Curso 2021/2022

Grado en Ingeniería Electrónica Industrial y Automática

Simulación y detección de fugas en depósitos de combustible

Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología

Autor:

Fernando Rodríguez Herrera

Tutores:

Pedro Antonio Toledo Delgado

Marta Sigut Saavedra

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. RESUMEN

Actualmente las estaciones de servicio poseen depósitos de combustibles enterrados, que pueden estar sometidos a corrosión y terminar provocando vertidos y contaminación del subsuelo. Sistemas de conciliación de inventarios de combustible pueden ayudar a tener alarmas tempranas para detectar si se está produciendo una fuga.

A partir de este problema, se quiere construir un prototipo físico que reproduzca el funcionamiento de un depósito de almacenamiento de una estación de servicios, partiendo del diseño previo de otro prototipo de menor escala. Se quiere además, construir un sistema de detección de fugas que, conectado a este prototipo, sea capaz de detectar fugas de la manera más temprana posible.

Para abordar el problema, se debe utilizar un diseño de bajo coste con los componentes más adecuados que se adapten a nuestro diseño. Para ello se va hacer uso de un microcontrolador que hará de software de la propia controladora y otro que funcione como sistema de detección de fugas.

Como resultado se ha realizado la simulación del funcionamiento de un depósito tanto sin fuga como con diferentes tipos de fugas con el objetivo de comprobar la precisión del sistema de detección.

A lo largo de este documento se hablará un poco de las estaciones de servicio de hoy en día. Aportaremos datos como códigos de la controladora de la planta, presupuestos y un estudio realizado a partir de los resultados obtenidos de las simulaciones que, posteriormente, analizaremos y sacaremos conclusiones.

2. ABSTRACT

Currently, service stations have buried fuel tanks, which may be subject to corrosion and end up causing spills and contamination of the subsoil. Fuel inventory reconciliation systems can help provide early warning to detect if a leak is occurring.

From this problem, we want to build a physical prototype that reproduces the operation of a storage tank of a service station, starting from the previous design of another smaller-scale prototype. It is also wanted to build a leak detection system that, connected to this prototype, is capable of detecting leaks as early as possible.

To address the problem, a low cost design should be used with the most suitable components that fit our design. To do this, a microcontroller will be used that will act as software for the controller itself and another that will function as a leak detection system.

As a result, the simulation of the operation of a tank has been carried out both without leaks and with different types of leaks in order to check the accuracy of the detection system.

Throughout this document we will talk a little about today's service stations. We will provide data such as plant controller codes, budgets and a study based on the results obtained from the simulations that we will later analyze and draw conclusions.

3. INTRODUCCIÓN

3.1. Detección de fugas en depósitos de combustibles

Actualmente los diferentes carburantes de las estaciones de servicio se almacenan en depósitos subterráneos situados bajo tierra. En el interior de cada uno de ellos se puede encontrar un tipo de carburantes diferente. Como se puede imaginar, el hecho de que los depósitos se encuentren bajo tierra, implica que debe existir un mecanismo que impulse el carburante hacia el exterior. En este sentido, las estaciones de servicio cuentan con una bomba sumergible o de succión.

Debido a la alta volatilidad e inflamabilidad de muchos carburantes, la opción más segura para su almacenaje son los tanques subterráneos. Estos tanques son lugares de almacenamiento que suelen trabajar con cantidades elevadas de estas sustancias. Estos suelen contar con un sistema de monitoreo que aumenta la seguridad detectando a tiempo los errores, como pueden ser las diferentes fugas de combustible.

Antes de instalar estos tanques en una estación de servicios, hay que tener en cuenta el tipo de suelo. No todos los terrenos son iguales y algunos presentan riesgos con respecto a la instalación de estos depósitos [1].

3.1.1. Funcionamiento de estaciones de servicio

El surtidor de combustible o bomba de gas es una máquina de las gasolineras que se utiliza para poner gasolina en los coches. La surtidor de una estación de servicio para poder funcionar se compone de tres elementos fundamentales:

Tanques de almacenamiento

Los combustibles en las estaciones de servicio se almacenan bajo tierra en depósitos subterráneos. Cada depósito tiene capacidad para miles de litros de combustible. La mayoría de las estaciones cuentan con cuatro de estos tanques y, por lo general, cada uno tiene un tipo diferente de gasolina. El tener estos depósitos bajo tierra, hace que se presente un problema obvio: la gasolina debe llegar a los surtidores que están ubicados por encima de estos, por lo que tienen que desafiar a la gravedad para llegar hasta arriba.

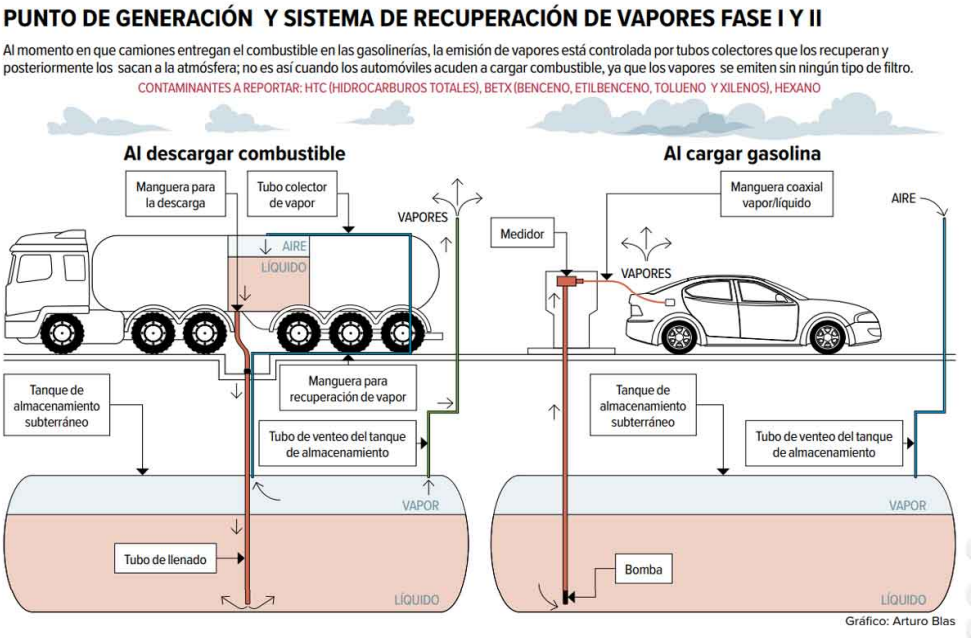


Figura 3.1. Funcionamiento de una estación de servicio

Para mover el combustible de estos depósitos a los surtidores, en la mayoría de estaciones de servicio se hace uso de uno de estos dos tipos de bomba:

- Bomba sumergible. Como su nombre indica, se sumerge por debajo de la superficie del líquido, donde se utiliza una hélice que impulsa el combustible hacia arriba [2].

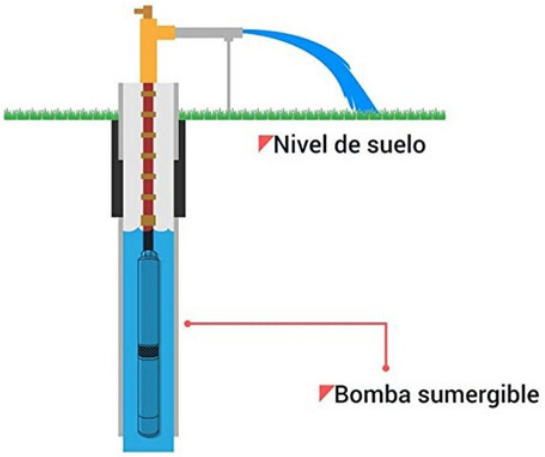


Figura 3.2. Bomba sumergible

- Bomba de succión. Mueve el combustible con el principio de la presión desigual. Un motor situado por encima del nivel del líquido elimina suficiente aire de la tubería para disminuir la presión del aire por encima del combustible. El motor continúa eliminando aire hasta que la presión del aire por encima del combustible es menor a la presión del aire que empuja hacia abajo el combustible fuera de la tubería. Una vez la presión del aire dentro del tubo es lo suficientemente baja, la gasolina subiría [2].

Figura 3.3. Bomba de succión

Válvula de retención

Una vez se apaga el motor de la bomba, la gasolina de dentro de la tubería no vuelve a caer en el tanque. Por el contrario, se queda atrapada dentro de la tubería gracias a la válvula de retención. Esta válvula está situada por encima de la gasolina del tubo y lo que hace es crear un sello hermético que impida que la gasolina vuelva a bajar. Utilizando esta válvula de retención, lo que conseguimos es evitar un desgaste innecesario de la bomba de succión [2].

Figura 3.4. Válvula de retención

Medidor de flujo

Una vez que el combustible está en camino hacia el vehículo, faltaría saber la cantidad de gasolina que se inyecta en el automóvil.

El combustible que viaja hacia el dispensador, pasa a través de una válvula reguladora que mide la velocidad del flujo de combustible. Esto lo hace a través de una membrana de plástico que se va comprimiendo por el tubo conforme el flujo de gasolina aumenta, dejando siempre suficiente espacio para que la cantidad correcta de gasolina pase a través de él [2].

El tubo contiene el medidor de flujo, que es un molde aluminio o hierro, el cual tiene una serie de engranes o un rotor simple por donde pasa el combustible. Estos leen el flujo de gasolina y pasan la información a un ordenador situado en el dispensador que mostrará la cantidad deseada [2].

Figura 3.5. Medidor de flujo

3.1.2. Sistema de detección de fugas

Para la detección de fugas y la prevención de contaminación, el uso de tanques y tuberías de doble pared se ha hecho obligatorio en la mayor parte de Europa. Para la detección de fugas es posible rellenar el espacio intersticial con un líquido o aire, y mediante su control es posible generar alarmas. Existen diferentes sistemas de detección de fugas, entre los que destacan los siguientes [3]:

- Por caída de presión. Esta prueba mide el cambio de presión entre la presión atmosférica y su muestra de prueba presurizada. A diferencia de otros métodos actuales de prueba de fugas, este método proporciona información cuantitativa, puntos de datos concretos que se pueden registrar y sobre los que se pueden tomar decisiones [4].

Figura 3.6. Sistema de detección de fugas por caída de presión

- Por vacío. Estos aparatos están diseñados para ser utilizados en todo tipo de depósitos con diámetro no superior a 2900 mm de doble pared con presión atmosférica. El equipo crea una depresión de -400 mbar entre las paredes del tanque, y cuando la depresión disminuye genera una alarma que indica una fuga [5].

Figura 3.7. Sistema de detector de fugas de vacío para depósitos de doble pared

- Por líquido. El líquido universal UV de detección de fugas ha sido diseñado para la detección de fugas, incluso las más pequeñas. El líquido ultravioleta localiza visualmente las fugas, lo que es ideal para su detección o para hacer un diagnóstico rápido. El colorante ultravioleta se mezcla con el líquido fugado, por lo que una fuga puede visualizarse con una lámpara ultravioleta en el lugar preciso en el que se encuentre [6].

Figura 3.8. Líquido universal ultravioleta de detección de fugas 100 ml

3.1.3. Ejemplos reales de sistemas de detección de fugas

La Conciliación Estadística de Inventarios permite detectar de forma rápida los posibles fallos en los almacenes de combustible y otros fluidos.

Desde FullGas, un equipo de analistas investiga estos fallos: fugas en tanques, tuberías y descargas, efecto de variación de temperatura, sobrellenado de los tanques, calibraciones incorrectas…, informando inmediatamente a los responsables de la estación de servicios. Todo ello se debe gracias a este sistema de alerta temprana de pérdidas de combustibles.

Diariamente la sonda detecta las diferencias entre las existencias teóricas y las existencias reales. Estas diferencias son investigadas por los analistas, informando inmediatamente en caso de una fuga para una rápida actuación y corrección.

Una rápida detección y corrección reduce las pérdidas de combustible y los costes operativos y de mantenimiento. Además protege el medioambiente [7].

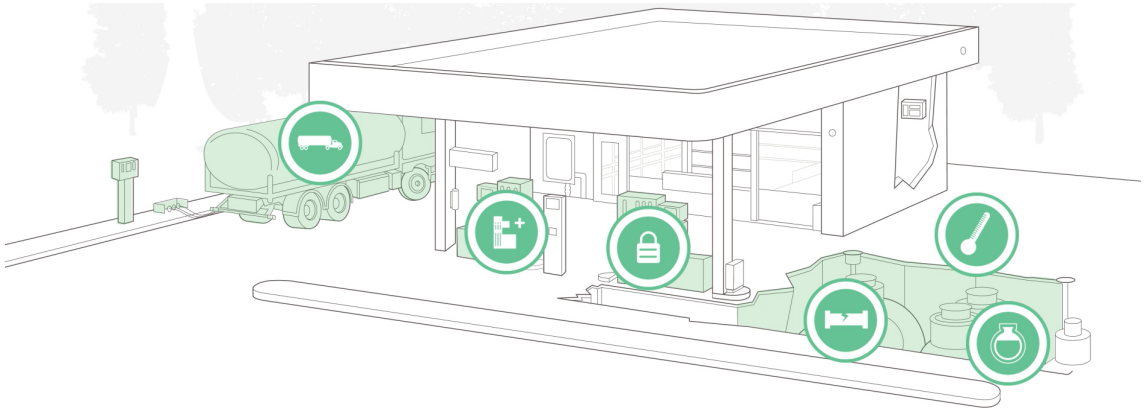


Figura 3.9. Sistema de control estadístico de inventario

3.2. Antecedentes

Este proyecto parte de una idea inicial basada en la implementación de un prototipo físico de una planta que simula el llenado y vaciado de depósitos de una estación de servicios. Nos hemos apoyado en dos proyectos anteriores, “Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos” [7] escrito por Luis Arriaga Campos y “Adquisición y tratamiento de datos de un sistema de simulación de fugas en depósitos de combustible” [8] escrito por Eduardo Miguel Gastón Quesada. Este último proyecto está pendiente de presentar.

El primer de ellos consistía en implementar un sistema autónomo que fuese capaz de simular fugas. La idea era construir un prototipo físico que representase el funcionamiento de una estación de servicios, pero debido a la circunstancias sanitarias, se priorizó la implementación de un simulador frente al prototipo físico. Se contaba con dos microcontroladores, los cuales, uno trabajaba como simulador de la planta y el otro como la propia controladora de la planta. La función del primero de ellos era enviar la información recogida de los sensores simulados al otro microcontrolador, y este segundo, en función de dicha información y de la lógica programada, enviaba sus respectivas órdenes a los actuadores del sistema. Finalmente se decidió hacer una simulación vía software usando alguno de los simuladores de internet.

El segundo de ellos partía de este primer proyecto. Una vez implementado el simulador, se decidió construir el prototipo físico que se iba a montar en un primer momento, por lo que se utilizaron ya componentes físicos como sensores y actuadores. El objetivo de este proyecto, al igual que en el anterior, era construir un modelo físico que fuese capaz de simular fugas. La implementación software de su controladora se ha hecho mediante el uso de un microcontrolador.

Este proyecto se centra en implementar un sistema de detección de fugas. Tomando la idea del prototipo físico anterior, se pasa a construir uno a mayor escala, con el objetivo de asimilarse más al funcionamiento de una estación de servicios real ya que se trabaja con mayores cantidades de agua, lo que conlleva más riesgos de fuga. Esto permite obtener resultados más acordes a un caso real y un análisis más preciso de los resultados de los experimentos. Debido a esto, se utilizan depósitos de mayor dimensión. Se nos ha aportado código para la controladora, el cual, se adapta a nuestro nuevo prototipo ya que cuenta con un depósito de ventas adicional al del proyecto anterior. También se toma la implementación del primer proyecto ya que este sí cuenta con dos de estos depósitos. Además, se implementa un segundo microcontrolador que, conectado al prototipo, trabaja como sistema de detección de fugas.

3.3. Objetivos propuestos

Para la consecución de este proyecto, se plantearon los siguientes objetivos a cumplir:

- Adaptar el diseño previo a uno a mayor a escala.
- Conseguir materiales que cumplan con los requisitos de prototipo de bajo coste.
- Construir el prototipo físico del sistema que se ha diseñado.
- Adaptar la controladora que gestiona el funcionamiento de la planta.
- Implementar un sistema de detección de fugas.
- Validación de la controladora y detector de fugas mediantes diferentes simulaciones del funcionamiento de la planta.

3.4. Planificación de las tareas a desarrollar

La planificación de las tareas es una actividad que se realizó desde el primer momento. A la hora de abordar el proyecto, lo primero fue analizar los diferentes objetivos a alcanzar y, en función de ellos, se realizó la planificación. De igual forma, una vez se comienza con el proyecto, siempre pueden ir surgiendo nuevas tareas como consecuencia de otras. A continuación se muestra la planificación que se desarrolló desde el principio.

Desde el primer momento se nos aportó código de la controladora del proyecto anterior. Al no haberse trabajado nunca con el lenguaje de programación de Arduino, se decidió crear un proyecto sencillo con el objetivo de coger soltura en este lenguaje de programación. A su vez, se tuvo que estudiar el código de la controladora ya que más adelante habría que modificarlo

debido a que este proyecto presenta algunas diferencias con respecto al anterior que ya explicaremos más adelante.

Lo siguiente fue realizar una búsqueda de los diferentes componentes necesarios para llevar a cabo el montaje de la planta. Esto hace referencia a los diferentes depósitos, sensores, actuadores y materiales necesarios para transportar el fluido de un tanque a otro.

Estas dos primeras tareas se realizarían al mismo tiempo. Una vez se aprendiera a trabajar con Arduino, se pasaría a diseñar el sistema de detección de fugas, por lo que se decidió empezar a crear el código del detector una vez se tuviese entendido y terminado el proyecto sencillo que se realizó en un primer momento.

Por último, tan pronto como se tuviese el material, se pasaría a construir el prototipo y, posteriormente, se comenzarían a realizar experimentos.

Como se mencionó anteriormente, esto es una planificación que se realizó desde el primer momento. Al adentrarse en el proyecto y comenzar con el desarrollo del mismo, han ido surgiendo nuevas tareas que han ido modificando la planificación.

4. DISEÑO DE LA PLANTA Y HARDWARE DE CONTROL

4.1. Diseño del prototipo físico

El diseño del prototipo físico se ha hecho tomando ideas del prototipo del proyecto anterior. Como mencionó el compañero Luis Arriaga en su proyecto "Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos" [7], a la hora de plantearse la construcción de una planta que representase el funcionamiento de una estación de servicio, incluyendo las posibles fugas, se tuvieron en cuenta dos aspectos. El primero de ellos, que contuviera todos los elementos básicos de una estación real y, el segundo, que el prototipo diseñado presentase unas características que permitieran su construcción en un laboratorio de docencia. Teniendo en cuenta los aspectos nombrados por el compañero, el prototipo físico se ha representado de la siguiente manera.

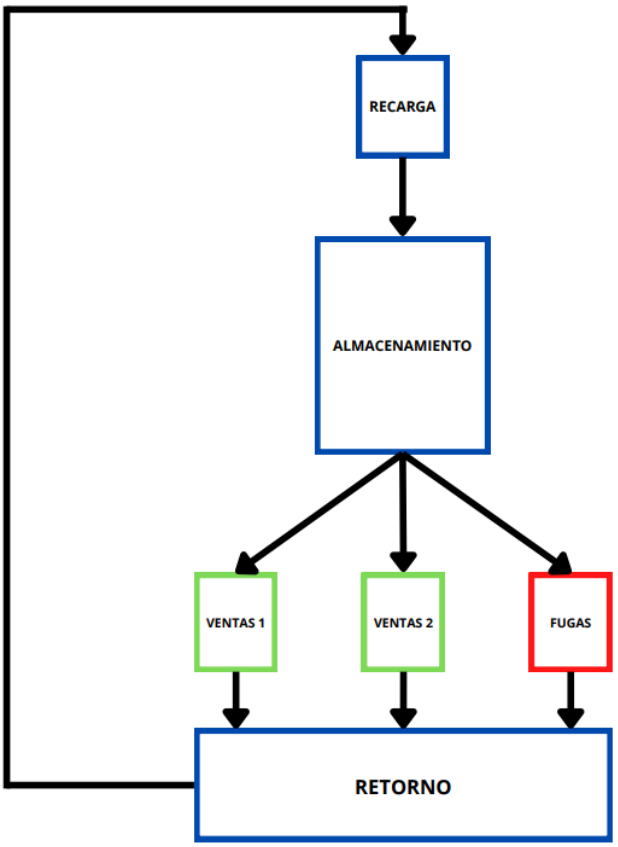


Figura 4.1. Esquema de la planta diseñada con sus diferentes tanques

- Tanque de recarga. Este depósito es equivalente al camión cisterna que descarga en el tanque de almacenamiento que se encuentra bajo tierra en las estaciones de servicio.

Figura 4.2. Camión cisterna

- Tanque de almacenamiento. Este depósito representa el tanque de almacenamiento subterráneo de una estación de servicio. Este es el depósito principal, por lo que es de grandes dimensiones, ya que de aquí obtenemos valores de ventas realizadas. Además, de este tanque extraeremos las fugas que analizaremos.

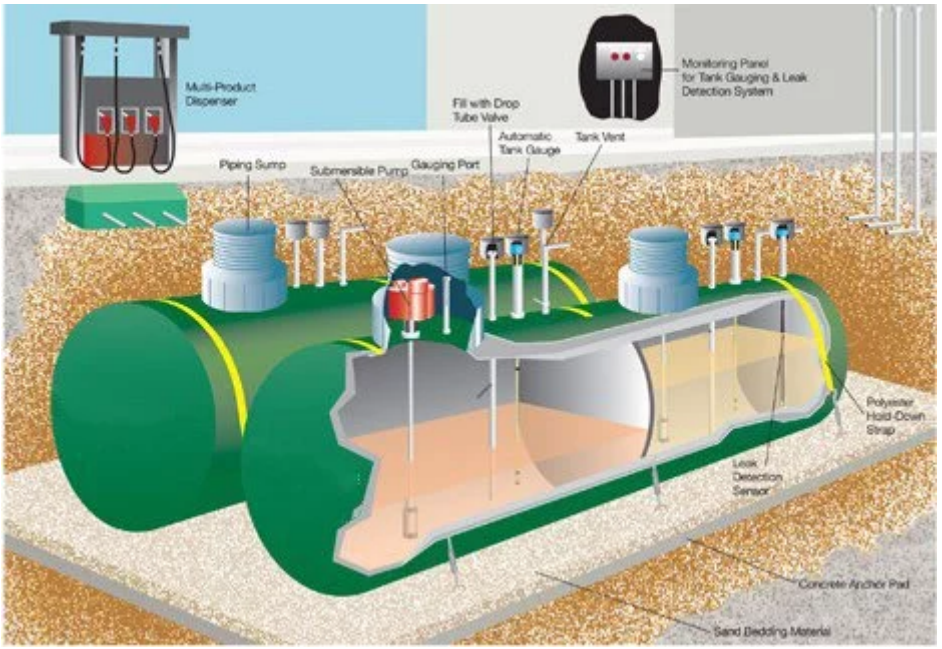


Figura 4.3. Representación de un tanque de almacenamiento

- Tanque de ventas. Estos son equivalentes a los surtidores de una estación de servicio. Estos recipientes son de un tamaño inferior y son los que simularán las ventas.

Figura 4.4. Dispensadores de una estación de servicios.

- Tanque de fugas. Este recipiente es del mismo tamaño que los depósitos de ventas. Aquí es donde iremos acumulando el agua que se fuga del tanque de almacenamiento.

- Tanque de retorno. Este tanque no representa ningún depósito real. Ha sido añadido para actuar como sumidero. En él almacenaremos el agua que sale de los tanques de ventas y fugas por lo que no tendrá relevancia a la hora de obtener resultados. A través de una bomba sumergida en este depósito, impulsaremos el agua hacia el depósito de recarga, creando así un circuito cerrado de agua que permitirá minimizar el consumo.

4.2. Materiales

A continuación se describen los materiales y componentes que se han utilizado en el montaje del prototipo. Se ha utilizado el proyecto de Eduardo Miguel Gastón Quesada [8] como referencia a la hora de escoger los materiales y componentes.

El movimiento del fluido de un depósito a otro se hace a través de mangueras que van conectadas a válvulas solenoides que son las que permiten o impiden el paso del mismo. A su vez, estas electroválvulas van acopladas a unas bombas que impulsan el fluido con mayor rapidez. La estructura es vertical, por lo que el agua termina depositándose en un tanque situado a nivel del suelo (tanque de retorno). Este tanque recoge el agua evacuada por los depósitos de ventas y fugas y, la devuelve hacia el depósito de recarga a través de una bomba de corriente continua, cerrando así el circuito de agua.

Para realizar la lectura de los volúmenes de los tanques se han utilizado unos sensores de nivel sin contacto para el depósito de recarga y unas básculas para los depósitos de almacenamiento, ventas y fugas. Para el depósito de retorno no se necesita saber la cantidad de agua que posee, ya que sólo funciona como sumidero para devolver el fluido de nuevo hacia el depósito de recarga. Respecto a los sensores de nivel, éstos van acoplados a la pared exterior del tanque recogiendo los diferentes valores de volumen. Para el resto de tanque se han utilizado diferentes básculas. A partir de la medida de masa recogida por las básculas, y sabiendo que la densidad del agua es 1 kg/L, podemos obtener el volumen de los depósitos. Debido a esta razón, se decidió usar básculas para la medición de los volúmenes de estos tanques.

Para fugar el agua del depósito de almacenamiento se ha utilizado una bomba peristáltica. La ventaja de estas bombas respecto a otras es que tienen una gran precisión en la dosificación de fluidos, es decir, pueden trabajar con diferentes tasas de flujo. Debido a esta razón, se decidió usar esta bomba para extraer el agua del depósito principal.

En cuanto a la capacidad de los depósitos, se decidió usar tanques de 4,2 litros para los depósitos de ventas y fugas. Para estos depósitos se han escogido los que fueran de sección pequeña y alargados para que pudiesen entrar por completo en las básculas. Además hay que tener en cuenta que su sección debe ser rectangular para facilitar el acople de las salidas de los depósitos. Para el tanque de recarga se utilizó un depósito de 33 litros y también de sección rectangular que permitiese incorporar con facilidad los sensores de nivel y la salida del mismo depósito. Para el tanque de almacenamiento se decidió usar un depósito de 125 litros capaz de almacenar grandes cantidades de agua. Este depósito es de sección circular aunque posee una pequeña parte de sección cuadrada donde se ha podido acoplar la salida del depósito. Además, este tanque posee una leyenda incorporada donde se pueden leer diferentes niveles de volumen. Por último, para el tanque de retorno se optó por escoger un depósito de 150 litros capaz de albergar el agua contenida en el resto de depósitos. Todos estos depósitos están conectados entre sí como se muestra en la figura 4.1.

Por último, para gestionar el funcionamiento de la controladora, se ha hecho uso de un panel de control con diferentes pulsadores que permiten cambiar de un estado a otro y, de 1 led que indica si el sistema se encuentra en estado de emergencia.

