



**Universidad  
de La Laguna**

**ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA**

**Trabajo de Fin de Grado**

**“Puesta en marcha y mejora de usabilidad  
de un prototipo físico para la simulación y  
detección de fugas en depósitos de  
combustible”**

**GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y  
AUTOMÁTICA**

**Autor:** Gregorio José Medina León

**Tutores:** Marta Sigut Saavedra, Pedro Antonio Toledo Delgado

**Fecha:** Julio de 2023

# Agradecimientos

Gracias a todas las personas que han contribuido en mi formación académica y desarrollo personal.

A los profesores que tuvieron la paciencia y dedicación de atender mis inquietudes.

A los compañeros que hicieron más llevaderas las largas horas de trabajo.

Y a la comunidad de creadores y entusiastas del 'hazlo tú mismo', por brindar la oportunidad de expandir mis conocimientos y ampliar mis horizontes.

# Índice de contenido

<b>Resumen</b> . . . . .	<b>5</b>
<b>Abstract</b> . . . . .	<b>5</b>
<b>1 Introducción</b> . . . . .	<b>6</b>
1.1 Marco teórico . . . . .	9
1.2 Antecedentes . . . . .	9
1.3 Objetivos propuestos . . . . .	10
<b>2 Hardware</b> . . . . .	<b>11</b>
2.1 Planta . . . . .	11
2.2 Sensores . . . . .	15
2.3 Actuadores . . . . .	18
2.4 Unidad de control . . . . .	21
2.5 Panel de control físico . . . . .	24
2.6 Instalación eléctrica y electrónica . . . . .	26
<b>3 Software</b> . . . . .	<b>31</b>
3.1 Controladora . . . . .	31
3.1.1 Inicialización . . . . .	31
3.1.2 Máquina de estado general (Controladora) . . . . .	33
3.1.3 Máquina de estado de depósitos . . . . .	34
3.2 Clase PanelControl . . . . .	37
3.2.1 Funciones <i>ActualizarEntradasPanel()</i> y <i>ActualizarSalidasPanel()</i> . . . . .	37
3.3 Cuadro de mando digital . . . . .	39
3.3.1 Funcionamiento del <i>dashboard</i> . . . . .	39
3.3.2 Información presentada . . . . .	40
3.4 Comunicación en red . . . . .	42
3.4.1 Funcionamiento del protocolo <i>MQTT</i> . . . . .	42
3.4.2 Configuración del bróker . . . . .	43
3.4.3 Envío de datos . . . . .	44
3.4.4 Recepción de datos . . . . .	44
3.5 Clase Debug . . . . .	45
3.6 Puesta en funcionamiento . . . . .	46
<b>4 Pruebas y resultados</b> . . . . .	<b>47</b>
<b>5 Conclusiones</b> . . . . .	<b>49</b>

<b>6</b>	<b>Conclusions</b>	<b>49</b>
<b>7</b>	<b>Líneas futuras</b>	<b>50</b>
7.1	Corto plazo	50
7.2	Medio plazo	50
7.3	Largo plazo	51
<b>8</b>	<b>Presupuesto</b>	<b>52</b>
<b>9</b>	<b>Bibliografía</b>	<b>54</b>
<b>Anexos</b>		<b>55</b>
	Anexo I. Disposición del bajante	56
	Anexo II. Disposición del cableado	57
	Anexo III. Panel de control físico	58
	Anexo IV. Diagrama esquemático de la instalación	59
	Anexo V. Diagrama detallado de la instalación	60
	Anexo VI. Proceso de Inicialización	61
	Anexo VII. Máquina de estados general	62
	Anexo VIII. Máquina de estados de depósito	63
	Anexo IX. Cuadro de mando digital	64

# Índice de figuras

1.1	Tanques subterráneos. . . . .	6
1.2	Camión cisterna. . . . .	7
1.3	Surtidores de combustible. . . . .	7
1.4	Probador de volúmenes <i>prover</i> . . . . .	8
2.1	Disposición de depósitos. . . . .	11
2.2	Nueva ubicación de las bombas. . . . .	13
2.3	Implementación del cambio en la planta. . . . .	13
2.4	Disposición general del bajante (vista trasera). . . . .	14
2.5	Conexión entre depósito y bajante. . . . .	15
2.6	Módulo HX711. . . . .	16
2.7	Sensor de nivel. . . . .	16
2.8	Disposición de los sensores. . . . .	17
2.9	Bomba centrífuga DC. . . . .	18
2.10	Electroválvula. . . . .	19
2.11	Bomba peristáltica. . . . .	19
2.12	Bomba centrífuga AC. . . . .	20
2.13	Válvula antirretorno. . . . .	20
2.14	Disposición de los actuadores. . . . .	21
2.15	Arduino Mega. . . . .	22
2.16	Módulo de relés. . . . .	23
2.17	Controlador L298N. . . . .	23
2.18	Elementos del panel de control. . . . .	25
2.19	Implementación del panel de control. . . . .	25
2.20	Conexiones del panel de control. . . . .	26
2.21	Cajas estancas de la balda superior. . . . .	27
2.22	Cajas estancas de la balda media. . . . .	27
2.23	Cajas estancas de la balda inferior. . . . .	27
2.24	Ubicación del armario eléctrico. . . . .	28
2.25	Regulador de voltaje de 5 V. . . . .	29
2.26	Diodo de descarga de inductancia. . . . .	29
2.27	Varistor . . . . .	30
2.28	Montaje provisional de la unidad de control . . . . .	30
3.1	Máquina de estados general. . . . .	34
3.2	Máquina de estados de depósito . . . . .	36
3.3	Elementos del cuadro de mandos digital. . . . .	40
3.4	Emergencia enviada desde el cuadro de mandos digital. . . . .	41
3.5	Pestaña de mensajes de depuración. . . . .	42
3.6	Recepción de datos en suscriptor. . . . .	45

## Índice de tablas

1	Relación de TipoDebug y mensajes mostrados . . . . .	46
2	Resumen de pruebas y resultados. . . . .	47
3	Presupuesto del proyecto. . . . .	52

## Resumen

En los últimos años, el Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna ha dedicado sus esfuerzos a una investigación enfocada en la detección de fugas de combustibles en estaciones de servicio.

En el contexto de esta investigación se han llevado a cabo diversos Trabajos de Fin de Grado y de Fin de Máster, consistentes en el diseño e implementación de un prototipo de bajo coste concebido para simular de manera autónoma las dispensaciones de una estación de servicio y recrear fugas bajo condiciones controladas.

El prototipo no sólo permitirá la generación de datos relevantes para la investigación en la detección de fugas de combustible, sino que también será utilizado como una herramienta docente, puesto que proporcionará a los estudiantes una plataforma de pruebas para los algoritmos que ellos mismos desarrollen.

A lo largo de este proyecto, se llevó a cabo la implementación de una serie de mejoras en materia de usabilidad y seguridad en dicho prototipo, incluyendo la reconfiguración del sistema de tuberías y cableado.

Asimismo, se desarrollaron nuevas funcionalidades para facilitar la interacción de los usuarios con el sistema y la transmisión de datos, tales como el diseño de un cuadro de mandos que permite el envío de información de la planta por red local.

## Abstract

In recent years, the Department of Computer Engineering and Systems at the University of La Laguna has dedicated its efforts to research focused on the detection of fuel leaks at service stations.

In the context of this research, various Bachelor's and Master's Thesis have been carried out, involving the design and implementation of a low-cost prototype designed to autonomously simulate the dispensing operations of a service station and recreate leaks under controlled conditions.

The prototype is not only expected to generate valuable data for fuel leak detection research but is also envisioned to serve as an educational tool, as it will provide students with a test platform for the algorithms they develop.

Throughout this project, several enhancements in terms of usability and safety were implemented in the prototype, including the reconfiguration of the piping and wiring system.

Moreover, new features were developed to improve user interaction with the system and to facilitate data transmission, such as the design of a *dashboard* that enables the transmission of plant information via a local network.

## 1. Introducción

Las estaciones de servicio son instalaciones complejas y altamente reguladas, diseñadas para almacenar, gestionar y dispensar combustible de forma segura y eficiente.

A grandes rasgos, una estación de servicio está conformada por las siguientes partes:

- **Depósitos subterráneos de combustible:**

Se trata de los depósitos en los que se almacena el combustible antes de ser entregado a los vehículos. Los depósitos pueden ser de diferentes capacidades, que normalmente están en el orden de las decenas de miles de litros, y normalmente están hechos de acero recubierto o de fibra de vidrio. A pesar de que la normativa no impone la obligatoriedad en todos los casos, cada depósito subterráneo puede contar con un sistema de monitorización de fugas encargado de la detección temprana y precisa de cualquier escape de combustible.



Figura 1.1: Tanques subterráneos.

- **Camiones cisterna:**

Son los vehículos de transporte que llevan el combustible desde la refinería o terminal de almacenamiento a la estación de servicio. Cuando el camión cisterna llega a la estación de servicio, se conecta a los puntos de llenado de los depósitos para realizar una descarga del orden de miles de litros, en un proceso que debe ser supervisado por un operario para asegurar que se realiza de forma segura.

Debido a la velocidad del flujo y a las características propias de los equipos de conexión utilizados, se generan variaciones en la medición del volumen de combustible transferido, lo que conlleva una disminución en la precisión de la cantidad registrada en los depósitos durante el proceso de llenado.



Figura 1.2: Camión cisterna.

■ **Surtidores de combustible:**

Los surtidores son la parte visible para el cliente de todo el sistema, conectando los depósitos subterráneos con los vehículos a través de mangueras y boquillas especializadas. Incluyen medidores de flujo para calcular la cantidad de combustible que se está dispensando. Las agencias reguladoras imponen normas que establecen cuánto puede desviarse un medidor de su calibración antes de requerir ajustes o reemplazos.



Figura 1.3: Surtidores de combustible.

En este aspecto, las estaciones de servicio también cuentan con un probador de volumen (*prover*) empleado para verificar la precisión de los medidores.



Figura 1.4: Probador de volúmenes *prover*.

Se trata de un instrumento en forma de matraz, con una escala graduada, en el que se vierte un volumen conocido del surtidor. Si las dos lecturas no coinciden dentro de un margen de error aceptable, el medidor de flujo requiere ser ajustado o calibrado.

En relación a las cuestiones de seguridad concernientes a una estación de servicio, se destaca la prevención de fugas, de suma importancia por el daño que puede causar en el medio a corto y largo plazo.

Los hidrocarburos que componen el combustible, como la gasolina y el diésel, poseen un alto grado de toxicidad y, una vez liberados en la naturaleza, pueden contaminar el suelo, las aguas subterráneas y superficiales, y la atmósfera.

Los recursos hídricos contaminados por filtraciones de combustible se vuelven inseguros para el consumo humano y la vida acuática. En el suelo, estos hidrocarburos pueden afectar adversamente la vida microbiana, lo que a su vez puede tener un efecto dominó en la salud del ecosistema terrestre local.

Asimismo, cuando estos hidrocarburos alcanzan la atmósfera, ya sea a través de la evaporación directa o como resultado de la biodegradación en el suelo, contribuyen a la formación de ozono troposférico, un gas que tiene efectos perjudiciales en la calidad del aire y la salud humana.

Estos daños, además, acarrean un riesgo de multas y sanciones regulatorias para las estaciones de servicio que no lleven a cabo las medidas preventivas adecuadas.

Algunos de estos procedimientos incluyen la realización de pruebas periódicas de los tanques y tuberías, para valorar su presión e integridad, y la implementación de Sistemas de Monitoreo Continuo de Fugas (CMLS), que constituyen una herramienta esencial para la detección de fugas en tiempo real.

## 1.1. Marco teórico

La dificultad que entraña la detección de fugas se hace patente en el artículo publicado por el departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna con el título *Applying pattern classification techniques to the early detection of fuel leaks in petrol stations* [1], que presenta la aplicación de técnicas de clasificación de patrones para la detección temprana de fugas de combustible en estaciones de servicio. En este trabajo, se emplean datos reales proporcionados por Repsol para construir objetos representativos de los días en que la estación de servicio está operativa, que se clasifican en dos posibles categorías: 'día sin fugas' o 'día con fugas', utilizando tanto clasificadores supervisados como no supervisados.

La investigación prosiguió con el artículo *Time windows: The key to improving the early detection of fuel leaks in petrol stations* [2], que propone el uso de ventanas de tiempo para mejorar la detección de fugas de combustible en las estaciones de servicio. Los autores emplean clasificadores supervisados de dos clases, que indican la presencia de fugas haciendo uso de variables de los libros de inventario de las estaciones. El estudio destaca cómo la aplicación de ventanas de tiempo, que implica el procesamiento de información acumulada durante varios días, puede utilizarse para resolver eficientemente el problema propuesto, cumpliendo plenamente con la normativa aplicable.

Dichos artículos han sido citados, entre otros, por *A new small leakage detection method based on capacitance array sensor for underground oil tank* [3], que desarrolla un nuevo método de detección de fugas basado en sensores de capacitancia; por *Feasibility of a TDR-based technique for fluid hydrocarbon leak detection* [4], que investiga la viabilidad de una técnica de detección de fugas rápida y no destructiva basada en los principios de la reflectometría; y por *Automatic meter error detection with a data-driven approach* [5], que propone un enfoque novedoso para la detección remota y automática de errores en los medidores de combustible a través de técnicas estadísticas y de aprendizaje profundo.

Considerando todos los aspectos mencionados, se puede apreciar que la detección de fugas constituye un desafío técnico complejo que está siendo objeto de multitud de investigaciones.

## 1.2. Antecedentes

El proyecto presentado en este documento se fundamenta en trabajos previos realizados por otros estudiantes, que se adscriben a la misma investigación acerca de la detección de fugas de combustible en los depósitos de estaciones de servicio.

El TFG *Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos* [6], escrito por Luis Arriaga Campos, presenta el diseño inicial de un prototipo de bajo coste de una planta que simula el proceso de llenado y vaciado de depósitos en una estación de servicio, aunque sin implementación física. Posteriormente, el TFM *Diseño e implementación de un sistema autónomo para la detección de fugas en depósitos* [7], de Nicolás Yanes Pérez, expuso el diseño de un simulador de dicha planta para la generación de datos de un inventario.

Por último, el TFG *Simulación y detección de fugas en depósitos de combustible* [8], presentado

por Fernando Rodríguez Herrera, así como el TFG aún sin presentar *Prototipo de bajo coste para la simulación de fugas en depósitos de una estación de servicio con fines docentes y de investigación*, por Eduardo Miguel Gastón Quesada, supusieron la implementación física del diseño, que sirve como punto de partida para el presente proyecto.

Es importante destacar que, si bien se han aprovechado numerosas propuestas en cuanto a diseño y selección de materiales del prototipo anterior, también se ha tenido que abordar y solucionar progresivamente ciertas dificultades que se dejaron desatendidas en su configuración y funcionamiento.

### 1.3. Objetivos propuestos

Para el presente proyecto, se plantean los siguientes objetivos:

1. Modificar la configuración física de la planta para mejorar su seguridad y usabilidad.
2. Renovar la instalación eléctrica y electrónica para mejorar su seguridad y facilitar la localización de errores y sustitución de componentes.
3. Realizar las modificaciones pertinentes en el código para una mejor legibilidad y usabilidad.
4. Diseñar un nuevo proceso de inicialización para agilizar la puesta en marcha de experimentos.
5. Diseñar un panel de control físico que permita interactuar con la planta y obtener información básica de la misma.
6. Diseñar un cuadro de mando digital (*dashboard*) que permita obtener información detallada del estado del experimento y de la planta, así como interactuar con la misma.
7. Diseñar un sistema de transmisión remota de los datos del experimento.
8. Diseñar un sistema básico de detección de fugas.

## 2. Hardware

A grandes rasgos, se ha aprovechado el hardware de versiones previas del prototipo, aunque con algunos cambios y adiciones que se explican en detalle a continuación:

### 2.1. Planta

La planta está conformada por cinco depósitos de agua dispuestos verticalmente en una estantería e interconectados como se aprecia en la figura 2.17.

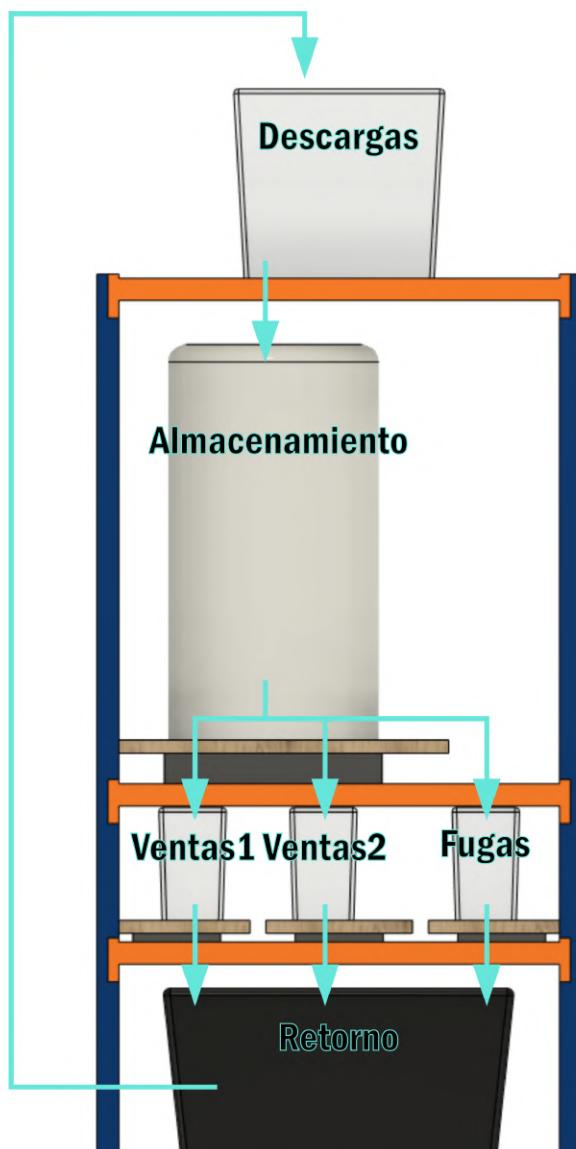


Figura 2.1: Disposición de depósitos.

El papel que cumple cada uno de estos depósitos es el siguiente:

- **Depósito de Descargas:** Este depósito, con una capacidad de 33 L, cumple la función de simular un camión cisterna que periódicamente efectúa descargas en el depósito de Almacenamiento.
- **Depósito de Almacenamiento o Principal:** El depósito de Almacenamiento, con una capacidad de 125 L, representa el tanque subterráneo de las estaciones de servicio. Desde este depósito se realizan las dispensaciones a los depósitos de Ventas y Fugas.
- **Depósitos de Ventas1 y Ventas2:** Estos depósitos, de 4 L de capacidad, desempeña el papel de surtidores de combustible, por lo que se van llenando y vaciando cíclicamente a distintos volúmenes.
- **Depósito de Fugas:** También con 4 L de capacidad, cumple la función de almacenar la fuga simulada en el depósito de Almacenamiento.
- **Depósito de Retorno:** El depósito de Retorno no tiene contraparte real, ya que su función es la de devolver el agua de Ventas y Fugas al depósito de Descargas. Por este motivo, se trata del único depósito del cual no se recopilan datos. Cuenta con una capacidad de 150 L, pues debe ser capaz de contener el volumen total de agua en el sistema.

