INSTITUTO TECNOLÓGICO SUPERIOR DE CAJEME

SISTEMA AUTOMATIZADO PARA CONTROL DE NIVEL DE REFRIGERANTE

TESIS

PARA OBTENER EL GRADO DE MAESTRO EN INGENIERÍA MECATRÓNICA

PRESENTA:

ING. RODOLFO EDUARDO GONZÁLEZ SALAS

DR. JOSÉ GUADALUPE CASTRO LUGO

Ciudad Obregón, Sonora, México AGOSTO 2024

ITESCA®
Instituto Tecnológico Superior de Cajeme

DICTAMEN DE LIBERACIÓN Y APROBACIÓN

F02PSA16.04

Fecha
28/1/2025

No. de Registro
MIM-075

Mtro. Horacio Valenzuela Martínez
Responsable de Maestría en Ingeniería Mecatrónica
PRESENTE

A través de este documento, me permito comunicarle que el estudiante ***ING. RODOLFO EDUARDO GONZÁLEZ SALAS*** cuyo proyecto de obtención de grado

SISTEMA AUTOMATIZADO PARA CONTROL DE NIVEL DE REFRIGERANTE

Cumple satisfactoriamente, con los Lineamientos para la Operación de Estudios de Posgrado en el Sistema Nacional de Educación Superior Tecnológica, por lo que ha sido aceptado para la obtención del grado:

Maestro en Ingeniería Mecatrónica

Sin más por el momento, quedo de Usted.

Dr. José Guadalupe Castro Lugo
Nombre y Firma del Director (a) de proyecto
Director (a) de proyecto de obtención de grado

24/01/25
Fecha

DICTAMEN DE ACEPTACION - COMISIÓN REVISORA

PRESIDENTE	**Dr. José Guadalupe Castro Lugo** (Nombre)	(Firma)	24/01/25 (Fecha)
MIEMBRO 1	**Dr. Juan Enrique Palomares Ruiz** (Nombre)	(Firma)	24/01/25 (Fecha)
MIEMBRO 2	**Dr. Flavio Muñoz Beltrán** (Nombre)	(Firma)	24/01/25 (Fecha)

AGRADECIMIENTOS

Culminar esta etapa de mi formación profesional como Maestro en Ingeniería en Mecatrónica es un logro que me llena de orgullo y satisfacción. Sin embargo, este camino no fue fácil y no hubiera sido posible sin el apoyo incondicional de personas muy importantes en mi vida, quienes me guiaron y me dieron ese empujoncito extra cuando mi ánimo flaqueaba ante los retos o cuando mis actividades me distraían de concluir este proyecto.

En primer lugar, quiero agradecer infinitamente a mi esposa Zaira y mis hijos Montserrat y Boris, cuyo amor y apoyo inquebrantables fueron mi mayor motivación en este largo camino. En especial a mi esposa, por ser mi roca y recordarme siempre que rendirme no era una opción. Este logro también es suyo.

A mi padre Rodolfo, cuyo apoyo inquebrantable y ejemplo de perseverancia han sido una luz constante en mi camino. A mi madre Alma que, aunque nos dejó antes de ver este sueño concretado, su amor y fe en mí fueron el cimiento de este logro. A mi hermana Denise, por su cariño, aliento y por ser compañera en este recorrido.

A mis profesores y mentores, quienes compartieron conmigo su sabiduría y experiencia, guiándome en el desarrollo de esta investigación. Su dedicación y compromiso fueron fundamentales para mi crecimiento profesional y personal.

A mis compañeros de estudio, con quienes compartí horas de estudio y experiencias enriquecedoras. Su apoyo y camaradería hicieron de este proceso un camino más ameno y estimulante.

Finalmente, quiero agradecer a mis compañeros de trabajo y amigos. Gracias por ser mi guía, sobre todo, por recordarme siempre que la meta estaba al alcance. Su apoyo incondicional, sus palabras de aliento y sus dosis de humor fueron un ingrediente esencial para llegar hasta aquí.

A todos ellos, mi más sincero agradecimiento por ser parte fundamental de este logro.