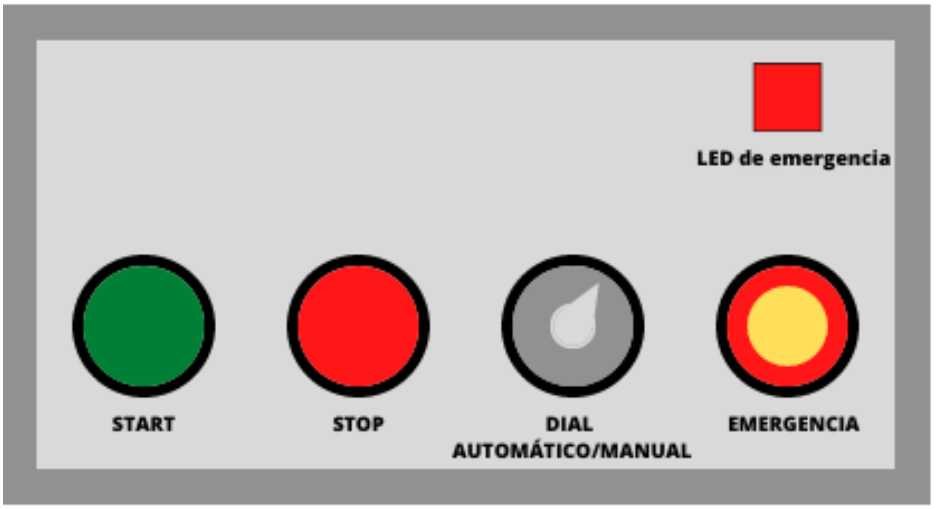


Figura 4.5. Diagrama del panel de control de la controladora

4.2.1. Electrónica de control

Atendiendo a las necesidades comentadas anteriormente, se han utilizado los siguiente componentes electrónicos de control:

- Arduino MEGA. Son las placas que se usarán para la controladora y el detector de fugas. Estas placas electrónicas son microcontroladores de código abierto basadas en el chip ATmega 2560 y formadas por 54 pines de E/S digitales, de los cuales 15 de ellos se pueden utilizar como salidas PWM, 16 pines analógicos y 4 unidades UART, que sirven para conectarse a otras placas o circuitos. Pueden ser alimentadas por un cable USB o por una batería externa de entre 6 y 20 voltios y su tensión de funcionamiento es de 5 voltios.

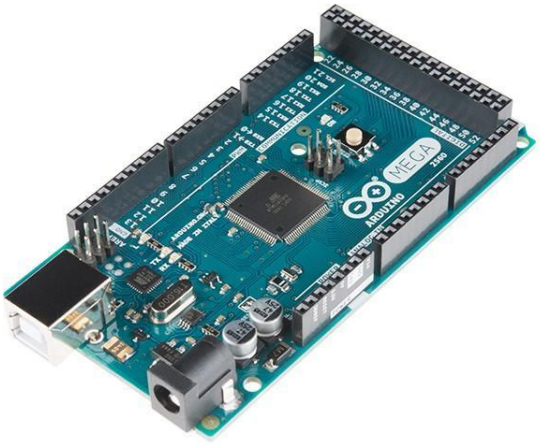


Figura 4.6. Placa Arduino MEGA

- Fuentes de alimentación. Encargadas de la alimentación de los actuadores y microcontroladores. La fuente de 12 voltios alimenta al módulo de relé, encargado del funcionamiento de las válvulas y bombas, y al driver del motor. Y la fuente de 5 voltios es la responsable de la alimentación de los microcontroladores.

- Módulos de relés. Son los encargados de permitir el paso de corriente eléctrica a las bombas y válvulas. Estos dispositivos están formados por un electroimán y una bobina que permiten abrir o cerrar el circuito eléctrico, funcionando como un interruptor controlado. En nuestro caso hemos utilizado un módulo de 16 relés, que se polarizan con una tensión continua de 12 voltios.

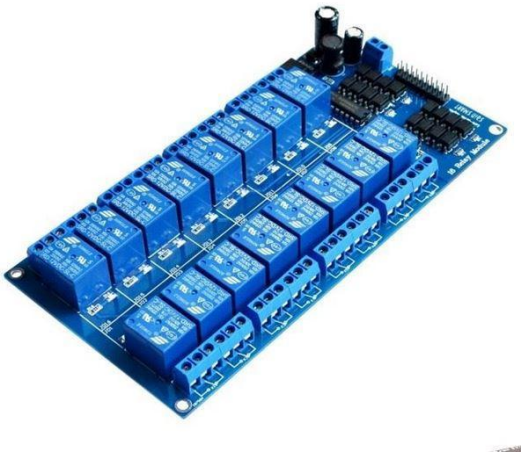


Figura 4.7. Módulo de 16 relés.

- Driver Motor L298N. Este pequeño módulo se ha utilizado para controlar la velocidad y dirección del motor de corriente continua de la bomba peristáltica. Posee un puente H formado por 4 transistores que permite invertir el sentido de la corriente y, de esta forma, permite invertir el sentido de giro del motor. Además, el módulo incluye un regulador, cuya tensión de funcionamiento es de 12 voltios, que permite obtener una tensión de 5 voltios para alimentar los microcontroladores. Aun así los microcontroladores se alimentan de la fuente de tensión de 5 voltios.

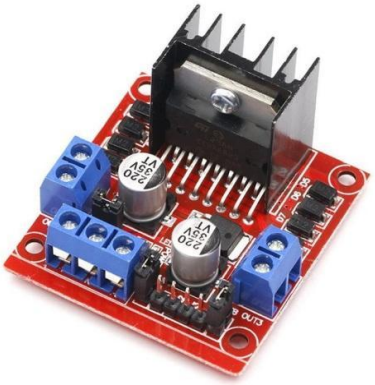


Figura 4.8. Módulo controlador de motor L298N

- Módulo HX711. Este módulo es un amplificador de célula de carga que permite leer el peso de manera sencilla en Arduino. Las células de carga se encuentran en las básculas y son transductores que convierten la fuerza aplicada sobre ellas en una señal eléctrica. Son los sensores de fuerza que utilizan las básculas. Respecto al módulo, este funciona como interfase entre las células de carga de las básculas y el microcontrolador. Esta placa se conecta al microcontrolador mediante dos pines (clock y data) y, por medio de otros cuatro pines, también se conecta a la báscula formando un puente de Wheatstone. En nuestro caso se han utilizado 4 de ellos para sus respectivas básculas.

Figura 4.9. Módulo de pesaje electrónico HX711

4.2.2. Sensores y actuadores

Con el objetivo de conseguir un correcto funcionamiento de nuestra planta, se ha hecho uso de una serie de sensores y actuadores. Los sensores son los encargados de detectar el nivel de agua, es decir, le aportan información al sistema de control sobre lo que está ocurriendo en el depósito de recarga. Una vez que la controladora recibe la información de los sensores, ésta la procesa y le envía una orden a los actuadores para que puedan realizar unas determinadas acciones. En relación a esto, se han utilizado los siguientes sensores y actuadores:

- Sensores de nivel de líquido sin contacto. Estos sensores funcionan sin la necesidad de estar en contacto con el líquido. La señal de salida puede ser de nivel alta o baja, dependiendo si se utiliza un cable más o no y, además, trabajan con un voltaje de entrada de entre 5 y 12 voltios. Se han utilizado 4 de estos sensores en el depósito de recarga, situados a la alturas equivalentes de 1, 5, 20 y 30 litros. De esta manera se pueden obtener diferentes volúmenes mediante una suma/resta entre las medidas de los sensores.

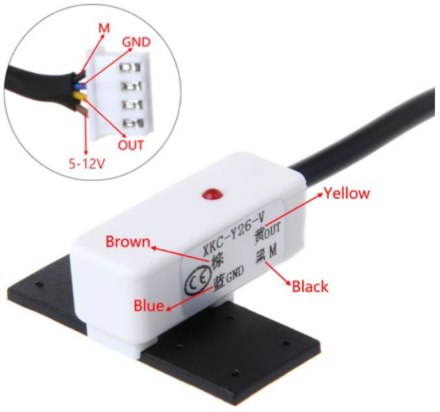


Figura 4.10. Sensor de nivel de líquido sin contacto

- Báscula del tanque de almacenamiento. Se ha utilizado una báscula de la marca WANT y modelo WT3002L de 300 kg de capacidad y 10 g de precisión ya que el depósito de almacenamiento es capaz de albergar 150 litros de agua. Como se comenta en el apartado 4.2, a partir de la medida de masa, se obtendrá el volumen mediante una conversión.

Figura 4.11. Báscula del depósito de almacenamiento

- Básculas de los tanques de ventas y fugas. Las básculas utilizadas para estos depósitos son de 5kg de capacidad y 1g de precisión. Poseen unas medidas de 20,5 x 16,5 cm, por lo que atendiendo a las medidas de los recipientes de ventas y fugas, encajan perfectamente bajo los depósitos. Como se mencionó en el apartado 4.2, también han sido utilizadas para obtener el volumen de estos recipientes a partir de la masa.

Figura 4.12. Básculas de los depósitos de ventas y fugas

- Bombas. Son las encargadas de impulsar el agua de un depósito a otro. Proporcionan un caudal de 800 L/h, se alimenta con 12 voltios y, además, son capaces de elevar el fluido hasta 5 metros.

Figura 4.13. Bomba de agua

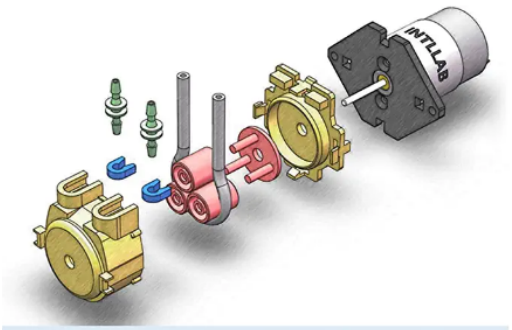
- Válvulas solenoides. Estas electroválvulas son las encargadas de permitir el paso del fluido de un tanque a otro. Funcionan normalmente cerradas y se alimentan con 12 voltios. Además, su presión mínima de operación es de 0 kg/cm^2, por lo que conmutan por el propio peso del fluido. Debido a esto, no haría falta incorporar las bombas, ya que conmutarían por el propio peso del agua, aun así, se han añadido ya que tras haber realizado diferentes experimentos con y sin bomba, se ha llegado a la conclusión de que al usar la bomba, el fluido se desplaza el doble de rápido. Se han obtenido unos tiempos de vaciado del depósito de recarga de 3 minutos y 42 segundos sin bomba, y de 1 minuto y 50 segundos con bomba

Figura 4.14. Válvulas solenoides

- Bomba peristáltica. Esta bomba es la responsable de producir las fugas del depósito de almacenamiento. Cuenta en su interior con un rotor sobre el que giran dos rodillos. Uno de ellos gira sobre el tubo, creando una depresión gracias a la cual el fluido es aspirado y arrastrado, es decir estos rodillos comprimen la manguera al girar, creando un vacío que succiona el fluido. En nuestro caso, se alimenta con 12 voltios.

Como se mencionó en el apartado 4.2, lo interesante de estas bombas es su gran capacidad de dosificación del flujo. Pueden hacer circular el agua con diferentes rangos de flujo. En este caso, las tuberías son de 3.0 x 5.0 mm, por lo que su tasa de flujo puede variar entre los 19 y 100 mL/min. En la figura 4.15. se puede observar una tabla con las diferentes tasas de flujo en función del ancho de la tubería utilizada.

Figura 4.15. Bomba peristáltica



ID×OD	FLOWRATE
1.0×3.0 (mm)	2~17 mL/min
2.0×4.0 (mm)	5~40 mL/min
3.0×5.0 (mm)	19~100 mL/min

Figura 4.16. Tasas de flujo de la bomba peristáltica

4.2.3. Planta

Respecto a los materiales utilizados para la construcción del prototipo, éstos han sido utilizados para pasar el fluido de un depósito a otro. Al tenerse que montar el prototipo dentro de una estantería con unas medidas limitadas y, para poder transportar el líquido a través de los sensores y actuadores, como son las bombas y válvulas, se requirió el uso de una serie de componentes compatibles con dichos elementos. Estos componentes se han mencionado en el Anexo 4. Además, en este anexo se pueden apreciar los diferentes croquis dibujados para cada depósito en el que se observan los distintos componentes utilizados por tanque.

4.2.4. Otros

Otros elementos a mencionar son la estantería que se ha utilizado para soportar la estructura de la planta. El mueble que se ha escogido ha sido de metal y de 2 metros de altura.

Los depósitos se han seleccionado de manera que cumplan con las especificaciones establecidas en el apartado 4.2. Además, se ha priorizado la elección de recipientes de plástico y que contengan tapa para evitar el desbordamiento de agua en caso de una emergencia.

En un primer momento, a la hora de montar los diferentes componentes a sus respectivos depósitos, surgió un problema relacionado con el tarado de los tanques. Los componentes al ir acoplados a los tanques, forman una estructura que es la que debe realizar el tarado. Debido al tamaño de las estructuras de los depósitos con sus componentes, estas se salían de las básculas por lo que podrían romperse debido a que quedaban colgando. A causa de esto, se decidió sujetar los respectivos componentes de los tanques de alguna manera.

El método más sencillo sería sujetar los diferentes componentes con nylon a la balda que tuviese encima. El problema de esto sería que a la hora de realizar el tarado del depósito, intervendrían fuerzas externas, como es en este caso, una fuerza de tensión que tira hacia arriba, por lo que la medida del tarado no sería válida. Debido a esto, se decidió buscar una solución para poder tarar

la estructura completa sin que intervinieran agentes externos. Se barajó la posibilidad de colgar los componentes del mismo depósito con nylon, como utilizó Eduardo Miguel Gastón Quesada en su proyecto [8]. El inconveniente de realizarlo de esta manera es que por un lado del depósito había más peso que por otro lado debido a la diferencia de peso entre una bomba y una válvula. Para ello se necesitaría del uso de contrapeso por un lado para equilibrar ambos lados del depósito. Esta opción requería del uso de más componentes, como son los contrapesos y, además, parecía más compleja.

También surgió la idea de utilizar una cinta de amarre con trinquetes y atarla al mismo depósito. De esta cinta irían colgados los componentes con nylon. Esta opción sólo sería válida para el depósito de almacenamiento ya que la cinta era bastante grande para acoplarse a los depósitos de ventas y fugas.

Por último, se decidió utilizar unas tablas que sirvieran como base de las estructuras. En un primer momento se pensó en utilizar bases de metacrilato para soportar las estructuras, pero por falta de tiempo, se decidió usar unos tablones cuadrados de madera en vez de estas tablas de metacrilato. Esta opción era la más cómoda y económica ya que sólo habría que situar las estructuras sobre sus bases y éstas sobre las básculas. Estos tablones sólo se han colocado bajo los depósitos de almacenamiento, ventas y fugas ya que son los únicos que presentan una báscula bajo ellos.

Figura 4.17. Base de madera para el tanque de almacenamiento

Respecto al panel de control de la controladora se ha utilizado una protoboard con 4 interruptores que representasen los pulsadores de START, STOP, EMERGENCIA y DIAL, y 1 led que se encendiera en caso de emergencia. En un primer momento se iba a colocar una caja botonera, pero por falta de tiempo se decidió colocar la protoboard con estos componentes.

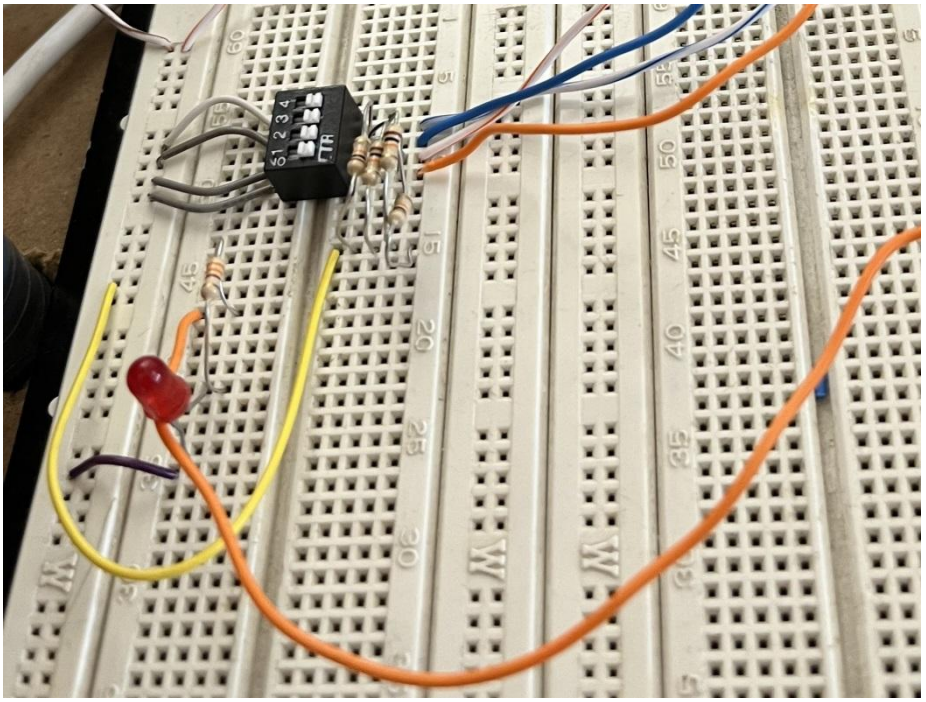


Figura 4.18. Panel de control de la controladora

5. CONTROLADORA

5.1. Funcionamiento

La controladora desempeña un papel fundamental en el desarrollo del trabajo. Ésta es la encargada del funcionamiento de la planta. Es la responsable de la apertura y cierre de las válvulas o de producir las fugas del depósito de almacenamiento. También se encarga de generar los valores de descargas y ventas de los tanques a partir de unos rangos de descarga y venta máxima y mínima que se le han asignado. Lo mismo ocurre con los tiempos de descargas y ventas.

Es la propia controladora la que irá generando nuevos eventos como pueden ser descargas o ventas del depósito de almacenamiento. Mediante una distribución aleatoria se han obtenido los valores de los volúmenes de descargas y ventas, y los tiempos en los que se han producido estos eventos. Estos eventos se generan aleatoriamente a partir de unas distribuciones de probabilidad previamente parametrizadas. Cada evento se caracteriza por el instante de tiempo en el que se produce y por el volumen de líquido que lleva asociado, tanto si es una descarga como una venta.

A partir de las controladoras que se nos entregaron de proyectos anteriores, se han tenido que fusionar ambas para adaptarla a este prototipo.

La primera de ellas cuenta con los mismos depósitos que se han utilizado en este prototipo, a diferencia de que esta se ha implementado en un simulador online, por lo que no cuenta con los mismos métodos y variables debido a que no se han utilizado componentes reales.

La segunda presenta un solo depósito de ventas en el que se realizan las descargas del depósito de almacenamiento y, un depósito de fugas encargado de acumular las filtraciones del depósito principal. Respecto a las funciones utilizadas en esta segunda controladora, se han usado las mismas debido a que en este segundo proyecto [8] si se ha construido un prototipo físico, por lo que las variables y los métodos son los mismos.

En este caso, se ha añadido un segundo depósito de ventas, encargado también de acumular descargas del tanque de almacenamiento, por lo que se han agregado nuevas variables para este segundo depósito de ventas. También se ha añadido y modificado algunas variables de volúmenes y tiempos, ya que al haber tratado con un prototipo de mayores dimensiones, se ha tenido que trabajar con mayores cantidades de agua. Además, se han agregado unas líneas de código cuya función ha sido transmitir información al sistema de detección de fugas que se explicará más adelante.

5.2. Máquina de estados de la controladora

Al igual que en los proyectos anteriores, para la controladora se ha utilizado una máquina de estados del tipo Moore, en el que las salidas del sistema dependen del estado actual en el que se encuentre. Esta máquina de estados representa el funcionamiento de la planta. En ella se puede observar como la planta cambia de estado en función de las condiciones establecidas. A su vez, esta máquina tiene asociada otra máquina de estados del tipo Moore por cada depósito. Esta máquina progresa en función de los cambios de estado de la máquina principal, por lo que una actúa en consecuencia de la otra.

Para la máquina de estados de la controladora, se ha implementado la máquina de los proyectos anteriores, la cual se intenta asemejar al diseño de una planta industrial. A continuación, se pasará a explicar el funcionamiento de esta máquina.

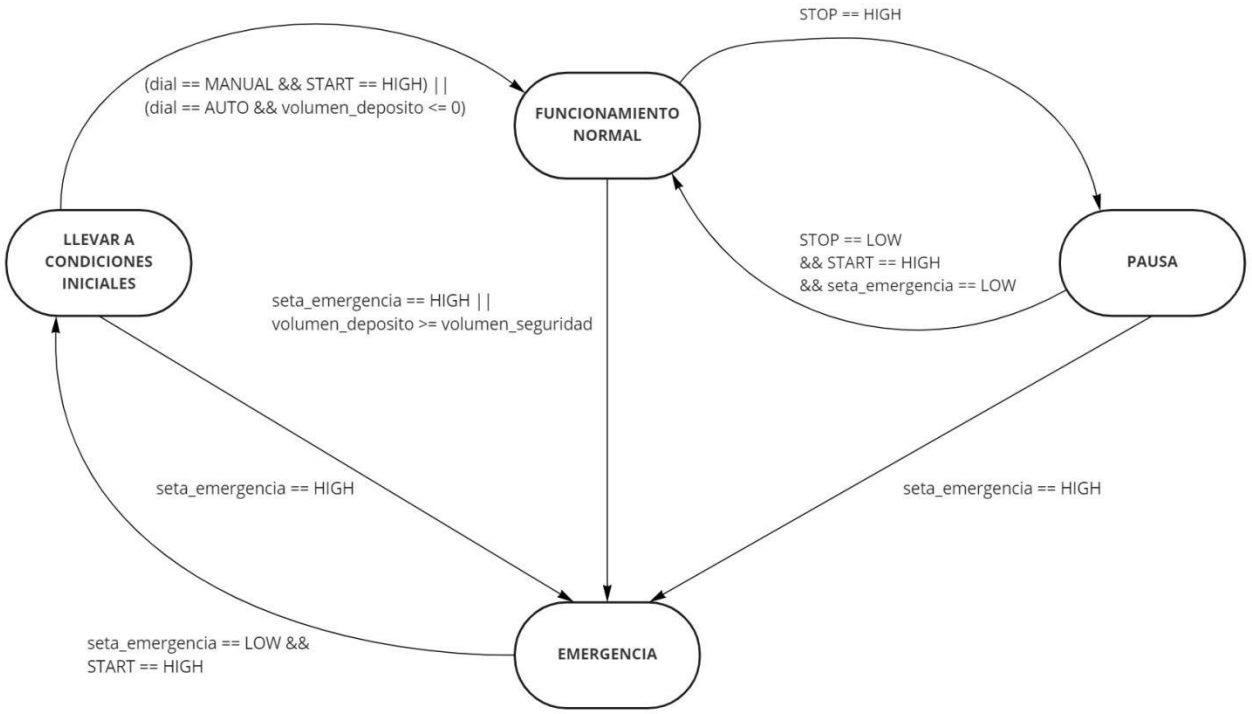


Figura 5.1. Máquina de estados de la controladora

Como se puede observar, la máquina de estados de la controladora está formada por 4 estados los cuales representan el funcionamiento de la planta. Estos estados consisten en lo siguiente:

- FUNCIONAMIENTO NORMAL. En este estado se encuentra la planta si no hay ningún problema. Los depósitos actualizan su estado con el fin de observar en qué momento del proceso se encuentran. A partir de este estado se puede entrar en los estados de PAUSA y EMERGENCIA.

- PAUSA. Accionando el pulsador de STOP se entra en el estado de PAUSA. En él, todas las válvulas y bombas se cierran a la espera de que se vuelva a accionar el pulsador de START para poder entrar de nuevo en FUNCIONAMIENTO NORMAL.

- EMERGENCIA. Como bien dice el nombre, para entrar en este estado debe haber un problema, ya sea activando la SETA DE EMERGENCIA o cuando el nivel de un depósito sobrepase su volumen de seguridad, por lo que se puede entrar en estado desde cualquier otro. Para volver al estado de FUNCIONAMIENTO NORMAL, se debe pasar primero por el estado de LLEVAR A CONDICIONES INICIALES. Para ello se debe desactivar la seta de emergencia y pulsar el botón de START.

- LLEVAR A CONDICIONES INICIALES. Una vez se salga de la posición de EMERGENCIA, se entrará en este estado. Para volver a arrancar la simulación, se debe reiniciar el sistema vaciando todos los depósitos y, una vez vaciados, podemos volver al estado de FUNCIONAMIENTO NORMAL de dos maneras. La primera de ellas, de forma manual accionando el pulsador de START o, de manera automática, cuando todos los depósitos estén vacíos.

Para controlar el funcionamiento de la controladora, se hará uso de un panel de control con cuatro pulsadores y un indicador led que se encenderá en caso de emergencia. La función de cada pulsador es la siguiente:

- START. Este pulsador permite reanudar la simulación en caso de PAUSA o, en caso de emergencia, reiniciar el sistema y, posteriormente y de forma manual, volver a reanudar la simulación.

- STOP. Accionando este botón se puede parar el sistema entrando en modo PAUSA. Para reanudar de nuevo el sistema, habría que presionar el pulsador de START.

- Dial manual/automático. Este pulsador determina el modo de funcionamiento de la planta. Normalmente se encuentra en la posición de automático, no obstante, en algunos casos interesa ponerlo en modo manual, cuando se esté en el estado de reinicio, si queremos reanudar la simulación sin haber vaciado los depósitos por completo.

- SETA DE EMERGENCIA. Es el interruptor encargado de colocar el sistema en estado de emergencia. Este botón se puede accionar desde cualquier estado. Una vez se active el pulsador, se encenderá automáticamente un led que indicará que se está en estado de emergencia. Para reanudar la simulación se debe volver a accionar este pulsador y activar el interruptor de START.

Figura 5.2 Panel de control de la controladora

5.3. Máquina de estados de los depósitos

Como mencionamos anteriormente, la máquina de estados de los depósitos actúa en consecuencia de la máquina de estados de la controladora. Los estados de esta máquina hacen referencia a las cantidades de agua de los tanques. Para explicar el comportamiento de ésta, se ha implementado la máquina de estados de los proyectos anteriores.

