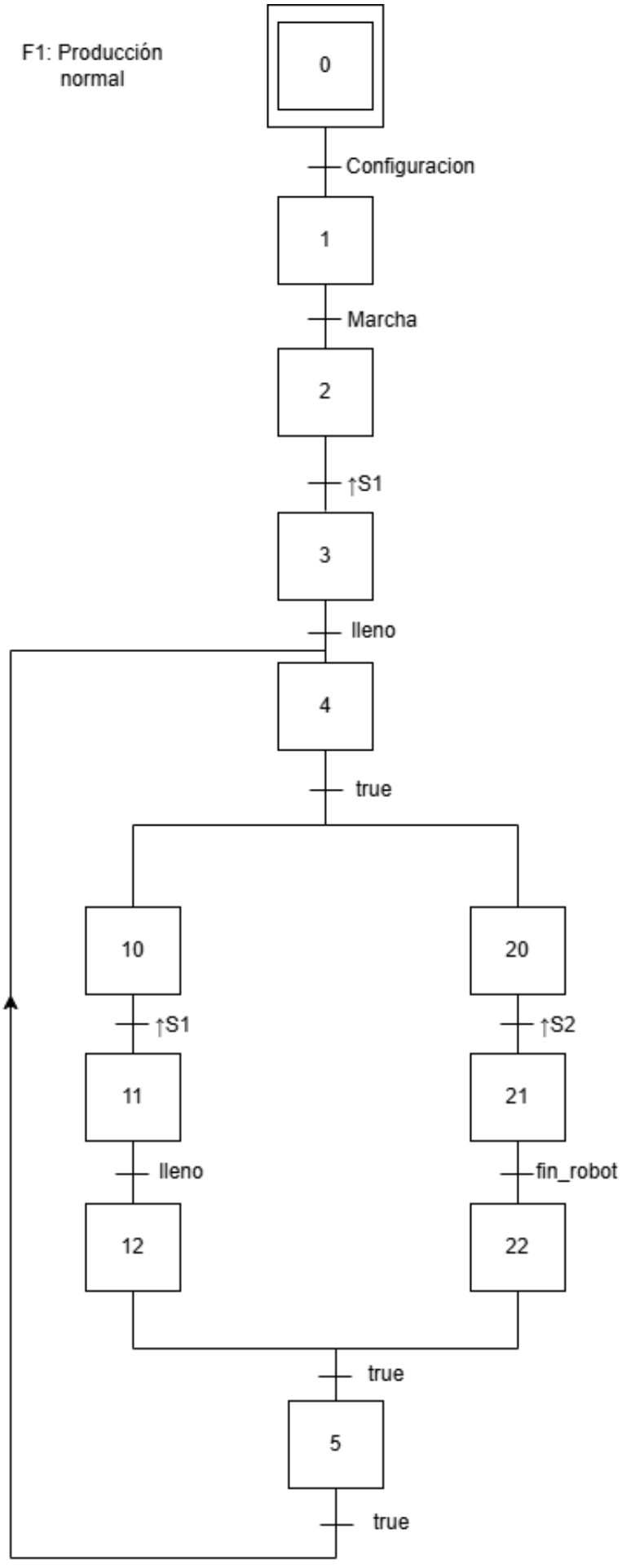
IN/OUT Properties					
General					
Name	IN/OUT	**Number**	1	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