Los depósitos están conectados por tuberías de PVC rígido de 20 mm de diámetro, a excepción de la unión entre el depósito de Retorno y el de Descargas, que se trata de una tubería de PVC flexible de 20 mm de diámetro. Además, el depósito de Almacenamiento cuenta en su salida con una válvula de esfera como medida de seguridad cuando la planta no está en funcionamiento.

Cabe destacar que, si bien no ha habido cambios en su uso con respecto a la iteración anterior del prototipo, sí que ha habido modificaciones en la ubicación de los depósitos y tuberías. Dichos cambios han sido motivados por la intención de instalar un cerramiento de metacrilato para versiones futuras del proyecto, lo que requiere que no haya elementos sobresalientes de la estantería.

Así, el depósito principal ha sido reubicado para ocupar una posición más central en la balda media, lo que a su vez también llevó la recolocación de las válvulas y bombas conectadas a dicho depósito (figura 2.3). Del mismo modo, la tubería del flexible que conecta el depósito de Retorno con el depósito de Descargas se ha reubicado para que pase por la parte interna de la estantería.

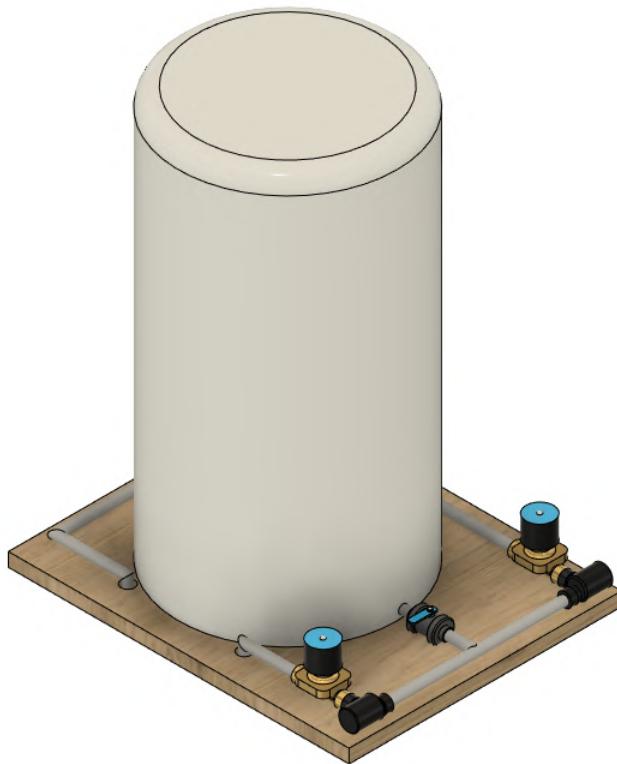


Figura 2.2: Nueva ubicación de las bombas.



Figura 2.3: Implementación del cambio en la planta.

Sin embargo, el cambio más importante con respecto al proyecto original que se ha realizado en la planta es la adición de un bajante de PVC rígido, que garantiza una capa de seguridad más ante posibles desbordamientos. El diámetro seleccionado para el bajante ha sido 40 mm, de modo que pueda contener sin problemas el caudal de múltiples depósitos rebosando simultáneamente.

Tal y como se puede apreciar en la figura 2.4 y en el anexo I, el bajante se ubica en la parte interior trasera de la estantería, y desde la vertical principal se desprenden ramificaciones que se conectan con cada depósito. Estas tuberías secundarias tienen una pendiente de un mínimo de  $3^{\circ}$ , lo que asegura un flujo adecuado del agua hacia el depósito de Retorno, evitando cualquier posible estancamiento en su trayecto.

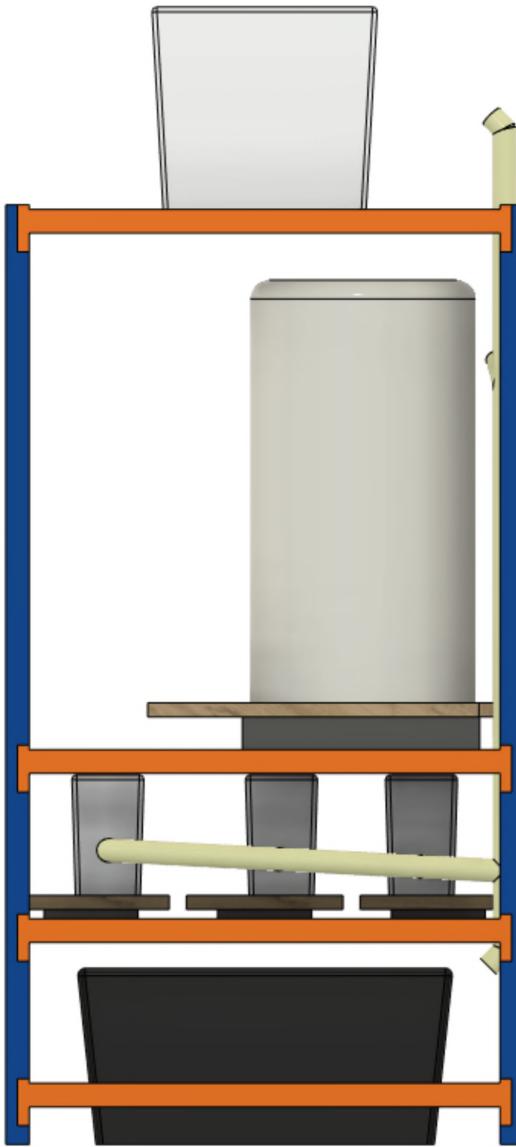


Figura 2.4: Disposición general del bajante (vista trasera).

Para evitar que la presencia de este bajante afecte al peso medido por las básculas, se tuvo la precaución de evitar el contacto directo entre los depósitos y las tuberías. Para ello, la salida de cada depósito, de PVC flexible de 20 mm de diámetro, se prolonga hasta ubicarse sobre su acceso al bajante, sin llegar a apoyarse en éste.



Figura 2.5: Conexión entre depósito y bajante.

En este aspecto cabe destacar que, aunque el depósito de Descargas actualmente no cuenta con báscula, su conexión con el bajante se ha dispuesto siguiendo la misma filosofía de cara a futuras versiones del proyecto. La ubicación y papel de las básculas se explica más detalladamente en el apartado 2.2.

Resulta digno de mención que la ejecución de estos cambios en la planta implicó inicialmente desmontar el prototipo dado en su totalidad, tanto para realizar una limpieza de todos los depósitos como para practicar las nuevas perforaciones de las salidas de desbordamiento y del bajante en las baldas.

## 2.2. Sensores

Los sensores encargados de obtener información sobre las variables de la planta son:

- **Básculas equipadas con módulos HX711:** Para tomar la medida del peso de cada depósito se hace uso de una báscula conformada por cuatro celdas de carga. El depósito de Almacenamiento, por sus dimensiones, cuenta con una báscula industrial WANT WT3002L de 300 kg de capacidad y 10 g de resolución, mientras que los depósitos de Ventas y Fugas cuentan con básculas de cocina genéricas, de 5 kg de capacidad y 1 g de resolución.

Independientemente del tipo de báscula, se requiere de una interfaz para la comunicación con la unidad de control. Para ello se emplea el módulo HX711, un convertidor analógico-digital de 24 bits especialmente diseñado para la comunicación con celdas de carga, que interpreta las señales analógicas proporcionales a la fuerza aplicada y las amplifica y convierte en señales digitales.

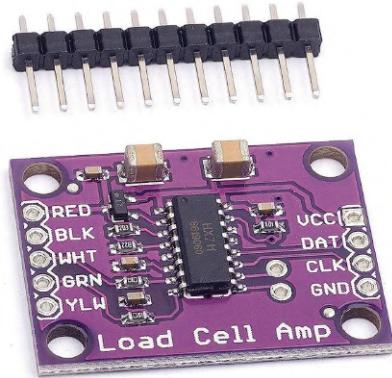


Figura 2.6: Módulo HX711.

- **Sensores de nivel XKC-Y26-V:** En lugar de utilizar una báscula, el depósito de Descargas está equipado con sensores de nivel que permiten determinar su volumen. Estos sensores requieren una alimentación de entrada de 5-24 V DC y operan mediante un principio capacitivo, lo que permite mantenerlos fijos en el exterior del depósito sin contacto directo con el fluido. Este modelo cuenta además con la posibilidad de invertir la lógica de salida a través de uno de sus pines, aunque para este prototipo se ha optado por conservar la lógica positiva.

Se ha colocado un total de cuatro sensores de nivel en el depósito, como se muestra en la figura 2.7, ubicados a alturas conocidas de 1, 5, 20 y 30 L, para la dispensación de volúmenes predeterminados.



Figura 2.7: Sensor de nivel.

Resulta oportuno señalar que los problemas de precisión inherentes a los sensores pueden ser beneficiosos para la generación de datos, dado que, en última instancia, estas limitaciones contribuyen a obtener una simulación más fiel de un entorno físico real.

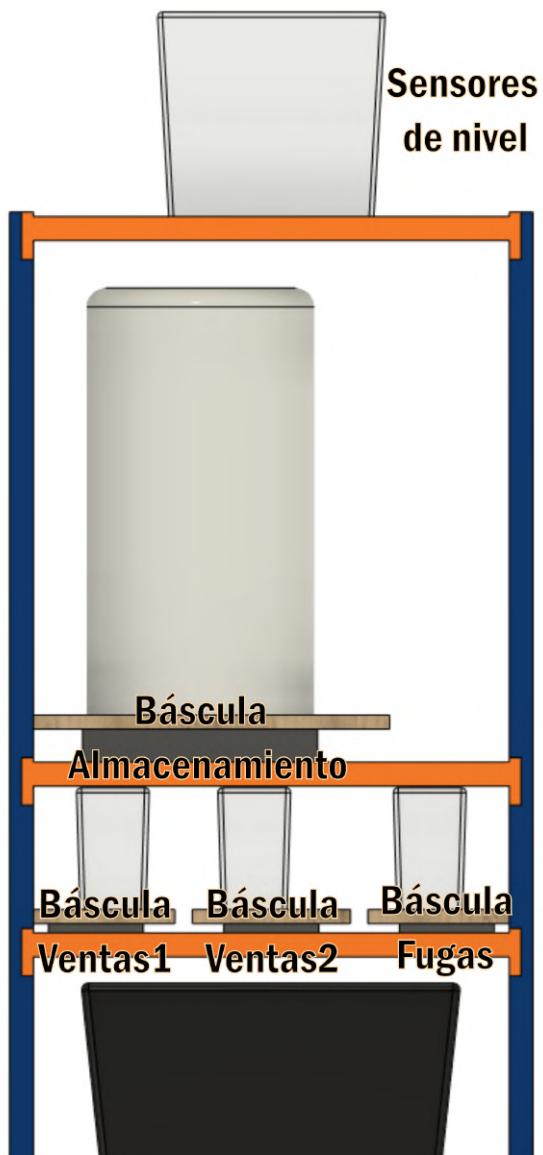


Figura 2.8: Disposición de los sensores.

## 2.3. Actuadores

Entre los actuadores que operan sobre la planta se encuentran:

- **Bomba centrífuga de DC:** Estas bombas sumergibles de 12 V DC y 19 W se caracterizan por un caudal teórico de 800 L/h y una altura de elevación de 5 m. Son empleadas como componente principal para el transporte de agua entre los tanques, ya que son los responsables del vaciado del depósito de Descargas y del llenado y vaciado de los depósitos de Ventas1 y Ventas2.



Figura 2.9: Bomba centrífuga DC.

- **Electroválvula:** Estas electroválvulas operan en conjunción con las bombas previamente mencionadas para facilitar la conducción del agua entre los tanques, ya sea permitiendo su flujo o conteniéndola según las instrucciones recibidas. Todas las bombas centrífugas de DC en el prototipo están equipadas con una electroválvula acoplada a su salida. Además, dado que también cuentan con una alimentación de entrada de 12 V, se hallan conectadas en paralelo con las bombas, por lo que se puede considerar que el binomio actúa como un único componente. Los detalles del cableado se exponen más detalladamente en el apartado 2.6.

Por último, además de las unidades emparejadas a las bombas, hay una electroválvula individual que permite el vaciado del depósito de Fugas, lo que es posible gracias a su capacidad de conmutar incluso bajo la escasa presión ejercida por el fluido. Sin embargo, cabe destacar que esto implica que el vaciado de dicho depósito es más lento en comparación con el resto, lo que no debería suponer un problema si se tiene en cuenta que el depósito de Fugas sólo se vacía al finalizar un experimento. En el apartado 3.1 se proporciona una descripción completa de los procesos de vaciado y llenado de los depósitos.



Figura 2.10: Electroválvula.

- **Bomba peristáltica:** Se trata de una bomba de 12 V DC capaz de administrar agua en cantidades reducidas, lo cual lo convierte en una opción óptima para el abastecimiento del depósito de Fugas. La bomba está equipada con una manguera de 1 mm de diámetro interno y tiene la capacidad de regular el flujo en un rango que oscila entre 2 y 17 mL/min. El caudal específico dentro de este intervalo está sujeto a la señal PWM que se proporciona como entrada a la bomba. Se proporciona más información sobre este aspecto en el apartado 3.1.3.



Figura 2.11: Bomba peristáltica.

- **Bomba centrífuga de AC:** Consiste en una bomba sumergible de 220-240 V AC y 85 W. Presenta un caudal teórico de 3800 L/h y una altura de elevación de 3,8 m, lo cual la hace idónea para su implementación como bomba de Retorno, reemplazando así a la bomba de 12 V DC utilizada en la versión anterior del prototipo.

Con el fin de aumentar aún más la velocidad el llenado del depósito de Descargas, dicha bomba sumergible ha sido equipada con una válvula antirretorno de bola a su salida, que evita el retroceso del flujo de agua una vez ha sido impulsado hacia el depósito.



Figura 2.12: Bomba centrífuga AC.



Figura 2.13: Válvula antirretorno.

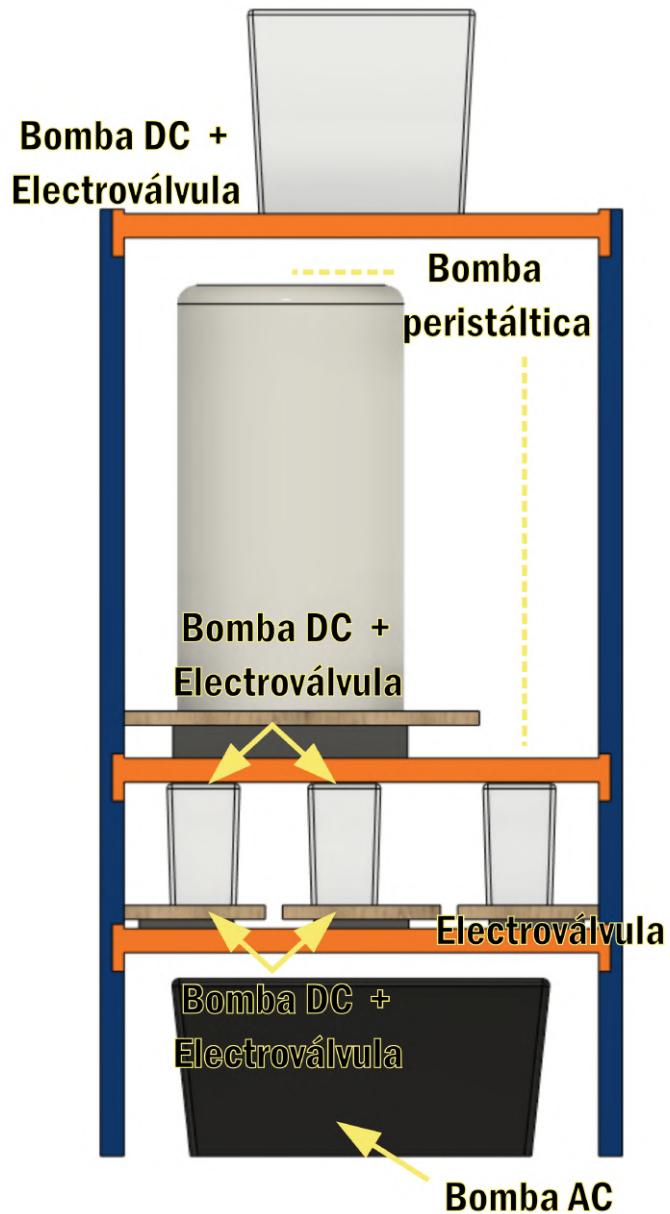


Figura 2.14: Disposición de los actuadores.

## 2.4. Unidad de control

La unidad de control que gobierna el funcionamiento de la planta está compuesta por:

- **Arduino Mega 2560:**

La Arduino Mega 2560 es una placa de microcontrolador basada en el ATmega2560. Proporciona 16 entradas analógicas y 54 pines de entrada/salida digital, de los cuales 15 pueden

utilizarse para salidas PWM y 4 como puertos de comunicación serial UART. La placa también presenta una conexión USB, un conector de alimentación, y un encabezado ICSP. Posee una memoria Flash de 256 KB para el almacenamiento de código (con 8 KB utilizados para el gestor de arranque), 8 KB de SRAM y 4 KB de EEPROM.

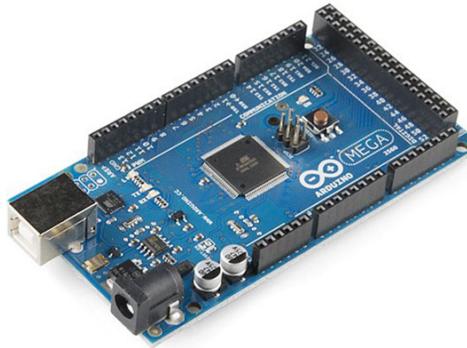


Figura 2.15: Arduino Mega.

Se ha seleccionado la Arduino Mega 2560 como plataforma principal debido a su coste asequible y versatilidad en una amplia variedad de aplicaciones. El valor de la Arduino se enfatiza por una comunidad activa que ofrece un amplio repositorio de librerías, tutoriales y soluciones a problemas comunes, lo que facilita sumamente su implementación.

Para este proyecto, la placa Arduino Mega 2560 se emplea no sólo para la lectura de los sensores y el control de los actuadores conforme al algoritmo desarrollado, sino también para la comunicación con otros dispositivos, tal como se describe en detalle en los apartados 3.3 y 3.4.