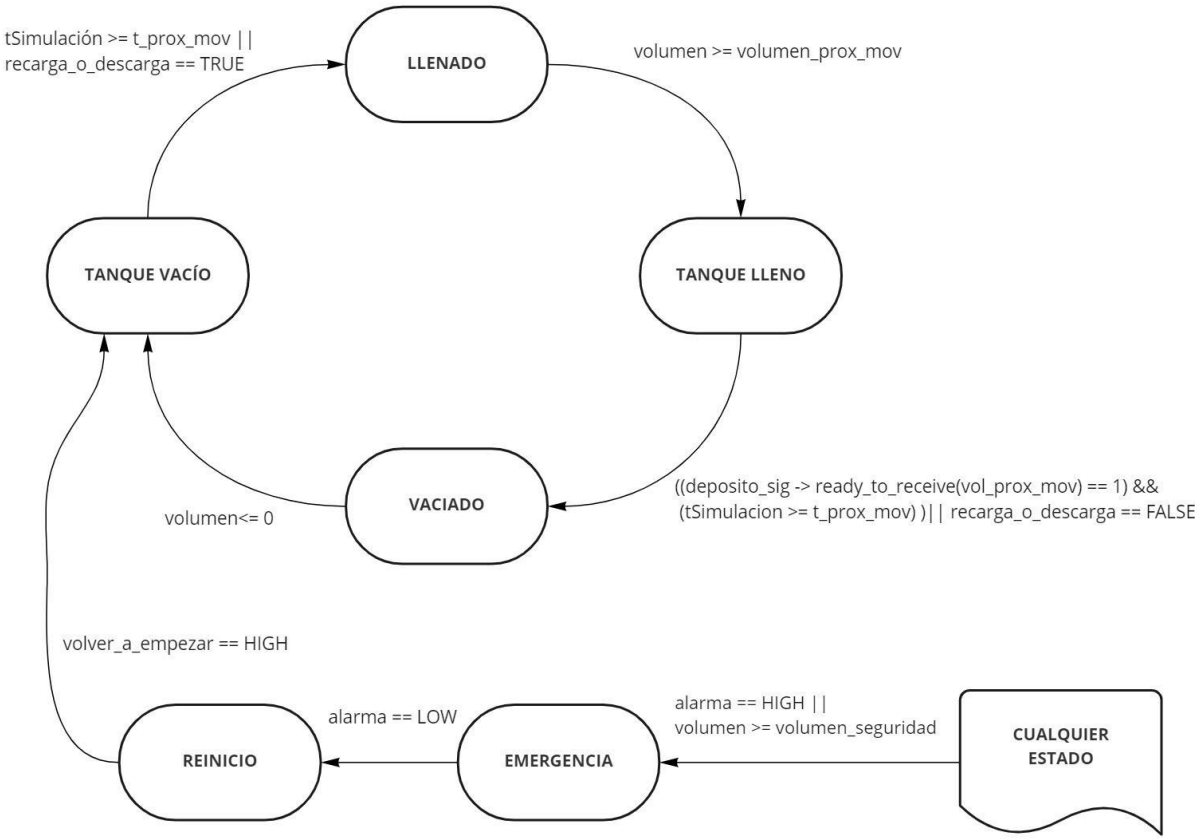


Figura 5.3. Máquina de estados de los depósitos

Los estados de esta máquina hacen referencia a los niveles de agua en los tanques. A continuación se pasará a explicar el funcionamiento de la misma, sin tener en cuenta los estados de emergencia y reinicio.

- TANQUE VACÍO. Es el estado inicial en el que se encuentran los tanques. Las válvulas de llenado y vaciado se encuentran cerradas hasta que el sistema cambie de estado.

- LLENADO. Situación en el que un tanque se encuentra llenándose. La válvula de llenado permanece abierta mientras que la de vaciado permanece cerrada. El sistema se sitúa en este estado hasta que el depósito se llene con la cantidad de agua establecida.

- TANQUE LLENO. Una vez que el depósito alcanza el nivel de agua establecido, las válvulas de llenado y vaciado permanecen cerradas, a esperas de que el sistema ordene una descarga del mismo.

- VACIADO. La válvula de vaciado se encuentra abierta y la de llenado cerrada. Una vez vaciado el depósito, ambas válvulas permanecen cerradas a esperas de una nueva recarga.

- EMERGENCIA. En este estado todas las válvulas permanecen cerradas. Hasta que no se vuelva a poner la seta de emergencia en su posición inicial, no se vuelven a abrir las válvulas

- REINICIO. Se abren todas las válvulas para vaciar los tanques y poder volver a empezar la simulación.

La máquina de estados anterior hace referencia a los depósitos en general. A parte, cada uno de ellos tiene sus propios atributos como puede ser volumen de seguridad, volumen límite, tiempo de recarga o tiempo de descarga. Al ser de diferentes tamaños, cada uno tiene sus propias cualidades. Operan de forma independiente, aunque pueden recoger datos de otros depósitos, ya sea de los tanques siguientes o anteriores, para ver si están preparados para recibir o realizar una descarga

6. MÉTODO DE DETECCIÓN DE FUGAS

6.1. Idea general

.

El sistema de detección de fugas es el elemento que nos permite detectar si se producen fugas en el depósito de almacenamiento. Al igual que para la controladora, hemos utilizado otro microcontrolador para el detector de fugas.

Este detector recibe, por parte de la controladora, una cadena de valores de volumen de los tanques periódicamente. La comunicación entre ambos microcontroladores se realiza mediante comunicación serial. Esto quiere decir que ambos microcontroladores van conectados entre sí mediante los puertos serie Tx y Rx, por lo que ambos pueden recibir y transmitir información. La controladora recoge los valores de volumen principal, volumen de recargas acumuladas, volumen de ventas acumuladas y volumen de fugas acumuladas, y los agrupa en una cadena que irá enviando periódicamente al detector de fugas. Para realizar la recogida de volúmenes, los tanques deben estar en estado de pausa y, una vez se encuentre la controladora en este estado, el datalogger de la controladora registra los datos y se los envía al detector de fugas. A través de una variable se le asignará el tiempo periódico al datalogger para la recogida de datos.

Por otro lado, el detector de fugas recibe la cadena que le proporciona la controladora y la descompone para, posteriormente, realizar una análisis de los valores de volumen recogidos. A partir de estos valores el detector obtiene las variaciones de volumen que se producen entre una cadena y otra y, una vez que las variaciones superen un cierto umbral que se le asignará, se activará un led indicando que se ha producido una alarma

Para contrastar los resultados, una de las básculas situada bajo el recipiente de fugas servirá de ayuda para ver si los resultados obtenidos son coherentes.

6.1.1. Diseño de hardware

Para construir el sistema de detección de fugas se ha hecho uso de un segundo microcontrolador. Como ya se ha mencionado, éste va conectado a la controladora mediante los cables TX, RX y GND. Estos cables le permiten interactuar con la controladora, siendo capaz de enviar y recibir información.

En cuanto a las salidas del detector de fugas, se ha conectado un led en serie con una resistencia de 330 ohmios para evitar que éste se queme. Ambos van conectados al microcontrolador creando un circuito cerrado por el que pasa corriente eléctrica cuando el detector da la orden.

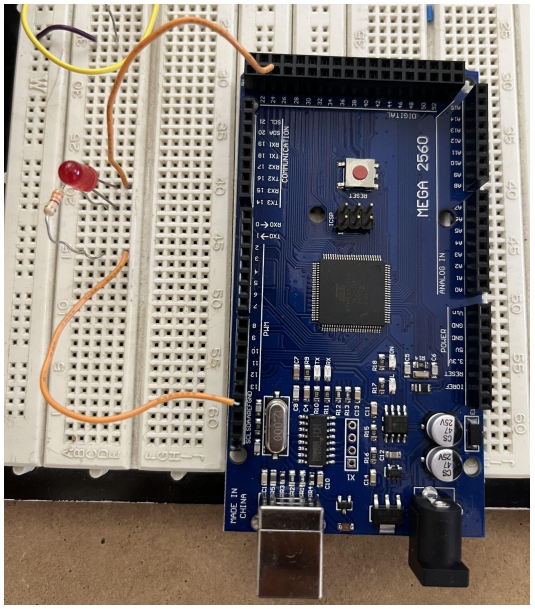


Figura 6.1. Sistema de detección de fugas

6.2. Funcionamiento

El sistema de detección de fugas trata de analizar los valores de volumen que le aporta la controladora periódicamente. Para ello, primero realiza una lectura de la cadena enviada por la controladora con los valores obtenidos de la simulación. A partir de estos datos, realiza un estudio y, en función de los resultados obtenidos, hace saltar la alarma indicando de que se ha producido una fuga. Para explicar el funcionamiento del detector haremos uso de una serie de imágenes que han sido diseñadas con el objetivo de entender bien la explicación. Los valores no son reales.

A través de la controladora, el detector de fugas recibe los resultados obtenidos de la simulación. Estos valores son los volúmenes acumulados de descargas y ventas del tanque principal. En la figura 7.1. se puede observar una primera cadena enviada por la controladora con los volúmenes mencionados.

Cadenas	Volumen_principal	Volumen-acumulado_descargas	Volumen_acumulado_ventas_1	Volumen_acumulado_ventas_2
nº1	100 L	20 L	10 L	5 L

Figura 6.2. Primera cadena de la controladora

A partir de las descargas y ventas acumuladas por muestra, el detector de fugas obtiene lo que debería ser el volumen final tras haber realizado estos procesos. En la figura 7.2. se observa el teórico volumen final del tanque principal tras haberse producido una descarga y venta.

Cadenas	Volumen_principal	Volumen-acumulado_descargas	Volumen_acumulado_ventas_1	Volumen_acumulado_ventas_2	Volumen final teórico
nº1	100 L	20 L	10 L	5 L	105 L

Figura 6.3. Primera cadena de la controladora y volumen final teórico

A partir del valor obtenido anteriormente, el detector realiza un estudio comparando el que debería ser el volumen de almacenamiento de la siguiente muestra, es decir, el volumen final teórico obtenido anteriormente, con el volumen real del tanque de almacenamiento que le llega de la siguiente muestra. Estas variaciones las irá acumulando el detector. En la figura 7.3. se observa el que debería ser el nuevo volumen de almacenamiento, con el volumen real de almacenamiento de la siguiente muestra.

Cadenas	Volumen_principal	Volumen-acumulado_descargas	Volumen_acumulado_ventas_1	Volumen_acumulado_ventas_2	Volumen final teórico
nº1	100 L	20 L	10 L	5 L	105 L
nº2	102 L	22 L	10 L	10 L	104 L

Figura 6.4. Primera y segunda cadena de la controladora y volúmenes finales teóricos

Por último, a partir de estas variaciones, el detector establece un umbral que servirá como condición para hacer saltar la alarma e indicar de que se ha producido una fuga en el tanque principal.

7. VALIDACIÓN Y RESULTADOS

7.1. Validación de la controladora

Con el objetivo de poder validar el funcionamiento del prototipo, se han realizado diferentes pruebas con el fin de comprobar si los códigos implementados funcionan correctamente.

Los dos primeros experimentos se han realizado solamente con un depósito de ventas, en lugar de dos. En estos ensayos sólo se ha utilizado un microcontrolador, el de la controladora. De esta manera se podrá comprobar si la controladora que se ha proporcionado funciona correctamente. A continuación se explicarán los dos primeros experimentos:

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas realizadas	Ventas acumuladas (L)	Variaciones (L)
20,01	0	0,00	0	0,00	0,00
14,95	0	0,00	2	5,02	0,04
14,95	1	1,00	1	6,03	0,01
13,93	0	1,00	1	7,05	0,00
13,93	0	1,00	0	7,05	0,00
13,92	0	1,00	0	7,05	0,01
16,65	1	3,76	0	7,05	0,03
16,68	0	3,76	0	7,05	0,00
13,82	0	3,76	2	10,03	0,12
13,74	0	3,76	0	10,03	0,08
13,70	0	3,76	0	10,03	0,00
13,70	0	3,76	0	10,03	0,00
13,70	0	3,76	0	10,03	0,00
13,69	0	3,76	0	10,03	0,01
13,70	0	3,76	0	10,03	0,01
13,70	0	3,76	0	10,03	0,00
16,39	1	6,51	0	10,03	0,06
14,47	0	6,51	1	12,04	0,09
12,42	0	6,51	1	14,05	0,04
12,42	0	6,51	0	14,05	0,00
10,53	0	6,51	1	16,05	0,11
8,43	0	6,51	1	18,06	0,09
8,42	0	6,51	0	18,06	0,01
5,63	0	6,51	3	21,05	0,20
5,44	0	6,51	0	21,05	0,20

Tabla 7.1. Resultados del 1º experimento sin fugas

En esta primera tabla se pueden apreciar los valores que se han ido recogiendo a lo largo del primer experimento. Se ha utilizado un tiempo de 20 segundos para realizar el datalogger, por lo que cada 20 segundos se ha recogido una nueva muestra. Hay que centrarse en tres de las cinco variables: volumen principal, descargas acumuladas y ventas acumuladas. Como se puede observar, inicialmente el tanque de almacenamiento cuenta con un volumen de 20,01 litros y, terminada la simulación, pasa a tener 5,44 litros. En este caso, el depósito principal ha disminuido su volumen respecto al que tenía al inicio de la simulación.

Para entender esto, hay que fijarse en el volumen final de las descargas acumuladas y en el volumen final de las ventas acumuladas. Como se ha mencionado, son los volúmenes acumulados, por lo que los últimos valores de ambas columnas, son las descargas y ventas que se han hecho en total a lo largo del experimento. Se puede ver como el volumen de ventas acumulado de 21,05 litros, es mayor que el volumen de descargas acumuladas de 6,51 litros. Esto lo que quiere decir es que al final del experimento, el volumen del depósito principal debería haber disminuido en relación a la diferencia entre las descargas y ventas acumuladas. Siendo exactos, el volumen del tanque de almacenamiento debería haber disminuido 14,54 litros, es decir, obtener un volumen final de 5,47 litros. Debido a la alta sensibilidad de las básculas, no se ha obtenido un valor teórico exacto, pero sí se ha acercado bastante a ello. Como resultado final se ha obtenido un volumen de 5,44 litros.

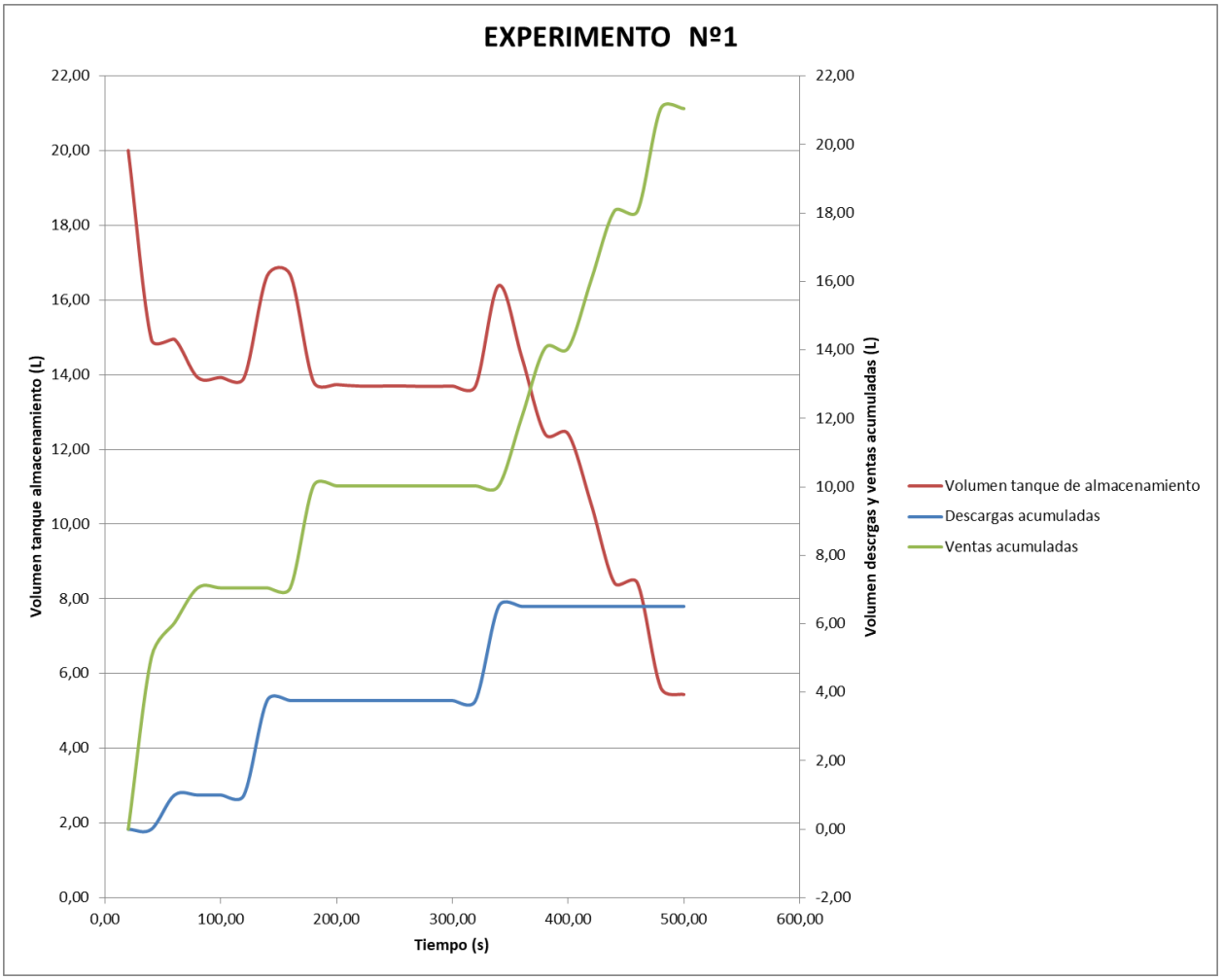


Figura 7.1. Gráfico de volúmenes del 1º experimento

En esta gráfica se observa el comportamiento de la controladora mencionado anteriormente.

En cuanto al segundo experimento, se han obtenido los siguientes valores:

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas realizadas	Ventas acumuladas (L)	Variaciones (L)
22,32	0	0,00	0	0,00	0,00
22,32	0	0,00	0	0,00	0,00
22,32	0	0,00	0	0,00	0,00
24,89	1	2,76	0	0,00	0,19
23,21	0	2,76	1	2,00	0,32
22,92	0	2,76	0	2,00	0,29
22,89	0	2,76	0	2,00	0,03
22,89	0	2,76	0	2,00	0,00
22,89	0	2,76	0	2,00	0,00
25,45	1	5,51	0	2,00	0,19
23,79	0	5,51	1	4,00	0,34
23,34	0	5,51	1	4,50	0,05
23,14	0	5,51	0	4,50	0,20
21,23	0	5,51	1	6,50	0,09
21,22	0	5,51	0	6,50	0,01
23,99	1	8,27	0	6,50	0,01
23,01	0	8,27	1	7,50	0,02

Tabla 7.2. Resultados del 2º experimento sin fugas

En este caso ocurre lo contrario, se parte de inicio con un volumen de 22,32 litros en el depósito principal y, una vez finalizado el experimento, se obtiene un volumen de 23,01 litros. En este caso, el volumen del tanque principal se ha incrementado 0,69 litros. Siguiendo los mismos pasos realizados en el primer experimento, se puede observar cómo las descargas acumuladas son mayores que las ventas acumuladas, por lo que al término del experimento debería haber incrementado el volumen del tanque principal. Teóricamente se debería estar en 23,09 litros, pero debido a la sensibilidad de las básculas, el volumen obtenido no es exactamente el mismo. En lugar de este valor, se ha obtenido un volumen de 23,01 litros.

A continuación, se muestra una gráfica para entender mejor la explicación.

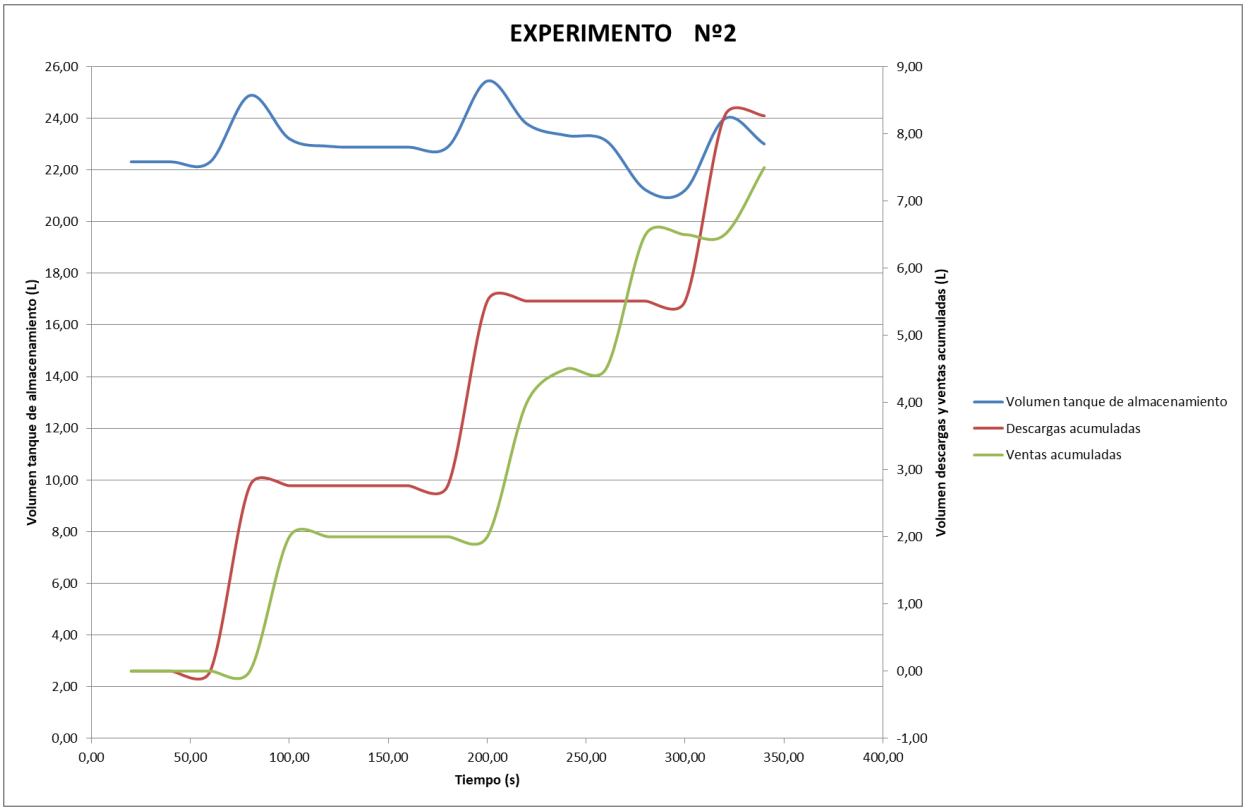


Figura 7.2. Gráfico de volúmenes del 2º experimento

En este caso, la gráfica del volumen del tanque de almacenamiento ha aumentado levemente. Se puede apreciar como las descargas acumuladas, al finalizar el experimento, son algo mayores que las ventas acumuladas. De todos modos, la diferencia entre las dos curvas es pequeña. En consecuencia a esto, se puede observar un pequeño incremento del volumen del depósito de almacenamiento al final del experimento con respecto al inicio.

En conclusión, con estos dos primeros experimentos se ha podido comprobar el correcto funcionamiento de la controladora. Se han realizado unos pequeños cambios en ella debido a que este prototipo es más grande y, debido a esto, se han modificado algunas variables de volumen y tiempo. Aun así, se ha conseguido que la controladora modificada funcionara correctamente. Estas pruebas deberían haberse hecho repetidamente para luego tener datos de la precisión del sistema pero, debido a los contratiempos que han surgido, no se ha podido realizar un mayor número de pruebas.

7.2. Simulaciones con 1 depósito de ventas

Una vez comprobado el funcionamiento de la controladora, se pasaron a generar las primeras fugas. Para hacer funcionar la bomba peristáltica se necesitó generar una señal PWM. Para ello se le asignó un ciclo de trabajo del 100%. Este parámetro permite controlar el voltaje promedio de la señal, es decir, variar el voltaje de salida. Si el ciclo de trabajo es del 100%, se obtiene un voltaje de salida de 12 voltios en este caso, por lo que la bomba ha trabajado a su máxima velocidad. Los siguientes experimentos han sido realizados utilizando un depósito de ventas y el depósito de fugas.

Respecto al primer experimento, se han obtenido los siguientes resultados:

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas realizadas	Ventas acumuladas (L)	Volumen de fuga medido acumulado (L)	Variaciones (L)
18,92	0	0,00	0	0,00	-0,01	0,00
13,87	0	0,00	2	5,02	0,00	0,03
13,87	1	1,00	1	6,03	0,00	0,01
12,84	0	1,00	1	7,05	0,01	0,01
12,84	0	1,00	0	7,05	0,01	0,00
12,84	0	1,00	0	7,05	0,02	0,00
15,58	1	3,76	0	7,05	0,03	0,02
15,59	0	3,76	0	7,05	0,03	0,01
12,59	0	3,76	2	10,03	0,04	0,02
12,60	0	3,76	0	10,03	0,04	0,01
12,59	0	3,76	0	10,03	0,05	0,01
12,59	0	3,76	0	10,03	0,05	0,00
12,58	0	3,76	0	10,03	0,06	0,01
12,57	0	3,76	0	10,03	0,06	0,01
12,56	0	3,76	0	10,03	0,06	0,01
12,56	0	3,76	0	10,03	0,07	0,00
15,32	1	6,51	0	10,03	0,08	0,01
13,30	0	6,51	1	12,04	0,09	0,01
11,28	0	6,51	1	14,05	0,10	0,01
11,31	0	6,51	0	14,05	0,10	0,03
9,27	0	6,51	1	16,05	0,10	0,04
7,26	0	6,51	1	18,06	0,10	0,00
7,27	0	6,51	0	18,06	0,11	0,01
4,30	0	6,51	3	21,05	0,11	0,02
4,29	0	6,51	0	21,05	0,13	0,01

Tabla 7.3. Resultados del 1º experimento con fugas y 1 depósito de ventas.

En las dos últimas columnas de la tabla 7.3. se puede observar el "Volumen de fuga medido acumulado", que refleja el volumen de agua que se ha ido acumulando en el depósito de fugas. Esta columna sirve de apoyo para visualizar la cantidad de agua que se va acumulando en el recipiente de fugas, es decir, el total de agua que se ha acumulado al finalizar el experimento. Y en la última columna "Variaciones", se han registrado las variaciones que se producen entre cada envío de datos de la controladora al detector de fugas. Como se ha explicado en el apartado 6.2, el volumen final teórico debería coincidir con el volumen final experimental. A continuación, se hará uso de la tabla 7.4. para explicar de dónde salen los valores de esta columna.

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas realizadas	Ventas acumuladas (L)	Volumen de fuga medido acumulado (L)	Variaciones (L)
18,92	0	0,00	0	0,00	-0,01	0,00
13,87	0	0,00	2	5,02	0,00	0,03

Tabla 7.4. 1º y 2º cadena del primer experimento con fugas y 1 depósito de ventas.

Al iniciar la simulación, se parte con un volumen de 18,92 litros en el volumen de almacenamiento. Pasados 20 segundos, llega la segunda cadena con nuevos valores de volúmenes. Ahora se puede apreciar cómo se ha producido una venta de 5,02 litros, pero no se ha realizado ninguna descarga. En este caso, el volumen del tanque de almacenamiento debería haber disminuido 5,02 litros, es decir, el volumen del tanque principal debería ser de 13,90 litros (volumen final teórico). En lugar de este valor, se obtiene un volumen de 13,87 litros (volumen final medido). Esta variación es la que se irá anotando en la última columna de la tabla.

A continuación se puede observar un histograma con las variaciones obtenidas del primer experimento.