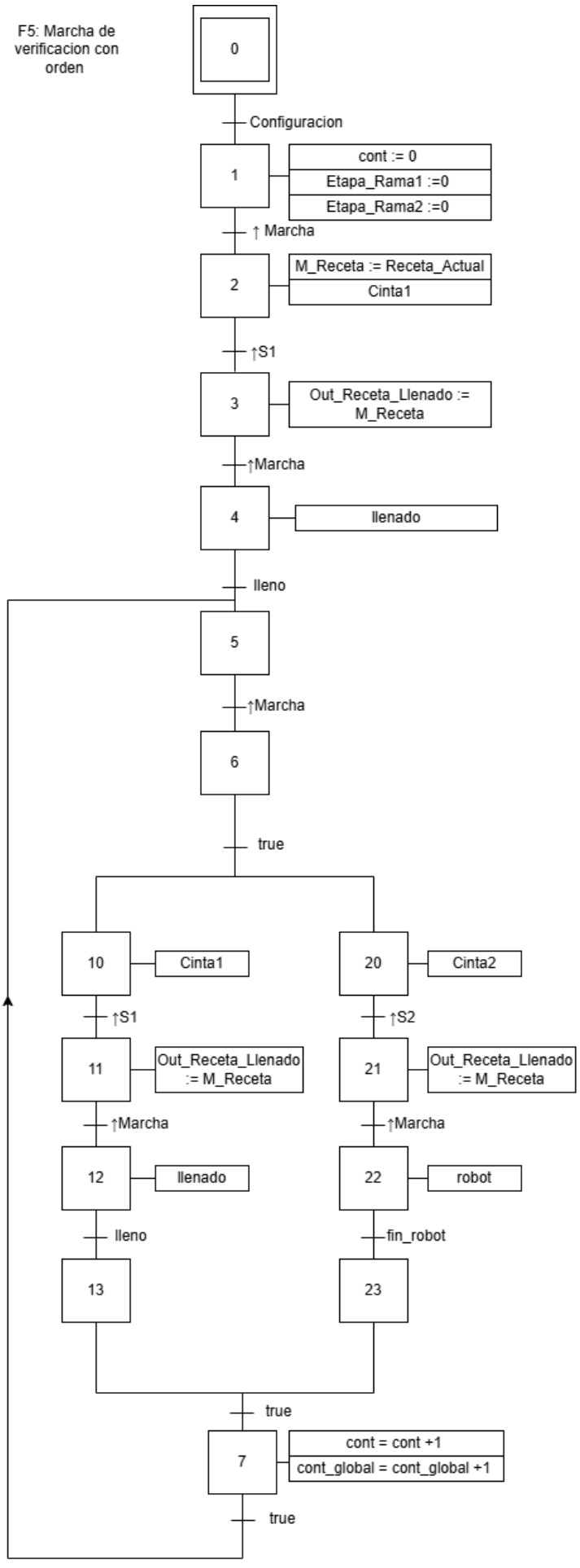
Name	Data type	Default value
Input		
Output		
InOut		
Temp		
Constant		
▼ Return		
IN/OUT	Void	

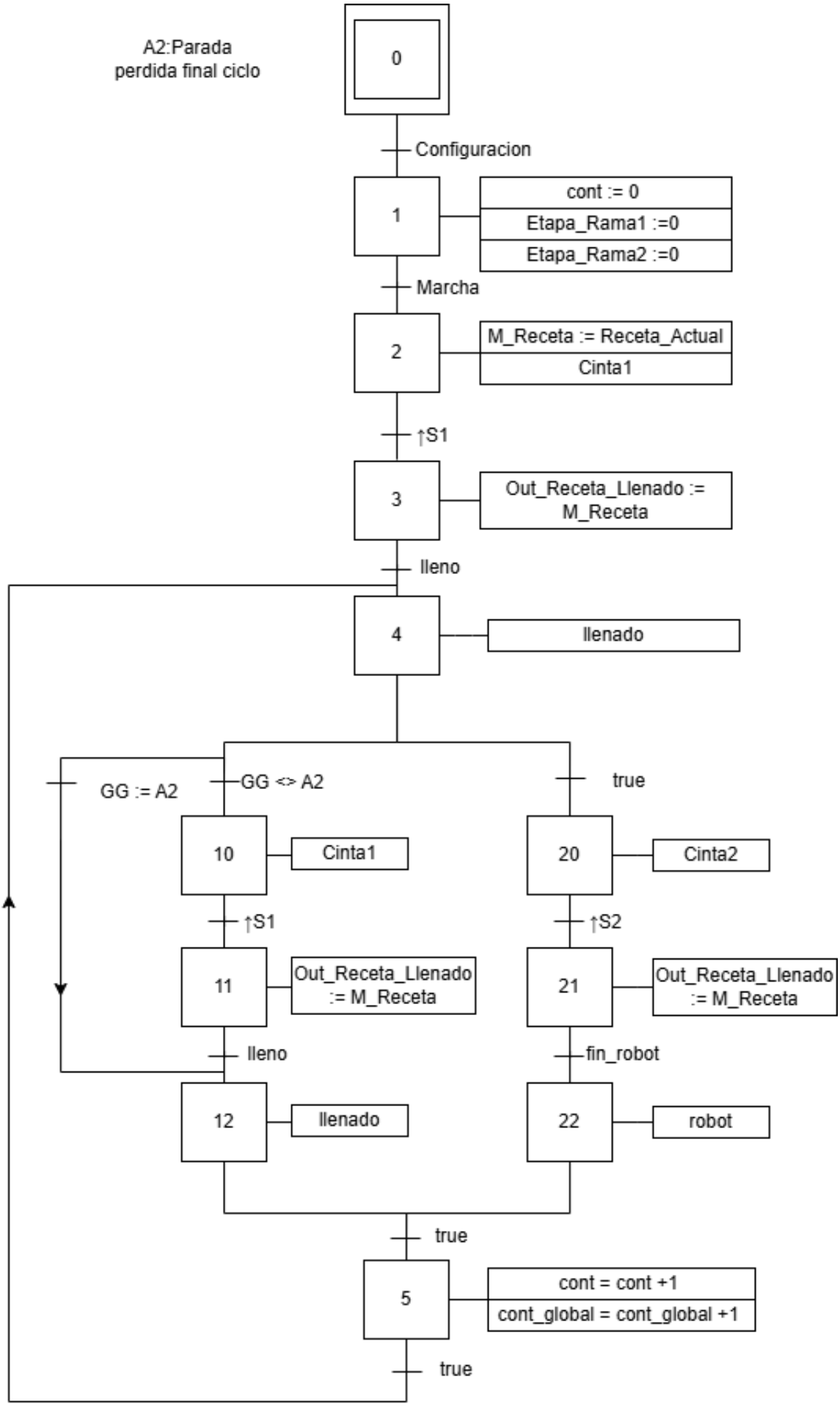
```
0001
0002
0003 //IN
0004 ////Botones
0005 "IN_Boton_Marcha" := "M_Boton_Marcha";
0006 "IN_Boton_Paro" := "M_Boton_Paro";
0007 "IN_Rearme" := "M_Rearme";
0008 //Sensores de proximidad
0009 "IN_S1" := "M_S1";
0010 "IN_S2" := "M_S2";
0011
0012 //Sincronizacion con elementos externos
0013 "IN_IN_lleno" := "M_IN_lleno";
0014 "IN_IN_fin_robot" := "M_IN_fin_robot";
0015
0016
0017 //OUT
0018 "OUT_Cinta1":= "M_Cinta1";
0019 "OUT_Cinta2":= "M_Cinta2";
0020 //Sincronizacion con elementos externos
0021 "OUT__Out_llenado":="M_Out_llenado";
0022 "OUT_Out_robot":="M_Out_robot" ;
0023 "OUT_OUT_robot_home":="M_OUT_robot_home";
0024 "OUT_Out_Receta_Llenado" :="M_Out_Receta_Llenado" ;
```