Una segunda placa Arduino cumple el papel de unidad de detección de fugas, recibiendo datos de la unidad principal para determinar si hay anomalías. En el ámbito académico, los estudiantes pondrían a prueba diversos algoritmos de detección de fugas para comparar su eficacia. En cualquier caso, debido a limitaciones de tiempo y disponibilidad de materiales, el desarrollo de estos algoritmos no pudo ser abordado durante este proyecto.

#### ■ Módulo de 16 relés:

Dada la incapacidad de la placa Arduino para realizar la conmutación directa de los actuadores, se hace uso de un módulo de 16 relés para controlarlos, independientemente de su voltaje de alimentación.

En este aspecto cabe destacar que las bombas centrífugas y las electroválvulas se conectan al pin normalmente abierto (NO) del módulo de relés. Además, es necesario tener en cuenta que el módulo de relés opera bajo una lógica negada, lo que implica que los pines de control del relé deben mantenerse en estado alto para mantener las bombas apagadas y las electroválvulas cerradas.

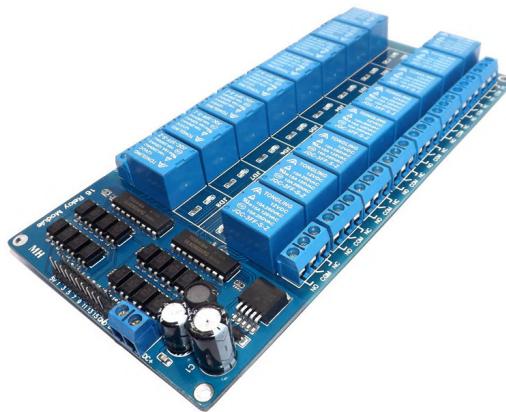


Figura 2.16: Módulo de relés.

■ **Controlador de motores:**

De manera análoga al módulo de relés, el controlador L298N permite la operación de la bomba peristáltica, con la capacidad adicional de controlar tanto la dirección como la velocidad de rotación mediante señales PWM.

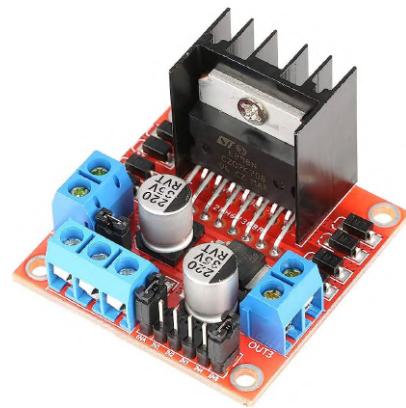


Figura 2.17: Controlador L298N.

Si bien en el contexto de este proyecto la dirección de rotación se mantiene constante para propiciar únicamente el llenado del depósito de Fugas, sí que se desea variar la velocidad de rotación para simular diferentes ratios de pérdida.

En cierto modo se podría considerar que el módulo de 16 relés y el controlador L298N constituyen una interfaz entre la unidad de control y los actuadores.

## 2.5. Panel de control físico

Una de las adiciones al hardware más notables llevadas a cabo es la de un panel de control físico, desarrollado con el fin de facilitar la supervisión en tiempo real de las condiciones de la planta, proporcionar un medio para el control de las operaciones y mejorar así la usabilidad general del dispositivo.

Tal y como se puede observar en la figura 2.18, el panel cuenta con las entradas y salidas siguientes:

- **Pulsadores:** Bombeo, Start y Stop.
- **Conmutadores de dos posiciones (interruptores):** Tara, Calibración, Tipo de ejecución, Cierre y Seta de emergencia.
- **Conmutadores de tres posiciones:** Vaciado y Recirculación.
- **Potenciómetros:** Altura de fuga y Dimensiones de fuga.
- **Leds:** Pilotos del estado del proceso, del estado de los depósitos y de la detección de fuga.

Estos elementos se pueden agrupar según su uso de la siguiente manera:

- **Inicialización:** Entradas encargados de la selección de los pasos de la inicialización que se llevarán a cabo antes de comenzar un experimento. Dicho proceso de inicialización se describe en profundidad en el apartado 3.1.1.
- **Control de proceso:** Incluye los pulsadores de Start y Stop, para comenzar y pausar experimentos, el Dial de ejecución manual o automática, el interruptor de Cierre para finalizar experimentos y la seta de Emergencias. También contiene el pulsador de Bombeo, empleado principalmente durante los procesos manuales de la inicialización.
- **Estado:** Consistente en cuatro leds indicativos del estado general de la planta y cinco leds representativos del estado de cada tanque. El código de luces, diseñado para simbolizar las distintas situaciones en las que se puede encontrar la planta, se explica detalladamente en el apartado 3.2.
- **Control de fugas:** Comprende los potenciómetros que regulan la altura y la magnitud de la fuga, así como el piloto de detección

En lo que a conexionado respecta, cada una de las entradas se ha provisto con una resistencia pull-down de  $10\text{ k}\Omega$ , y cada uno de los leds cuenta con una resistencia de  $330\text{ }\Omega$ . En el anexo III se puede encontrar el pin de la placa Arduino vinculado a cada uno de los elementos del panel de control.

Para el montaje físico del panel se hizo uso de materiales de bajo coste, tales como tableros de MDF y componentes electrónicos de grado comercial. El resultado obtenido se puede observar en las figuras 2.19 y 2.20, así como en los vídeos demostrativos disponibles en el [repositorio](#) de GitHub asociado a este proyecto (<https://github.com/GregorioML/TFG-DeteccionFugas>).

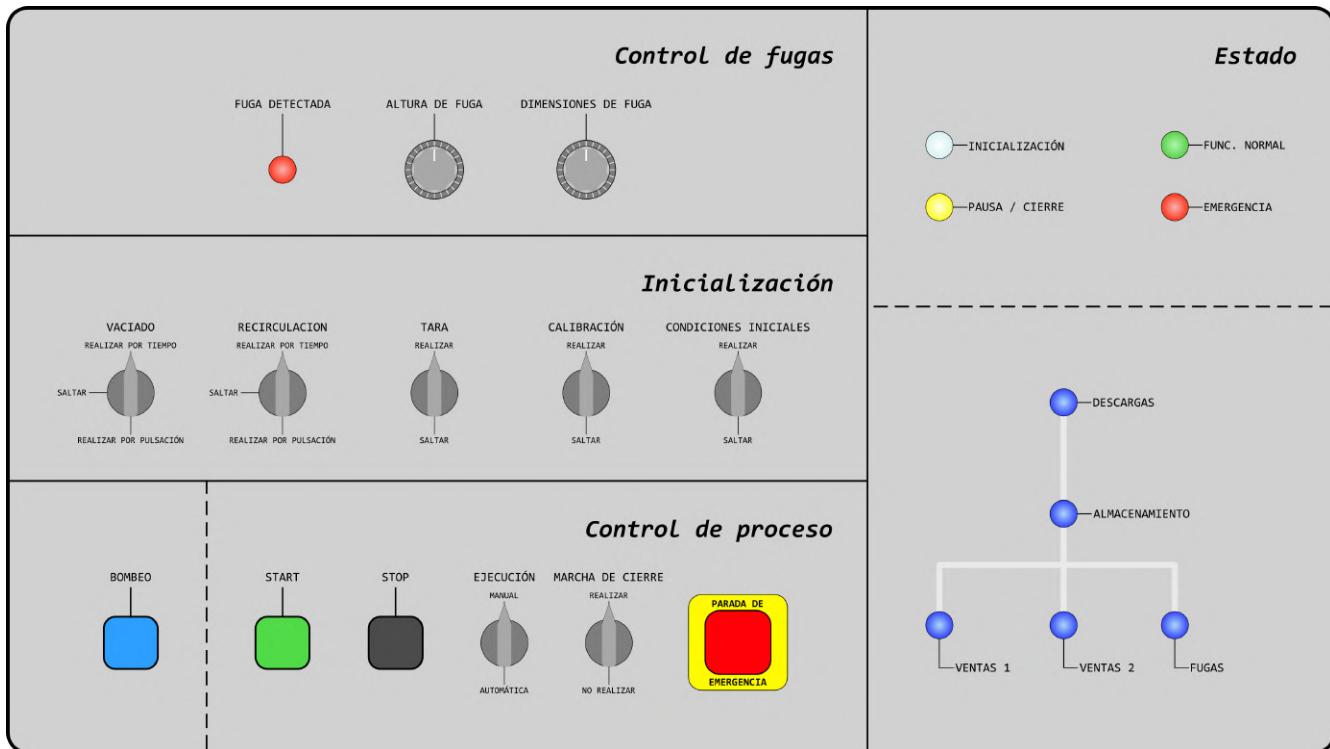


Figura 2.18: Elementos del panel de control.



Figura 2.19: Implementación del panel de control.

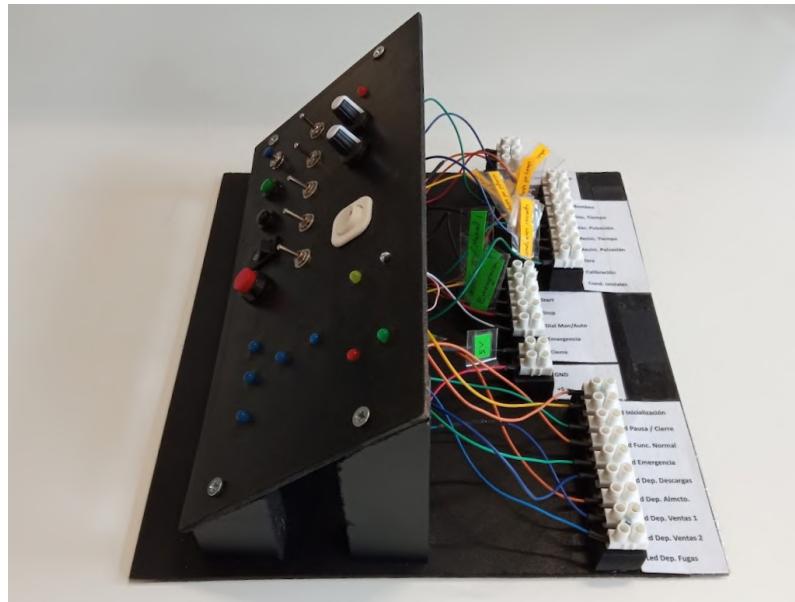


Figura 2.20: Conexiones del panel de control.

## 2.6. Instalación eléctrica y electrónica

Para el presente proyecto, el cableado en su totalidad ha sido renovado con la usabilidad, la seguridad y la flexibilidad en mente.

Si bien para el presente proyecto no se ha podido contar con estos materiales a tiempo, la primera medida tomada para incrementar la seguridad y claridad del cableado consistirá en ubicar cajas estancas en cada balda para aislar las conexiones de los potenciales peligros de la planta, tales como salpicaduras, contacto accidental...

De este modo, en cada balda se instalará una caja estanca para albergar las clemas de alimentación. Además, se dispondrán las cajas necesarias en cada caso para garantizar un almacenamiento seguro de la electrónica, al mismo tiempo que se logra una separación adecuada en función de su uso específico.

A su vez, se implementará protección adicional para el cableado que interconecta cada una de las cajas estancas mediante el uso de canaletas, lo que reforzará aún más el aislamiento. Para el conexionado entre baldas, el cableado se ubicará en el interior de la estructura vertical de la estantería, en la vertical trasera derecha, que no está ocupado por tuberías.

Con ello, se obtendrá la distribución por balda que se observa en las figuras 2.21, 2.22 y 2.23.



Figura 2.21: Cajas estancas de la balda superior.



Figura 2.22: Cajas estancas de la balda media.



Figura 2.23: Cajas estancas de la balda inferior.

Además, para el nuevo diseño se ha optado por mantener la unidad de control separada del resto de componentes, contenida en una caja de equipos eléctricos de uso industrial con protección IP66. Dicho armario se apoyará en el lateral exterior derecho de la estantería, lo que facilitará su acceso en caso de sustitución o adición de componentes.

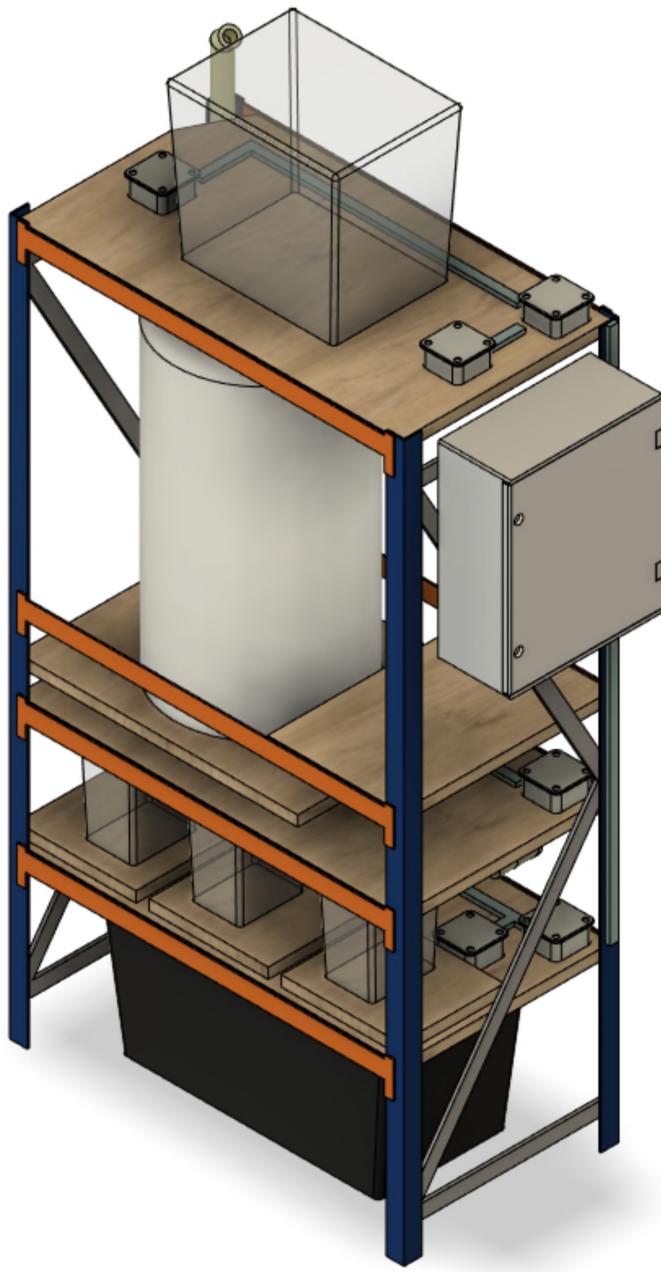


Figura 2.24: Ubicación del armario eléctrico.

Internamente, en el margen derecho de la caja se ubicará la regleta que suministra la alimentación a todos los componentes, mientras que a su izquierda se posicionarán todos los elementos de la unidad de control. En el margen superior se ubicarán las clemas destinadas al conexionado con los sensores y actuadores de cada balda.

En relación a la alimentación, al igual que en la iteración anterior del prototipo, se cuenta con una fuente de 12 V que suministra el voltaje necesario para la mayoría de actuadores y sensores.

Sin embargo, para este proyecto, en lugar de alimentar la placa Arduino con 12 V y usar los 5 V que entrega como alimentación, se ha optado por emplear un regulador externo LM7805 con sus respectivos condensadores de acople y desacople. Esta decisión se basa en la necesidad de aliviar parte de la carga que soporta la placa Arduino, incrementando así su vida útil y mejorando la estabilidad de la señal.

Cabe destacar que, alternativamente, también es posible emplear una segunda fuente que entregue 5 V de manera independiente.

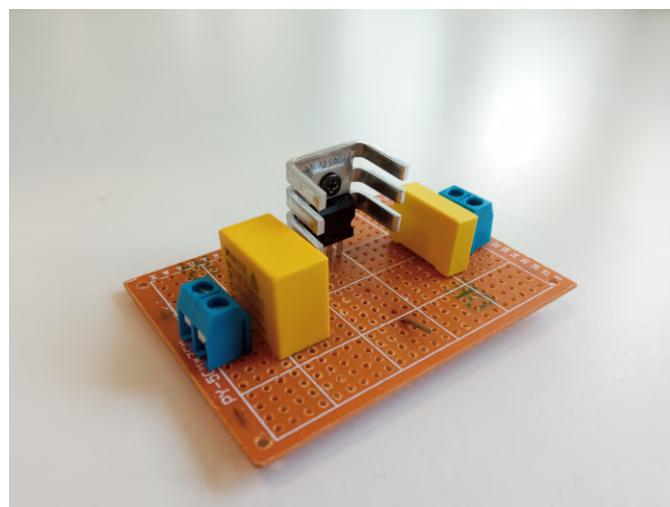


Figura 2.25: Regulador de voltaje de 5 V.

Como medida adicional para incrementar la seguridad de la instalación, se han instalado diodos de descarga de inductancia (*flyback*) en las bombas de DC. Estos diodos se encargan de proteger los componentes sensibles contra los picos de voltaje generados por la inductancia del motor al interrumpir la corriente. Por otro lado, en el caso de la bomba de AC, se ha instalado un varistor que cumple la función de salvaguardar el sistema contra posibles fluctuaciones y sobretensiones indeseadas.



Figura 2.26: Diodo de descarga de inductancia.



Figura 2.27: Varistor

Debido a la indisponibilidad de materiales a tiempo, de manera provisional, se ha dispuesto todo el cableado de la caja eléctrica en un panel de cartón rígido, que constituye un soporte poco protegido, pero suficientemente fiable en el corto plazo.

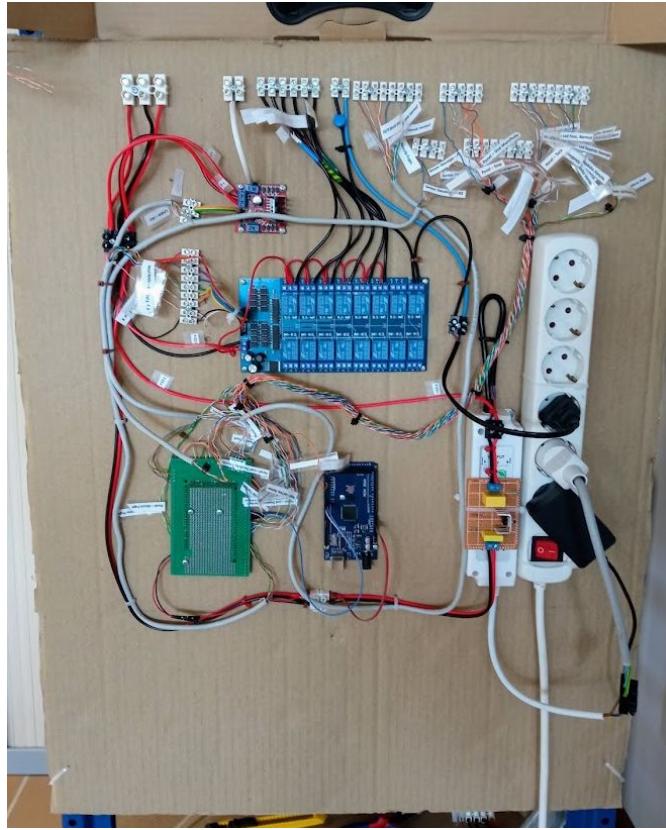


Figura 2.28: Montaje provisional de la unidad de control

De manera similar, se ha llevado a cabo la instalación del cableado de cada balda considerando la posición de las cajas estancas. De este modo, se han colocado las clemas en las ubicaciones previstas para dichas cajas, evitando así conexiones expuestas fuera de estos puntos.