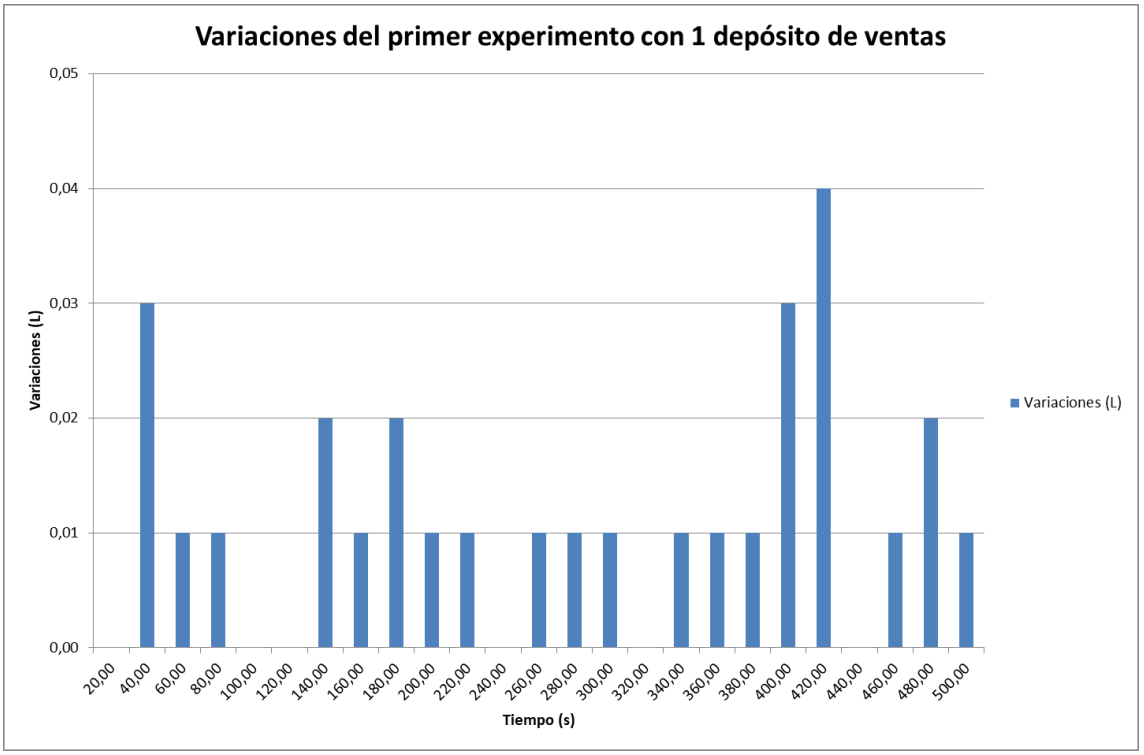


Figura 7.3. Histograma nº1 de variaciones obtenidas con 1 depósito de ventas

En cuanto al segundo experimento, se han extraído los siguientes resultados

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas realizadas	Ventas acumuladas (L)	Volumen de fuga medido acumulado (L)	Variaciones (L)
18,90	0	0,00	0	0,00	0,00	0,00
18,90	0	0,00	0	0,00	0,00	0,00
18,89	0	0,00	0	0,00	0,00	0,01
21,65	1	2,76	0	0,00	0,00	0,00
19,63	0	2,76	1	2,00	0,01	0,02
19,64	0	2,76	0	2,00	0,01	0,01
19,65	0	2,76	0	2,00	0,01	0,01
19,64	0	2,76	0	2,00	0,02	0,01
19,64	0	2,76	0	2,00	0,02	0,00
22,37	1	5,51	0	2,00	0,03	0,02
20,36	0	5,51	1	4,00	0,03	0,01
19,84	0	5,51	1	4,50	0,04	0,02
19,85	0	5,51	0	4,50	0,04	0,01
17,82	0	5,51	1	6,50	0,04	0,03
17,86	0	5,51	0	6,50	0,04	0,04
20,60	1	8,27	0	6,50	0,05	0,02

Tabla 7.5. Resultados del 2º experimento con fugas y 1 depósito de ventas.

En este segundo experimento también se puede comprobar como el volumen final teórico (20,67 litros) no coincide con el volumen final medido (20,60 litros). En este caso, la diferencia de volumen es de 0,07 litros y, como se puede observar, el volumen acumulado de las fugas, al término de la simulación, es de 0,05 litros. En algunos casos no coinciden exactamente estos valores, pero esto es debido a pequeños fallos que se producen, como es el caso de alguna gota de agua que no llegue a tiempo al depósito de fugas cuando se ha realiza el datalogger o algún pequeño error de la báscula al medir debido a su alta sensibilidad.

A partir de la tabla 7.5, se ha realizado un histograma con las variaciones adquiridas de este experimento:

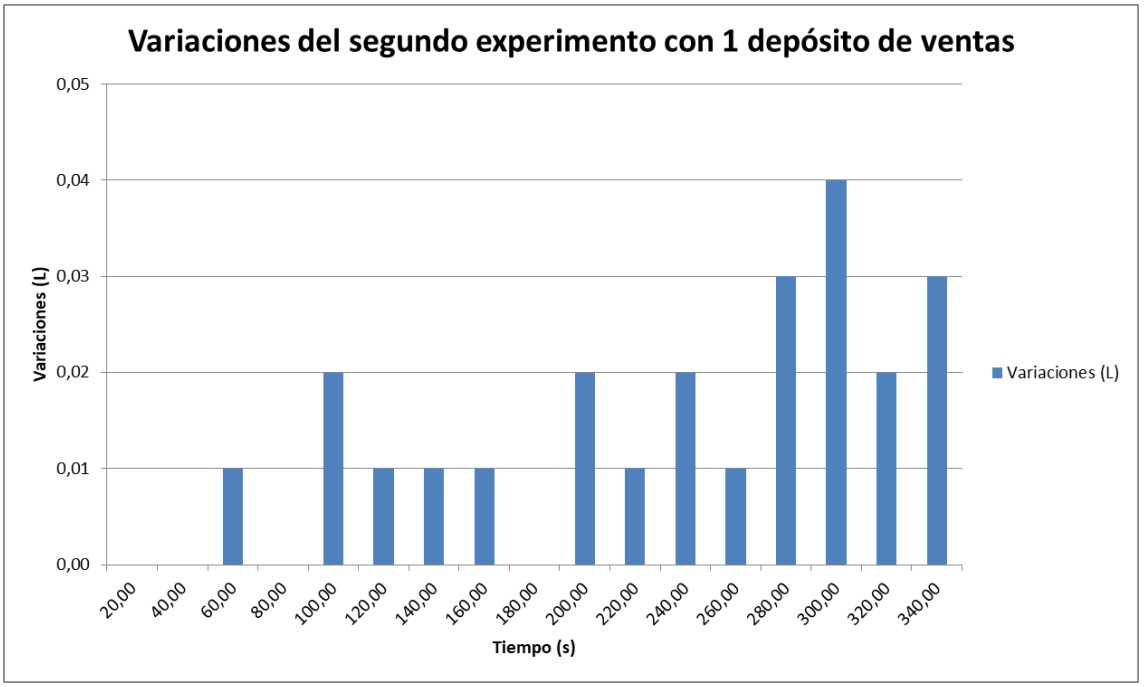


Figura 7.4. Histograma nº2 de variaciones obtenidas con 1 depósito de ventas

7.3. Simulaciones con 2 depósito de ventas

Una vez se haya comprendido el sistema de detección de fugas con un solo depósito de ventas, se procederá a añadir un segundo tanque de ventas. El prototipo físico consta con dos de estos recipientes, por lo que de esta manera se conseguirá validar el modelo real. De nuevo, se han realizado dos experimentos que permitirán realizar un estudio con el fin de obtener el porcentaje de probabilidad de que se produzcan fugas en el depósito principal.

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas 1 realizadas	Ventas 1 acumuladas (L)	Ventas 2 realizadas	Ventas 2 acumuladas (L)	Volumen de fuga medido acumulado (L)	Variaciones (L)
24,78	0	0,00	0	0,00	0	0,00	0,00	0,00
24,78	0	0,00	0	0,00	0	0,00	0,00	0,00
24,78	0	0,00	0	0,00	0	0,00	0,01	0,00
26,66	1	2,92	1	0,33	1	0,70	0,01	0,01
26,66	0	2,92	0	0,33	1	0,70	0,02	0,00
26,66	0	2,92	0	0,33	1	0,70	0,02	0,00
26,66	0	2,92	0	0,33	1	0,70	0,02	0,00
26,65	0	2,92	0	0,33	1	0,70	0,03	0,01
26,65	0	2,92	0	0,33	1	0,70	0,03	0,00
26,64	0	2,92	0	0,33	1	0,70	0,04	0,01
26,64	0	2,92	0	0,33	1	0,70	0,04	0,00
26,64	0	2,92	0	0,33	1	0,70	0,04	0,00
26,64	1	2,92	0	0,33	1	0,70	0,05	0,00
35,92	0	13,76	0	0,33	2	2,25	0,05	0,01
30,19	0	13,76	1	3,31	4	5,00	0,06	0,00
27,28	0	13,76	0	3,31	5	7,90	0,06	0,01
25,36	0	13,76	1	5,21	5	7,90	0,07	0,02
25,36	0	13,76	0	5,21	5	7,90	0,07	0,02

Tabla 7.6. Resultados del 1º experimento con fugas y 2 depósitos de ventas.

Este experimento se inicia con un volumen principal de 24,78 litros y se termina con un volumen 25,36 litros en el tanque de almacenamiento. Se ha terminado el experimento con 0,58 litros más respecto del inicio del ensayo. Fijándose en la variación final, obtenida de calcular la diferencia entre las descargas y las ventas acumuladas, debería haber incrementado el volumen en 0,65 litros. En este caso, la diferencia de volumen es de 0,07 litros que, como se puede observar, coincide con el volumen final acumulado de las fugas.

A partir de esta tabla, se ha obtenido un histograma que ha servido de ayuda para concluir el estudio.

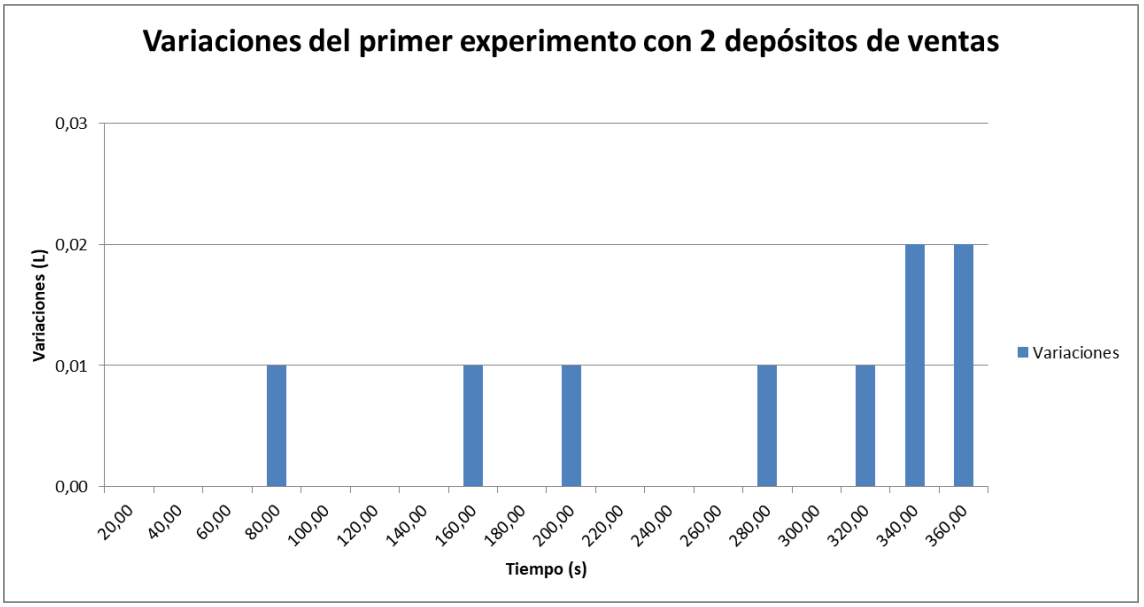


Figura 7.5. Histograma nº1 de variaciones obtenidas con 2 depósitos de ventas

En cuanto al segundo experimento, se ha intentado ser más preciso que en el anterior, por lo que se han recogido más valores que antes. Debido al tamaño de la imagen de este nuevo experimento, se ha decidido dividir la siguiente tabla en dos.

Volumen principal (L)	Descargas realizadas	Descargas acumuladas (L)	Ventas 1 realizadas	Ventas 1 acumuladas (L)	Ventas 2 realizadas	Ventas 2 acumuladas (L)	Volumen de fuga medido acumulado (L)	Variaciones (L)
25,53	0	0,00	0	0,00	0	0,00	0,00	0,00
25,53	0	0,00	0	0,00	0	0,00	0,00	0,00
25,53	0	0,00	0	0,00	0	0,00	0,01	0,00
28,29	1	2,77	0	0,00	0	0,00	0,01	0,01
27,29	0	2,77	0	0,00	1	1,00	0,01	0,00
27,29	0	2,77	0	0,00	1	1,00	0,02	0,00
27,29	0	2,77	0	0,00	1	1,00	0,02	0,00
27,28	0	2,77	0	0,00	1	1,00	0,03	0,01
27,28	0	2,77	0	0,00	1	1,00	0,03	0,00
27,26	0	2,77	0	0,00	1	1,00	0,04	0,02
27,26	0	2,77	0	0,00	1	1,00	0,04	0,00
27,26	0	2,77	0	0,00	1	1,00	0,04	0,00
27,26	0	2,77	0	0,00	1	1,00	0,05	0,00
27,25	0	2,77	0	0,00	1	1,00	0,05	0,01
27,25	0	2,77	0	0,00	1	1,00	0,06	0,00
39,28	1	14,83	0	0,00	1	1,00	0,06	0,03
32,67	0	14,83	2	2,62	6	4,99	0,06	0,00
29,67	0	14,83	1	4,62	7	5,99	0,06	0,00

Tabla 7.7.1. Resultados del 2º experimento con fugas y 2 depósitos de ventas.

23,38	0	14,83	1	6,62	10	10,27	0,07	0,01
20,88	0	14,83	1	7,12	13	12,27	0,07	0,00
20,88	0	14,83	0	7,12	13	12,27	0,07	0,00
20,88	0	14,83	0	7,12	13	12,27	0,08	0,00
20,87	0	14,83	0	7,12	13	12,27	0,08	0,01
20,87	0	14,83	0	7,12	13	12,27	0,09	0,00
20,87	0	14,83	0	7,12	13	12,27	0,09	0,00
20,87	0	14,83	0	7,12	13	12,27	0,10	0,00
20,86	0	14,83	0	7,12	13	12,27	0,10	0,01
20,86	0	14,83	0	7,12	13	12,27	0,10	0,00
20,86	0	14,83	0	7,12	13	12,27	0,11	0,00
20,86	0	14,83	0	7,12	13	12,27	0,11	0,00
30,91	1	26,88	2	9,12	13	12,27	0,12	0,00
22,26	0	26,88	2	10,62	19	19,42	0,12	0,00
17,46	0	26,88	1	12,62	23	22,21	0,12	0,01
13,46	0	26,88	1	14,62	24	24,21	0,12	0,00
13,45	0	26,88	0	14,62	24	24,21	0,13	0,01

Tabla 7.7.2. Resultados del 2º experimento con fugas y 2 depósitos de ventas. (Continuación)

En este último caso, se comienza el experimento con 25,53 litros y se termina con 13,45 litros. Ahora se ha producido un fuerte decrecimiento del volumen principal respecto del inicio de la simulación. Fijándose de nuevo en la variación final, obtenida de calcular la diferencia entre las descargas y las ventas acumuladas, se debería haber reducido el volumen 11,95 litros, es decir, haber terminado el experimento con un volumen de 13,58 litros. En este caso, la diferencia entre el volumen final medido y el volumen final teórico es de 0,13 litros que, como se puede observar, coincide con el volumen final acumulado de las fugas.

A partir de esta última tabla se obtendrá el histograma que, posteriormente, se utilizará para realizar el estudio y así poder conseguir el porcentaje de probabilidad de que se produzcan fugas.

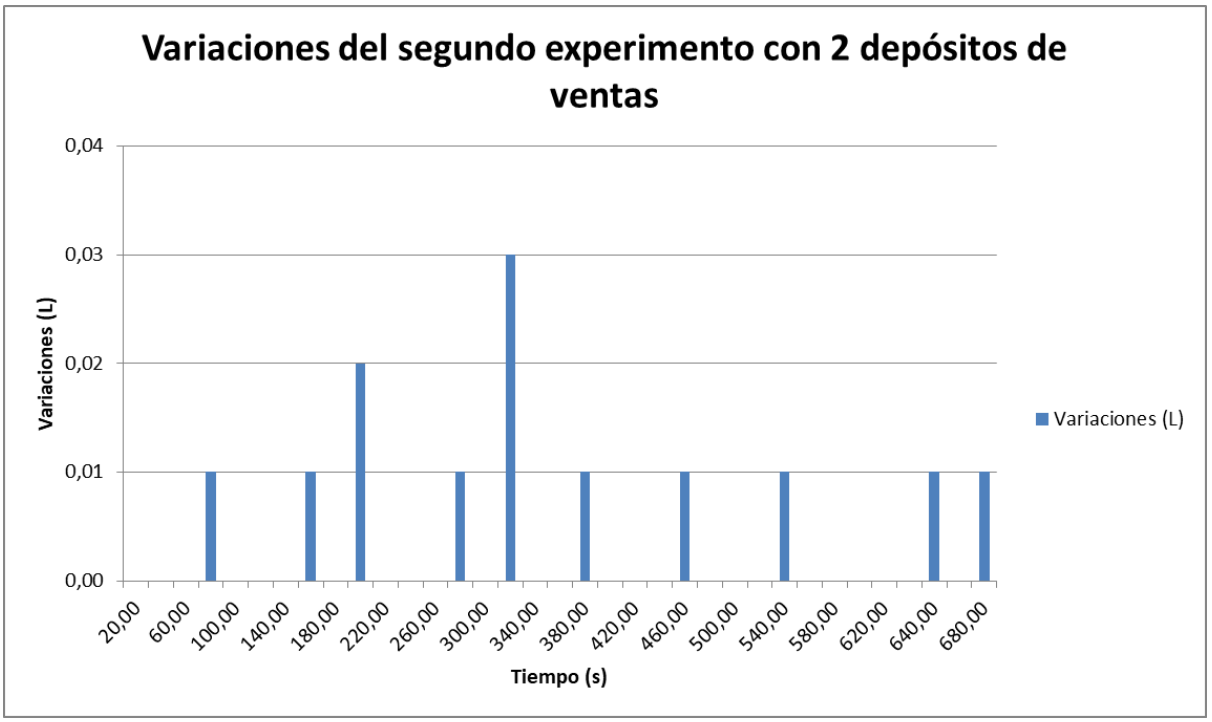


Figura 7.6. Histograma nº2 de variaciones obtenidas con 2 depósitos de ventas

8. PRESUPUESTO

Para la realización de este proyecto se han necesitado diferentes componentes. Muchos de ellos no estaban pensados en un principio, pero a medida que ha ido avanzando el proyecto y, para conseguir los resultados deseados, han sido indispensables. Por ello, el presupuesto de este proyecto ha sido el siguiente:

Orden	Artículo	Unidades	Precio unidad	Precio
1	Bomba	4	13,40 €	53,60 €
2	Electroválvula	6	24,20 €	145,20 €
3	Sensores de nivel sin contacto	4	15,99 €	63,96 €
4	Bomba peristáltica	1	32,68 €	32,68 €
5	Depósito retorno	1	49,99 €	49,99 €
6	Depósito recarga	1	10,90 €	10,90 €
7	Depósitos ventas y fugas	3	3,00 €	9,00 €
8	Básculas pequeñas	3	12,00 €	36,00 €
9	Báscula grande	1	350,00 €	350,00 €
10	Arduino MEGA	2	20,00 €	40,00 €
11	Módulo HX711	4	13,99 €	55,96 €
12	Módulos relés	1	20,00 €	20,00 €
13	Fuente de alimentación 12v	1	30,00 €	30,00 €
14	Driver Motor Peristáltica	1	5,00 €	5,00 €
15	Fuente de alimentación 5v	1	10,00 €	10,00 €
16	Material de ferretería (mangueras, acoples, etc.)	1	67,19 €	67,19 €
17	Estantería	1	59,00 €	59,00 €
18	Bases de madera	4	12,00 €	12,00 €
			TOTAL:	**1.050,48 €**

Tabla 8.1. Presupuesto del proyecto

Además de los materiales necesarios para construir el prototipo, también se ha incluido el presupuesto por mano de obra en función de las horas empleadas en el desarrollo del proyecto.

Nº horas	Precio/hora	Precio total
300	18,00 €	5.400,00 €

Tabla 8.2. Presupuesto de mano de obra

9. CONCLUSIONES

9.1. Relación de problemas planteados y procedimientos de resolución

A lo largo de estos meses han surgido diferentes problemas que se ha tenido que ir solucionando.

El primero de ellos ha tenido que ver con el montaje del prototipo físico. Se ha tenido que construir el prototipo con diferentes componentes de ferretería. A primera vista no parecía complicado llevar a cabo esta tarea ya que tan solo se tenía que ir uniendo piezas de acuerdo a los diseños realizados de los diferentes depósitos. Al ejecutar esta tarea se observó que algunos componentes no entraban fácilmente en la manguera a la que iban acoplados, por lo que se tuvo que usar un decapador, dicho de otra manera, una pistola de calor. Esta actividad no parecía difícil ya que tan solo se tenía que aplicar calor a la manguera y la pieza entraría perfectamente, pero no fue el caso. Se investigó en internet sobre cómo hacer esto, se preguntó en diferentes ferreterías si había manera de acoplar los componentes en la manguera y, en casi todas decían lo mismo, que la manguera no era compatible con los componentes que se tenían. A pesar de esto, se estuvo probando con la manguera hasta que al final se consiguió acoplar a las diferentes piezas.

Una vez montados los depósitos con sus respectivos componentes, se tuvo que realizar un ensayo de estanqueidad para cada uno de ellos. A priori, no parecía que fuesen a producirse fugas ya que se añadió bastante teflón en las uniones “macho” y, en las uniones a presión, se añadió también bastante pegamento. Cuando se le añadió agua a los depósitos, no parecía haber problema pero, una vez se agregó una mayor cantidad de agua, empezaron a aparecer las primeras fugas. La primera solución que se tomó fue desmontar de nuevo todo y volver a montarlo añadiendo más teflón y pegamento, pero seguían apareciendo fugas. Después se pasó a adjuntar abrazaderas en las uniones donde se escapaba el agua. En algunos casos se añadieron

hasta tres abrazaderas juntas. Aun así seguía escapándose algo de agua. Y ya por último, se decidió pasar a un nivel más. Se le añadió una cinta autosoldante a las uniones débiles. Esta cinta se colocaría por fuera entre los dos componentes en los que se producía la fuga. Con esto se consiguió reducir las fugas casi en su totalidad. A día de hoy sigue existiendo alguna pequeña fuga pero que no interviene en la obtención de resultados.

Otro problema que surgió fue el fallo de uno de los cables que conectan la bomba del depósito de retorno con el módulo de 16 relés. Se estuvo realizando experimentos para la obtención de resultados y, en un momento dado, la planta dejó de funcionar como lo estaba haciendo antes. Se estuvo buscando durante semanas dónde podría estar el fallo. Primero se probó a alimentar el módulo de 16 relés directamente de la fuente de 12 voltios. Se observó que el problema seguía presente. Posteriormente se pasó a comprobar el funcionamiento de cada relé por separado y, se observó que la bomba del depósito de retorno fallaba. Este error provocaba que el microcontrolador al que iba asociada la controladora, se reiniciara. Para solucionar este error se fueron cambiando de uno en uno los cables que tuviesen algo que ver con esta bomba, ya sean los cables que van del módulo de 16 relés al microcontrolador, como los cables que conectan la bomba al relé. Una vez cambiado uno de los que conecta la bomba con el módulo de relés, desapareció el problema.

9.2. Conclusiones

Este proyecto se ha centrado especialmente en diseñar un sistema de detección de fugas. Como se ha mencionado en los objetivos propuestos, se ha conseguido adaptar el diseño previo a uno a mayor escala y, además, se ha podido construir el prototipo físico con los materiales más adecuados para el diseño. Se ha conseguido adaptar la controladora aportada a este proyecto, por lo que se ha podido gestionar correctamente el funcionamiento de la planta. Además, se ha implementado un sistema de detección de fugas encargado de obtener las variaciones de volumen del tanque principal. Atendiendo a los objetivos que se pusieron en un primer momento, se puede decir que la mayoría de ellos se han cumplido.

Respecto a la validación de la controladora y del detector de fugas, no se ha podido realizar un mayor número de pruebas debido a los diferentes contratiempos que han surgido. En un primer momento se realizaron diferentes experimentos para validar ambos componentes pero, llegados a agosto, la universidad cerró y, cuando volvió a abrir a finales de mes y se volvió a probar el funcionamiento de la planta, ésta no respondía. Se estuvo buscando el fallo durante días hasta que al final resultó ser un fallo de hardware debido a un cable dañado. La resolución de este problema llevó días ya que costó encontrar el fallo. Debido a esto, no se han podido realizar más pruebas pero, se entiende la comprobación que se debería haber hecho para validar ambos componentes.

Los experimentos deberían haber sido más largos y el período de observación se debería haber ajustado más para obtener más movimientos entre cada observación. A partir de los histogramas realizados y conociendo el caudal de fuga que está sacando la bomba peristáltica en cada experimento, se debería haber planteado un umbral de detección razonable. Para comprobar el funcionamiento del detector de fugas se deberían haber hecho simulaciones más largas sin fugas para comprobar el número de veces que salta la alarma sin producirse fugas. Posteriormente, se deberían haber hecho simulaciones con fugas y se hubiese observado cuánto volumen de agua hay en el depósito de fugas cuando el detector hizo saltar la alarma. A partir de los resultados obtenidos en cada experimento se hubiese sacado una estadística del funcionamiento del detector.

Personalmente, no me imaginé que este proyecto fuera tan complejo. La construcción del prototipo físico me llevó bastante tiempo ya que iban surgiendo problemas continuamente. La validación de la controladora también conllevó bastantes dificultades ya que al incluir tantos componentes en la construcción del prototipo y, al haber trabajado con un código tan amplio, a cualquier error que apareciera, costaba mucho encontrar el error. Aun así, esto me ha servido de ayuda para aprender a enfrentarme a los diferentes problemas que pueden surgir en el futuro, ya que esto será algo que me acompañará en el día de mañana.

Conclusions

This project has focused especially on designing a leak detection system. As mentioned in the proposed objectives, it has been possible to adapt the previous design to one on a larger scale and, in addition, it has been possible to build the physical prototype with the most suitable materials for the design. It has been possible to adapt the controller provided to this project, so it has been possible to correctly manage the operation of the plant. In addition, a leak detection system has been implemented in charge of obtaining the volume variations of the main tank. Taking into account the objectives that were set at the beginning, it can be said that most of them have been met.

Regarding the validation of the controller and the leak detector, it has not been possible to carry out a greater number of tests due to the different setbacks that have arisen. Initially, different experiments were carried out to validate both components, but by August the university closed and, when it reopened at the end of the month and the operation of the plant was tested again, it did not respond. The fault was searched for days until it finally turned out to be a hardware fault due to a damaged cable. The resolution of this problem took days as it was difficult to find the fault. Due to this, no further tests could be carried out, but it is understood the check that should have been done to validate both components.

The experiments turned out to have been longer and the observation period should have been adjusted more to get more movements between each observation. From the histograms made and knowing the leak rate that the peristaltic pump is extracting in each experiment, a reasonable detection threshold should have been established. To check the operation of the leak detector, longer simulations without leaks have probably been done to check the number of times the alarm goes off without leaks. Subsequently, simulations with leaks should have been carried out and it was observed how much water there was in the leak tank when the detector triggered the alarm. From the results obtained in each experiment a statistic of the functioning of the detector had been drawn.

Personally, I did not imagine that this project would be so complex. The construction of the physical prototype took me quite a long time as problems were constantly arising. The validation of the controller also led to quite a few difficulties since by including so many components in the construction of the prototype and, having worked with such a large code, any error that appeared took a lot of time to find the error. Even so, this has helped me to learn to face the different problems that may arise in the future, since this will be something that will accompany me in the future.