Capítulo D: Esquemas GRAFCETS de la programación del PLC

En las siguientes páginas se presentan los diagramas GRAFCET que modelan la lógica secuencial y los estados de funcionamiento implementados para el control del automatismo.


F1: Producción normal
0
Configuracion
1
Marcha
2
↑S1
3
lleno
4
true
10
↑S1
11
lleno
12
20
↑S2
21
fin_robot
22
true
5
true


F5: Marcha de verificacion con orden
0
Configuracion
1
cont := 0
Etapa_Rama1 :=0
Etapa_Rama2 :=0
↑ Marcha
2
M_Receta := Receta_Actual
Cinta1
↑S1
3
Out_Receta_Llenado := M_Receta
↑Marcha
4
llenado
lleno
5
↑Marcha
6
true
10
Cinta1
↑S1
11
Out_Receta_Llenado := M_Receta
↑Marcha
12
llenado
lleno
13
20
Cinta2
↑S2
21
Out_Receta_Llenado := M_Receta
↑Marcha
22
robot
fin_robot
23
true
7
cont = cont +1
cont_global = cont_global +1
true


A2:Parada
perdida final ciclo
0
Configuracion
1
cont := 0
Etapa_Rama1 :=0
Etapa_Rama2 :=0
Marcha
2
M_Receta := Receta_Actual
Cinta1
↑S1
3
Out_Receta_Llenado :=
M_Receta
lleno
4
llenado
GG := A2
GG <> A2
true
10
Cinta1
20
Cinta2
↑S1
↑S2
11
Out_Receta_Llenado
:= M_Receta
21
Out_Receta_Llenado
:= M_Receta
lleno
fin_robot
12
llenado
22
robot
true
5
cont = cont +1
cont_global = cont_global +1
true