Los esquemas del cableado en su totalidad se pueden encontrar en los anexos IV y V.

### 3. Software

En relación al software, si bien se ha mantenido la idea central del diseño, se han llevado a cabo múltiples modificaciones y adiciones importantes que se detallan a continuación.

#### 3.1. Controladora

Al igual que en el diseño original, el sistema en su totalidad se rige por dos máquinas de estado de tipo Moore, que controlan respectivamente el estado de ejecución general y la gestión individual de los depósitos.

##### 3.1.1. Inicialización

Una de las principales adiciones que se ha realizado en el código de la controladora es el diseño de un proceso de inicialización que agiliza el arranque de los experimentos. Este proceso de inicialización se lleva a cabo después de encender el dispositivo o tras una situación de emergencia, garantizando así que se inicie un experimento desde un estado conocido y adecuado para la simulación.

El proceso de inicialización consta de siete etapas que se exponen a continuación.

###### ■ Pre-Vaciado:

Al comenzar la Inicialización, el sistema entra en un estado en que se inicializan las variables y se espera una entrada desde el panel de control para dar inicio al proceso de vaciado. El panel de control físico, mediante el cual se gestiona este proceso, ha sido presentado en detalle en el apartado 2.5.

###### ■ Vaciado:

En esta etapa, se presentan dos posibilidades dependiendo de la selección realizada en el panel de control. En el caso del vaciado por pulsación, se abren las válvulas de salida del depósito seleccionado mientras el botón de bombeo se mantenga presionado. El usuario tiene la opción de cambiar el depósito seleccionado utilizando el botón de Start y puede indicar que el vaciado ha finalizado mediante el botón de parada Stop. En el modo de vaciado por tiempo, se abrirán todas las válvulas de salida durante un período de tiempo predeterminado.

###### ■ Recirculación:

El propósito de esta etapa es garantizar que el nivel de agua en cada depósito alcance al menos la altura de la válvula de salida correspondiente. La etapa de Recirculación presenta similitudes con la etapa de Vaciado, pero además se abren las válvulas 'aguas arriba' y 'aguas abajo' de los depósitos. Esto implica que, en el modo de vaciado por pulsación, el usuario puede elegir recircular a través del depósito de Ventas1, Ventas2 o Fugas.

■ **Tara:**

La etapa de Tara tiene como objetivo establecer el peso actual registrado por cada báscula como referencia para el resto de la ejecución.

■ **Calibración:**

Durante esta etapa, se lleva a cabo el llenado del depósito de Descargas en cada uno de sus niveles con el propósito de realizar su posterior dispensación y pesaje en el depósito de Almacenamiento. De esta manera, es posible determinar con precisión el volumen asociado a cada uno de los sensores de nivel.

■ **Puesta en Condiciones iniciales:**

Finalmente, en esta etapa se llevan a cabo dos tareas:

- Se depositan márgenes de agua en todos los depósitos para evitar que las bombas centrífugas operen en vacío durante el experimento.
- Se establecen niveles aleatorios en los depósitos de Descargas y Almacenamiento, lo cual enriquece el valor de los datos generados al introducir variabilidad en las condiciones iniciales.

■ **Pre-Funcionamiento Normal:**

De manera similar a la etapa de Pre-Vaciado, al concluir la etapa de Puesta en Condiciones Iniciales, el sistema se mantiene a la espera de una entrada proveniente del panel de control para dar inicio al experimento.

Cabe destacar que, del mismo modo en que es posible seleccionar la modalidad de Vaciado y Recirculación mediante las entradas del panel de control, también se brinda la opción de omitir cada una de las etapas de la Inicialización. Esto permite al usuario ejecutar únicamente las etapas estrictamente necesarias para comenzar el experimento en un estado conocido en función de la situación previa de los depósitos, evitando así dispensaciones innecesarias, con el consecuente ahorro de tiempo.

Sin embargo, cabe resaltar que existen etapas en la inicialización que en ninguna circunstancia sería recomendable omitir, tales como la etapa de Tara. A pesar de lo cual, se ha decidido permitir la opción de omitirlas todas por motivos didácticos, para que los estudiantes que operen con la planta puedan comprender las implicaciones de cada una de las decisiones de diseño inherentes a cada etapa.

Si bien el proceso de inicialización no constituye un proceso independiente con su propia máquina de estados, sino que forma parte de la máquina de estados general de la controladora, en el anexo VI se proporciona un diagrama de las etapas que lo componen.

Por último, cabe destacar que cada una de las etapas y subetapas de la inicialización se gestionan mediante llamadas a funciones desde la controladora, como se puede apreciar en el [repositorio](#) de GitHub de este proyecto. Este enfoque permite una encapsulamiento del código que facilita la legibilidad y la edición de funcionalidades.

### 3.1.2. Máquina de estado general (Controladora)

La máquina de estados general actúa como controlador principal del proceso, con lo que determina el estado de la ejecución y gestiona las máquinas de estados de los depósitos.

Esta máquina de estados se implementa en el bucle principal Controladora.ino, y se compone por los siguientes estados:

- **Inicialización:**

Estado en el que, tal y como se expuso en el apartado 3.1.1, permite comenzar el experimento bajo condiciones conocidas.

- **Funcionamiento Normal:**

Se trata del estado principal en el que, de manera cíclica, se ordena el llenado y vaciado de los depósitos para la extracción de datos. Esto se realiza a través de los métodos de depósito *actualizar\_estado()*, como se expone en el apartado 3.1.3.

- **Pausa:**

Al presionar el pulsador de Pausa en el panel de control, el sistema entrará en este estado, en el que cerrará todas sus válvulas y se mantendrá a la espera de una entrada del pulsador Start, momento en el que continuará el proceso desde el punto en el que se detuvo.

Una de las modificaciones realizadas en el código consiste en la incorporación del seguimiento del tiempo transcurrido durante el estado de pausa. De esta manera, dicho tiempo se tiene en cuenta al reanudar la ejecución, asegurando un funcionamiento de acuerdo al diseño.

En la versión anterior del proyecto, nada más reanudar la marcha tras una pausa de cierta duración, con total seguridad se darían nuevas dispensaciones que en condiciones normales hubieran sucedido en un momento posterior. Este comportamiento que no respeta la dinámica del proceso diseñado y supone una alteración de los datos generados, por lo que se ha corregido contabilizando dicho tiempo e incluyéndolo como parámetro de entrada en el método *actualizar\_estado()* de los depósitos.

- **Fin de experimento o Cierre:**

Este estado ha sido añadido en esta versión del proyecto con el propósito de finalizar los experimentos de manera controlada. Al activar el interruptor de Cierre en el panel de control, se detendrá el registro de datos y los depósitos comenzarán a vaciarse.

Adicionalmente, se ha implementado la capacidad de configurar un tiempo máximo de simulación. Una vez transcurrido este tiempo, el estado del proceso cambiará automáticamente a Fin de experimento, permitiendo así, una vez más, un cierre programado y controlado de la simulación.

- **Emergencia:**

Cuando se entra al estado de Emergencia, ya sea por la activación de la seta de emergencia o por alcanzar un volumen crítico en alguno de los depósitos, se procede al cierre de todas las válvulas y se mantiene el sistema en espera. En este estado, se aguarda una confirmación

a través del panel de control para reiniciar el proceso de inicialización y retomar así el funcionamiento normal del sistema.

Las transiciones entre los estados se detallan en la figura 3.1 y en el anexo VII. El código correspondiente a esta máquina de estados se puede encontrar en el en el [repositorio](#) de GitHub de este proyecto.

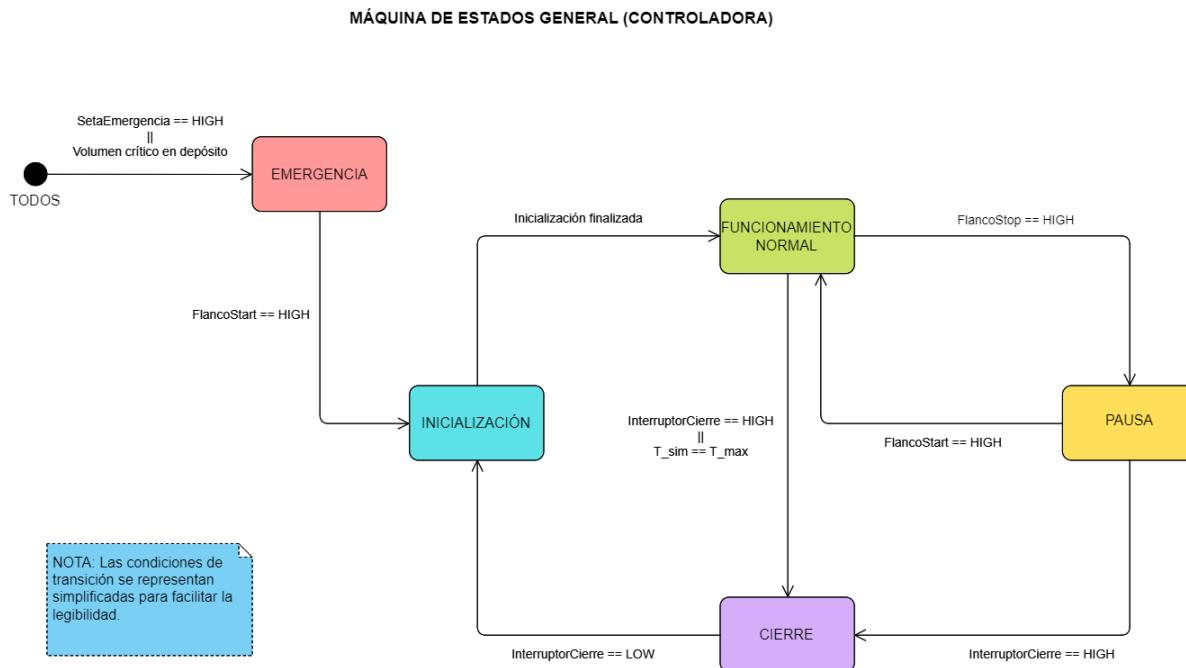


Figura 3.1: Máquina de estados general.

### 3.1.3. Máquina de estado de depósitos

Esta máquina de estado gestiona de forma individual el proceso de llenado y vaciado de cada uno de los cuatro depósitos activos, que incluyen los depósitos de Descarga, Ventas y Fugas. En contraste, los depósitos de Almacenamiento y Retorno se consideran pasivos, ya que su principal función consiste en almacenar y dispensar volúmenes de agua según las necesidades de los depósitos activos.

La máquina de estados de los depósitos se implementa a través de la clase Deposito.cpp, y se instancia para cada depósito. Para realizar la transición entre los estados, se invoca el método *actualizar\_estado()* de cada depósito.

A continuación, se presenta los estados que la conforman:

■ **Comprobación Inicial:**

En este estado, al comienzo de un experimento, se consulta el volumen de cada depósito. Los depósitos que contienen volumen, transicionan al estado Tanque Lleno, mientras que el resto pasan al estado Tanque Vacío. La finalidad de esta comprobación es determinar el volumen aleatorio que se le fue asignado al depósito de Descargas durante la etapa final de la Inicialización. Además, esta medida adicional de seguridad contribuye a prevenir desbordamientos en todos los depósitos.

■ **Tanque Vacío, Llenado, Tanque Lleno y Vaciado:**

Se tratan de los cuatro estados que, durante el funcionamiento normal de la controladora, se recorren cíclicamente para ejecutar las dispensaciones en cada depósito.

Durante el estado de vaciado, se determina el volumen de la próxima dispensación y el instante en el que se producirá utilizando los métodos *GenerarVolumenDisp()* y *GenerarInstanteProxDisp()*. En el caso de los depósitos de Ventas y Descargas, se generan de forma cíclica múltiples volúmenes e instantes a lo largo de la ejecución del experimento. Por otro lado, en el depósito de Fugas, se genera una dispensación continua que comienza transcurrido un lapso de tiempo aleatorio tras el comienzo del experimento.

Cabe destacar que este comportamiento del depósito de Fugas es una de las nuevas adiciones al código. En la versión de partida, el depósito de Fugas se llenaba desde el comienzo del experimento, lo cual disminuye la calidad de los datos generados y el valor del algoritmo de detección de fugas.

Otra mejora en este aspecto es la implementación de un sistema para controlar el caudal de fugas a través de un potenciómetro del panel de control. Esta funcionalidad permite simular diferentes dimensiones del orificio de fugas.

Asimismo, también se ha implementado la posibilidad de controlar la altura del orificio mediante un potenciómetro adicional en el panel de control. De este modo, la bomba peristáltica se activará sólo cuando el nivel del depósito de Almacenamiento sea superior al valor establecido por dicho parámetro.

■ **Fin experimento o Cierre:**

Cuando la máquina de estados general pasa a Fin de Experimento, se le es comunicado a la máquina de estado de cada depósito a través de un parámetro de entrada, lo que le lleva a meterse en un estado del mismo nombre. En este estado se vacían todos los depósitos y se mantienen a la espera de que la controladora reinicie el proceso.

Una vez que la máquina de estados general transita al estado Fin de Experimento, se le es comunicado a la máquina de estado de cada depósito mediante un parámetro de entrada. Esto provoca que todas las máquinas de estado de los depósitos pasen al estado Fin de Experimento, en el que se procede al vaciado completo de todos los tanques y se espera a que la controladora reinicie el proceso.

■ **Emergencia:**

En el caso de que se alcance un volumen crítico en un depósito, la máquina de estados asociada entrará en estado de Emergencia. En este estado, se procederá al cierre de todas las válvulas y se pondrá el sistema en espera. Además, el método `actualizar_estado()` correspondiente generará una señal que será recibida por la controladora, lo que provocará que todas las máquinas de estado, incluida la controladora misma, entren en estado de emergencia.

Así, se puede apreciar que el sistema en su conjunto presenta bastante redundancia en relación a la detección de emergencias, pues el ingreso de una máquina de estados en el estado de emergencia conlleva a que el resto se metan en el mismo.

Con ello, se puede apreciar que el sistema en su totalidad exhibe una notable redundancia en cuanto a la detección de emergencias, lo que asegura una mayor robustez y fiabilidad en situaciones críticas.

La máquina de estados de los depósitos, con todas sus transiciones, se puede observar en la figura 3.2 y el anexo VIII, mientras que el código correspondiente se encuentra en el [repositorio](#) de GitHub de este proyecto.

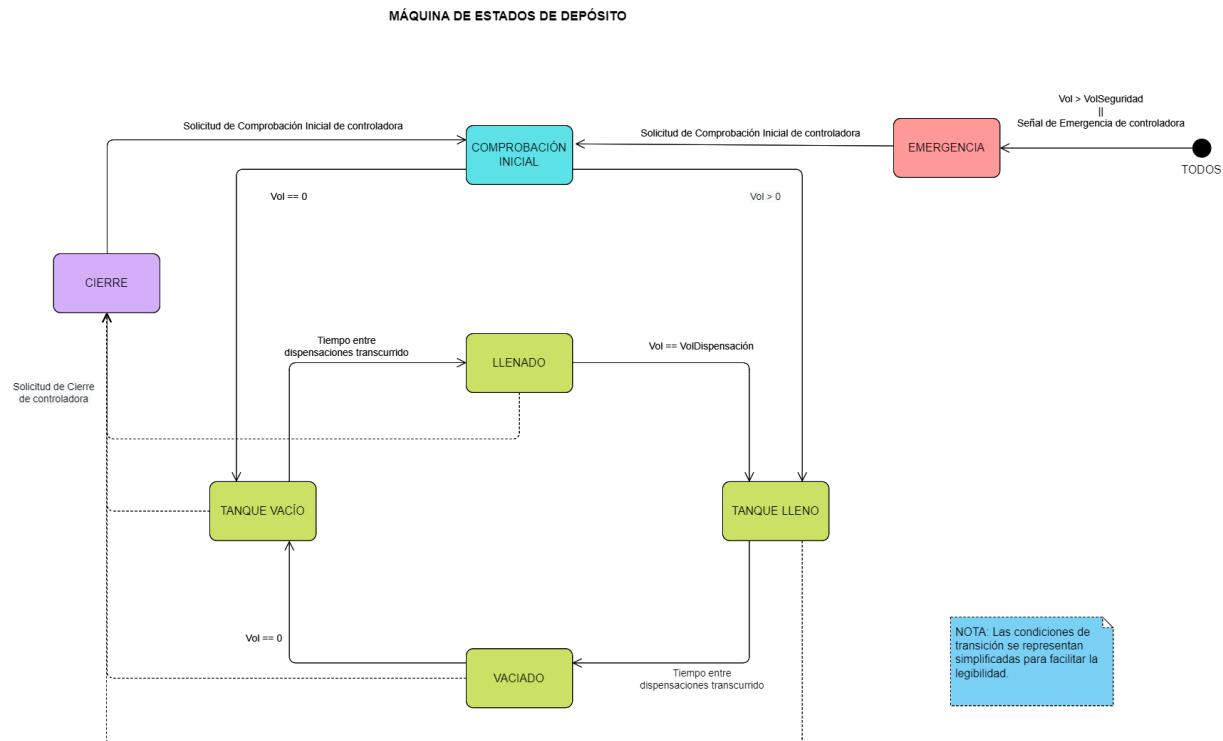


Figura 3.2: Máquina de estados de depósito

### 3.2. Clase PanelControl

Como complemento al panel de control físico expuesto en el apartado 2.5, se ha desarrollado una clase de Arduino para facilitar su uso, proporcionando métodos para la consulta de entradas y control de salidas.

La clase PanelControl está diseñada para ser instanciada con cada elemento del panel físico, de modo que acciones como la lectura de entradas y el encendido o apagado de salidas se gestionan a través de métodos de esta clase. Así, en el constructor de la clase se especifica el pin de Arduino al que el elemento está conectado, así como si se trata de una entrada o una salida. Una vez instanciada, se pueden llamar los métodos que se exponen a continuación.

Para el caso de las entradas del panel, se encuentran disponibles los siguientes métodos:

- **ComprobarEstado()**: Devuelve el estado actual de una entrada, como interruptores y pulsadores (HIGH o LOW).
- **ComprobarFlanco()**: Devuelve el flanco que se haya producido en la entrada, ya sea de subida o de bajada (1, 0 o -1). El tiempo de rebote para ajustar la sensibilidad del flanco es un atributo modificable.
- **LeerValor()**: Devuelve la lectura de una entrada analógica, como potenciómetros, en el rango 0-1023.