RESUMEN

Las tareas fáciles y repetitivas en el lugar de trabajo pueden ocupar hasta el 20% de nuestro tiempo en lo que podríamos llamar "tareas manuales repetitivas basadas en reglas". Este proyecto demuestra una forma sencilla de automatizar un proceso, centrándose en la parte lógica del sistema en lugar de la parte física. La principal tarea repetitiva es rellenar un contenedor con agua hasta un cierto nivel. Por lo tanto, considerando que esta tarea debe realizarse cada vez que se va a operar la máquina, podríamos concluir fácilmente que esta operación es una excelente candidata para la automatización. La tarea secundaria sería mezclar agua y lubricante (refrigerante) en una proporción específica basada en las especificaciones de la máquina. Esta tarea, que se lleva a cabo cada vez que realizamos una limpieza completa del refrigerante, lo cual ocurre cada varios meses, no es una tarea repetitiva, pero también se puede automatizar con la misma lógica y los mismos productos físicos que la operación de rellenado. Como el presupuesto siempre es una gran limitante, este proyecto pretende utilizar electrónica sencilla y asequible, algunas de las cuales se pueden comprar en paquetes como kits de DIY o kits de proyecto. El dispositivo terminó proporcionando una buena detección de nivel y una fácil instalación en varios tipos de contenedores. Después de investigar y probar, surgieron nuevas ideas, pero también se plantearon más preguntas sobre qué más se le puede agregar al dispositivo como futuras actualizaciones y qué nuevas características se podrían incluir en las nuevas versiones.

ABSTRACT

Easy and repetitive tasks on the workplace can take up to 20% of our time in what we could call “manual repetitive rule-based tasks”. This project demonstrates an easy way of automating a process, with a focus primarily on the logic part of the system rather than the physical part. The primary repetitive task is refilling a container with water up to a certain level. So, considering that this task needs to be performed every time the machine is going to be operated, we could easily conclude that this operation is a great candidate for automation. The secondary task would be to mix accordingly water and lubricant (coolant) to a specific proportion based on the machine’s specifications. This task being carried out every time we have a complete coolant clean-up, which occurs every several months is not a repetitive task, but can also be automated by the same logic as and physical products as the refill operation. As budget is always a big hold-up, this project intends to make use of easy and affordable electronics, some which can be bought on bundle as DIY Kits or Project Kits. The device ended up providing a good level sensing and easy installation on various types of containers. After researching and testing, new ideas came to mind, but also raised more questions as to what else can be added to the device as future upgrades, and what new features could be included on new versions.

ÍNDICE

I.INTRODUCCIÓN 7
1.1 Antecedentes.....7
1.2 Problema.....7
1.3 Justificación.....8
1.4 Objetivos.....13
1.4.1 Objetivo General.....13
1.4.2 Objetivos Específicos 13
1.5 Alcances y Limitaciones.....13
II. FUNDAMENTO TEÓRICO 15
Máquinas de estado finito.....15
Estructuras en lenguaje de Arduino IDE. 16
Módulo de sensor ultrasónico HC-SR04.....18
Velocidad del sonido 19
Cálculo de la distancia.....19
Ultrasonido. 19
Encoder rotativo 20
Arduino.....21
Volumen de un líquido 21
Volumen de un ortoedro.....21
Principio de Arquímedes 21
El refractómetro.....22
Medición de la concentración de los lubricantes refrigerantes 22
III. DESARROLLO.....23
3.1 Determinar dispositivos electrónicos y software a utilizar.....23
3.2 Bosquejos de elementos de control. 23
3.3 Desarrollo del prototipo de los elementos electrónico y de software.....23
3.4 Pruebas de funcionamiento y mejoras.....24

IV. RESULTADO 25
4.1 Determinar dispositivos electrónicos y software a utilizar. 25
4.2 Bosquejos de elementos de control. 30
4.21 Bosquejos de circuitos eléctricos. 30
4.3 Desarrollo del prototipo de los elementos electrónico y de software. 32
4.3.1 Desarrollo del sistema lógico del proyecto 32
4.3.2 Desarrollo del contenedor del sistema electrónico 38
4.4 Pruebas de funcionamiento y mejoras 39
4.4.1 Navegación de menú 39
4.4.2 Prueba de sensor de distancia 41
4.4.3 Mejoras y propuestas 43
V. CONCLUSIONES Y RECOMENDACIONES 45
5.1 Conclusiones 45
5.2 Recomendaciones 45
VI. ANEXOS 46
VII. BIBLIOGRAFÍA 69