Capítulo E: Código RAPID del robot

Se adjunta en este anexo el código de programación en el lenguaje RAPID para el controlador del brazo robótico.

```
MODULE Module1
    CONST robtarget Target_30:=[[704.990919137,0,705.74730631],[0.190808996,0,0.981627183,0],
[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[563.169,341.222,250],[0,0,1,0],[0,-1,1,0],[9E+09,9E+09,9E+09,9E
+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[549.352,-277.081,250],[0,0,1,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E
+09,9E+09,9E+09]];
    CONST robtarget M3_3:=[[-406.555,-406.628,120],[0,0.707106781,0.707106781,0],[-1,-1,0,0],[9E
+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Bote:=[[-116.935,106.667,100],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E
+09,9E+09]];
    CONST robtarget Tapa:=[[-107.014,102.06,20],[0,-0.707106781,0.707106781,0],[0,1,-2,0],[9E+09,9E
+09,9E+09,9E+09,9E+09,9E+09]];

    PROC main()
        VAR num i;
        VAR num j;
        VAR num n_bote;
        CONST num dist_x := 150;
        CONST num dist_y := 150;


        FOR i FROM 0 TO 2 DO
            FOR j FROM 0 TO 2 DO

                MoveL Target_30,v1000,fine,MyTool\WObj:=wobj0;

                WaitDI DI_S2,1;
                SetDO Digital1 ,1 ;
                WaitTime 0.5;
                SetDO Digital1,0 ;
                WaitDI DI_S2,0;

                MoveL Offs(Tapa,0,0,100),v1000,z10,MyTool\WObj:=WO_Tapas;
                MoveL Tapa,v500,fine,MyTool\WObj:=WO_Tapas;
                WaitTime 0.5;
                SetDO DO_Ventosa,1 ;
                MoveL Offs(Tapa,0,0,100),v500,z10,MyTool\WObj:=WO_Tapas;

                MoveL RelTool(Offs(Bote,0,0,100),0,0,0\Rz:=90), v500, z10, MyTool\WObj:=WO_Cinta;
                MoveL RelTool(Offs(Bote,0,0,21),0,0,0\Rz:=90), v500, z10, MyTool\WObj:=WO_Cinta;
                MoveL Offs(Bote,0,0,20),v500,fine,MyTool\WObj:=WO_Cinta;
                WaitTime 0.5;
                MoveL Offs(Bote,0,0,100),v500,z10,MyTool\WObj:=WO_Cinta;

                MoveL Offs(M3_3, i*dist_x, j*dist_y,50),v500, z10, MyTool\WObj:=WO_Caja;
                MoveL Offs(M3_3, i*dist_x, j*dist_y,0),v500, fine, MyTool\WObj:=WO_Caja;
                WaitTime 0.5;
                SetDO DO_Ventosa,0 ;
                MoveL Offs(M3_3, i*dist_x, j*dist_y,50),v500, z10, MyTool\WObj:=WO_Caja;

                !para la simulacion y mostrar los botes
                n_bote := (i*3)+j+1;
                TEST n_bote
                    CASE 9: PulseDO DO_Show1;
```

```
                    CASE 8: PulseDO DO_Show1;
                    CASE 7: PulseDO DO_Show3;
                    CASE 6: PulseDO DO_Show4;
                    CASE 5: PulseDO DO_Show5;
                    CASE 4: PulseDO DO_Show6;
                    CASE 3: PulseDO DO_Show7;
                    CASE 2: PulseDO DO_Show8;
                    CASE 1: PulseDO DO_Show9;
                ENDTEST



            ENDFOR
        ENDFOR
    ENDPROC
    PROC Path_10()
        MoveL Target_30,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Tapa,v1000,fine,MyTool\WObj:=WO_Tapas;
        MoveL Target_30,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Bote,v1000,fine,MyTool\WObj:=WO_Cinta;
        MoveL Target_30,v1000,z100,MyTool\WObj:=wobj0;
        MoveL M3_3,v1000,fine,MyTool\WObj:=WO_Caja;
        MoveL Target_30,v1000,z100,MyTool\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

Capítulo F: Datasheet de MOSFET Canal N IRLZ44N

Documento técnico - MOSFET IRLZ44N (AllDatasheet)

Capítulo G: Datasheet de diodo

Documento técnico - Diodo 1N4007 (AllDatasheet)

Capítulo H: Sensor de la célula de carga de 1kg

Documento técnico - Célula de carga TAL220 (SparkFun)

Capítulo I: Bomba peristáltica

Documento técnico - Bomba peristáltica Kamoer NKP

Capítulo J: Datasheet de ESP32

Documento técnico - Microcontrolador ESP32 (AllDatasheet)

Capítulo K: Datasheet de HX711

Documento técnico - Convertidor A/D HX711 (SparkFun)

Capítulo L: Diodo Flyback

Documento técnico - Diodo de protección 1N4007

Capítulo M: Estudio económico

Hoja de cálculo - Desglose del Presupuesto Económico

Capítulo N: Archivos del proyecto del PLC

Carpeta compartida de Google Drive - Proyecto PLC y Vídeo

Capítulo Ñ: Archivos del proyecto del robot

Carpeta compartida de Google Drive - Proyecto Robot y Vídeo