Por otro lado, para el caso de las salidas del panel, se disponen de los métodos siguientes:

- **Encender()**: Realiza una escritura digital de la salida a HIGH.
- **Apagar()**: Realiza una escritura digital de la salida a LOW.
- **Parpadear()**: Enciende y apaga la salida a una frecuencia dependiente del parámetro de entrada TipoParpadeo. Se han implementado tres posibilidades para este parámetro (*PARPADEO\_LENTO*, *PARPADEO\_RAPIDO* y *PARPADEO\_ERROR*), de modo que el led pueda representar diferentes situaciones que se detallan en el apartado 3.2.1.
- **EncenderPorNivel()**: Realiza una escritura analógica de la salida, de modo que el led se enciende con una intensidad dependiente del porcentaje llenado del depósito. Existen cinco niveles de luminosidad, además de una función de parpadeo que se activa cuando el nivel del depósito está próximo a su capacidad máxima.

Cabe destacar que, tanto para la detección de flancos de un pulsador como para el parpadeo de un led es necesario considerar el tiempo transcurrido con respecto a una referencia, por lo que muchos de los métodos son invocados con la variable InstanteActual como parámetro de entrada.

#### 3.2.1. Funciones *ActualizarEntradasPanel()* y *ActualizarSalidasPanel()*

Para el uso de la clase PanelControl en este proyecto, se instancia, antes del *setup()* del programa principal, un objeto por cada uno de los elementos del panel.

Posteriormente, al principio de cada ciclo del *loop()*, se invoca la función *ActualizarEntradasPanel()*, que, haciendo uso de los métodos anteriormente presentados, almacena todos los estados, flancos y lecturas del panel en variables que se consultarán durante la ejecución del programa.

De manera análoga, al final del bucle principal, se llama la función *ActualizarSalidasPanel()*, que actualiza el estado de los leds del panel en función del estado del proceso y del volumen de los depósitos. En esta función se distinguen dos secciones: una para el control de los cuatro leds de estado y otra para la gestión de los cinco leds de depósito.

Para representar el estado actual de la controladora, los leds de Inicialización, Funcionamiento Normal y Pausa/Cierre permanecen encendidos con luz fija, mientras que el led de Emergencia parpadea rápidamente. La gestión de los leds de depósitos es algo más compleja, pues representan distinta información en función del estado de la controladora:

- Durante los estados de espera previo y posterior a la inicialización, los leds parpadean lentamente.
- Durante la etapa de vaciado de la inicialización, en caso de realizarse por tiempo, se iluminan los leds actualmente vacíandose. En caso de llevarse a cabo por pulsación, el led correspondiente al depósito seleccionado para vaciar parpadea lentamente. Mientras se presione el pulsador de bombeo para su vaciado, el led permanecerá encendido con luz fija.
- La etapa de recirculación de la inicialización es similar al vaciado, con la particularidad de que también se encienden los leds 'aguas abajo' del seleccionado, indicando que dichas válvulas también se abrirán.
- Durante la calibración, se indica con una luz fija el abastecimiento del depósito de Descargas y con un parpadeo lento su vaciado. El led de Almacenamiento se ilumina durante su llenado y parpadea lentamente durante su pesaje y evacuación, al igual que el depósito de Ventas1 a través del cual se desaloja.
- En la etapa de establecimiento de condiciones iniciales, durante la dispensación de márgenes, el led de Almacenamiento se enciende durante su llenado y parpadea lentamente durante el abastecimiento de los márgenes de depósitos de Ventas y Fugas. Los leds de dichos depósitos inferiores se mantienen encendidos una vez se ha vertido su volumen de margen. Posteriormente, durante la dispensación de volúmenes aleatorios, los leds de Descargas y Almacenamiento parpadean lentamente durante su carga y se mantienen encendidos una vez alcanzan el nivel establecido.
- Por último, durante los estados de funcionamiento normal, pausa, fin de experimento y emergencia, los leds se mantienen encendidos con un nivel de luminosidad proporcional a su volumen, haciendo uso del método *EncenderPorNivel()*, tal y como se expuso en el apartado 3.2.

### 3.3. Cuadro de mando digital

Además del diseño del panel de control físico expuesto en el apartado 2.5, se ha diseñado un cuadro de mandos digital (*dashboard*) que facilita una interacción más avanzada con la planta, permitiendo obtener información más detallada del experimento (tal como el tiempo transcurrido, los volúmenes de los depósitos, la cantidad de dispensaciones...) y el envío de instrucciones a la unidad de control.

Además, el cuadro de mandos permite leer los mensajes de depuración provenientes de la Arduino y llevar un registro permanente de los mismos.

#### 3.3.1. Funcionamiento del *dashboard*

Las herramientas seleccionadas para el desarrollo del cuadro de mandos son las librerías de Python Plotly [9], para la creación de gráficos interactivos , y Dash [10], para la construcción de la aplicación web interactiva en la que se mostrarán los gráficos.

Además, se hace uso de la librería Pyserial [11] para facilitar la comunicación a través de puertos serie desde la placa Arduino Mega.

Con dichas herramientas, la construcción del cuadro de mandos comienza con la definición de la estructura de la aplicación. Esto se realiza mediante *dash.html*, módulo que proporciona clases para todas las etiquetas HTML (como *html.div*), empleadas para crear contenedores en la página web. Los contenedores pueden alojar componentes de Dash, incluyendo gráficos, tablas y entradas. La disposición de estos elementos en la página se gestiona a través del estilo CSS asignado a cada contenedor *Div*.

Por otro lado, para actualizar la información visualizada, se hace uso de la funcionalidad *callback*, una función que se dispara en respuesta a un cambio en el estado de la aplicación, como la interacción del usuario con un elemento de entrada o la llegada de nuevos datos. Los *callbacks* se definen utilizando la función *app.callback*, vinculando un componente de entrada a un componente de salida. Así, en caso de un cambio en el valor del componente de entrada, el *callback* se activa y recalcula la salida basándose en la nueva entrada. Este proceso permite la actualización automática del tablero de instrumentos sin la necesidad de recargar la página.

Para la recepción de datos, se emplea la librería Pyserial, con la que se leen e interpretan los datos desde puerto serie. Los mensajes enviados por la Arduino que comiencen por la palabra 'Debug' se almacenan en un vector de mensajes de depuración para su posterior presentación en el *dashboard* y su registro en un documento de texto. El resto de mensajes, que contienen valores numéricos, son segmentadas utilizando una coma como delimitador y posteriormente se guardan en las variables correspondientes destinadas para la visualización en el cuadro de mandos.

### 3.3.2. Información presentada

En cuanto a la disposición de los elementos en el *dashboard* diseñado, se ha optado por mostrar la información separada en dos pestañas:

- **Monitorización de la planta:**

Aquí se presenta la información principal del proceso. Tal y como se puede observar en la figura 3.3, en el margen izquierdo se representa gráfica y numéricamente el volumen actual de cada depósito, además de la altura del orificio de fugas junto al caudal de pérdidas.

En el lado derecho se presenta, en forma tabular, el instante de la próxima dispensación y la cantidad de dispensaciones de cada depósito. Además, se encuentra un gráfico circular que representa el volumen acumulado de las dispensaciones.

En la parte inferior se representa un histórico de los volúmenes de dispensación de cada depósito, mientras que en el margen superior se muestra el tiempo transcurrido en el experimento.



Figura 3.3: Elementos del cuadro de mandos digital.

Por último, en la esquina inferior izquierda se puede encontrar una entrada designada como seta de emergencia. Al ser presionada, se envía a la unidad de control una señal de emergencia idéntica a la del panel de control físico.



Figura 3.4: Emergencia enviada desde el cuadro de mandos digital.

Cabe destacar que estos datos del experimento también son enviados por red local a otros dispositivos, lo que se aborda en el apartado 3.4.

#### ■ Mensajes de depuración:

Esta pantalla presenta los últimos 15 mensajes de depuración enviados por la Arduino. Aunque sólo se visualicen los mensajes más recientes, todos los mensajes de depuración se registran en un archivo de texto que no se borra al finalizar la ejecución. Esto asegura que la información del experimento se conserve para su análisis posterior.



```

Mensajes recibidos por comunicación serial

Debug N1 - 70827. Fin línea Debug
Debug N1 - 71465. Fin línea Debug
Debug N1 - 72107. Fin línea Debug
Debug N1 - 72750. Fin línea Debug
Debug N1 - 73388. Fin línea Debug
Debug N1 - 74031. Fin línea Debug
Debug N1 - 74671. Fin línea Debug
Debug N1 - 75313. Fin línea Debug
Debug N1 - 75952. Fin línea Debug
Debug N1 - 76594. Fin línea Debug
Debug N1 - 77235. Fin línea Debug
Debug N1 - 77877. Fin línea Debug
Debug N1 - 78518. Fin línea Debug
Debug N1 - 79160. Fin línea Debug
Debug N1 - 79801. Fin línea Debug

```

Figura 3.5: Pestaña de mensajes de depuración.

En el apartado 3.5 se puede encontrar una explicación más detallada de los mensajes de depuración recibidos. Por último, resulta significativo señalar que para ejecutar y mostrar el cuadro de mandos se hace uso de un miniPC PIPO X10S con el sistema operativo Windows 10.

### 3.4. Comunicación en red

Además de presentar los datos gráficamente en el cuadro de mandos digital, se ha desarrollado un sistema para su envío a otros dispositivos en la misma red, los cuales pueden utilizar dicha información para almacenarla, analizarla y utilizarla en distintas aplicaciones.

#### 3.4.1. Funcionamiento del protocolo *MQTT*

El sistema diseñado hace uso de las comunicaciones *MQTT* (*Message Queuing Telemetry Transport*), un protocolo de mensajería ligero diseñado para la comunicación máquina a máquina (*M2M*) que constituye un estándar en el Internet de las cosas (*IoT*). Se trata de un protocolo eficiente en lo que ha ancho de banda y uso de energía se refiere, además de lo cual es capaz de manejar conexiones intermitentes.

El protocolo *MQTT* permite la transmisión de mensajes entre dispositivos a través de un modelo de publicación-suscripción. Esto implica que los dispositivos pueden 'publicar' (enviar) información en un 'tema' (o *topic*) específico, mientras que otros dispositivos pueden 'suscribirse' a estos temas para recibir la información. Cabe señalar que un mismo dispositivo puede cumplir

los roles de publicador y suscriptor según las necesidades.

Además, es necesaria la figura de un bróker como intermediario que facilita la transmisión de mensajes entre los publicadores y los suscriptores. El broker recibe todos los mensajes de los publicadores, y los reenvía a los dispositivos que se han suscrito a los temas correspondientes. El bróker también gestiona las sesiones de los clientes, incluyendo aspectos como la autenticación y el mantenimiento del estado de la sesión.

### 3.4.2. Configuración del bróker

Para posibilitar el envío y recepción de datos, se ha seleccionado el bróker Mosquitto [12] como gestor de comunicaciones. Se trata de un bróker ampliamente utilizado por su ligereza, rendimiento y fiabilidad, que además ofrece opciones de autenticación que permiten implementaciones seguras, lo que lo hace apto para todo tipo de dispositivos.

La instalación del bróker se realiza en el miniPC PIPO, que actuará como servidor. Tras la instalación, se establece un protocolo de autenticación de usuario en el que cada cliente debe suministrar un identificador válido con una contraseña. Esta medida garantiza la seguridad del servidor frente a accesos no autorizados permitiendo únicamente que los clientes con credenciales válidas puedan publicar o suscribirse a los temas.

Posteriormente, el servidor *MQTT* se configura para recibir conexiones externas, permitiendo que los clientes fuera de la red local se conecten. Para salvaguardar contra amenazas potenciales, se definen reglas de cortafuegos para permitir tráfico únicamente a través del puerto específico empleado por *MQTT*, en este caso, el puerto 1883. Este procedimiento limita el acceso al servidor a aquellos clientes que se conecten mediante este puerto en particular, reduciendo así la posibilidad de ataque potencial.

Por último, se cifran las contraseñas de los usuarios de modo que queden almacenadas de forma segura e ilegible, salvaguardando así la información de autenticación de los usuarios.

En resumen, para llevar estas medidas a cabo se ha de añadir en el archivo de configuración de Mosquitto, las siguientes instrucciones

```
1      allow_anonymous false
2      password_file passwordfile.txt
```

y sustituyendo 'listener 1883 localhost', por:

```
1      listener 1883 0.0.0.0
```

Posteriormente, en la línea de comandos de Mosquitto, se añaden los usuarios y contraseñas, a través de las siguientes operaciones:

```
1      mosquitto_passwd passwordfile.txt usuario_publicador
2      fuga_publicador
3      mosquitto_passwd passwordfile.txt usuario_suscriptor
4      fuga_suscriptor
```

Y finalmente, antes de iniciar la comunicación, se reinicia el bróker con el siguiente comando:

```
1 mosquito -v -c mosquitto.conf
```

### 3.4.3. Envío de datos

Para emplear el protocolo *MQTT* en Python, se puede hacer uso de la librería paho-mqtt [13], que proporciona todas las funcionalidades necesarias para crear clientes capaces de publicar mensajes a un bróker, suscribirse a temas y recibir mensajes de estos.

Haciendo uso de dicha librería, es posible enviar al bróker los datos que se reciben por puerto serie en el *dashboard*.

Para ello, los datos almacenados en el diccionario *cleaned\_data\_dict* se convierten a formato JSON mediante el método *json.dumps()*. Así, el conjunto de datos se serializa a una cadena de texto en formato JSON, la cual se publica en el tema almacenado en la constante *TOPIC* a través del comando *client.publish()*.

Esta publicación permite que cualquier cliente suscrito al tema especificado por *TOPIC* en el bróker *MQTT* reciba estos datos.

### 3.4.4. Recepción de datos

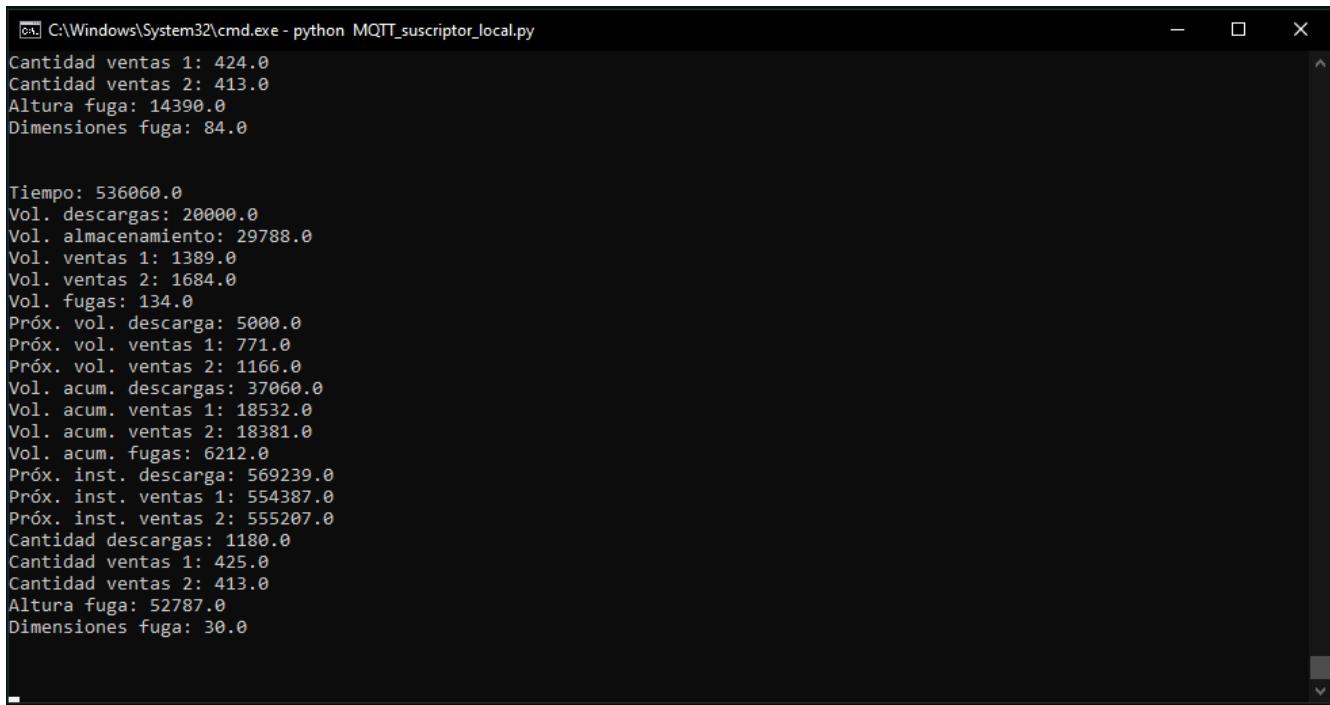
Si bien es posible suscribirse a un tema a través de la línea de comandos con la instrucción

```
1 mosquitto_sub -h host -t tema -u usuario -P contrasena
```

se ha creado un script de recepción de datos como prueba de concepto para un sistema capaz de recibir la información de la planta y procesarla para usos más complejos.

Dicho script, que se puede encontrar en el [repositorio](#) de GitHub de este proyecto, establece y autentica una conexión con el bróker usando los datos del mismo y las credenciales correspondientes, para posteriormente suscribirse al tema *MQTT* definido por la constante *TOPIC*. Una vez se establece la conexión con el bróker, se entra en un bucle de procesamiento de mensajes que mantiene al cliente escuchando y procesando los mensajes de manera continua.

Cuando se recibe un mensaje en el tema suscrito, se activa la función *on\_message*. Este mensaje, codificado como una cadena JSON, es decodificado y cargado en un objeto de Python, para luego imprimir los diferentes valores de sus campos.



```
C:\Windows\System32\cmd.exe - python MQTT_suscriptor_local.py
Cantidad ventas 1: 424.0
Cantidad ventas 2: 413.0
Altura fuga: 14390.0
Dimensiones fuga: 84.0

Tiempo: 536060.0
Vol. descargas: 20000.0
Vol. almacenamiento: 29788.0
Vol. ventas 1: 1389.0
Vol. ventas 2: 1684.0
Vol. fugas: 134.0
Próx. vol. descarga: 5000.0
Próx. vol. ventas 1: 771.0
Próx. vol. ventas 2: 1166.0
Vol. acum. descargas: 37060.0
Vol. acum. ventas 1: 18532.0
Vol. acum. ventas 2: 18381.0
Vol. acum. fugas: 6212.0
Próx. inst. descarga: 569239.0
Próx. inst. ventas 1: 554387.0
Próx. inst. ventas 2: 555207.0
Cantidad descargas: 1180.0
Cantidad ventas 1: 425.0
Cantidad ventas 2: 413.0
Altura fuga: 52787.0
Dimensiones fuga: 30.0
```

Figura 3.6: Recepción de datos en suscriptor.