10. ANEXOS

Anexo 1

Guía del desarrollo. Controladora

En este anexo se detalla la estructura del código implementado en la controladora, nombrando los diferentes archivos que se han utilizado. El código de la controladora está compuesto por 4 archivos:

- Controladora2.ino. Es el archivo principal de la controladora. Alberga la máquina de estados de la controladora.

- Controladora2.h. Archivo que contiene las constantes, variables locales y la definición de pines.

- Depósito2.cpp. Archivo que contiene los métodos de la clase “Depósito” y su máquina de estados.

- Depósito2.h. Archivo asignado a la clase “Depósito2” el cual contiene la parte pública y privada.

- DetectaFlanco2.cpp. Archivo que gestiona el los cambios de estado de los pulsadores

- DetectaFlanco2.h. Archivo asignado a la clase “DetectaFlanco2” el cual contiene la parte pública y privada.

Controladora2.ino

```
#include "Controladora2.h"

#include "Deposito2.h"

//#include <stdio.h>

//#include <string.h>

//#include <stdlib.h>

//#include <TimeLib.h>

//#define DEBUG(a) Serial.println(a);


#include <HX711_ADC.h>

#if defined(ESP8266)|| defined(ESP32) || defined(AVR)

#include <EEPROM.h>

#endif


#include "DetectaFlanco2.h"

DetectaFlanco dfEmergencia(PIN_EMERGENCIA);



void setup () {

  Serial.begin(9600);

  Serial.println("Comenzamos...");
```

```
  Serial.println("Comenzamos...");

  Serial.println("Comenzamos...");

  Serial.println("Comenzamos...");



// Asociación de pines de entradas y salidas //

  pinMode (alarma_rebosamiento, OUTPUT);

  pinMode (PIN_START, INPUT);

  pinMode (PIN_STOP, INPUT);

  pinMode (PIN_EMERGENCIA, INPUT);

  pinMode (PIN_DIAL, INPUT);


  dfEmergencia.inicio(INPUT_PULLUP);


  pinMode (PIN_Rele1_Llenado_recarga, OUTPUT);

  pinMode (PIN_Rele1_Vaciado_recarga, OUTPUT);

  pinMode (PIN_Rele1_Llenado_dispensador_1, OUTPUT);

  pinMode (PIN_Rele1_Vaciado_dispensador_1, OUTPUT);

  pinMode (PIN_Rele1_Llenado_dispensador_2, OUTPUT);

  pinMode (PIN_Rele1_Vaciado_dispensador_2, OUTPUT);

  pinMode (PIN_Rele1_Vaciado_fuga, OUTPUT);


  pinMode (PIN_ENA_fuga, OUTPUT);
```

```
  pinMode (PIN_IN1_fuga, OUTPUT);

  pinMode (PIN_IN2_fuga, OUTPUT);


  pinMode (sensor_0_recarga, INPUT);

  pinMode (sensor_1_recarga, INPUT);

  pinMode (sensor_2_recarga, INPUT);

  pinMode (sensor_3_recarga, INPUT);


// Hacemos que no conmute los reles debido a su logica inversa enviando un HIGH desde el Arduino //

// ACTIVAMOS RELE (LOW) PARA QUE LA VALVULA ESTÉ CONECTADA A TIERRA Y NO SE CALIENTE LA PLACA REGULADORA DE 4,5 V

  digitalWrite (PIN_Rele1_Llenado_recarga, HIGH);

  digitalWrite (PIN_Rele1_Vaciado_recarga, HIGH);


  digitalWrite (PIN_Rele1_Llenado_dispensador_1, HIGH);

  digitalWrite (PIN_Rele1_Llenado_dispensador_2, HIGH);


  digitalWrite (PIN_Rele1_Vaciado_dispensador_1, HIGH);

  digitalWrite (PIN_Rele1_Vaciado_dispensador_2, HIGH);


  digitalWrite (PIN_Rele1_Vaciado_fuga, HIGH);
```

```
}


//  Tanques declarados a partir de la clase deposito  //


/*

Deposito (int _PIN_Rele1_Llenado, int _PIN_Rele1_Vaciado, int _PIN_IN1, int _PIN_IN2, int
_PIN_ENA, int _HX711_dout, int _HX711_sck, float _valor_calibracion_basculas, int
_sensor_0, int _sensor_1,  int _sensor_2, int _sensor_3,

float _capacidad_maxima, float _vol_max_seguridad, float _volumen_descarga_max, float
_volumen_descarga_min, float _t_prox_descarga_max, float _t_prox_descarga_min,

bool _recarga_o_descarga, Deposito *deposito_ant, Deposito *deposito_sig, bool _es_fugas,
bool _es_recarga, float _vol_seguridad_anterior, float _vol_limite_anterior, float
_vol_min_comienzo_bomba);

*/


Deposito tanque_retorno (PIN_Rele1_Vaciado_dispensador_1, PIN_Rele1_Llenado_recarga, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,

volumen_retorno_limite, volumen_retorno_seguridad, 0, 0, 0, 0, 0, &tanque_retorno
,&tanque_retorno, 0, 0, 0, 0, 0);


Deposito tanque_almacenamiento (PIN_Rele1_Vaciado_recarga,
PIN_Rele1_Llenado_dispensador_1, 0, 0, 0, dout_almacenamiento, sck_almacenamiento,
calibracion_almacenamiento, 0, 0, 0, 0,

volumen_almacenamiento_limite, volumen_almacenamiento_seguridad, 0, 0, 0, 0, 1,
&tanque_almacenamiento, &tanque_almacenamiento, 0, 0, 0, 0, 0);
```

Deposito tanque_recarga (PIN_Rele1_Llenado_recarga, PIN_Rele1_Vaciado_recarga, 0, 0, 0, 0, 0, 0, sensor_0_recarga, sensor_1_recarga, sensor_2_recarga, sensor_3_recarga, volumen_recarga_limite, volumen_recarga_seguridad,

volumen_recarga_max, volumen_recarga_min, t_prox_recarga_max, t_prox_recarga_min, 1, &tanque_retorno, &tanque_almacenamiento, 0, 1, 0, 0, 0);

Deposito dispensador_1 (PIN_Rele1_Llenado_dispensador_1, PIN_Rele1_Vaciado_dispensador_1, 0, 0, 0, dout_dispensador_1, sck_dispensador_1, calibracion_dispensador_1, 0, 0, 0, 0, volumen_ventas_limite,

volumen_ventas_seguridad, volumen_ventas_max, volumen_ventas_min, t_prox_venta_max, t_prox_venta_min, 0, &tanque_almacenamiento, &tanque_retorno, 0, 0, volumen_almacenamiento_seguridad, volumen_almacenamiento_limite, vol_min_alm_comienzo_bomba);

Deposito dispensador_2 (PIN_Rele1_Llenado_dispensador_2, PIN_Rele1_Vaciado_dispensador_2, 0, 0, 0, dout_dispensador_2, sck_dispensador_2, calibracion_dispensador_2, 0, 0, 0, 0, volumen_ventas_limite,

volumen_ventas_seguridad, volumen_ventas_max, volumen_ventas_min, t_prox_venta_max, t_prox_venta_min, 0, &tanque_almacenamiento, &tanque_retorno, 0, 0, volumen_almacenamiento_seguridad, volumen_almacenamiento_limite, vol_min_alm_comienzo_bomba);

Deposito tanque_fugas (0, PIN_Rele1_Vaciado_fuga, PIN_IN1_fuga, PIN_IN2_fuga,

PIN_ENA_fuga, dout_fugas, sck_fugas, calibracion_fugas, 0, 0, 0, 0, volumen_fugas_limite, volumen_fugas_seguridad, 0, 0, 0, 0, 1, &tanque_almacenamiento, &tanque_almacenamiento, 1, 0, 0, 0, 0);

```
//
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------

/*

Deposito::Deposito (int _PIN_bomba_llenado, int _PIN_bomba_vaciado, int
_PIN_Rele1_Llenado, int _PIN_Rele2_Llenado, int _PIN_Rele1_Vaciado, int
_PIN_Rele2_Vaciado, int _PIN_IN1, int _PIN_IN2, int _PIN_ENA, int _HX711_dout,

int _HX711_sck, float _valor_calibracion_basculas, int _sensor_0, int _sensor_1,  int _sensor_2,
int _sensor_3, float _capacidad_maxima, float _vol_max_seguridad, float
_volumen_descarga_max, float _volumen_descarga_min,

float _t_prox_descarga_max, float _t_prox_descarga_min, bool _recarga_o_descarga, Deposito
*deposito_ant, Deposito *deposito_sig, bool _es_fugas, bool _es_recarga, float
_vol_seguridad_anterior, float _vol_limite_anterior, float _vol_min_comienzo_bomba) :
LoadCell(_HX711_dout, _HX711_sck) {

*/


/*Deposito tanque_retorno (PIN_bomba_vaciado_dispensador_1, PIN_bomba_llenado_recarga,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

volumen_retorno_limite, volumen_retorno_seguridad, 0, 0, 0, 0, 0, &tanque_retorno
,&tanque_retorno, 0, 1, 0, 0, 0);


Deposito tanque_almacenamiento (222, 222, 0, 0, 0, 0, 0, 0, 0, dout_almacenamiento,
sck_almacenamiento, calibracion_almacenamiento, 0, 0, 0, 0,

volumen_almacenamiento_limite, volumen_almacenamiento_seguridad, 0, 0, 0, 0, 1,
&tanque_almacenamiento, &tanque_almacenamiento, 0, 0, 0, 0, 0);
```

Deposito tanque_recarga (PIN_bomba_llenado_recarga, PIN_bomba_vaciado_recarga, 0, 0, PIN_Rele1_Vaciado_recarga,

PIN_Rele2_Vaciado_recarga, 0, 0, 0, 0, 0, 0, sensor_0_recarga, sensor_1_recarga, sensor_2_recarga, sensor_3_recarga, volumen_recarga_limite, volumen_recarga_seguridad,

volumen_recarga_max, volumen_recarga_min, t_prox_recarga_max, t_prox_recarga_min, 1, &tanque_retorno, &tanque_almacenamiento, 0, 1, 0, 0, 0);

Deposito dispensador_1 (PIN_bomba_llenado_dispensador_1, PIN_bomba_vaciado_dispensador_1, PIN_Rele1_Llenado_dispensador_1, PIN_Rele2_Llenado_dispensador_1, PIN_Rele1_Vaciado_dispensador_1,

PIN_Rele2_Vaciado_dispensador_1, 0, 0, 0, dout_dispensador_1, sck_dispensador_1, calibracion_dispensador_1, 0, 0, 0, 0, volumen_ventas_limite,

volumen_ventas_seguridad, volumen_ventas_max, volumen_ventas_min, t_prox_venta_max, t_prox_venta_min, 0, &tanque_almacenamiento,&tanque_retorno, 0, 0, volumen_almacenamiento_seguridad, volumen_almacenamiento_limite, vol_min_alm_comienzo_bomba);

Deposito tanque_fugas (0, PIN_bomba_vaciado_fugas, 0, 0, PIN_Rele1_Vaciado_fuga, PIN_Rele2_Vaciado_fuga, PIN_IN1_fuga, PIN_IN2_fuga,

PIN_ENA_fuga, dout_fugas, sck_fugas, calibracion_fugas, 0, 0, 0, 0, volumen_fugas_limite, volumen_fugas_seguridad, 0, 0, 0, 0, 1, &tanque_almacenamiento, &tanque_retorno, 1, 0, 0, 0, 0);

*/

//----------

```
//  Bucle principal donde se irán recorriendo los valores de los sensores y se ejecutaran las llamadas a las funciones pertinentes  //

void loop () {

  start = digitalRead (PIN_START);

  stop_ = digitalRead (PIN_STOP);

  seta_emergencia = digitalRead (PIN_EMERGENCIA);

  dial = digitalRead (PIN_DIAL);


  int flancoEmergencia = dfEmergencia.comprueba();



//  Si la seta_emergencia está presionada al iniciar arduino, se queda a la espera de iniciar el CICLO que deja los limites inferiores de los depositos

  if ((seta_emergencia == HIGH) && (estado_actual == COMPROBAR_CICLO_INICIAL)) {

    estado_actual = CICLO_FONDO_TANQUES;

    if (TIPO_DEPURACION == 5){

      Serial.println( "Estado: CICLO_FONDO_TANQUES" );

    }

  }


//  Al dejar de prersionar la seta_emergencia y producirse un flanco de bajada, se inicia el CICLO. Abrimos todas las valvulas  //
```

```
// Comenzamos llenando el tanque de recarga y a su vez el tanque de almacenamiento, LLENANDO_RECARGA_ALMACEN //

 if ((flancoEmergencia == -1) && (estado_actual == CICLO_FONDO_TANQUES)) {

   digitalWrite (PIN_Rele1_Llenado_recarga, LOW);      // Los Relés tienen logica inversa con Arduino, al mandar un LOW = Relé conmuta/activado

   digitalWrite (PIN_Rele1_Vaciado_recarga, LOW);

   t_inicio_valvulas_abiertas = millis();              // Tomamos el tiempo de inicio para establecer el tiempo de CONMUTADO de las valvulas, en este caso abiertas

   t_inicio_tanque_recarga_almacen = millis();         // Tomamos el tiempo de inicio del estado: LLENANDO_RECARGA_ALMACEN

   estado_actual = LLENANDO_RECARGA_ALMACEN;

 }



// Se está llenando el tanque de recarga y a su vez el tanque de almacenamiento //

 if (estado_actual == LLENANDO_RECARGA_ALMACEN) {

   if (TIPO_DEPURACION == 5){

    Serial.println( "Estado: LLENANDO_RECARGA_ALMACEN" );      // Mensaje utilizado para depurar errores mediante el Monitor Serial

   }

 // Una vez pasado el tiempo definido en T_PARTE_RECARGA_ALMACEN dejamos de llenar el tanque de recarga y comenzamos a llenar el tanque de fugas //

 // Comenzamos el siguiente estado: LLENANDO_FUGA //

   if (millis() > (T_PARTE_RECARGA_ALMACEN + t_inicio_tanque_recarga_almacen)) {

    digitalWrite (PIN_Rele1_Llenado_recarga, HIGH);
```

```
      digitalWrite (PIN_IN1_fuga, HIGH);

      digitalWrite (PIN_IN2_fuga, LOW);

      analogWrite (PIN_ENA_fuga, 255);

      t_inicio_tanque_fuga = millis();                //  Tomamos el tiempo de inicio del estado: LLENANDO_FUGA

      estado_actual = LLENANDO_FUGA;

    }

  }


//  Se está llenando el tanque de almacenamiento y a su vez el tanque de fugas //

  if (estado_actual == LLENANDO_FUGA) {

    if (TIPO_DEPURACION == 5){

      Serial.println( "Estado: LLENANDO_FUGA" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial

    }

  //  Una vez pasado el tiempo definido en T_PARTE_FUGAS dejamos de llenar el tanque de almacenamiento y el tanque de fugas  //

  //  Comenzamos el siguiente estado: LLENANDO_DEPOSITO_1y2, llenamos el deposito de ventas 1 y 2 (vaciamos el tanque de almacenamiento) a la vez que vaciamos el mismo //

    if (millis() > (T_PARTE_FUGAS + t_inicio_tanque_fuga)) {

      digitalWrite (PIN_IN1_fuga, LOW);

      digitalWrite (PIN_IN2_fuga, LOW);

      analogWrite (PIN_ENA_fuga, 0);

      digitalWrite (PIN_Rele1_Vaciado_recarga, HIGH);
```

```
      digitalWrite (PIN_Rele1_Vaciado_dispensador_1, LOW);

      digitalWrite (PIN_Rele1_Llenado_dispensador_1, LOW);

      digitalWrite (PIN_Rele1_Vaciado_dispensador_2, LOW);

      digitalWrite (PIN_Rele1_Llenado_dispensador_2, LOW);

      t_inicio_tanque_deposito_1y2 = millis();           //  Tomamos el tiempo de inicio del estado: LLENANDO_DEPOSITO_1

      estado_actual = LLENANDO_DEPOSITO_1y2;

    }

  }


//  Se está llenando (vaciando tanque de almacenamiento)  y vaciando el deposito de ventas 1  //

  if (estado_actual == LLENANDO_DEPOSITO_1y2) {

    if (TIPO_DEPURACION == 5){

      Serial.println( "Estado: LLENANDO_DEPOSITO_1y2" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial

    }

  //  Una vez pasado el tiempo definido en T_PARTE_DEPOSITO_1y2 dejamos de llenar el deposito de ventas 1 y 2, y seguimos vaciando dicho deposito  //

  //  Comenzamos el siguiente estado: COLOCANDO_FONDOS_FINALES, vaciamos tambien el tanque de fugas //

    if (millis() > (T_PARTE_DEPOSITO_1y2 + t_inicio_tanque_deposito_1y2)) {

      digitalWrite (PIN_Rele1_Llenado_dispensador_1, HIGH);

      digitalWrite (PIN_Rele1_Llenado_dispensador_2, HIGH);

      digitalWrite (PIN_Rele1_Vaciado_fuga, LOW);
```

```
    t_inicio_fondos_finales = millis();              //  Tomamos el tiempo de inicio del estado: COLOCANDO_FONDOS_FINALES

    estado_actual = COLOCANDO_FONDOS_FINALES;

   }

 }


//  Se están vaciando el deposito de ventas 1 y el deposito de fugas //

 if (estado_actual == COLOCANDO_FONDOS_FINALES) {

   if (TIPO_DEPURACION == 5){

    Serial.println( "Estado: COLOCANDO_FONDOS_FINALES" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial

   }

 //  Una vez pasado el tiempo definido en T_PARTE_FONDO_FINAL dejamos de vaciar el deposito de ventas 1 y tanque de fugas  //      //  PREGUNTAR!!!! ///

 //  Cerramos todas las valvulas. Comenzamos el siguiente estado: HACER_TARA  //

   if (millis() > (T_PARTE_FONDO_FINAL + t_inicio_fondos_finales)) {

    digitalWrite (PIN_Rele1_Vaciado_dispensador_1, HIGH);

    digitalWrite (PIN_Rele1_Vaciado_dispensador_2, HIGH);

    digitalWrite (PIN_Rele1_Vaciado_fuga, HIGH);

    t_inicio_valvulas_cerradas = millis();           //  Tomamos el tiempo de inicio para establecer el tiempo de CONMUTADO de las valvulas, en este caso cerradas


    estado_actual = HACER_TARA;

   }
```

```
  }


// Si la seta_emergencia no está presionada al iniciar arduino, se supone que ya tenemos realizados los fondos de los depositos de la planta //

// Cerramos todas las valvulas. Comenzamos el siguiente estado: HACER_TARA //

  if ((seta_emergencia == LOW) && (estado_actual == COMPROBAR_CICLO_INICIAL)) {

    estado_actual = HACER_TARA;

    t_inicio_valvulas_cerradas = millis();              // Tomamos el tiempo de inicio para establecer el tiempo de CONMUTADO de las valvulas, en este caso cerradas


  }



// Realizamos la tara de los depositos con el fondo ya establecido //

  if (estado_actual == HACER_TARA) {

    if (TIPO_DEPURACION == 5){

     Serial.println( "Estado: HACER_TARAS" );      // Mensaje utilizado para depurar errores mediante el Monitor Serial

    }

    bool tara_almacenam = tanque_almacenamiento.realizar_tara();

    bool tara_disp_1 = dispensador_1.realizar_tara();

    bool tara_disp_2 = dispensador_2.realizar_tara();

    bool tara_fugas = tanque_fugas.realizar_tara();

  // Una vez realizadas las taras de los depositos, entramos en funcionamiento normal //
```

```
    if ((tara_disp_1) && (tara_disp_2) && (tara_almacenam) && (tara_fugas) == true) {
      estado_actual = LLENAR_VOLUMEN_RECARGA_1;
      estado_taras = HECHO;
      Serial.println(" TARAS HECHAS");
    }
  }



// Si se produce un flanco de subida en la seta_emergencia entramos en el estado de EMERGENCIA //
  if (flancoEmergencia == 1) {
    estado_actual = EMERGENCIA;
  }


// Tomamos el volumen de los depositos //
  if (estado_taras == HECHO) {
    tanque_almacenamiento.get_volumen();
    dispensador_1.get_volumen();
    dispensador_2.get_volumen();
    tanque_recarga.get_volumen();
    tanque_fugas.get_volumen();
  }
```

```
//************************************  FUNCIONAMIENTO NORMAL
**********************************************************************************
***********

  if (estado_actual == FUNC_NORMAL) {

    if (TIPO_DEPURACION == 5){

 //     Serial.println( "Estado: FUNCIONAMIENTO_NORMAL" );      //  Mensaje utilizado para
depurar errores mediante el Monitor Serial

    }

    if (stop_ == HIGH) {

      estado_actual = PAUSA;

    }

    estado_recarga = tanque_recarga.actualizar_estado(tSimulacion, seta_emergencia, 0);

    estado_disp_1 = dispensador_1.actualizar_estado(tSimulacion, seta_emergencia, 0);

    estado_disp_2 = dispensador_2.actualizar_estado(tSimulacion, seta_emergencia, 0);

    estado_fugas = tanque_fugas.actualizar_estado(tSimulacion, seta_emergencia, 0);


    if ((estado_recarga == true) || (estado_disp_1 == true) || (estado_disp_2 == true) ||
(estado_fugas == true)) {

      estado_actual = EMERGENCIA;

    }

  }
```

```
//*****************************************  PAUSA
*****************************************************************************
************

  if (estado_actual == PAUSA) {

    if (TIPO_DEPURACION == 5){

 //     Serial.println( "Estado: PAUSA" );      //  Mensaje utilizado para depurar errores mediante
el Monitor Serial

    }

    tanque_recarga.cerrar_valvula_llenado();

    tanque_recarga.cerrar_valvula_vaciado();

    dispensador_1.cerrar_valvula_llenado();

    dispensador_1.cerrar_valvula_vaciado();

    dispensador_2.cerrar_valvula_llenado();

    dispensador_2.cerrar_valvula_vaciado();

    if ((stop_ == LOW) && (seta_emergencia == LOW) && (start == HIGH)) {

      estado_actual = FUNC_NORMAL;

    }

  }



//*****************************************  EMERGENCIA
*****************************************************************************
************

  if (estado_actual == EMERGENCIA) {

    if (TIPO_DEPURACION == 5){
```

```
//    Serial.println( "Estado: EMERGENCIA" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
   }
   digitalWrite (alarma_rebosamiento, HIGH);
   tanque_recarga.actualizar_estado(tSimulacion, HIGH, 0);
   dispensador_1.actualizar_estado(tSimulacion, HIGH, 0);
   dispensador_2.actualizar_estado(tSimulacion, HIGH, 0);
   tanque_fugas.actualizar_estado(tSimulacion, HIGH, 0);
   if ((seta_emergencia == LOW) && (start == HIGH)) {
     estado_actual = LLEVAR_COND_INIC;
   }
 }


//************************************  LLEVAR COND INICIALES ********************************************************************************************
  if (estado_actual == LLEVAR_COND_INIC) {
   if (TIPO_DEPURACION == 5){
//    Serial.println( "Estado: LLEVAR_A_COND_INICIALES" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
   }
   digitalWrite (alarma_rebosamiento, LOW);
   tanque_recarga.actualizar_estado(tSimulacion, seta_emergencia, 0);
```

```
    tanque_recarga.cerrar_valvula_llenado();

    dispensador_1.actualizar_estado(tSimulacion, seta_emergencia, 0);

    dispensador_2.actualizar_estado(tSimulacion, seta_emergencia, 0);

    tanque_fugas.actualizar_estado(tSimulacion, seta_emergencia, 0);

    if (((tanque_recarga.get_volumen() <= UMBRAL_VACIO) &&
(tanque_almacenamiento.get_volumen() <= UMBRAL_VACIO) &&
(dispensador_1.get_volumen() <= UMBRAL_VACIO) && (dispensador_2.get_volumen() <=
UMBRAL_VACIO) &&

      (dial == 0)) || ((dial == 1) && (start == HIGH))) {

      tanque_recarga.actualizar_estado(tSimulacion, seta_emergencia, 1);

      dispensador_1.actualizar_estado(tSimulacion, seta_emergencia, 1);

      dispensador_2.actualizar_estado(tSimulacion, seta_emergencia, 1);

      tanque_fugas.actualizar_estado(tSimulacion, seta_emergencia, 1);

      estado_actual = FUNC_NORMAL;

    }

  }


//  Toma de valores de volumen de los depositos de manera periódica //

  if ( (millis() > (T_INTERVALO_DATALOGGER + t_datalogger)) && (estado_actual ==
FUNC_NORMAL) ) {

    String str = "";

    tanque_recarga.solicitud_pausa_recarga();

    dispensador_1.solicitud_pausa_venta();

    dispensador_2.solicitud_pausa_venta();
```

```
   tanque_fugas.solicitud_pausa_fuga();

   if ( (tanque_recarga.solicitud_pausa_recarga() && dispensador_1.solicitud_pausa_venta()
&& dispensador_2.solicitud_pausa_venta() && tanque_fugas.solicitud_pausa_fuga() ) == 1) {

     // # tiempo, volumen principal, volumen total_descarga, volumen total_ventas, volumen
fugas_ac

   //Serial.print("# ");



     Serial.print( tanque_almacenamiento.get_volumen() );

     Serial.print(" , ");

     Serial.print( tanque_recarga.recargas_realizadas() );

     Serial.print(" , ");

     Serial.print( tanque_recarga.volumen_acumulado_recarga() );

     Serial.print(" , ");

     Serial.print( dispensador_1.ventas_realizadas() );

     Serial.print(" , ");

     Serial.print( dispensador_1.volumen_acumulado_venta() );

     Serial.print(" , ");

     Serial.print( dispensador_2.ventas_realizadas() );

     Serial.print(" , ");

     Serial.print( dispensador_2.volumen_acumulado_venta() );

     Serial.print(" , ");

     Serial.println( tanque_fugas.get_volumen() );
```

```
/*
  str.concat( tanque_almacenamiento.get_volumen() );
  str.concat(",");
  str.concat( tanque_recarga.recargas_realizadas() );
  str.concat(",");
  str.concat( tanque_recarga.volumen_acumulado_recarga() );
  str.concat(",");
  str.concat( dispensador_1.ventas_realizadas() );
  str.concat(",");
  str.concat( dispensador_1.volumen_acumulado_venta() );
  str.concat(",");
  str.concat( dispensador_2.ventas_realizadas() );
  str.concat(",");
  str.concat( dispensador_2.volumen_acumulado_venta() );
  str.concat(",");
  str.concat( tanque_fugas.get_volumen() );
  Serial.println(str);
*/

  t_datalogger = millis();
  tanque_recarga.reset_recargas_realizadas();
  dispensador_1.reset_ventas_realizadas();
  dispensador_2.reset_ventas_realizadas();
```

```
    tanque_recarga.quitar_pausa();
    dispensador_1.quitar_pausa();
    dispensador_2.quitar_pausa();
    tanque_fugas.quitar_pausa();
  }
 }
```

```
            //  PASOS DESPUES DE HABER REALIZADO LA TARA Y ANTES DE ENTRAR EN FUNC. NORMAL  //

/**************************************************************************************************/

/*************************************     LLLENAR Y MEDIR VOLUMEN SENSOR 1    ************************************************/

/**************************************************************************************************/

 if (estado_actual == LLENAR_VOLUMEN_RECARGA_1) {
  tanque_recarga.llenar_recarga_1();
  if (TIPO_DEPURACION == 5){
    Serial.println( "Estado: LLENAR_VOLUMEN_RECARGA_1" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
  }
```

```
  if ( tanque_recarga.llenar_recarga_1() == 1 ) {
    tanque_recarga.cerrar_valvula_llenado();
    estado_actual = VACIAR_VOLUMEN_RECARGA_1;
  }
 }

 if (estado_actual == VACIAR_VOLUMEN_RECARGA_1) {
  tanque_recarga.vaciar_recarga_1();
     if (TIPO_DEPURACION == 5){
   Serial.println( "Estado: VACIAR_VOLUMEN_RECARGA_1" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
  }
  if (  tanque_recarga.vaciar_recarga_1() == 1 ) {
    tanque_recarga.cerrar_valvula_vaciado();
    estado_actual = MEDIR_VOLUMEN_RECARGA_1;
  }
 }

 if (estado_actual == MEDIR_VOLUMEN_RECARGA_1) {
  tanque_recarga.medir_recarga_1();
  Serial.println( "Estado: MEDIR_VOLUMEN_RECARGA_1" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
  estado_actual = VACIAR_ALMACENAMINETO_1;
```

```
 }


 if (estado_actual == VACIAR_ALMACENAMINETO_1) {

   dispensador_1.vaciar_almacemiento();

      if (TIPO_DEPURACION == 5){

    Serial.println( "Estado: VACIAR_ALMACENAMINETO_1" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial

   }

    if ( dispensador_1.vaciar_almacemiento() == 1 ) {

      dispensador_1.cerrar_valvula_llenado();

      dispensador_1.cerrar_valvula_vaciado();

      estado_actual = LLENAR_VOLUMEN_RECARGA_2;

   }

 }



/*************************************************************************
**************************************************/

/*************************************     LLLENAR Y MEDIR VOLUMEN SENSOR 2    **********************************************/

/*************************************************************************
**************************************************/


 if (estado_actual == LLENAR_VOLUMEN_RECARGA_2) {
```

```
    dispensador_1.resetar_vaciar_almacemiento();
    dispensador_1.ciclos_cero();
    tanque_recarga.llenar_recarga_2();
    if (TIPO_DEPURACION == 5){
      Serial.println( "Estado: LLENAR_VOLUMEN_RECARGA_2" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
    }
    if ( tanque_recarga.llenar_recarga_2() == 1 ) {
      tanque_recarga.cerrar_valvula_llenado();
      estado_actual = VACIAR_VOLUMEN_RECARGA_2;
    }
  }

  if (estado_actual == VACIAR_VOLUMEN_RECARGA_2) {
    tanque_recarga.vaciar_recarga_2();
       if (TIPO_DEPURACION == 5){
      Serial.println( "Estado: VACIAR_VOLUMEN_RECARGA_2" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
    }
    if (  tanque_recarga.vaciar_recarga_2() == 1 ) {
      tanque_recarga.cerrar_valvula_vaciado();
      estado_actual = MEDIR_VOLUMEN_RECARGA_2;
    }
```

```
}

if (estado_actual == MEDIR_VOLUMEN_RECARGA_2) {
  tanque_recarga.medir_recarga_2();
  Serial.println( "Estado: MEDIR_VOLUMEN_RECARGA_2" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
  estado_actual = VACIAR_ALMACENAMINETO_2;
}

if (estado_actual == VACIAR_ALMACENAMINETO_2) {
  dispensador_1.vaciar_almacemiento();
     if (TIPO_DEPURACION == 5){
   Serial.println( "Estado: VACIAR_ALMACENAMINETO_2" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
  }
   if ( dispensador_1.vaciar_almacemiento() == 1 ) {
     dispensador_1.cerrar_valvula_llenado();
     dispensador_1.cerrar_valvula_vaciado();
     estado_actual = LLENAR_VOLUMEN_RECARGA_3;
  }
}
```

```
/*************************************************************************
**************************************************/

/************************************     LLLENAR Y MEDIR VOLUMEN
SENSOR 3    ***********************************************/

/*************************************************************************
**************************************************/


 if (estado_actual == LLENAR_VOLUMEN_RECARGA_3) {

   dispensador_1.resetar_vaciar_almacemiento();

   dispensador_1.ciclos_cero();

   tanque_recarga.llenar_recarga_3();

   if (TIPO_DEPURACION == 5){

    Serial.println( "Estado: LLENAR_VOLUMEN_RECARGA_3" );      //  Mensaje utilizado
para depurar errores mediante el Monitor Serial

   }

   if ( tanque_recarga.llenar_recarga_3() == 1 ) {

    tanque_recarga.cerrar_valvula_llenado();

    estado_actual = VACIAR_VOLUMEN_RECARGA_3;

   }

 }


 if (estado_actual == VACIAR_VOLUMEN_RECARGA_3) {

   tanque_recarga.vaciar_recarga_3();
```

```
      if (TIPO_DEPURACION == 5){
    Serial.println( "Estado: VACIAR_VOLUMEN_RECARGA_3" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
   }
   if (  tanque_recarga.vaciar_recarga_3() == 1 ) {
    tanque_recarga.cerrar_valvula_vaciado();
    estado_actual = MEDIR_VOLUMEN_RECARGA_3;
   }
 }

 if (estado_actual == MEDIR_VOLUMEN_RECARGA_3) {
   tanque_recarga.medir_recarga_3();
   Serial.println( "Estado: MEDIR_VOLUMEN_RECARGA_3" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
   estado_actual = VACIAR_ALMACENAMINETO_3;
 }

 if (estado_actual == VACIAR_ALMACENAMINETO_3) {
   dispensador_1.vaciar_almacemiento();
      if (TIPO_DEPURACION == 5){
    Serial.println( "Estado: VACIAR_ALMACENAMINETO_3" );      //  Mensaje utilizado para depurar errores mediante el Monitor Serial
   }
    if ( dispensador_1.vaciar_almacemiento() == 1 ) {
```

```
      dispensador_1.cerrar_valvula_llenado();

      dispensador_1.cerrar_valvula_vaciado();

      estado_actual = CONDICIONES_FUNC_NORMAL;

   }

 }
```

```
/*************************************************************************
**************************************************/

/*************************************  CONDICIONES PARA ENTRAR EN
FUNC NORMAL  ***********************************************/

/*************************************************************************
**************************************************/


 if (estado_actual == CONDICIONES_FUNC_NORMAL) {

   dispensador_1.resetar_vaciar_almacemiento();

   Serial.println( "Estado: RESETEAR_VACIAR_ALMACENAMINETO_" );     // Mensaje
utilizado para depurar errores mediante el Monitor Serial

   dispensador_1.ciclos_cero();

   Serial.println( "Estado: CICLOS_CERO" );     // Mensaje utilizado para depurar errores
mediante el Monitor Serial

   tanque_recarga.nuevo_vol_descarga ();

   Serial.println( "Estado: NUEVO_VOL_DESCARGA" );     // Mensaje utilizado para depurar
errores mediante el Monitor Serial

   estado_actual = FUNC_NORMAL;
```

```
 }

/*****************************************************************************
***************************************************/

/************************************** SERIAL
************************************************/

/*****************************************************************************
***************************************************/

  if (TIPO_DEPURACION == 2){              // EDUARDDO 2

    Serial.print( "Volumen: " );     // Mensaje utilizado para depurar errores mediante el
Monitor Serial

    Serial.println( tanque_recarga.serial_volumen() );     // Mensaje utilizado para depurar
errores mediante el Monitor Serial

    Serial.print( "Volumen Proximo Mov: " );     // Mensaje utilizado para depurar errores
mediante el Monitor Serial

    Serial.println( tanque_recarga.serial_prox_vol()  );     // Mensaje utilizado para depurar
errores mediante el Monitor Serial

  }
```

```
//  Incrementamos tiempos de simulación //

  tSimulacion = millis();

  tSimulacion = tSimulacion/1000;



}
```