I.INTRODUCCIÓN

1.1 Antecedentes

La Universidad Tecnológica de Nogales es una institución de educación superior cuya actividad se centra en formar técnicos superiores universitarios e ingenieros que puedan desarrollar y aplicar en forma práctica tecnologías, metodologías y herramientas que requiera el sector productivo y de servicios.

Se ofrecen varias opciones como lo es la carrera de Manufactura Aeronáutica área Maquinado de Precisión, donde se prepara a futuros profesionistas para coordinar y desarrollar procesos de manufactura de piezas de aeronáutica mecanizadas a partir de documentación técnica de ingeniería, métodos y técnicas de fabricación, herramientas de planeación y supervisión por medio de la normatividad aplicable.

El área de interés para este proyecto se enfoca en los talleres de maquinado por control numérico por computadora (CNC), específicamente en un equipo de maquinado FANUC de la serie Robodrill Alpha-DiA5 con controlador FANUC. Este equipo cuenta con un contenedor de refrigerante con bombeo automático para facilitar el control de temperaturas de maquinado, y es en este contenedor donde se pretende utilizar un sistema automatizado programable para el control de llenado y mezcla del refrigerante.

El procedimiento utilizado actualmente para la revisión de la calidad de refrigerante para este equipo se hace de forma manual, se utilizan tiras para probar el nivel de pH del refrigerante. La concentración de la mezcla del refrigerante se mide por medio de un refractómetro manual. Y el nivel de refrigerante se monitorea a través de un indicador de nivel de vidrio, que por el lugar donde ha sido instalado el equipo, no queda fácilmente visible para el usuario.

1.2 Problema

El centro de maquinado CNC FANUC Robodrill Alpha-DiA5, con el cual cuenta la Universidad Tecnológica de Nogales está instalado en laboratorio 1-5 donde por su infraestructura, el clima afecta directamente los equipos, esto provoca que el agua contenida en la mezcla del refrigerante tienda a evaporarse rápidamente, aunado a esto el equipo no es

utilizado constantemente como en el sector industrial, por lo que no se hace el proceso de llenado y mezcla de refrigerante de manera continua. Contar con una mezcla incorrecta o falta de refrigerante puede provocar daño a la herramienta de corte, oxidación en los elementos mecánicos del equipo y maquinados de baja calidad. Por otra parte, realizar el llenado y mezclado de refrigerante consume bastante tiempo como para ser considerado dentro del tiempo destinado a realizar la práctica de maquinado, por eso nace la idea de desarrollar un prototipo de sistema de llenado y mezclado de refrigerante.

1.3 Justificación

Dentro de las actividades de mantenimiento preventivo de un equipo de maquinado CNC Robodrill Alpha-DiA5, se encuentra el monitoreo del estado del refrigerante para maquinado. Existen varios factores que debemos considerar para mantener una buena calidad de refrigerante: concentración de mezcla del refrigerante, nivel de refrigerante y nivel de pH. No llevar un control de estas variables puede ocasionar daños a herramienta (Figura 1), al material maquinado y al equipo en general (Figura 2).



Figura 1 - Daño en herramienta



Figura 2 - Daño en maquinaria

El refrigerante utilizado en este equipo es TRIM E206, un refrigerante de aceite soluble utilizado para el maquinado de materiales ferrosos y no ferrosos. La concentración recomendada en la hoja de datos del producto (Figura 3) es del 3% al 10% de concentración en agua, sin embargo, durante la puesta en marcha del equipo, el proveedor recomendó hacer una proporción de 1 a 20 entre refrigerante y agua para trabajo general.