### 3.5. Clase Debug

Como complemento adicional a todas las modificaciones y adiciones presentadas en materia de programación, y como medio para facilitar el desarrollo de nuevas funcionalidades y detección de errores, se ha desarrollado una clase Debug dedicada al envío de mensajes por puerto serial.

Esta clase permite enviar un mensajes personalizado y una lista de datos del experimento, cuya extensión depende del atributo TipoDebug, que representa el nivel de depuración de los cinco posibles. En la siguiente tabla se puede consultar qué datos son mostrados en cada nivel de depuración:

Se debe tener en cuenta que en cada uno de los niveles de depuración, se envía la lista de datos correspondientes a dicho nivel, junto a los mensajes de todos los niveles previos.

El envío de los mensajes como tal se lleva a cabo mediante el método *EnviarMensajeDebug()*, que se llama con el string del mensaje personalizado como parámetro de entrada.

Si bien al instanciar la clase se establece un nivel de depuración predeterminado, es posible cambiarlo manualmente a través del método *CambiarTipoDebug()*. Algo similar sucede con la frecuencia de envío de mensajes, pudiéndose modificar a través del método *CambiarIntervaloDebug()*.

Por último, cabe destacar que, al tratarse de una clase, es posible instanciarla una única vez para enviar los mismos mensajes a lo largo de todo el programa, o instanciar distintos objetos enfocados en la depuración de ciertos procesos. Asimismo, también existe la posibilidad de cambiar el intervalo y tipo de depuración en determinados fragmentos del código.

Tabla 1: Relación de TipoDebug y mensajes mostrados

TipoDebug	Nivel de depuración	Datos mostrados
0	Nulo	Ninguno
1	Mínimo	Mensaje personalizado
2	Normal	Estado de la controladora, volumen actual de depósitos y próximos volúmenes de dispensación
3	Avanzado	Volúmenes de seguridad de los depósitos y volúmenes correspondientes a los niveles de descarga
4	Todo	Cantidad de dispensaciones por depósito y sus volúmenes acumulados

### 3.6. Puesta en funcionamiento

Además de los cambios y adiciones expuestos anteriormente, se procedió a realizar una reestructuración del código (*refactoring*) con el objetivo de mejorar tanto su claridad como su funcionalidad.

De esta manera, se llevó a cabo un cambio de nomenclatura en múltiples variables y funciones con el propósito de mejorar la legibilidad del código y eliminar ambigüedades. Un ejemplo de este cambio consistió en distinguir las variables relacionadas con un instante de tiempo específico (ahora precedidas por la palabra 'Instante') de las variables relacionadas con un lapso de tiempo (ahora precedidas por la palabra 'Tiempo').

Por otro lado, se ha implementado la convención de notación PascalCase en el nuevo código, así como en varias secciones del código previo que presentaban una notación inconsistente. De igual forma, todas las constantes ahora están escritas en letras mayúsculas.

Asimismo, se ha seguido la convención ampliamente utilizada de preceder los atributos con el símbolo '`_`' en la declaración e implementación de clases, y no usarlo para los parámetros de entrada. Inicialmente, esta convención estaba invertida en algunos casos del código dado, lo que dificultaba considerablemente la comprensión del mismo.

Adicionalmente, se han corregido diversos errores que indudablemente influyeron en los problemas de funcionamiento en versiones anteriores del algoritmo. Estos errores incluyen el uso incorrecto de punteros, la utilización de variables de tiempo en diferentes unidades, la gestión inadecuada de salidas PWM y la falta de verificación de condiciones para el cambio de estados, entre otros.

## 4. Pruebas y resultados

Se han realizado distintas pruebas para comprobar el funcionamiento en cada uno de los ámbitos del proyecto, que han arrojado resultados diversos.

Se logró garantizar de manera satisfactoria el correcto funcionamiento de las partes del proyecto relacionadas con el uso directo del hardware, tales como la disposición del bajante y la instalación eléctrica. Del mismo modo, resultó relativamente sencillo llevar a cabo pruebas en los aspectos de programación que eran más independientes del proceso de puesta en marcha del sistema. Ejemplos de ello incluyen el desarrollo del cuadro de mandos digital y la implementación de la comunicación a través de *MQTT*.

No obstante, en el caso de los cambios en las máquinas de estados y la adición de nuevos procesos, surgió la dificultad de la disponibilidad tardía de ciertos materiales, lo cual retrasó el inicio de las pruebas con el prototipo completo.

Por consiguiente, para estos aspectos del proyecto, las pruebas se centraron principalmente en trazas exhaustivas, que evaluaron diversas situaciones con el fin de obtener una cobertura completa. A pesar de esto, es importante tener en cuenta que, en última instancia, solo es posible descubrir la totalidad de los errores de diseño al someter el código a pruebas reales durante ejecuciones prolongadas.

En el caso de la fase de Inicialización, debido a su naturaleza secuencial y su desarrollo desde cero, se considera que las trazas realizadas son altamente representativas del comportamiento real del proceso. En cambio, en lo que concierne a los demás cambios y adiciones en las máquinas de estado, debido a la amplia cantidad de factores involucrados, resulta más complejo llegar a una conclusión sin llevar a cabo pruebas adicionales en la planta.

Finalmente, debido a la imposibilidad de generar los datos requeridos y sobre todo a la falta de tiempo, se tuvo que prescindir del desarrollo del algoritmo de detección de fugas. Dicho algoritmo de detección simple se basaría en analizar los datos de múltiples ejecuciones para seleccionar un umbral de diferencia de volumen que dispararía el aviso de fuga.

A modo de resumen, en la tabla 2 se puede encontrar una relación de pruebas realizadas para cada uno de los ámbitos del proyecto, resultados obtenidos y grado de consecución del objetivo asociado.

Por último, es digno de mención que, aunque no se pudo realizar una prueba exhaustiva de las nuevas máquinas de estado, sí se llevaron a cabo ejecuciones del código completo. Estas ejecuciones resultaron en la aparición de errores como el reinicio continuo de la placa Arduino y el comportamiento errático de los mensajes enviados por puerto serial, entre otros errores ajenos al diseño del proceso.

Los esfuerzos para identificar y solucionar estos errores acabaron por consumir el tiempo originalmente destinado para realizar pruebas reales de las máquinas de estado. Si bien con el tiempo disponible no se pudo localizar el origen exacto del problema, se sospecha que puede estar relacionado con la gestión de los strings en Arduino y el uso de la memoria RAM que conlleva en la placa.

Tabla 2: Resumen de pruebas y resultados.

<b>Objetivo</b>	<b>Pruebas realizadas</b>	<b>Resultados</b>	<b>Cumplimiento</b>
1. Modificación de la configuración de la planta.	Uso habitual de la nueva configuración de tuberías, realizando dispensaciones entre todos los depósitos.	Uso normal sin eventualidades, se considera que ahora la planta se puede operar con mayor seguridad gracias al bajante.	100%
2. Renovación de instalación eléctrica y electrónica.	Uso habitual del sistema, operando con todos los elementos de la unidad de control, sensores y actuadores.	Instalación perfectamente funcional, sólo a falta de emplear los materiales ahora disponibles.	90%
3. Reestructuración y mejora del código.	Trazas exhaustivas realizadas, a falta de realizar pruebas reales en profundidad.	Si bien no se han localizado errores, es necesario ejecutar más pruebas reales.	50%
4. Diseño de proceso de Inicialización.	Trazas exhaustivas realizadas, a falta de realizar pruebas reales en profundidad.	Es necesario ejecutar más pruebas reales, pero, por las características del diseño, se considera que las trazas son muy significativas.	70%
5. Diseño de panel de control físico.	Uso de entradas y salidas del panel para interactuar con Arduino.	La Arduino responde ante cada una de las entradas, y todas las salidas del panel responden como se espera.	100%
6. Diseño de cuadro de mandos digital.	Presentación de datos simulados enviados por puerto serial y envío de señales desde el cuadro.	La información se muestra correctamente y se actualiza tras cada cambio, y las señales son enviadas correctamente.	100%
7. Diseño de sistema para la transmisión remota.	Envío y recepción de datos simulados desde dos dispositivos en la misma red.	La lista de datos se envía y recibe correctamente y de manera estable.	100%
8. Diseño de algoritmo de detección de fugas.	Ninguna prueba realizada.	-	0%

## 5. Conclusiones

A la vista de la tabla 2, una vez finalizado este Trabajo de Fin de Grado se concluye que se han cumplido exitosamente los objetivos 1, 2, 5, 6 y 7.

No obstante, una gestión inadecuada del tiempo y la disponibilidad tardía de ciertos materiales obstaculizaron la consecución de los objetivos 3 y 4, los cuales sólo pudieron ser cumplidos parcialmente. Si bien se realizaron cambios importantes en el código y se diseñó un nuevo proceso de inicialización, no pudieron ser validados con pruebas reales del funcionamiento del prototipo. El objetivo 8 no se pudo acometer por falta de tiempo.

A pesar de los desafíos mencionados, se considera que se ha logrado mejorar significativamente la usabilidad de la planta y llevar el proyecto a un estado más avanzado en general. Esto será aprovechado por futuros estudiantes como sólido punto de partida, lo que en definitiva constituye el propósito fundamental del proyecto.

Además, durante el desarrollo de este trabajo se ha tenido la oportunidad de aplicar multitud de conocimientos inherentes a la titulación, y especialmente de adquirir diversas habilidades técnicas y competencias transversales, lo que en última instancia ha resultado en un valioso crecimiento personal.

## 6. Conclusions

Upon reviewing the table 2, it is concluded that upon the completion of this Bachelor's Thesis, objectives 1, 2, 5, 6, and 7 have been successfully met.

However, improper time management and late availability of certain materials hindered the fulfillment of objectives 3 and 4, which were only partially met. Despite making significant changes to the code and designing a new initialization process, they could not be validated with real tests of the prototype's functionality. Objective 8 was unattainable due to a lack of time.

Despite the aforementioned challenges, it is considered that the usability of the plant has been significantly improved, and the project has reached a more advanced state overall. This will be utilized by future students as a solid starting point, which ultimately embodies the fundamental purpose of the project.

Moreover, throughout the development of this work, there has been an opportunity to apply a multitude of knowledge inherent to the degree, and particularly to acquire various technical skills and transversal competences, which in the end has resulted in valuable personal growth.

## 7. Líneas futuras

Considerando el estado actual del proyecto, se propone las siguientes líneas de mejora, organizadas en tres categorías según su nivel de complejidad y prioridad:

### 7.1. Corto plazo

Una de las mejoras más urgentes en el prototipo actual consiste en llevar a cabo una depuración pormenorizada de las distintas partes del código con el fin de identificar con precisión el origen de los problemas de funcionamiento.

Además, si bien se han realizado avances en la reestructuración del código, aún queda un amplio margen de mejora para aumentar la eficiencia y escalabilidad del programa. Otra mejora relacionada con la controladora es la incorporación de funciones adicionales para la ejecución manual, tales como desarrollar una modo 'paso a paso' para analizar detalladamente el proceso.

Una medida que contribuiría al desarrollo de estos aspectos es habilitar la capacidad de modificar el nivel de depuración y la frecuencia de envío de mensajes durante las ejecuciones, invocando los métodos *CambiarTipoDebug()* y *CambiarIntervaloDebug()* desde el cuadro de mandos digital. Esto permitiría ajustar de manera más flexible la configuración de depuración según las necesidades específicas en cada circunstancia.

En lo que se refiere a hardware, se identifica como mejora inmediata la incorporación de una báscula al depósito de Descargas, lo cual posibilitaría la generación de datos más diversos durante los experimentos. Del mismo modo, se propone utilizar los materiales recientemente disponibles con el objetivo de aislar completamente la electrónica. Específicamente, se sugiere emplear las cajas estancas y el armario eléctrico mencionados en el apartado 2.6 para dicho propósito.

Además, se contempla la posibilidad de incorporar un carril DIN para asegurar de manera adecuada los elementos de la unidad de control, junto con la inclusión de un diferencial y un magnetotérmico para la alimentación eléctrica. Estas adiciones contribuirían a garantizar una mayor seguridad y protección eléctrica en el sistema.

Asimismo, modificar las básculas para que los módulos HX711 se encuentren en su interior, podría proporcionar una mayor fiabilidad en los datos tomados. Alternativamente, se recomendaría emplear cables RJ45 con apantallamiento para la conexión entre cada báscula y su módulo.

Por último, con el prototipo ya funcional y capaz de ejecutar simulaciones largas, se acometería el desarrollo de un algoritmo de detección de fugas simple.

### 7.2. Medio plazo

En el medio plazo, se podrían realizar modificaciones en el prototipo para incrementar la vida útil de ciertos componentes. Una de estas modificaciones consistiría en reemplazar el regulador lineal LM7805 por un convertidor DC/DC como el K78L05-1000R3, menos propensos a calentarse.

Otros cambios incluyen la sustitución del panel de control físico actual por una botonera industrial de mayor fiabilidad. Además, se contempla la posibilidad de implementar el panel de control en su totalidad en el *dashboard*, lo cual eliminaría el desgaste de los componentes mecánicos y aumentaría la versatilidad de las entradas.

En esta misma línea, se propone la utilización de una placa ESP32 como módulo de comunicación *MQTT*, conectada ya sea a la placa Arduino Mega o al miniPC PIPO. Esto sería posible gracias a las capacidades integradas de la placa ESP32, que incluyen conectividad *Wi-Fi*, *Bluetooth*, *UART*, *SPI* e *I2C*, con bajo consumo de energía.

Con respecto a la detección de fugas, a estas alturas del proyecto ya se podrían implementar algoritmos más avanzados de detección, que no se basen únicamente en la umbralización de los datos tomados.

### 7.3. Largo plazo

Finalmente, a largo plazo, con respecto al hardware, tal como se mencionó previamente, se propone implementar un cerramiento de metacrilato para la estantería, lo cual brindaría un mayor aislamiento a los componentes, minimizando la posibilidad de contactos accidentales. Además, uno de los cambios más destacados implica la sustitución de los depósitos existentes con el fin de ajustar de forma más precisa sus capacidades a las proporciones de los elementos encontrados en una estación de servicio real. Esta actualización tiene como objetivo mejorar la representación y la fidelidad del sistema en relación a su equivalente real.

En cuanto a la interacción con la planta, se plantea la instalación de una cámara que posibilite la visualización remota del estado de los depósitos. Asimismo, se sugiere configurar *MQTT* para establecer comunicaciones a través de internet, no limitándose únicamente a la red local. Estas incorporaciones permitirían, en última instancia, alcanzar un control telemático completo de la planta, brindando la capacidad de supervisar y gestionar el sistema de forma remota y en tiempo real.

Por último, en relación a los algoritmos de detección, se plantea la posibilidad de modelar la histéresis de los sensores y abordar otras no linealidades presentes en el sistema, lo que garantizaría la adquisición de datos de calidad.

## 8. Presupuesto

La tabla 3 ofrece un desglose presupuestario del proyecto. Esto incluye no solo el costo de los materiales y los costos de mano de obra, sino también el beneficio industrial, y los costes indirectos e impuestos.

Tabla 3: Presupuesto del proyecto.

Concepto	Uds.	Precio / Ud.	Precio
<b>Planta</b>			
Estantería	1	59,00 €	59,00 €
Bases de madera	3	12,00 €	36,00 €
Depósito de Descargas	1	10,90 €	10,90 €
Depósito de Almacenamiento	1	115,00 €	115,00 €
Depósitos de Ventas y Fugas	3	3,00 €	9,00 €
Depósito de Retorno	1	49,99 €	49,99 €
Tuberías y uniones varias	1	106,80 €	106,80 €
<b>Sensores y Actuadores</b>			
Báscula industrial	1	350,00 €	350,00 €
Báscula de cocina	3	12,00 €	36,00 €
Módulo HX711	4	1,49 €	5,96 €
Sensor nivel	4	15,99 €	63,96 €
Electroválvula	6	24,20 €	145,20 €
Bomba centrífuga DC	5	12,00 €	60,00 €
Bomba centrífuga AC	1	39,70 €	39,70 €
Bomba peristáltica	1	32,68 €	32,68 €
<b>Unidad de control e interacción</b>			
Arduino Mega 2560	2	20,00 €	40,00 €
Módulo de 16 relés	1	20,00 €	20,00 €
Módulo L298N	1	5,00 €	5,00 €
Panel de control	1	17,70 €	17,70 €
miniPC PIPO X10S	1	372,70 €	372,70 €
<b>Instalación eléctrica y electrónica</b>			
Armario eléctrico 530x430x200	1	142,23 €	142,23 €
Cajas estancas 105x105x55	10	1,63 €	16,30 €
Canaletas 32x16	6	2,12 €	12,73 €
Fuente de 12v	1	30,00 €	30,00 €
Regulador LM7805 y condensadores	1	1,70 €	1,70 €
Diodos y varistor de protección	6	1,20 €	7,20 €
Regleta de tomas	1	6,99 €	6,99 €
Clemas de 12 polos	10	0,49 €	4,90 €
Rollo de cable ethernet 25m	1	19,79 €	19,79 €
Rollo de cable 1,5 mm 20 m	1	10,29 €	10,29 €
Tornillería	1	10,35 €	10,35 €
<b>Mano de obra</b>			
Ingeniero (hora)	300	48,00 €	14.400,00 €