Controladora2.h

```
//  DECLARACION DE VARIABLES  //


//  Espacio para detectar y depurar errores del codigo mediante el Monitor Serial //

#define TIPO_DEPURACION 5

// 1 = Volumen Tanque Recarga

// 2 = Volumen Tanque Almacenamiento

// 3 = Volumen Dispensador ventas 1

// 4 = Volumen Tanque de Fugas

// 5 = Estado Actual



//  Tiempos utilizados  //

long t_inicio_tanque_recarga_almacen = 0;

#define T_PARTE_RECARGA_ALMACEN 20000                    //  Tiempo utilizado para
llenar tanque de almacenamiento  /  Eduardo = 10000


long t_inicio_tanque_fuga = 0;

#define T_PARTE_FUGAS 10000                              //  Tiempo utilizado para llenar deposito
de fugas y vaciar t. recargas / Eduardo = 10000
```

```
long t_inicio_tanque_deposito_1y2 = 0;

#define T_PARTE_DEPOSITO_1y2 40000   //  Tiempo utilizado para llenar deposito de ventas 1 y vaciar t. almacenamiento  / Eduardo = 400000


long t_inicio_tanque_deposito_2 = 0;

#define T_PARTE_DEPOSITO_2 40000                    //  Tiempo utilizado para llenar deposito de ventas 1 y vaciar t. almacenamiento  / Eduardo = 400000


long t_inicio_fondos_finales = 0;

#define T_PARTE_FONDO_FINAL 30000                   //  Tiempo utilizado para terminar de vaciar los depositos   /  Eduardo = 30000


long t_inicio_valvulas_abiertas = 0;

long t_inicio_valvulas_cerradas = 0;

#define T_VALVULAS_CONMUTADAS 2000                  //  Tiempo para conmutar las valvulas, tanto para abrir como cerrar


long t_datalogger = 0;                              //  Tiempo en el que se realizó el Datalogger

#define T_INTERVALO_DATALOGGER 20000                //  Tiempo periodico para realizar el Datalogger    AJUSTAR MAS EL TIEMPO, ES DECIR, PONER UN VALOR MÁS PEQUEÑO


//  Variables asociadas a las entradas y salidas analógicas del arduino, encargadas de las lectura de los sensores  //
```

```
#define sensor_0_recarga 8                    //  Sensor optico de nivel situado mas abajo en
el tanque de recarga

#define sensor_1_recarga 9                    //  Sensor optico de nivel situado por encima
del sensor anterior (500)g

#define sensor_2_recarga 10                   //  Sensor optico de nivel situado por encima
del sensor anterior (1000)g

#define sensor_3_recarga 11                   //  Sensor optico de nivel situado mas arriba
en el tanque de recarga (1500)g


#define dout_almacenamiento 27                //  Sensor de volumen (báscula) asociado
al tanque de almacenamiento

#define sck_almacenamiento 26

float calibracion_almacenamiento = 10600;     //  Valor de calibracion asosiado a la
bascula del tanque de almacenamiento


#define dout_dispensador_1 35                 //  Sensor de volumen (báscula) asociado al
dispensador 1

#define sck_dispensador_1 34

float calibracion_dispensador_1 = 303401.4;   //  Valor de calibracion asosiado a la
bascula del dispensador 1


#define dout_dispensador_2 36                 //  Sensor de volumen (báscula) asociado al
dispensador 1

#define sck_dispensador_2 37

float calibracion_dispensador_2 = 291262.75;  //  Valor de calibracion asosiado a la
bascula del dispensador 1
```

```
#define dout_fugas 39                    //  Sensor de volumen (báscula) asociado al
tanque de fugas

#define sck_fugas 38

float calibracion_fugas = 302908.4;              //  Valor de calibracion asosiado a la
bascula del tanque de fugas


//  Variables asociadas a las salidas digitales del arduino para controlar los reles utilizados en el
cambio de sentido del voltaje en la valvulas  //


#define PIN_Rele1_Vaciado_recarga 28


#define PIN_Rele1_Llenado_dispensador_1 29


#define PIN_Rele1_Llenado_dispensador_2 30


#define PIN_Rele1_Vaciado_fuga 50


#define PIN_Rele1_Vaciado_dispensador_1 48


#define PIN_Rele1_Vaciado_dispensador_2 49


#define PIN_Rele1_Llenado_recarga 52
```

```
//  Variables asociadas a las salidas del arduino y las PINes de apertura y cierre de bombas, así como las señales de emergencia  //

                                                    //  Señal que activa la bomba de vaciado del tanque de fugas

#define PIN_ENA_fuga 7                              //  Señal analógica que controla la velocidad del caudal que se está fugando

#define PIN_IN1_fuga 44                             //  Señal que controla el sentido de la bomba peristaltica del tanque de fugas

#define PIN_IN2_fuga 45                             //  Señal que controla el sentido de la bomba peristaltica del tanque de fugas


//  Variables que definen los límites de los depositos  //


float volumen_recarga_limite = 33;                  //  Valor máximo de volumen del tanque de recarga

float volumen_recarga_seguridad = 30;               //  Valor de seguridad de volumen del tanque de recarga


float volumen_almacenamiento_seguridad = 100;       //  Valor de seguridad de volumen del tanque de almacenamiento

float volumen_almacenamiento_limite = 125;          //  Valor máximo de volumen del tanque de almacenamiento

float vol_min_alm_comienzo_bomba = 2;               //  Valor de volumen minimo en t. almacenamiento para que comience a rotar la bomba


float volumen_ventas_limite = 2.5;                  //  Valor máximo de volumen de los dispensadores
```

```
float volumen_ventas_seguridad = 2.3;                    //  Valor de seguridad de volumen de los dispensadores


float volumen_fugas_limite = 2.5;                        //  Valor máximo de volumen del tanque de fugas

float volumen_fugas_seguridad = 2.3;                     //  Valor de seguridad de volumen del tanque de fugas


float volumen_retorno_limite = 150;

float volumen_retorno_seguridad = 125;



//  Variables que definen los límites de cada una de las distribuciones aleatorias que se usarán a lo largo del código  //


#define volumen_recarga_min 1                            //  Valor mínimo del volumen de descarga para su distribución aleatoria  // EDUARDO 10000 recarga en vez de descarga

#define volumen_recarga_max 28                           //  Valor máximo del volumen de descarga para su distribución aleatoria   // EDUARDO 20000 recarga en vez de descarga

#define volumen_ventas_min 0.1                           //  Valor mínimo del volumen de ventas para su distribución aleatoria     // EDUARDO 1000

#define volumen_ventas_max 2                             //  Valor máximo del volumen de ventas para su distribución  aleatoria    // EDUARDO 4000

#define t_prox_recarga_min 25000                         //  Valor mínimo del tiempo de la próxima descarga para su distribución aleatoria   // EDUARDO 25000  recarga en vez de descarga
```

```
#define t_prox_recarga_max 50000                    //  Valor máximo del tiempo de la
próxima descarga para su distribución aleatoria   // EDUARDO 50000  recarga en vez de
descarga

#define t_prox_venta_min 13000                      //  Valor mínimo del tiempo de la próxima
ventas para su distribución aleatoria     // EDUARDO 13000

#define t_prox_venta_max 20000                      //  Valor máximo del tiempo de la próxima
ventas para su distribución aleatoria     // EDUARDO 20000


//  Indica si se ha realizado o no la TARA del deposito

bool tara_recarga;                          //  Indica si se ha realizado la tara de la bascula
('1') o no ('0') del tanque de recarga

bool tara_disp_1;                           //  Indica si se ha realizado la tara de la bascula
('1') o no ('0') del deposito 1

bool tara_disp_2;                           //  Indica si se ha realizado la tara de la bascula
('1') o no ('0') del deposito 1

bool tara_fugas;                            //  Indica si se ha realizado la tara de la bascula ('1')
o no ('0') del tanque de fungas


//  Variables temporales que influyen en la simulación de los eventos aleatorios  //

bool estado_recarga;

bool estado_disp_1;

bool estado_disp_2;

bool estado_fugas;

long tSimulacion = 0;                       //  Tiempo de la simulación


//  MAQUINAS DE ESTADOS  //
```

```
//  Variables de la máquina de estados de la controladora y del panel de control  //

#define FUNC_NORMAL 1

#define PAUSA 2

#define EMERGENCIA 3

#define LLEVAR_COND_INIC 4


//  Variables de la maquina de estados al realizar el llenado de los fondos de los depositos  //

#define COMPROBAR_CICLO_INICIAL 5

#define CICLO_FONDO_TANQUES 6

#define LLENANDO_RECARGA_ALMACEN 7

#define LLENANDO_FUGA 8

#define LLENANDO_DEPOSITO_1y2 9

#define COLOCANDO_FONDOS_FINALES 10

int estado_actual = COMPROBAR_CICLO_INICIAL;


//  Variables de la maquina de estados al realizar la tara  //

#define HACER_TARA 11

#define SIN_HACER 12                    //  Estados del tarado de los depositos

#define HECHO 13

int estado_taras = SIN_HACER;


//  Variables de la maquina de estados al calcular el volumen de las recargas //
```

```
#define LLENAR_VOLUMEN_RECARGA_1 15

#define VACIAR_VOLUMEN_RECARGA_1 16

#define MEDIR_VOLUMEN_RECARGA_1 17

#define VACIAR_ALMACENAMINETO_1 18


#define LLENAR_VOLUMEN_RECARGA_2 19

#define VACIAR_VOLUMEN_RECARGA_2 20

#define MEDIR_VOLUMEN_RECARGA_2 21

#define VACIAR_ALMACENAMINETO_2 22


#define LLENAR_VOLUMEN_RECARGA_3 23

#define VACIAR_VOLUMEN_RECARGA_3 24

#define MEDIR_VOLUMEN_RECARGA_3 25

#define VACIAR_ALMACENAMINETO_3 26


#define CONDICIONES_FUNC_NORMAL 27


//  Botonera de la planta  //

#define PIN_START 2                    //  START

#define PIN_STOP 3                     //  Botón STOP

#define PIN_DIAL 4                     //  Posición del dial (0->automático,
1->manual)
```

```
#define PIN_EMERGENCIA 5                    //  Reset de la planta

#define alarma_rebosamiento 6               //  Señal de alarma en caso de rebosamiento

int start;                                  //  Señal de START

int stop_;                                  //  Señal de STOP

int seta_emergencia;                        //  Señal de reset

int dial;                                   //  Señal que recoge la posición del dial
(0->automático, 1->manual)
```

Deposito2.cpp

```
#include "Arduino.h"

#include "Deposito2.h"


#define TANQUE_VACIO 0

#define LLENADO 1

#define TANQUE_LLENO 2

#define DESCARGANDO 3

#define EMERGENCIA 4

#define REINICIO 5

#define bit_PIN 1023

#define bit_PIN_OUT 255

#define t_ciclo 0.003

#define r_max 51

#define v_r_max 512


#include <HX711_ADC.h>

#if defined(ESP8266)|| defined(ESP32) || defined(AVR)

#include <EEPROM.h>

#endif

const int calVal_eepromAdress = 0;
```

```
unsigned long t = 0;


Deposito::Deposito (int _PIN_Rele1_Llenado, int _PIN_Rele1_Vaciado, int _PIN_IN1, int
_PIN_IN2, int _PIN_ENA, int _HX711_dout,

int _HX711_sck, float _valor_calibracion_basculas, int _sensor_0, int _sensor_1,  int _sensor_2,
int _sensor_3, float _capacidad_maxima, float _vol_max_seguridad, float
_volumen_descarga_max, float _volumen_descarga_min,

float _t_prox_descarga_max, float _t_prox_descarga_min, bool _recarga_o_descarga, Deposito
*deposito_ant, Deposito *deposito_sig, bool _es_fugas, bool _es_recarga, float
_vol_seguridad_anterior, float _vol_limite_anterior, float _vol_min_comienzo_bomba) :
LoadCell(_HX711_dout, _HX711_sck) {


 estado_actual = TANQUE_VACIO;

 abrir_valv_llenado_activada_antes = 0;

 cerrar_valv_llenado_activada_antes = 0;

 abrir_valv_vaciado_activada_antes = 0;

 cerrar_valv_vaciado_activada_antes = 0;


 //Añadido nuevo


 PIN_Rele1_Llenado = _PIN_Rele1_Llenado;

 PIN_Rele1_Vaciado = _PIN_Rele1_Vaciado;

 PIN_IN1 = _PIN_IN1;

 PIN_IN2 = _PIN_IN2;

 PIN_ENA = _PIN_ENA;
```

```
sensor_0 = _sensor_0;

sensor_1 = _sensor_1;

sensor_2 = _sensor_2;

sensor_3 = _sensor_3;


HX711_dout = _HX711_dout;

HX711_sck = _HX711_sck;

valor_calibracion_basculas = _valor_calibracion_basculas;


capacidad_maxima = _capacidad_maxima;

vol_max_seguridad = _vol_max_seguridad;

volumen_descarga_max = _volumen_descarga_max;

volumen_descarga_min = _volumen_descarga_min;

t_prox_descarga_max = _t_prox_descarga_max;

t_prox_descarga_min = _t_prox_descarga_min;

vol_prox_mov = 0;

t_prox_mov = 0;

nuevo_vol_descarga();

nuevo_t_prox_mov();


recarga_o_descarga = _recarga_o_descarga;

es_fugas = _es_fugas;
```

```
  es_recarga = _es_recarga;


  this->deposito_sig = deposito_sig;

  this->deposito_ant = deposito_ant;


  vol_seguridad_anterior = _vol_seguridad_anterior;

  vol_limite_anterior = _vol_limite_anterior;

  vol_min_comienzo_bomba = _vol_min_comienzo_bomba;


// Asignación de los pines a las válvulas correspondientes //


  pinMode(sensor_0, INPUT);

  pinMode(sensor_1, INPUT);

  pinMode(sensor_2, INPUT);

  pinMode(sensor_3, INPUT);


  pinMode(PIN_Rele1_Llenado, OUTPUT);

  pinMode(PIN_Rele1_Vaciado, OUTPUT);


  pinMode(PIN_IN1, OUTPUT);

  pinMode(PIN_IN2, OUTPUT);

  pinMode(PIN_ENA, OUTPUT);
```

```
  pinMode(HX711_dout, INPUT);

  pinMode(HX711_sck, INPUT);


}


//  Método que comprueba el estado del deposito //


bool Deposito::actualizar_estado (int tSimulacion, int alarma, int volver_a_empezar) {
//  El parámetro volver_a_empezar indíca que el conjunto de todos los depositos se han vaciado
y podemos salir del estado de reinicio


if ((es_fugas == 1) && (tSimulacion > 7200)) {

 //if (es_fugas == 1) {

    velocidad = genera_caudal();

  }


  if (alarma == HIGH) {                                                    //  Condición de
emergencia en función del reset

      estado_actual = EMERGENCIA;

      abrir_valv_llenado_activada_antes = 0;

      cerrar_valv_llenado_activada_antes = 0;

      abrir_valv_vaciado_activada_antes = 0;

      cerrar_valv_vaciado_activada_antes = 0;

  }
```

```
  if ((volumen >= vol_max_seguridad) && (estado_actual != REINICIO)) {
//  Condición de emergencia

      estado_actual = EMERGENCIA;

      abrir_valv_llenado_activada_antes = 0;

      cerrar_valv_llenado_activada_antes = 0;

      abrir_valv_vaciado_activada_antes = 0;

      cerrar_valv_vaciado_activada_antes = 0;

  }


  if (estado_actual == TANQUE_VACIO) {                                    //  Estado
que define las condiciones de cuando el tanque espera una descarga

    cerrar_valvula_llenado();                                             //  Cerramos la
válvula de llenado del tanque

    cerrar_valvula_vaciado();                                             //  Cerramos la
válvula de vaciado del tanque

   if ( ( ( (tSimulacion >= t_prox_mov) && (deposito_ant->ready_to_send(vol_prox_mov) == 1)
) || (recarga_o_descarga == true) ) && (peticion_pausa_venta == 0) ) {                //
Condición de cambio de estado, tanque vacio a llenado

   //if  ( ( (tSimulacion >= t_prox_mov) || (recarga_o_descarga == true) ) &&
(peticion_pausa_venta == 0) ) {                                           //  Condición de cambio de estado,
tanque vacio a llenado

      abrir_valv_llenado_activada_antes = 0;

      cerrar_valv_llenado_activada_antes = 0;

      abrir_valv_vaciado_activada_antes = 0;

      cerrar_valv_vaciado_activada_antes = 0;
```

```
    estado_actual = LLENADO;
   }
  }

  if ( (estado_actual == LLENADO) && (peticion_pausa_fuga == 0) ) {          //  Estado que define el llenado del tanque
   abrir_valvula_llenado();                                                  //  Abrimos la válvula de llenado del tanque
   cerrar_valvula_vaciado();                                                 //  Cerramos la válvula de vaciado del tanque
   if ((volumen >= vol_prox_mov) && (es_fugas == 0)) {
    estado_actual = TANQUE_LLENO;
    abrir_valv_llenado_activada_antes = 0;
    cerrar_valv_llenado_activada_antes = 0;
    abrir_valv_vaciado_activada_antes = 0;
    cerrar_valv_vaciado_activada_antes = 0;
    volumen_total_venta = volumen_total_venta + volumen;
    ventas_totales++;
   }
  }

  if (estado_actual == TANQUE_LLENO) {                                       //  Estado que define las condiciones de cuando el tanque ha recibido una descarga
   cerrar_valvula_llenado();                                                 //  Cerramos la válvula de llenado del tanque
```

```
   cerrar_valvula_vaciado();                                          //  Cerramos la
válvula de vaciado del tanque

   if ((deposito_sig->ready_to_receive(vol_prox_mov) == 1) && ((tSimulacion >=
t_prox_mov) ||          //  Condición de cambio de estado, tanque lleno a descargando

   (recarga_o_descarga == false)) && (peticion_pausa_recarga == 0) ) {

    // toDo generalizar la condicion de que existe suficiente comb. en el principal cuando haya
mas de 1 dep. de ventas

     estado_actual = DESCARGANDO;

     abrir_valv_llenado_activada_antes = 0;

     cerrar_valv_llenado_activada_antes = 0;

     abrir_valv_vaciado_activada_antes = 0;

     cerrar_valv_vaciado_activada_antes = 0;

     volumen_total_recarga = volumen_total_recarga + volumen;

     recargas_totales++;

   }

 }


 if (estado_actual == DESCARGANDO) {                                  //
Estado que define cuando el tanque está descargando

   cerrar_valvula_llenado();                                          //  Cerramos la
válvula de llenado del tanque

   abrir_valvula_vaciado();                                           //  Abrimos la
válvula de vaciado del tanque

   if (volumen < UMBRAL_VACIO) {                                      //
Condición de cambio de estado, descargando a tanque vacio
```

```
      estado_actual = TANQUE_VACIO;
      abrir_valv_llenado_activada_antes = 0;
      cerrar_valv_llenado_activada_antes = 0;
      abrir_valv_vaciado_activada_antes = 0;
      cerrar_valv_vaciado_activada_antes = 0;
      nuevo_vol_descarga();                                    // Asignamos
nuevamente valores aleatorios a las variables de simulación
      nuevo_t_prox_mov();
    }
  }

  if (estado_actual == EMERGENCIA) {                           // Estado
que define la entrada a los límites de seguridad del tanque
    cerrar_valvula_llenado();                                  // Cerramos la
válvula de llenado del tanque
    cerrar_valvula_vaciado();                                  // Cerramos la
válvula de vaciado del tanque
    if (alarma == LOW) {                                       // Condición de
cambio de estado, emergencia a reinicio
      estado_actual = REINICIO;
      abrir_valv_llenado_activada_antes = 0;
      cerrar_valv_llenado_activada_antes = 0;
      abrir_valv_vaciado_activada_antes = 0;
      cerrar_valv_vaciado_activada_antes = 0;
    }
```

```
    return true;
  }

  if (estado_actual == REINICIO) {                                    // Estado que define el reinicio de la planta
    abrir_valvula_llenado();                                          // Abrimos la válvula de llenado del tanque
    abrir_valvula_vaciado();                                          // Abrimos la válvula de vaciado del tanque
    if (volver_a_empezar == 1) {                                      // Condición de cambio de estado, de reinicio a tanque vacio
      estado_actual = TANQUE_VACIO;
      abrir_valv_llenado_activada_antes = 0;
      cerrar_valv_llenado_activada_antes = 0;
      abrir_valv_vaciado_activada_antes = 0;
      cerrar_valv_vaciado_activada_antes = 0;
      nuevo_vol_descarga();                                           // Asignamos nuevamente valores aleatorios a las variables de simulación
      nuevo_t_prox_mov();
    }
  }

  return false;
}
```

```
//  Método que abre la válvula de llenado //


void Deposito::abrir_valvula_llenado () {
 if (es_fugas == 0) {
 /*  if (abrir_valv_llenado_activada_antes == 0) {
     digitalWrite(PIN_Rele1_Llenado, HIGH);
     t_inicio_valv_llenado_abierta = millis();
     abrir_valv_llenado_activada_antes = 1;
   }  */
   digitalWrite(PIN_Rele1_Llenado, LOW);  //Motor activado
 } else {
   digitalWrite(PIN_IN1, HIGH);
   digitalWrite(PIN_IN2, LOW);
   analogWrite(PIN_ENA, velocidad);
 }
}


//  Método que abre la válvula de vaciado //


void Deposito::abrir_valvula_vaciado () {
/*  if (abrir_valv_vaciado_activada_antes == 0) {
   digitalWrite(PIN_Rele1_Vaciado, HIGH);
```

```
    t_inicio_valv_vaciado_abierta = millis();

    abrir_valv_vaciado_activada_antes = 1;              // INDICA QUE YA SE HA ACTIVADO LA
VALVULA ANTERIORMENTE  //

   }  */

   digitalWrite(PIN_Rele1_Vaciado, LOW);

}


//  Método que cierra la válvula de llenado //


void Deposito::cerrar_valvula_llenado () {

   if (es_fugas == 0) {

   /*  if (cerrar_valv_llenado_activada_antes == 0) {

      digitalWrite(PIN_Rele1_Llenado, LOW);

      t_inicio_valv_llenado_cerrada = millis();

      cerrar_valv_llenado_activada_antes = 1;              // INDICA QUE YA SE HA ACTIVADO
LA VALVULA ANTERIORMENTE  //

    }  */

    digitalWrite(PIN_Rele1_Llenado, HIGH);


   } else {

    analogWrite(PIN_ENA, 0);

    digitalWrite(PIN_IN1, LOW);

    digitalWrite(PIN_IN2, LOW);
```

```
  }
}

//  Método que cierra la válvula de vaciado //

void Deposito::cerrar_valvula_vaciado () {
/*  if (cerrar_valv_vaciado_activada_antes == 0) {
    digitalWrite(PIN_Rele1_Vaciado, LOW);
    t_inicio_valv_vaciado_cerrada = millis();
    cerrar_valv_vaciado_activada_antes = 1;              // INDICA QUE YA SE HA ACTIVADO
LA VALVULA ANTERIORMENTE  //
  }   */
  digitalWrite(PIN_Rele1_Vaciado, HIGH);
}


//  Método que generará valores aleatorios de tiempo a las variables de la simulación //

void Deposito::nuevo_t_prox_mov () {
  t_prox_mov = t_prox_mov + random (t_prox_descarga_min, t_prox_descarga_max)/1000;
}

//  Método que nos permite ver si el tanque está listo para que se realice una descarga en él //
```

```
int Deposito::ready_to_receive (float vol_prox_mov) {
  if(vol_max_seguridad > (volumen + vol_prox_mov)) {
    return 1;
  } else {
    return 0;
  }
}
```

```
//  Método que nos permite ver si el tanque anterior está listo para poder realizar una descarga  //
```

```
int Deposito::ready_to_send (float vol_prox_mov) {
  if ( (deposito_ant->get_volumen() > vol_prox_mov) ) {
    return 1;
  } else {
    return 0;
  }
}
```

```
//  Método de conversión de volumen medido por las basculas //
```

```
float Deposito::conversion_volumen () {
  float resultado;
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0;                    //  Aumentar el valor para ralentizar la actividad de impresión en serie
  if (LoadCell.update()) newDataReady = true;                  //  Comprobar si hay nuevos datos / iniciar la siguiente conversión
  if (newDataReady) {                                  //  Obtener un valor suavizado del conjunto de datos:
      resultado = LoadCell.getData();
  }
  return resultado;
}

/*
float Deposito::conversion_volumen () {
  float resultado;
  resultado = analogRead(puerto_sensor_vol);
  resultado = resultado -10 - (r_max - (((v_r_max - resultado)/bit_PIN)*(v_r_max - resultado)/bit_PIN)*bit_PIN/5);
  if (es_fugas == 1) {
    resultado = resultado + 10;
  }
  resultado = (resultado*capacidad_maxima)/bit_PIN;
```

```
  return resultado;
}

*/

// ********** REVISAR GENERAR CAUDAL FUGAS ********** //
//  Método genera un caudal para las fugas  //

int Deposito::genera_caudal() {
  int caudal;

  if (deposito_sig->get_volumen() > vol_seguridad_anterior) {
    caudal = bit_PIN_OUT;
  }
  if ((deposito_sig->get_volumen() < vol_seguridad_anterior) &&
(deposito_sig->get_volumen() > vol_min_comienzo_bomba))  {
    caudal = (deposito_sig->get_volumen() / vol_limite_anterior)*bit_PIN_OUT;      // Apartir de
1110 g es cuando comienza la bomba peristaltica (PIN_ENA > 150)
   // caudal = bit_PIN_OUT;
  }
  if (deposito_sig->get_volumen() < vol_min_comienzo_bomba) {
    caudal = 0;
  }
```

```
  return caudal;
}
```

```
/*
int Deposito::genera_caudal() {
  int caudal;
  if (deposito_sig->get_volumen() > 100) {
    caudal = bit_PIN_OUT;
  }
  if ((deposito_sig->get_volumen() < 100) && (deposito_sig->get_volumen() > 1))  {
    caudal = (deposito_sig->get_volumen()/125)*bit_PIN_OUT;
  }
  if (deposito_sig->get_volumen() < 1) {
    caudal = 0;
  }
  return caudal;
}

*/
```

```
//  Método que realiza la tara de las basculas  //


bool Deposito::realizar_tara() {
  LoadCell.begin();
  float calibrationValue;
  bool tara_realizada;
  calibrationValue = valor_calibracion_basculas;                  //  Valor de calibración
  #if defined(ESP8266)|| defined(ESP32)
    //EEPROM.begin(512); // uncomment this if you use ESP8266/ESP32 and want to fetch the calibration value from eeprom
  #endif
    //EEPROM.get(calVal_eepromAdress, calibrationValue); // uncomment this if you want to fetch the calibration value from eeprom


  unsigned long stabilizingtime = 3000;                           //  La precisión justo después del encendido se puede mejorar agregando unos segundos de tiempo de estabilización
  boolean _tare = true;                                           //  Establezca esto en falso si no desea que se realice la tara en el siguiente paso
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag()) {
    while (1);
    tara_realizada = false;


  } else {
```

```
    LoadCell.setCalFactor(calibrationValue);                    //  set calibration value (float)
    Serial.println("Tara Realizada correctamente");
    tara_realizada = true;
  }
  return tara_realizada;
}



//  Funciones para dejar en estado de pausa los depositos para el Datalogger  //


bool Deposito::solicitud_pausa_recarga() {
  peticion_pausa_recarga = 1;
  bool recarga_pausada;
  if (estado_actual != DESCARGANDO) {
    recarga_pausada = 1;
  }
  return recarga_pausada;
}


bool Deposito::solicitud_pausa_venta() {
  peticion_pausa_venta = 1;
  bool venta_pausada;
  if (estado_actual != LLENADO) {
```

```
    venta_pausada = 1;
  }
  return venta_pausada;
}

//Comprobar si bomba peristaltica se para
bool Deposito::solicitud_pausa_fuga() {
  bool fuga_pausada;
  peticion_pausa_fuga = 1;
  if(  (peticion_pausa_fuga == 1)&&(estado_actual == LLENADO) ){
    cerrar_valvula_llenado();
    fuga_pausada = 1;
  }
  return fuga_pausada;
}

void Deposito::quitar_pausa() {
  peticion_pausa_recarga = 0;
  peticion_pausa_venta = 0;
  peticion_pausa_fuga = 0;

}
```

```
float Deposito::volumen_acumulado_recarga() {
  return volumen_total_recarga;
}

float Deposito::volumen_acumulado_venta() {
  return volumen_total_venta;
}

int Deposito::recargas_realizadas() {
  return recargas_totales;
}

int Deposito::ventas_realizadas() {
  return ventas_totales;
}

void Deposito::reset_recargas_realizadas(){
  recargas_totales = 0;
}

void Deposito::reset_ventas_realizadas(){
  ventas_totales = 0;
}
```

```
/*************************************************************************
**********************************/

/*************************************  VOLUMEN SENSORES RECARGA
*************************************************/

/*************************************************************************
**********************************/

float Deposito::recarga_sensor_1_volumen(){

 return descarga_sensor_1;

}


float Deposito::recarga_sensor_2_volumen(){

 return descarga_sensor_2;

}


float Deposito::recarga_sensor_3_volumen(){

 return descarga_sensor_3;

}



/*************************************************************************
**********************************/

/*************************************  VOLUMEN Y VOL PROXIMO
*************************************************/
```

```
/*************************************************************************
***********************************/

float Deposito::serial_prox_vol(){

  return vol_prox_mov;

}


float Deposito::serial_volumen(){

  return volumen;

}
```

// CODIGO AÑADIDO PARA MEDIR VOLUMEN DE LAS DESCARGAS //

```
/*************************************************************************
***********************************/

/************************************     RECARGA SENSOR 1
*********************************************/
```

```
/*************************************************************************
************************************/

bool Deposito::llenar_recarga_1() {

  abrir_valvula_llenado();

  if (digitalRead(sensor_1) == 1) {

    llenar_sensor_1 = 1;

  }

  return llenar_sensor_1;

}


bool Deposito::vaciar_recarga_1() {

  abrir_valv_llenado_activada_antes = 0;

  cerrar_valv_llenado_activada_antes = 0;

  abrir_valvula_vaciado();

  if (digitalRead(sensor_0) == 0) {

    vaciar_sensor_1 = 1;

  }

  return vaciar_sensor_1;

}


void Deposito::medir_recarga_1()  {

  abrir_valv_vaciado_activada_antes = 0;

  cerrar_valv_vaciado_activada_antes = 0;
```

```
  descarga_sensor_1 = deposito_sig->get_volumen();  //float descarga_sensor_1; en el .h
}

/*************************************************************************
**********************************/
/*************************************     RECARGA SENSOR 2
*********************************************/
/*************************************************************************
**********************************/
bool Deposito::llenar_recarga_2() {
  abrir_valvula_llenado();
  if (digitalRead(sensor_2) == 1) {
    llenar_sensor_2 = 1;
  }
  return llenar_sensor_2;
}

bool Deposito::vaciar_recarga_2() {
  abrir_valv_llenado_activada_antes = 0;
  cerrar_valv_llenado_activada_antes = 0;
  abrir_valvula_vaciado();
  if (digitalRead(sensor_0) == 0) {
    vaciar_sensor_2 = 1;
  }
```

```
  return vaciar_sensor_2;
}

void Deposito::medir_recarga_2()  {
  abrir_valv_vaciado_activada_antes = 0;
  cerrar_valv_vaciado_activada_antes = 0;
  descarga_sensor_2 = deposito_sig->get_volumen();  //float descarga_sensor_1; en el .h
}

/*************************************************************************
************************************/
/*************************************     RECARGA SENSOR 3
*************************************************/
/*************************************************************************
************************************/
bool Deposito::llenar_recarga_3() {
  abrir_valvula_llenado();
  if (digitalRead(sensor_3) == 1) {
    llenar_sensor_3 = 1;
  }
  return llenar_sensor_3;
}
```

```
bool Deposito::vaciar_recarga_3() {
  abrir_valv_llenado_activada_antes = 0;
  cerrar_valv_llenado_activada_antes = 0;
  abrir_valvula_vaciado();
  if (digitalRead(sensor_0) == 0) {
    vaciar_sensor_3 = 1;
  }
  return vaciar_sensor_3;
}

void Deposito::medir_recarga_3()  {
  abrir_valv_vaciado_activada_antes = 0;
  cerrar_valv_vaciado_activada_antes = 0;
  descarga_sensor_3 = deposito_sig->get_volumen();  //float descarga_sensor_1; en el .h
}

/**********************************************************************
**********************************/
/*********************************     VACIAR TANQUE ALMACENAMIENTO
*****************************************/
/**********************************************************************
**********************************/
bool Deposito::vaciar_almacemiento() {
  abrir_valvula_llenado();
```

```
  abrir_valvula_vaciado();

  if ( (deposito_ant->get_volumen() < UMBRAL_VACIO) && (volumen < UMBRAL_VACIO)
) {

      vacio_almacenamiento = 1;

  }

  return vacio_almacenamiento;

}


void Deposito::resetar_vaciar_almacemiento() {

  vacio_almacenamiento = 0;

}


void Deposito::ciclos_cero() {

      abrir_valv_llenado_activada_antes = 0;

      cerrar_valv_llenado_activada_antes = 0;

      abrir_valv_vaciado_activada_antes = 0;

      cerrar_valv_vaciado_activada_antes = 0;

}




/**********************************************************************
**********************************/
```

```
/*********************************     SENSORES TANQUE RECARGA
****************************************/

/*************************************************************************
***********************************/


//  Método que nos permite saber el volumen del tanque en todo momento  //


float Deposito::get_volumen () {


 if (es_recarga == 0) {

   volumen = conversion_volumen ();


 } else if (digitalRead(sensor_3) == 1) {

   volumen = recarga_sensor_3_volumen();            // o descarga_sensor_3


 } else if (digitalRead(sensor_2) == 1) {

   volumen = recarga_sensor_2_volumen();           // o descarga_sensor_2


 } else if (digitalRead(sensor_1) == 1) {

   volumen = recarga_sensor_1_volumen();           // o descarga_sensor_1


 } else if (digitalRead(sensor_0) == 1) {

   volumen = UMBRAL_VACIO;
```

```
  } else {
    volumen = 0;
  }

  return volumen;
}

//  Método que generará valores aleatorios de volumen a las variables de la simulación  //

void Deposito::nuevo_vol_descarga () {
  int numero_sensor;
  if (es_recarga == 0) {
    vol_prox_mov = random (volumen_descarga_min, volumen_descarga_max);

  } else {
    numero_sensor = random (1,N_SENSORES_REC);      //N_SENSORES_REC

    if (numero_sensor == 3){
      vol_prox_mov = recarga_sensor_3_volumen();
    }
```

```
    if (numero_sensor == 2){
      vol_prox_mov = recarga_sensor_2_volumen();
    }

    if (numero_sensor == 1) {
      vol_prox_mov = recarga_sensor_1_volumen();
    }
  }
}
```

Deposito2.h

```
#include "Arduino.h"

#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

// VARIABLES TANQUE RECARGA //
#define UMBRAL_VACIO 0.1              //  Volumen cuando sensor_0 está activado, g = ml
#define VOL_REC_1 5                   //  Volumen cuando sensor_1 está activado, g = ml
#define VOL_REC_2 20                  //  Volumen cuando sensor_2 está activado, g = ml
#define VOL_REC_3 30                  //  Volumen cuando sensor_3 está activado, g = ml

#define N_SENSORES_REC 3              //  Numero de sensores utilizados en el Tanque de Recarga
#define RESOLUCION_REC 500            //  Resolucion entre cada sensor

#define T_VAL_CONMUT 2000             //  Tiempo para conmutar las valvulas, tanto para abrir como cerrar
```

```
class Deposito {


  private:


  //Atributos estáticos//

  bool recarga_o_descarga;              //  Variable que nos indica si, a la hora de simular, nos interesa si el deposito se recarga o descarga agua en el siguiente//

  bool es_fugas ;                       //  Parámetro que nos indicará si el tanque es el de fugas (1)//

  bool es_recarga ;                     //  Parámetro que nos indicará si el tanque es el de recarga (1)//


  bool peticion_pausa_recarga;

  bool peticion_pausa_venta;

  bool peticion_pausa_fuga;


  Deposito *deposito_sig;               //  Puntero que apunta al deposito siguiente con objetivo de obtener el permiso para realizar una descarga//

  Deposito *deposito_ant;               //  Puntero que apunta al deposito siguiente con objetivo de obtener el permiso para realizar una descarga//



  int PIN_Rele1_Llenado;
```

```
  int PIN_Rele1_Vaciado;
  int PIN_IN1;
  int PIN_IN2;
  int PIN_ENA;

  int sensor_0;
  int sensor_1;
  int sensor_2;
  int sensor_3;

  HX711_ADC LoadCell;                    //Declaracion de la bascula
  int HX711_dout;
  int HX711_sck;
  float valor_calibracion_basculas;

// Atributos referidos al tanque anterior //
  float vol_seguridad_anterior;
  float vol_limite_anterior;
  float vol_min_comienzo_bomba;

 /*
// Tomamos el tiempo de inicio para establecer el tiempo de CONMUTADO de las valvulas //
  long t_inicio_valv_llenado_abierta = 0;
```

```
  long t_inicio_valv_llenado_cerrada = 0;

  long t_inicio_valv_vaciado_abierta = 0;

  long t_inicio_valv_vaciado_cerrada = 0;

*/


//  Condicion para no repetir conmutado de las valvulas //

  bool abrir_valv_llenado_activada_antes = 0;

  bool cerrar_valv_llenado_activada_antes = 0;

  bool abrir_valv_vaciado_activada_antes = 0;

  bool cerrar_valv_vaciado_activada_antes = 0;



//  Atributos de simulación //

  float capacidad_maxima;              //  Límite máximo que puede alcanzar el deposito//

  float vol_max_seguridad;             //  Límite hasta el cual se podrá llenar el deposito sin
llegar al umbral máximo//

  float volumen_descarga_max;          //  Límites de la simulación//

  float volumen_descarga_min;          //  Límites de la simulación//

  float t_prox_descarga_max;           //  Límites de la simulación//

  float t_prox_descarga_min;           //  Límites de la simulación//

  float vol_prox_mov;                  //  Cantidad de agua que descargará el deposito cuando la
simulación lo diga//

  float t_prox_mov;                    //  Tiempo en el que se realizará el próximo movimiento del
tanque//
```

```
  //Atributos de estado//
  int estado_actual;                    //  Recoge el estado en el que se encuentra el deposito//
  float volumen;                        //  Señal que almacena el valor de volumen del deposito//
  float velocidad;                      //  En caso de que el tanque sea el de fugas, esta variable
guardará la velocidad de cuanto fugará//


  float volumen_total_recarga = 0;
  float volumen_total_venta = 0;
  int ventas_totales = 0;
  int recargas_totales = 0;



  int NIVEL_DEPURACION = 0;


//  Volumenes calculados antes de entrar en FUNC. NORMAL
  float descarga_sensor_3 = 0;
  float descarga_sensor_2 = 0;
  float descarga_sensor_1 = 0;


  bool vacio_almacenamiento;


  bool llenar_sensor_1;
```

```
  bool vaciar_sensor_1;


  bool llenar_sensor_2;

  bool vaciar_sensor_2;


  bool llenar_sensor_3;

  bool vaciar_sensor_3;




  public:


  //Constructor de la clase deposito//


   Deposito (int _PIN_Rele1_Llenado, int _PIN_Rele1_Vaciado, int _PIN_IN1, int _PIN_IN2,
int _PIN_ENA, int _HX711_dout, int _HX711_sck, float _valor_calibracion_basculas, int
_sensor_0, int _sensor_1,  int _sensor_2, int _sensor_3,

float _capacidad_maxima, float _vol_max_seguridad, float _volumen_descarga_max, float
_volumen_descarga_min, float _t_prox_descarga_max, float _t_prox_descarga_min,

bool _recarga_o_descarga, Deposito *deposito_ant, Deposito *deposito_sig, bool _es_fugas,
bool _es_recarga, float _vol_seguridad_anterior, float _vol_limite_anterior, float
_vol_min_comienzo_bomba);
```

```
//  Método que comprueba el estado del deposito//

bool actualizar_estado (int tSimulacion, int alarma, int volver_a_empezar);



//  Método que abre la válvula de llenado //

void abrir_valvula_llenado ();



//  Método que abre la válvula de vaciado //

void abrir_valvula_vaciado ();



//Método que cierra la válvula de llenado //

void cerrar_valvula_llenado ();



//  Método que cierra la válvula de vaciado //

void cerrar_valvula_vaciado ();



//  Método que generará valores aleatorios de volumen a las variables de la simulación  //

void nuevo_vol_descarga ();
```

```
//  Método que generará valores aleatorios de tiempo a las variables de la simulación //

void nuevo_t_prox_mov ();



//  Método que nos permite ver si el tanque está listo para que se realice una descarga en él //

int ready_to_receive (float vol_prox_mov);



//  Método que nos permite ver si el tanque está listo para que se realice una descarga en él//

int ready_to_send (float vol_prox_mov);



//  Método que nos permite saber el volumen del tanque en todo momento  //

float get_volumen ();



//  Método que convierte la señal recibida por los sensores a volumen //

float conversion_volumen ();



//  Método genera un caudal para las fugas  //

int genera_caudal ();
```

```
//  Método que realiza el tarado inicial  //

bool realizar_tara ();



//  Método que realiza la solicitud neceraria para el Datalogger  dependiendo del Deposito  //

bool solicitud_pausa_recarga ();

bool solicitud_pausa_venta ();

bool solicitud_pausa_fuga ();


void quitar_pausa();



//  Método que nos permite saber el volumen acumulado del tanque de recarga  //

float volumen_acumulado_recarga();



//  Método que nos permite saber el volumen acumulado del deposito de ventas  //

float volumen_acumulado_venta();


//  Método que nos permite saber las recargas realizadas  //

int recargas_realizadas();
```

```
//  Método que nos permite saber las ventas realizadas  //

int ventas_realizadas();


//  Método que resetea las recargas realizadas  //

void reset_recargas_realizadas();


//  Método que resetea las ventas realizadas  //

void reset_ventas_realizadas();



//  Método que realiza el llenado de las recargas y su posterior medicion  //

bool llenar_recarga_1 ();

bool vaciar_recarga_1 ();

void medir_recarga_1 ();


bool llenar_recarga_2 ();

bool vaciar_recarga_2 ();

void medir_recarga_2 ();


bool llenar_recarga_3 ();

bool vaciar_recarga_3 ();

void medir_recarga_3 ();
```

```
bool vaciar_almacemiento ();

void resetar_vaciar_almacemiento();


void ciclos_cero ();



float recarga_sensor_1_volumen();

float recarga_sensor_2_volumen();

float recarga_sensor_3_volumen();


float serial_prox_vol();

float serial_volumen();



};
```

DetectaFlanco2.cpp

```
#include "Arduino.h"

#include "DetectaFlanco2.h"


DetectaFlanco::DetectaFlanco(int pin) {

  _pin = pin ;

}


void DetectaFlanco::inicio(int input) {  //solo admitre los parámetros INPUT e
INPUT_PULLUP

  pinMode(_pin, input);

  _anterior_estado = digitalRead(_pin);

}


int DetectaFlanco::comprueba() {

  _estado = digitalRead(_pin);


  if (_anterior_estado != _estado) {

    if (_estado == HIGH) {

      _anterior_estado = _estado;
```

```
      return 1; //Flanco Ascendente
    }
    else {
      _anterior_estado = _estado;
      return -1; //Flanco Descendente
    }
  }
  else
    return 0;
}
```

DetectaFlanco2.h

```
#ifndef DetectaFlanco_h

#define DetectaFlanco_h


#include "Arduino.h"


class DetectaFlanco {

  private:

    int _pin;

    boolean _anterior_estado;

    boolean _estado;


  public:

    DetectaFlanco(int pin);

    void inicio(int input);  //INPUT or INPUT_PULLUP

    int comprueba();

};


#endif
```

Anexo 2

Guía de desarrollo. Sistema de detección de fugas

En este anexo se detalla la estructura del código implementado en el detector de fugas, nombrando los diferentes archivos que se han utilizado. El código del detector está compuesta por un solo archivo:

- Fugas2.ino. Archivo que contiene los métodos y constantes que definen el detector de fugas.