<b>PEM</b>	16.238,07 €
------------	-------------

<b>BI</b>	974,28 €
<b>CI</b>	2.110,95 €
<b>Total sin IGIC</b>	19.323,30 €

<b>TOTAL</b>	<b>20.675,93 €</b>
--------------	--------------------

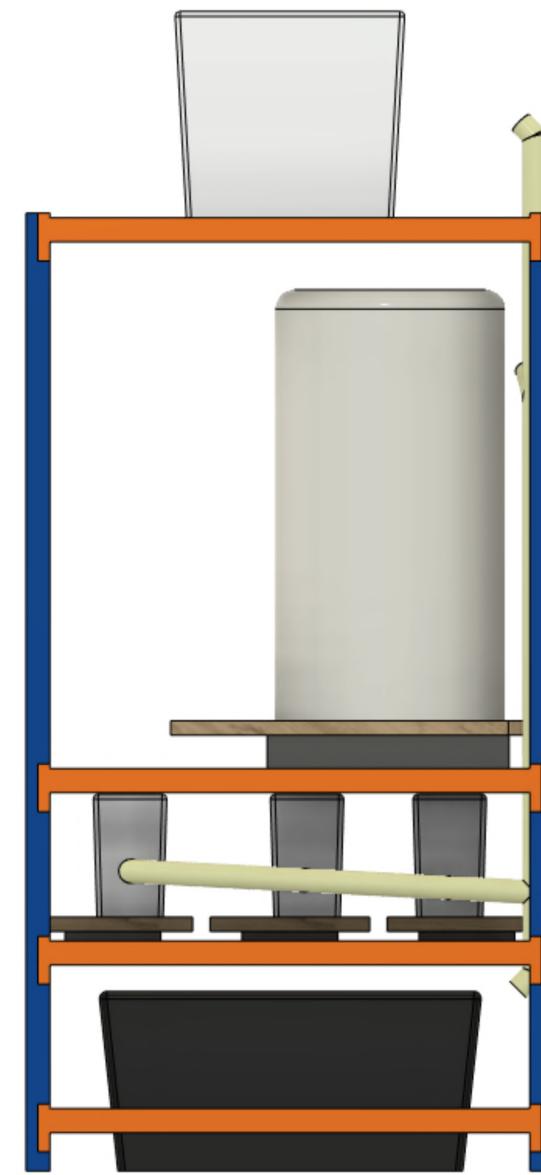
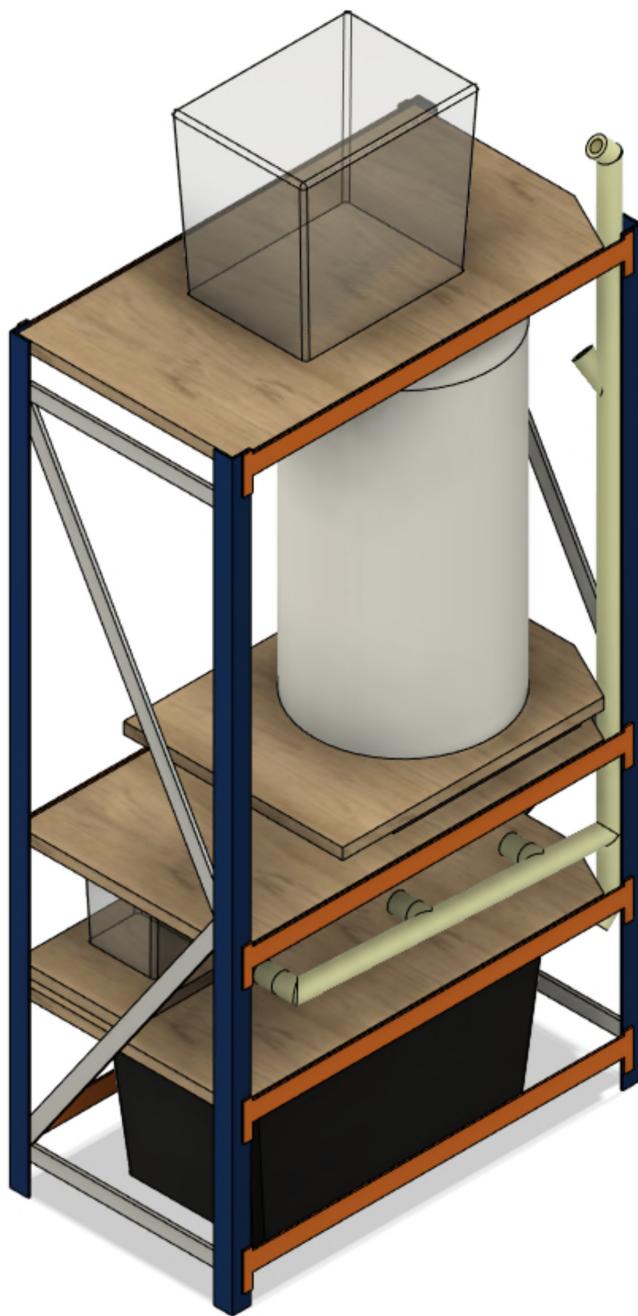
## 9. Bibliografía

- [1] M. Sigut, S. Alayón, y E. Hernández, 'Applying pattern classification techniques to the early detection of fuel leaks in petrol stations', Journal of Cleaner Production, vol. 80, pp. 262-270, 2014. [Online]. Disponible: <https://doi.org/10.1016/j.jclepro.2014.05.070>
- [2] S. Alayón, M. Sigut, R. Arnay, y P. Toledo, 'Time windows: The key to improving the early detection of fuel leaks in petrol stations', Safety Science, vol. 130, 104874, 2020. [Online]. Disponible: <https://doi.org/10.1016/j.ssci.2020.104874>
- [3] L. Li, H. Chen, Y. Huang, G. Xu, y P. Zhang, 'A new small leakage detection method based on capacitance array sensor for underground oil tank', Process Safety and Environmental Protection, vol. 159, pp. 616-624, 2022. [Online]. Disponible: <https://doi.org/10.1016/j.psep.2022.01.020>
- [4] C. Lin, Y. J. Ngu, Y. Chan, y A. T. Yeung, 'Feasibility of a TDR-based technique for fluid hydrocarbon leak detection', Process Safety and Environmental Protection, vol. 175, pp. 732-743, 2023. [Online]. Disponible: <https://doi.org/10.1016/j.psep.2023.05.087>
- [5] R. Chu, L. Chik, J. Chan, K. Gutzmann, y X. Li, 'Automatic meter error detection with a data-driven approach', Engineering Applications of Artificial Intelligence, vol. 123, part C, 106466, 2023. [Online]. Disponible: <https://doi.org/10.1016/j.engappai.2023.106466>
- [6] L. Arriaga Campos, 'Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos', Trabajo Final de Grado, Repositorio institucional de la Universidad de La Laguna, 2020. [Online]. Disponible: <http://riull.ull.es/xmlui/handle/915/20640>
- [7] N. Yanes Perez, 'Diseño e implementación de un sistema autónomo para la detección de fugas en depósitos', Trabajo Final de Máster, Máster Universitario en Ingeniería Industrial, Repositorio institucional de la Universidad de La Laguna, 2022. [Online]. Disponible: <http://riull.ull.es/xmlui/handle/915/31568>
- [8] F. Rodríguez Herrera, 'Simulación y detección de fugas en depósitos de combustible', Trabajo Final de Grado, Grado En Ingeniería Electrónica Industrial Y Automática, Repositorio institucional de la Universidad de La Laguna, 2022. [Online]. Disponible: <http://riull.ull.es/xmlui/handle/915/30329>
- [9] Plotly, 'Plotly Python Open Source Graphing Library', 2023. [Online]. Disponible: <https://plotly.com/python/>
- [10] Plotly, 'Dash by Plotly', 2023. [Online]. Disponible: <https://dash.plotly.com>
- [11] PyPI, 'PySerial', 2023. [Online]. Disponible: <https://pypi.org/project/pyserial/>
- [12] Eclipse, 'Eclipse Mosquitto', 2023. [Online]. Disponible: <https://mosquitto.org>
- [13] PyPI, 'Paho-MQTT', 2023. [Online]. Disponible: <https://pypi.org/project/paho-mqtt/>

## Anexos

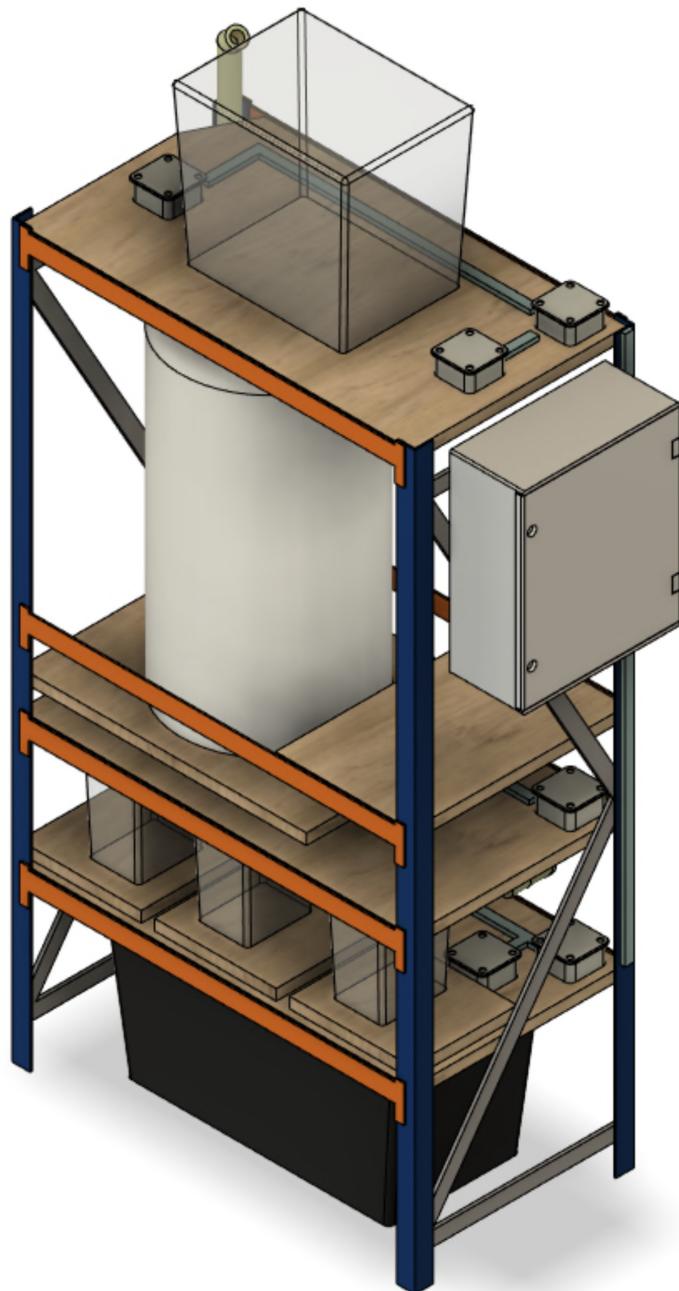
## Anexo I. Disposición del bajante

## Disposición del bajante



## Anexo II. Disposición del cableado

# Ubicación de cajas estancas



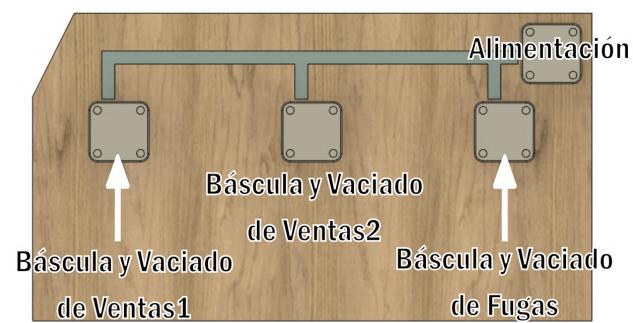
Balda superior



Balda media

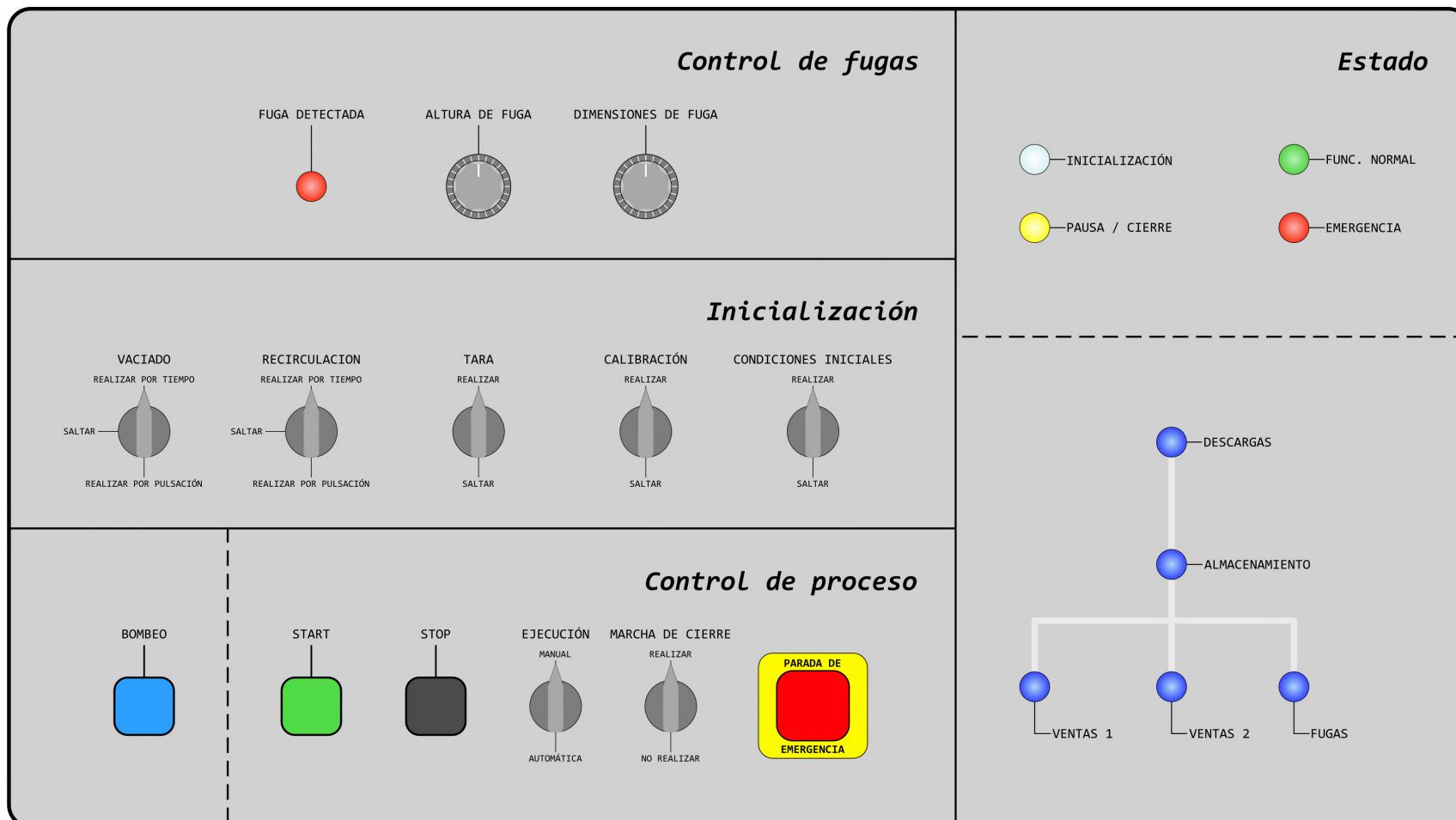


Balda inferior

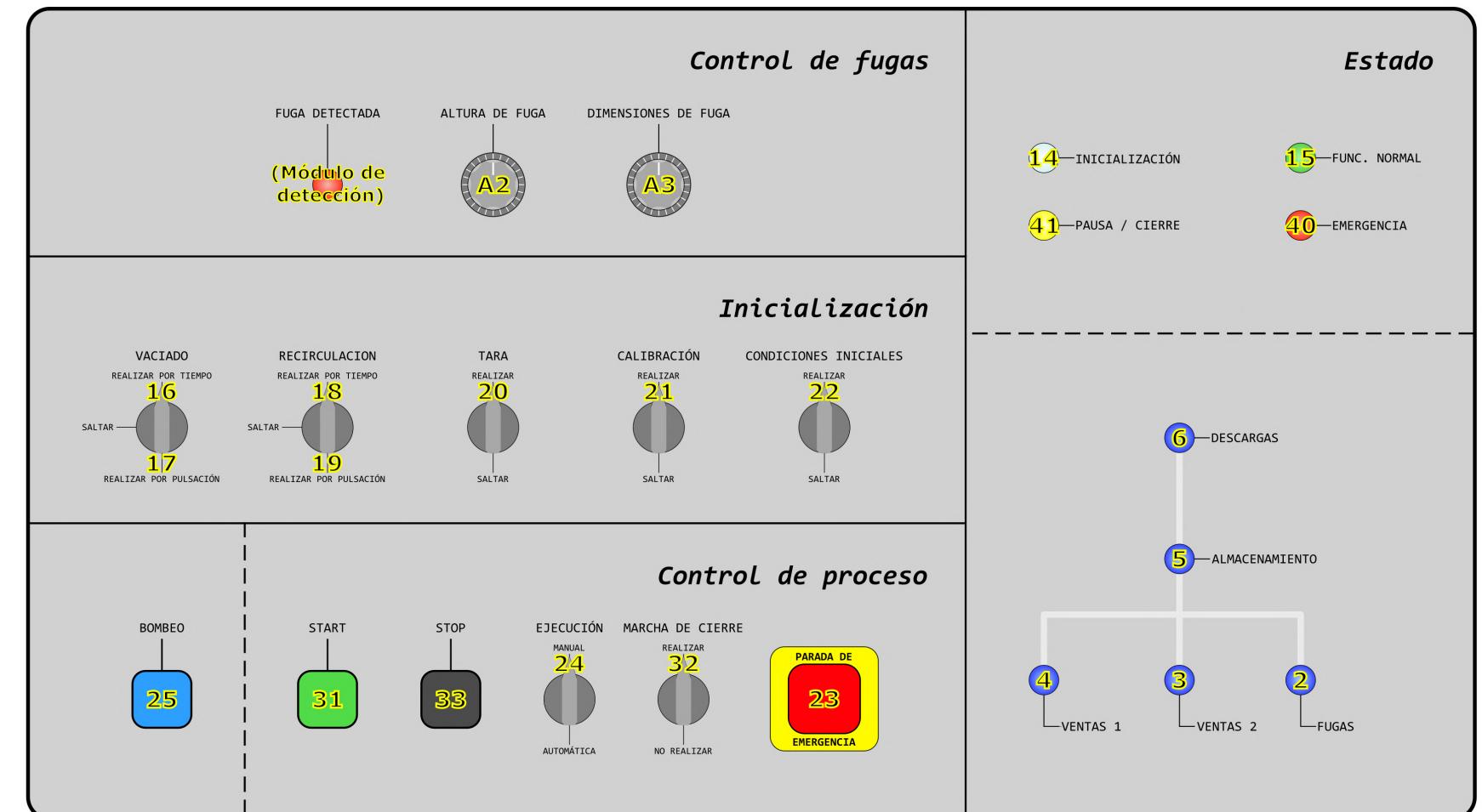


## Anexo III. Panel de control físico

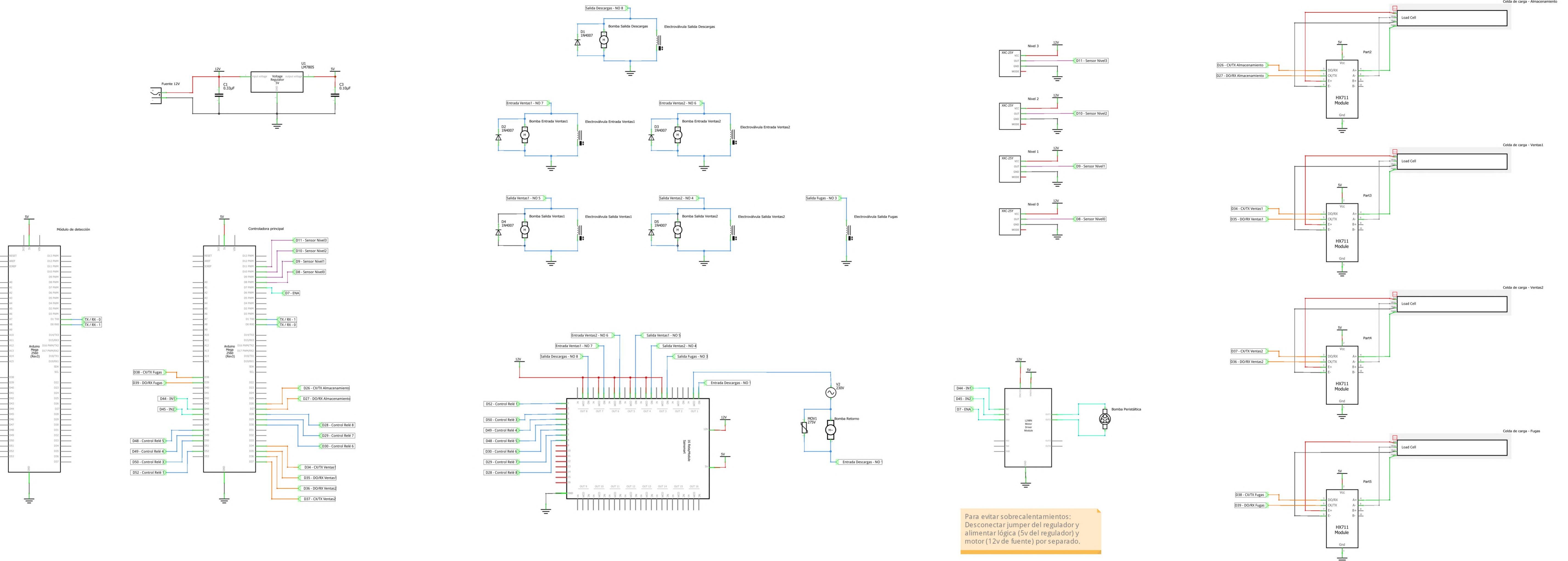
## Disposición de elementos del panel de control