Fugas2.ino

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <TimeLib.h>

#define DEBUG(a) Serial.println(a);


String myString = "";

String str1 = "";

char* nums[8];

float volumenes[8];

float teorico[50];

float experimental[50];

float resultado[50];

float fugas_total;


int count = 0;

int contador = 1;

float actualizacion;

int a=0;

int b=0;
```

```
int x=0;



//unsigned long tiempo_1 = 7000;


char cadena[100];

char *name = NULL;

//char *token;

//const char *delimiter =",";



int LED1 = 22;


time_t fecha;


void setup(){


  Serial.begin(9600);


  setTime(00,00,00,27,07,2022);

  fecha = now();

  Serial.println(fecha);
```

```
  pinMode(LED1, OUTPUT);


}

void loop() {

if (Serial.available()){

  RecibirPaquete();
  LecturaVolumenes();
  VariacionVolumenFinal();
  x++;

  }
}

float RecibirPaquete(){

  myString = Serial.readString();
  Serial.println("PAQUETE RECIBIDO: ");
  Serial.println("Volumen principal , Recargas realizadas, Volumen acumulado_descargas , Ventas_1 realizadas, Volumen acumulado_ventas_1, Ventas_2 realizadas, Volumen acumulado_ventas_2, Volumen acumulado_fugas");
```

```
  Serial.println(myString);


}


float LecturaVolumenes(){


  int count = 0;
  Serial.println("Los valores son: ");
  myString.toCharArray(cadena, 100);
  name = strtok(cadena, ",");


    while(name != NULL) {
      // Serial.println(name);
       nums[count] = name;
       name = strtok(NULL, ",");
       count++;
    }


for (int i = 0; i<8; i++){
  volumenes[i] = String(nums[i]).toInt();
  Serial.println(volumenes[i]);
}
  Serial.print("\n");
```

```
}


float VariacionVolumenFinal(){


    teorico[a]=(volumenes[0] + volumenes[2] - (volumenes[4] + volumenes[6]));  // Volumen teórico que sale de: (volumen_almacenamiento + volumen_acumulado_descargas) - volumen_acumulado_ventas


    if(x >= 1){

      experimental[b] = volumenes[0];                                  // Volumen_almacenamiento (volumen inicial) del siguiente ciclo. Este volumen (suponiendo que no hay fugas) debería ser igual al volumen teórico calculado anteriormente

      Serial.print("Volumen final anterior: ");

      Serial.println(teorico[a-1]);                                    // Volumen teórico del ciclo anterior

      Serial.print("Volumen inicial nuevo: ");

      Serial.println(experimental[b]);

      resultado[b] = teorico[a-1] - experimental[b];                   //Comparo el volumen_almacenamiento nuevo (volumen inicial del ciclo nuevo) con el volumen teórico anterior

      Serial.print("Diferencia entre experimental y teórico: ");

      Serial.println(resultado[b]);

      fugas_total = fugas_total + resultado[b];                        // Acumlación de variaciones


       if (fugas_total >= 5) {

         Serial.println(" ¡ FUGA DETECTADA ! ");
```

```
            digitalWrite(LED1 , HIGH);
          }
         b++;
       }

        Serial.print("\n");

a++;

}
```

Anexo 3

Diario de actividades

Lo primero fue realizar un estudio de la planta. Para ello, tuvimos que tomar medidas de la estantería en la que montaríamos el prototipo. Una vez recogidas estas medidas, realizamos un estudio de los depósitos más adecuados para nuestro proyecto. Escogimos los depósitos más económicos posibles, siempre y cuando cumplieran con los requisitos nombrados en el apartado 4.2. Y una vez determinados los modelos de los recipientes, pasamos a elaborar un listado con todos ellos para poder empezar a buscar por las diferentes empresas.

El siguiente paso fue empezar a trabajar con arduino. En este proyecto trabajaríamos con el lenguaje de programación de arduino por lo que debíamos aprender a operar en él. Al no haber tratado nunca con este lenguaje, comenzamos creando un proyecto sencillo que nos permitiera desenvolvernos en este programa. El objetivo de este mini proyecto era conectar dos arduinos para que uno fuera capaz de recibir información del otro y, en función de dicha información, ejecutar unas determinadas órdenes. Esta tarea nos llevó tiempo ya que para nosotros era algo nuevo. Una vez efectuáramos esta tarea, ya seríamos capaces de realizar actividades más complejas en este entorno de trabajo, por lo que pasamos a implementar el módulo de detección de fugas de nuestro proyecto.

Lo siguiente fue empezar con el diseño del prototipo físico. Teníamos una idea del diseño del proyecto anterior, por lo que esto nos serviría como referencia. Comenzamos realizando un croquis de cada estantería con sus respectivos depósitos (ver Anexo 4). En cada uno de ellos podemos observar un esquema de cada balda con su respectivo depósito y los componentes necesarios para conducir el fluido de un tanque a otro. La finalidad de hacer estos croquis era conseguir un listado con el material final que se necesitaríamos. Una vez terminado el listado de materiales, comenzamos a buscar el material en las distintas empresas.

Ya adquiridos los materiales necesarios, empezamos a construir el prototipo físico. Como mencionamos anteriormente, ya contábamos con el diseño del proyecto anterior, por lo que teníamos una idea de cómo iría situado cada depósito, a diferencia de que nuestro proyecto tenía un recipiente de ventas más que tendríamos que adaptar al diseño.

La construcción de los tanques se realizó por separado. Lo primero sería agujerear el depósito con el que estuviésemos trabajando y, una vez tuviésemos el agujero hecho, colocaríamos las piezas necesarias para obtener la salida del depósito. El siguiente paso fue colocar el resto de componentes de acuerdo a los croquis (ver Anexo 4). Para evitar pequeñas fugas de agua añadimos teflón a las piezas en las que uno de sus lados fuese "macho". Las uniones entre componentes las hicimos mediante un pegamento de PVC en el caso de piezas a presión o, en el caso de uniones "macho – hembra", tan solo añadimos teflón y enroscamos.

Figura 10.1. Componentes depósito de almacenamiento

Tan pronto como tuviésemos montada la estructura de un depósito, comenzamos a llenar de agua para comprobar si se producía alguna fuga. Este paso nos dio bastante problema ya que el agua se fugaba por cualquier junta entre componentes. La solución a este problema fue desmontar las uniones donde se producía alguna fuga y, en caso de que fuese una unión "macho–hembra" añadimos más teflón. En los casos en los que las uniones fueran a presión, la solución fue añadir abrazaderas para ajustar más las tuberías o, añadir la cinta autosoldante que mencionamos en el Anexo 4. Estos pasos los repetiríamos tantas veces como fuese necesario.

Una vez consiguiéramos que la estructura completa fuera estanca, ya podríamos situarla en su respectiva balda y pasar al montaje del siguiente depósito. Estos pasos los seguimos con todos los recipientes excepto con el tanque de retorno ya que este depósito no sería perforado, ya que el fluido salía directamente por la tapa con una bomba que lo impulsa directamente hacia arriba por medio de la tubería flexible.

Lo siguiente fue trabajar con los sensores. Para medir las cantidades de agua se contó con sensores de nivel sin contacto para el depósito de recarga y con básculas para los depósitos de almacenamiento, ventas y fugas. Respecto al tanque de recarga, sólo tuvimos que pegar los sensores con cinta aislante en 4 alturas diferentes con el fin de obtener diferentes volúmenes a la hora de producir descargas mediante una suma/resta de estas lecturas, como mencionamos en el apartado 4.2.2.

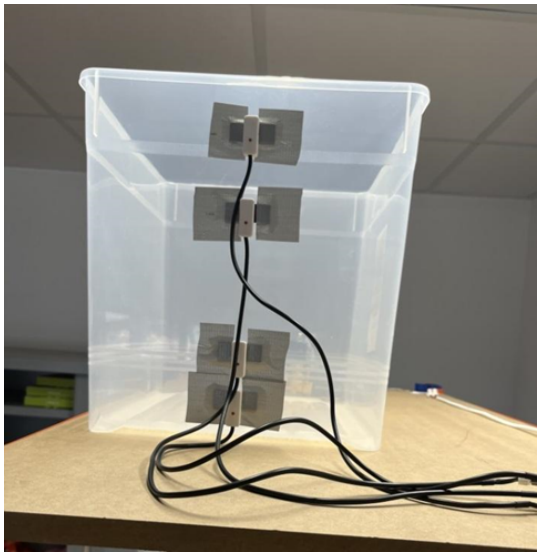


Figura 10.2. Sensores de nivel sin contacto del depósito de recarga

Para el resto de depósitos mencionados utilizamos las básculas. Para ello tuvimos que abrir las balanzas y soltar los 4 cables asociados a los pines E+, E-, S+ y S-. Estos 4 cables los conectamos directamente a sus correspondientes pines del módulo HX711 y, de estos módulos sacamos 4 cables más que irían directamente a los microcontroladores. De esta manera ya podríamos obtener la cantidad de fluido de estos depósitos con nuestro microcontrolador. Ya

preparadas las básculas, lo siguiente fue colocar sus correspondientes depósitos y componentes sobre unas tablas de madera y, estas tablas de madera, sobre las balanzas.

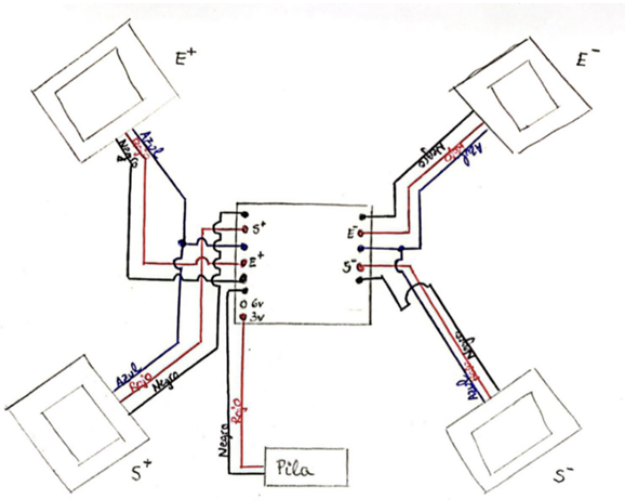


Figura 10.3. Configuración del interior de las básculas

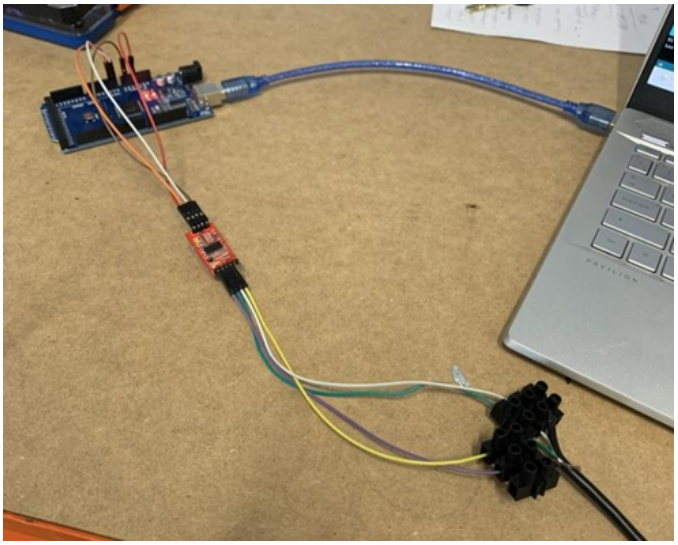


Figura 10.4. Configuración del conexionado de la báscula, módulo HX711 y microcontrolador

Llegados a este punto, ya teníamos montados todos los tanques con sus respectivos sensores sobre sus respectivas baldas, por lo que pasamos a perforar las baldas para permitir el paso del fluido de un depósito a otro. Teniendo ya todos los agujeros hechos, procedimos a situar la bomba peristáltica sobre el depósito de almacenamiento y a pasar una tubería del fondo del tanque de almacenamiento hasta la bomba y otra desde la bomba hasta el tanque de fugas.

Figura 10.5. Colocación de la bomba peristáltica

Con esto ya tendríamos montada toda la planta a falta de realizar las conexiones.

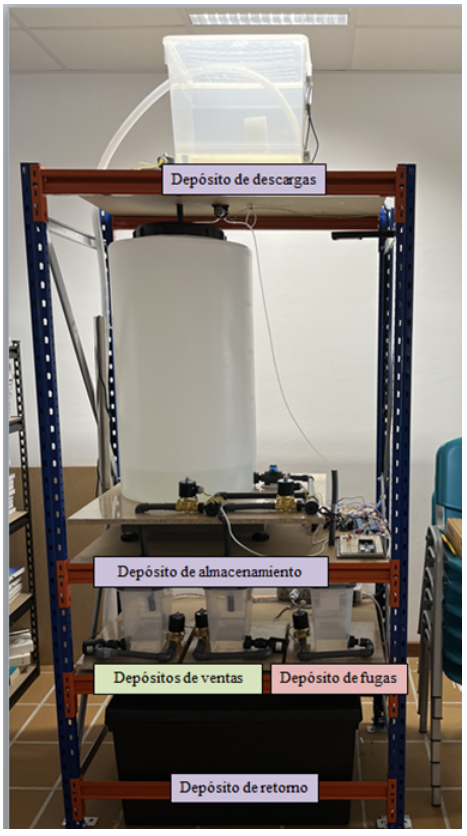


Figura 10.6. Prototipo físico de la planta

Por último, solo nos faltaría el conexionado de los sensores y actuadores. En esta etapa tendríamos que conectar los mencionados sensores y actuadores al módulo de 16 relés, al

microcontrolador y al driver del motor. Para ello comenzamos midiendo las distancias entre los componentes y la balda donde colocaríamos todos estos elementos electrónicos, la cual es la repisa donde se encuentra el depósito de almacenamiento. A la hora de medir estas distancias, tuvimos en cuenta que los cables se pasarían por detrás de los tanques, es decir, al fondo de los estantes y, además, estos cables irían en línea recta y pasarían de una repisa a otra por las barras que se encuentran en las esquinas. Teniendo las distancias de todos los componentes, lo siguiente fue empezar a conectar cables. Las bombas y válvulas irían conectadas al módulo de 16 relés. Al activarse a la vez la bomba y válvula de un mismo depósito, éstas irían conectadas juntas a un mismo relé. La bomba peristáltica iría conectada al driver del motor y, este a su vez al microcontrolador. Y los sensores, ya sean las básculas y sensores de nivel, irían conectados al Arduino. A su vez, el módulo de 16 relés y el driver del motor estarían alimentados por la fuente de 12 voltios y el microcontrolador por la fuente de 5 voltios.

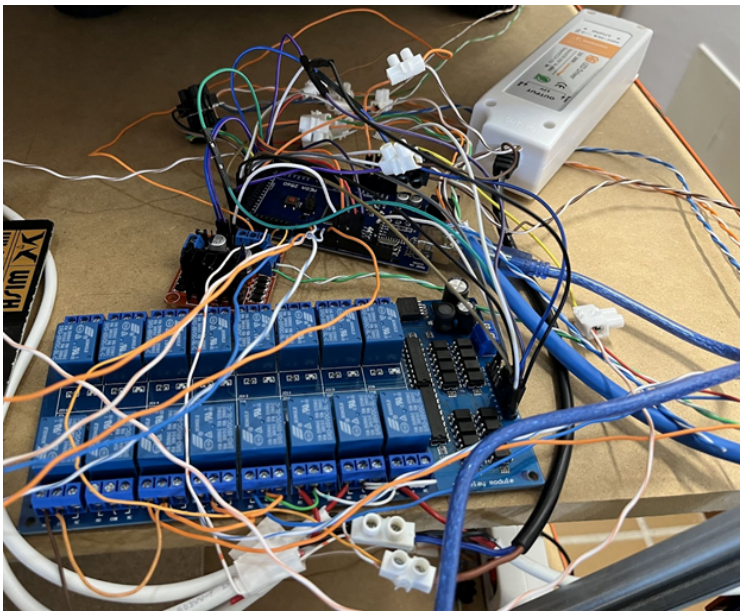
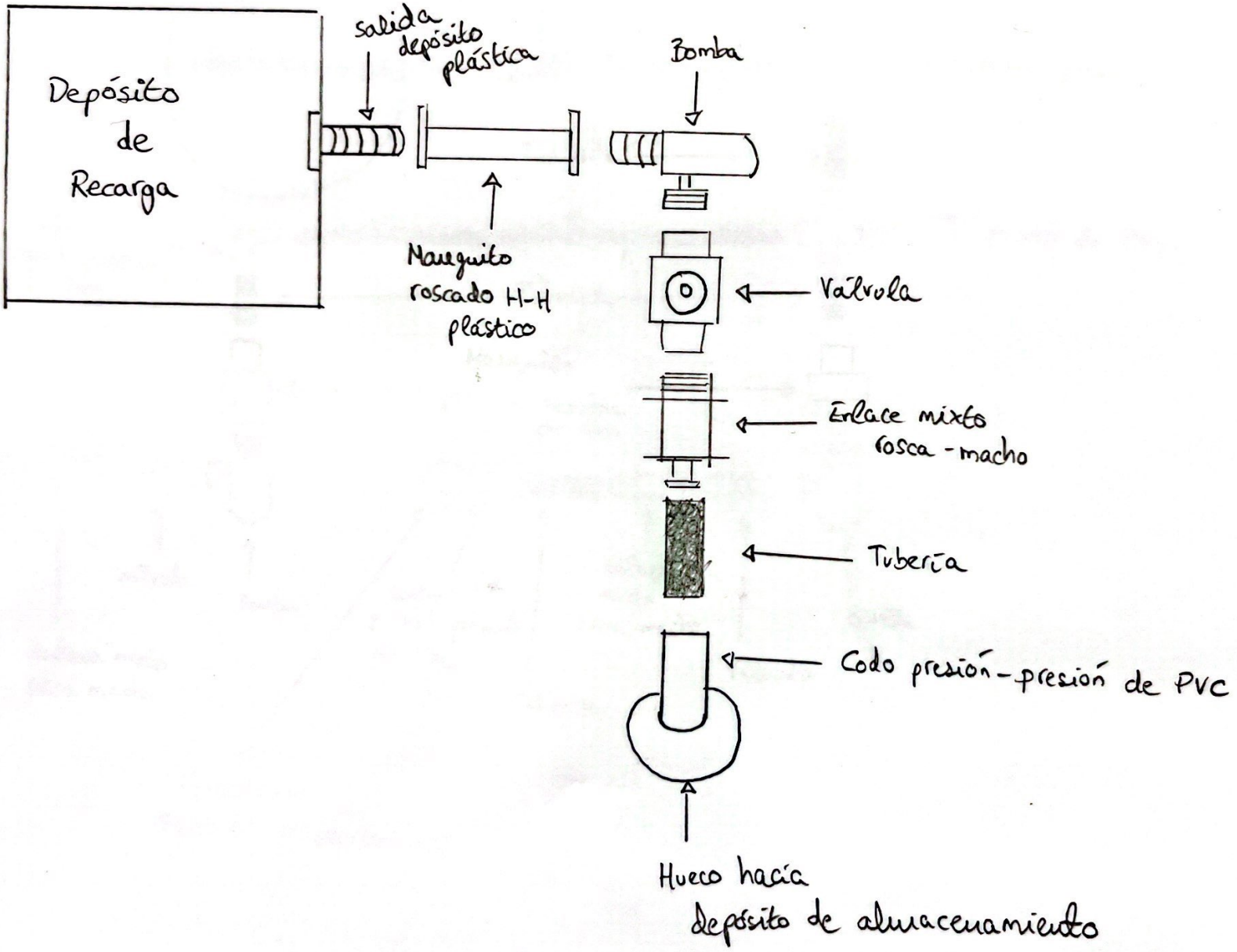


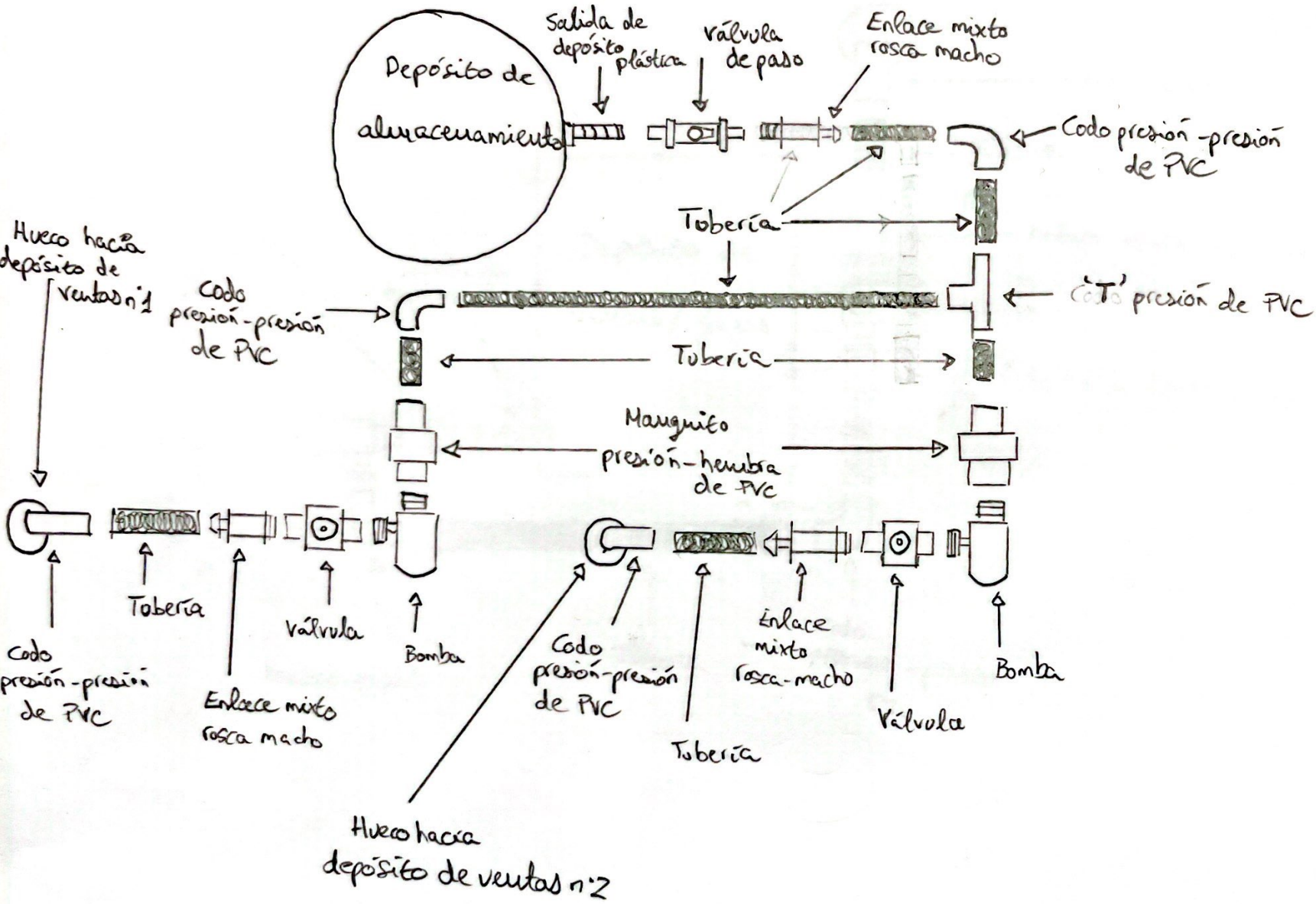
Figura 10.7. Conexiones del módulo de 16 relés, microcontrolador y driver del motor

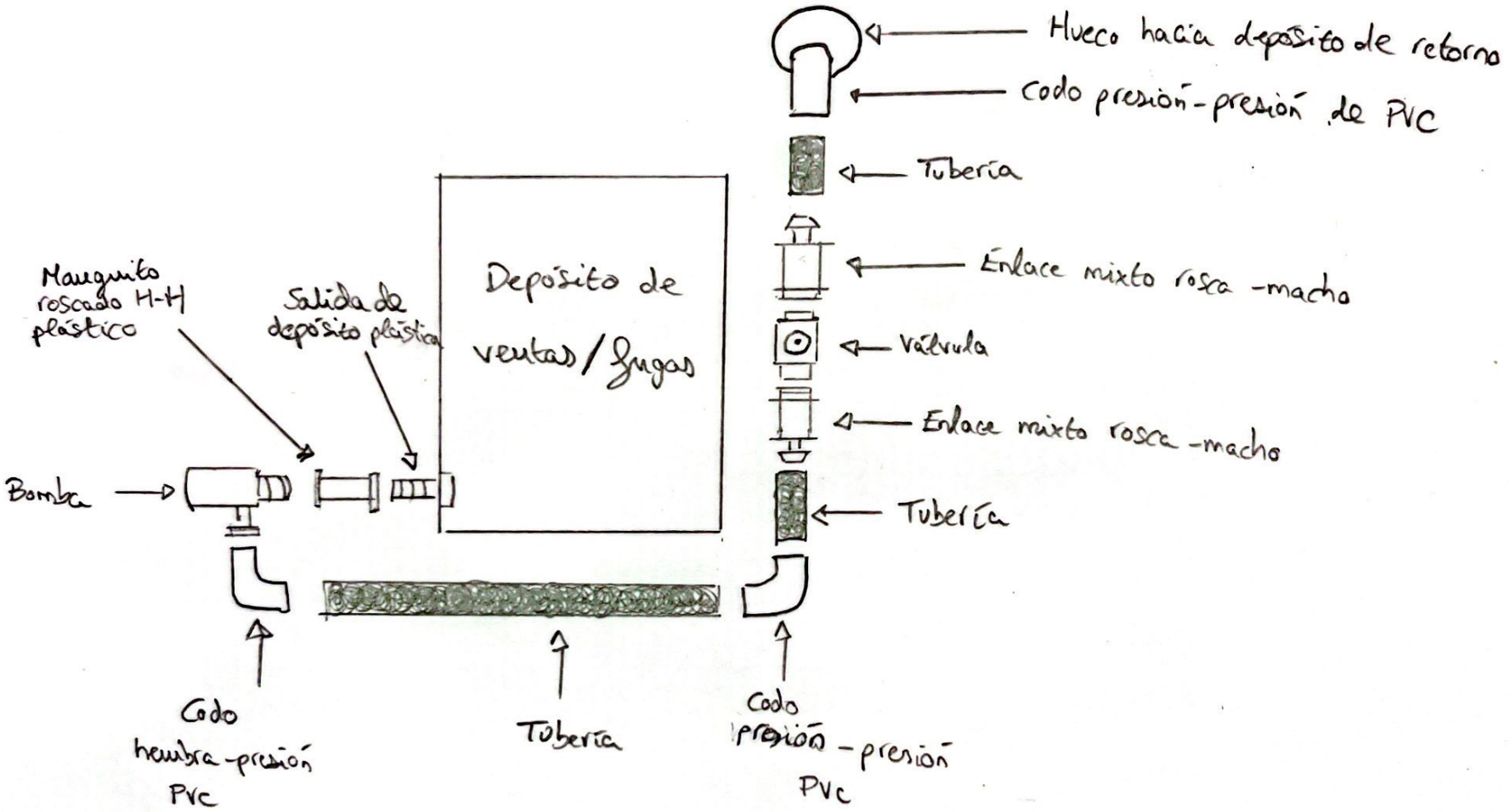
Por último, ya montado el prototipo, empezamos a meter agua en la planta y pasamos a depurar el código de la controladora, adaptándolo a nuestro proyecto ya que presentaba cambios con respecto al proyecto anterior. Lo siguiente fue ir probando partes del código para comprobar su funcionamiento y, posteriormente, empezar a hacer los diferentes experimentos.

Anexo 4

Croquis y listado del material


Depósito de Recarga
salida depósito plástica
Bomba
Manguito roscado H-H plástico
Válvula
Enlace mixto rosca - macho
Tubería
Codo presión - presión de PVC
Hueco hacia depósito de almacenamiento


Depósito de almacenamiento
Salida de depósito plástica
válvula de paso
Enlace mixto rosca macho
Codo presión-presión de PVC
Tubería
Hueco hacia depósito de ventas nº1
Codo presión-presión de PVC
'T' presión de PVC
Tubería
Manguito presión-hembra de PVC
Codo presión-presión de PVC
Tubería
válvula
Bomba
Codo presión-presión de PVC
Enlace mixto rosca macho
Hueco hacia depósito de ventas nº2
Codo presión-presión de PVC
Tubería
Enlace mixto rosca-macho
Válvula
Bomba


Hueco hacia depósito de retorno
Codo presión-presión de PVC
Tubería
Enlace mixto rosca-macho
Válvula
Enlace mixto rosca-macho
Tubería
Manguito roscado H-H plástico
Salida de depósito plástico
Depósito de ventas/fugas
Bomba
Codo hembra-presión PVC
Tubería
Codo presión-presión PVC

MATERIAL

- 5 salidas de depósitos de plástico
- 6 electroválvulas
- 7 bombas
- 1 bomba peristáltica
- 4 manguitos roscados hembra-hembra de plástico
- 10 enlaces mixto rosca-macho de plástico
- 11 codos presión-presión de plástico
- 1 válvula de paso
- 1 ‘T’ de plástico
- 2 manguitos presión-hembra de plástico
- 3 codos hembra-presión de plástico
- Teflón
- Pegamento industrial
- Abrazaderas
- Tubería flexible PVC hidrotubo 20 x 16
- Cinta autosoldante
- 4 tablas de madera para bases de depósitos
- 4 módulos HX711

11. BIBLIOGRAFÍA

[1] https://alasestaciones.com/funcionamiento-gasolinera/

[2] https://noticias.autocosmos.com.mx/2011/09/01/como-funciona-la-bomba-de-una-gasolinera

[3] https://www.tsg-solutions.com/es/detectores-de-fugas/

[4] https://industrialphysics.com/es/base-de-conocimientos/articulos/comprobadores-de-fugas-por-caida-de-presion/

[5] https://www.evoprotect.net/catalogo/depositos-accesorios/sondas-de-nivel-y-accesorios/detector-de-fugas-para-tanque-de-doble-pared/

[6] https://toralin.es/liquido-universal-uv-de-deteccion-de-fugas-100-ml.html

[7] https://fullgas.es/conciliacion-estadistica-de-inventario/

[8] "Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos" por Luis Arriaga Campos. <https://riull.ull.es/xmlui/handle/915/20640>

[9] "Adquisición y tratamiento de datos de un sistema de simulación de fugas en depósitos de combustible" por Eduardo Miguel Gastón Quesada [Pendiente de publicación].