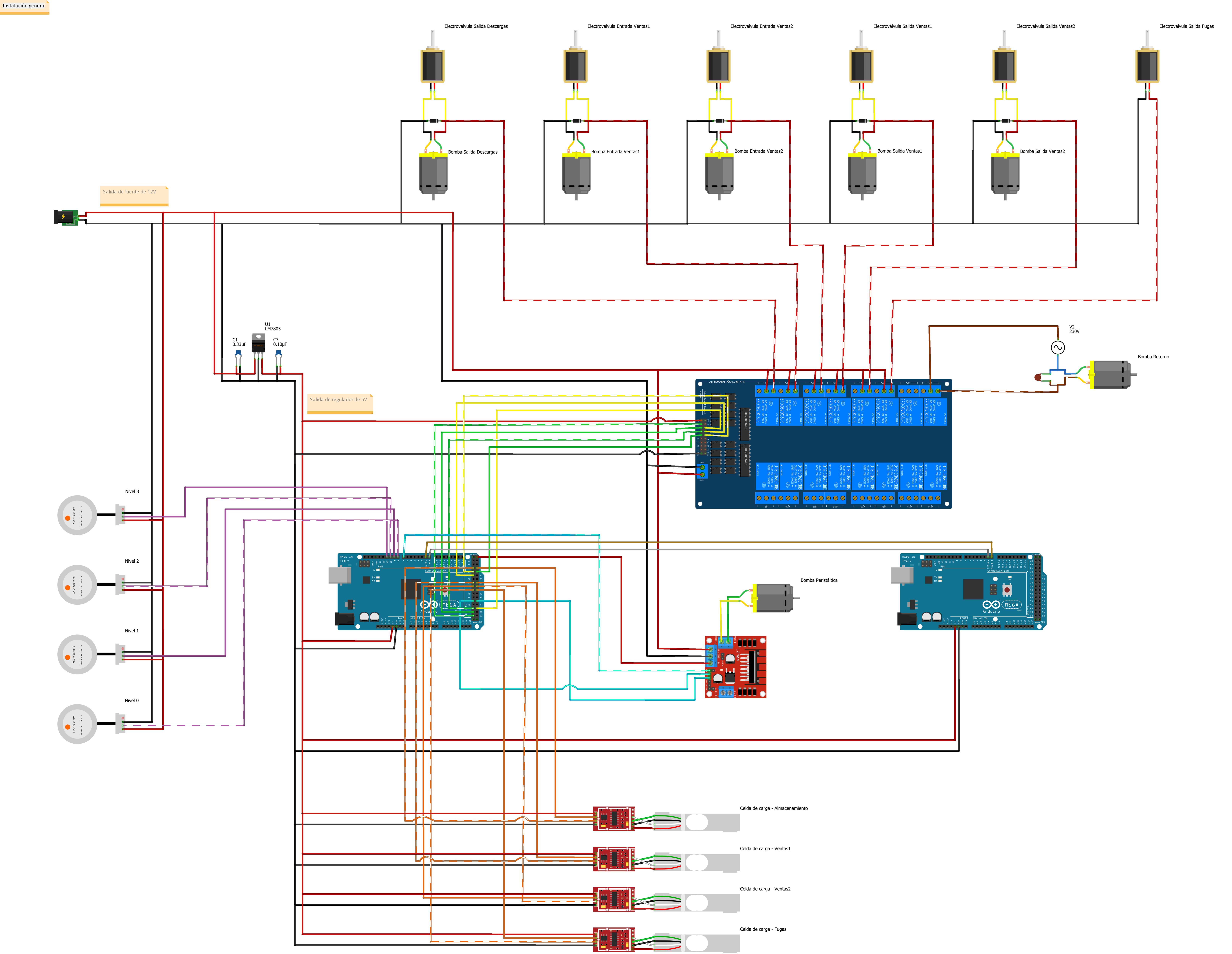
## Pin de placa Arduino de cada elemento



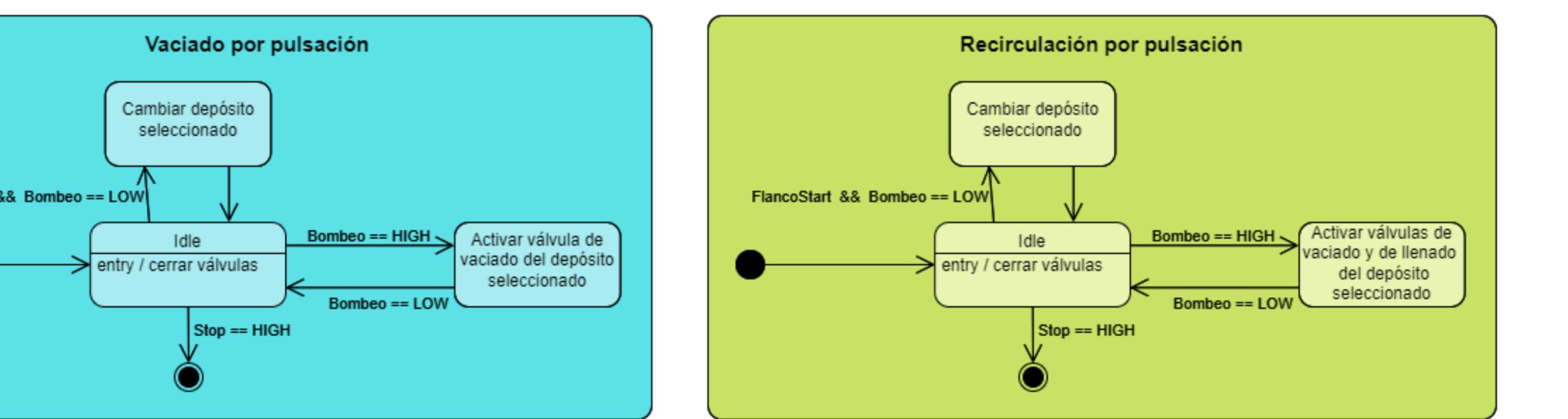
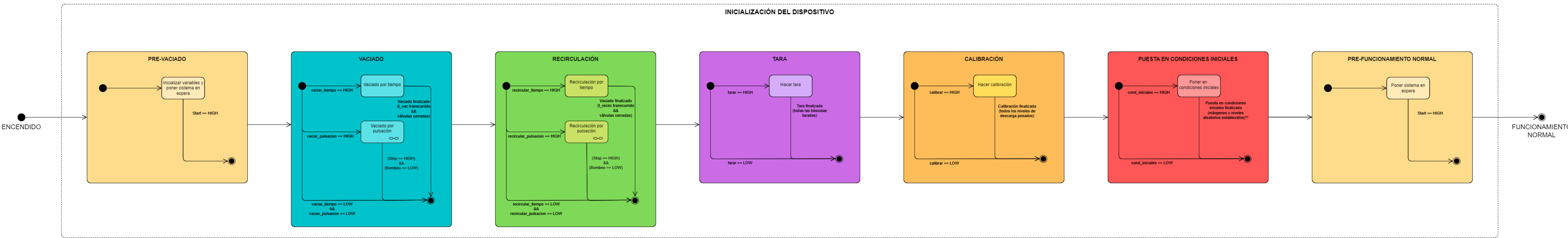
## Anexo IV. Diagrama esquemático de la instalación



## Anexo V. Diagrama detallado de la instalación



## Anexo VI. Proceso de Inicialización



NOTA: En caso de no indicarse ninguna condición en una transición, se trata de una transición automática.

Interruptores del panel de mando:  

- vaciar\_tiempo
- vaciar\_pulsacion
- recircular\_tiempo
- recircular\_pulsacion
- tarar
- calibrar
- cond\_iniciales

Pulsadores del panel de mando:  

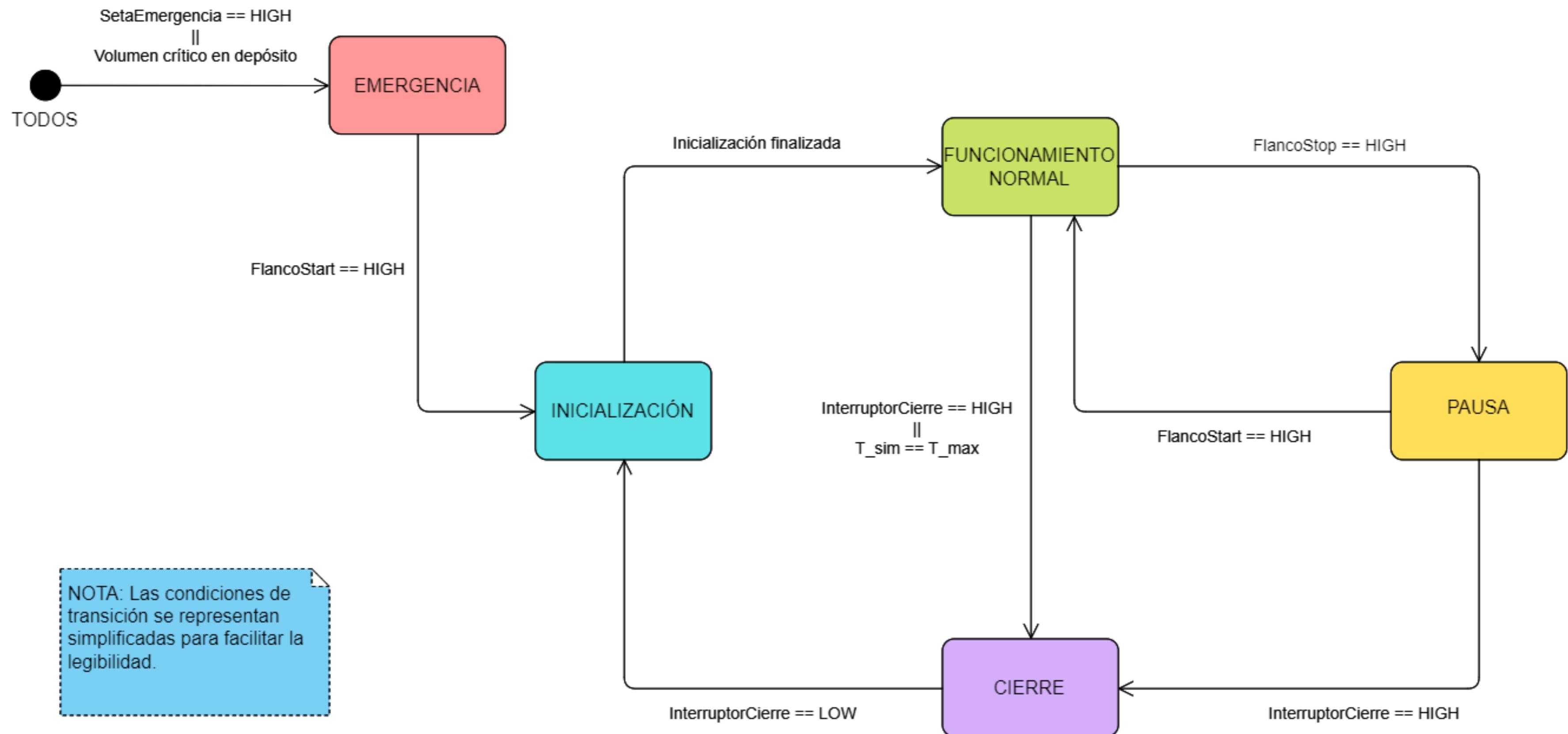
- Start
- Stop
- Bombeo

\*\*La puesta en condiciones iniciales cumple dos funciones:  

1. Añade a cada depósito un margen de agua que evita el funcionamiento de las bombas en vacío.
2. Llena los tanques de almacenamiento y de descarga a un nivel aleatorio

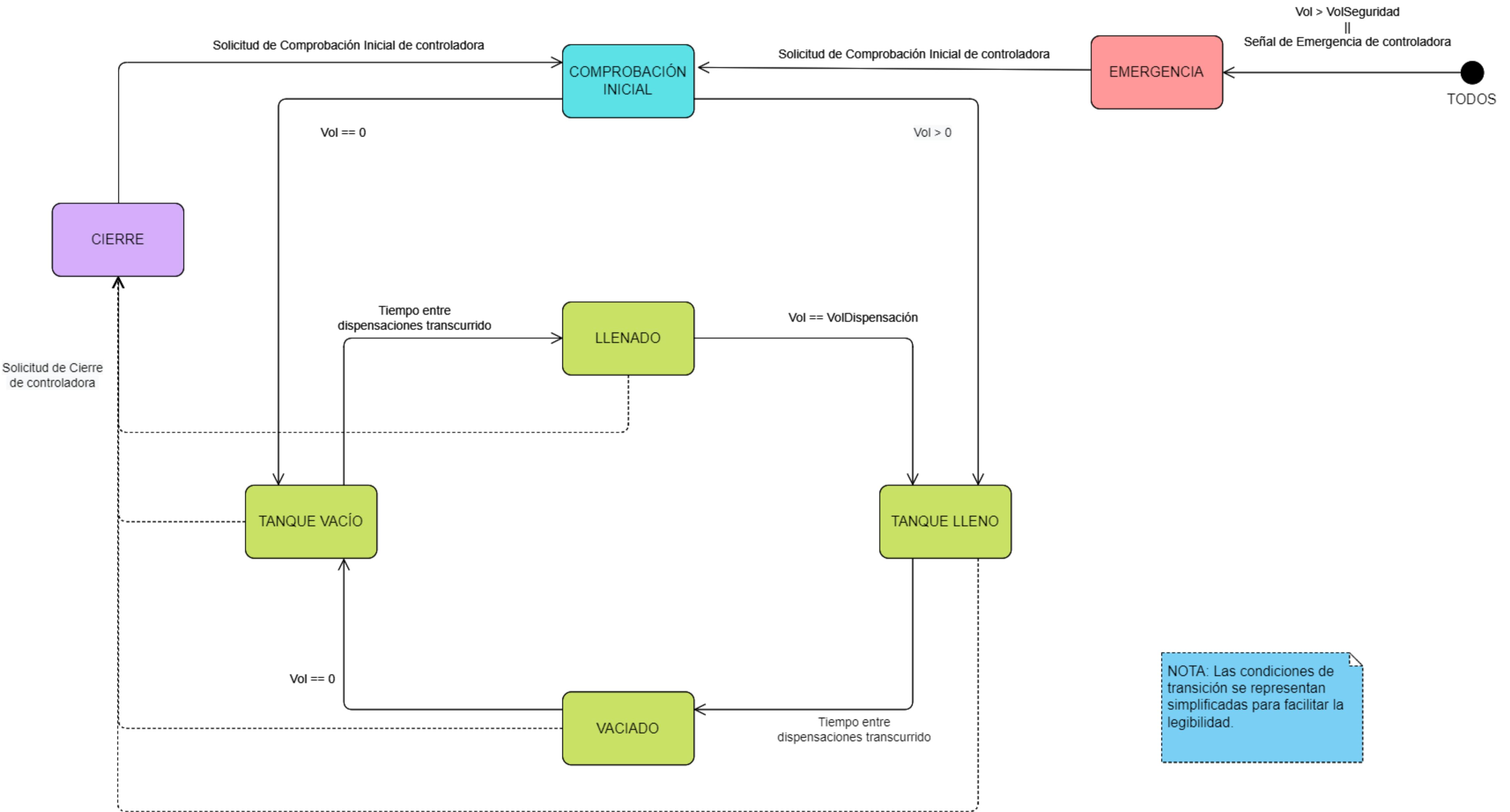
## Anexo VII. Máquina de estados general

## MÁQUINA DE ESTADOS GENERAL (CONTROLADORA)



## Anexo VIII. Máquina de estados de depósito

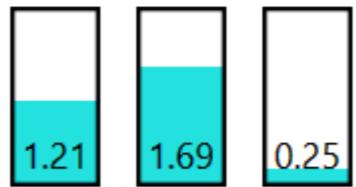
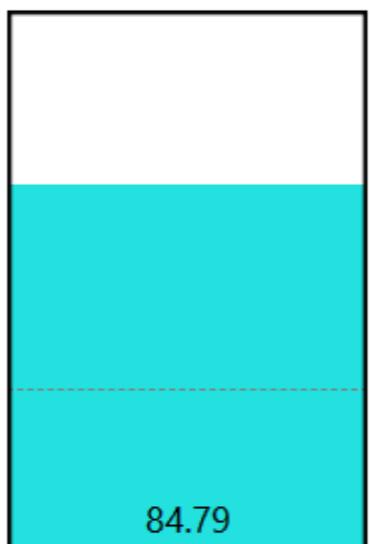
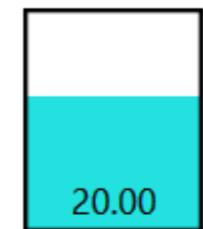
## MÁQUINA DE ESTADOS DE DEPÓSITO



## Anexo IX. Cuadro de mando digital

# MONITORIZACIÓN DE LA PLANTA

T = 00:00:44



Emergencia

Depósito	Prox. Dispensación	Dispensaciones
Descargas	00:01:24	100
Ventas1	00:01:01	30
Ventas2	00:00:57	35