Si contamos con una mezcla de refrigerante muy concentrada (>10%), ésta se tornará gruesa lo que podría ocasionar que algún dispositivo de bombeo pueda taparse o forzarse de más y ocasionar daños al mismo. El costo aproximado de una bomba para refrigerante oscila los $3000 más accesorios y mano de obra para su instalación. Una mezcla muy poco concentrada (<3%) provocaría que se genere óxido en las partes metálicas y fomente la aparición de bacterias. Estas bacterias destruyen el lubricante y en caso de tener aditivos anticorrosión también los deshabilita generando un ambiente acídico que contribuye al desgaste del equipo, sobre todo en pinturas y sellos de goma que estén en contacto con el refrigerante. Se recomienda tener un pH alrededor de 8.6 ya que normalmente los refrigerantes son ligeramente alcalinos. Tener una concentración óptima de refrigerante y agua (3% - 10%) nos garantizará maquinados de mayor calidad, pero sobre todo mayor vida para la herramienta y el equipo.

Concentraciones recomendadas para metalurgia:	
Trabajo ligero	3.0% - 6.0%
Trabajo moderado	6.0% - 8.0%
Trabajo pesado	8.0% - 10.0%
Rango de concentración por diseño	3.0% - 10.0%

Figura 3 - Tabla de concentraciones de refrigerante

El proyecto pretende excluir de manera parcial al recurso humano que está realizando el proceso de llenado y mezcla de refrigerante de forma manual, ya que es un proceso que consume bastante tiempo y se pretende que el proceso de mezcla pueda ejecutarse de manera autónoma y el proceso de llenado cada vez que sea necesario utilizar el equipo.

En la universidad existen varios equipos de maquinado convencional y CNC, en los cuales agregar el mismo tipo de sistema de llenado y mezcla de refrigerantes podría generar una gran ayuda para el laboratorista encargado de dar mantenimiento al equipo y de la misma manera poder mantener en buen funcionamiento el equipo. Cabe mencionar que la cantidad de tiempo que estos equipos están en uso en la universidad dista mucho de la cantidad de tiempo que equipo similares se encuentran en uso en la industria.

Existen tres métodos de mezcla de refrigerantes para equipos de maquinado, estos son: manual, semiautomático y automático.

El método de mezcla manual (Figura 4), el cual es el método utilizado en la universidad, consiste en realizar la mezcla en un depósito controlando la dosificación de forma manual utilizando contenedores graduados. Este método es el más simple y económico, sin embargo, puede provocar variaciones en la calidad de mezcla.



Figura 4 - Método de mezcla manual

El método semiautomático consiste en utilizar un sistema de bombeo (Figura 5) para mezclar el refrigerante. Este método es más preciso, pero también más costoso. Un sistema de bombeo proporcionado puede llegar a tener un costo de hasta $5000 USD. En él se ajustan los reguladores de las bombas para obtener la proporción deseada. Existen otros métodos semiautomáticos para la mezcla, uno de ellos es el sistema tipo Venturi (Figura 6) que puede llegar a tener un costo de hasta $1000 USD. En este tipo de sistema se requiere tener una presión constante de agua alrededor de 35-75 psi para poder funcionar correctamente, lo que obliga a tener un sistema de bombeo.



Figura 5 - Sistema de bombeo



Figura 6 - Sistema semiautomático Venturi

Por último, tenemos el método automático de mezcla (Figura 7), que consiste en instalar un sistema de dosificación en el depósito de refrigerante y configurar el sistema para obtener la proporción deseada. Este método es el mas preciso y fiable, pero también es el mas costoso de los tres llegando a costar hasta $12000 USD por sistema más un costo extra de $2000 USD por instalación.



Figura 7 - Sistema automático

Desventajas de utilizar sistemas semi automáticos y automáticos del mercado:

- Al ser una institución educativa pública, no existe una relación costo/beneficio al realizar una inversión adquiriendo un sistema nuevo, ya que no se están generando ganancias por operar el equipo de maquinado. Considerando los costos de equipos automáticos de dosificación y semiautomáticos de bombeo proporcionado en el mercado, sería muy difícil descargar recursos estatales o federales justificando estas compras.
- Dentro de los sistemas más económicos como el sistema Venturi, es necesario contar con cierta presión de agua durante el proceso de mezclado y esto conlleva a la necesidad de instalar un sistema de bombeo aparte, el cual garantice los 35-75 psi necesarios para su operación, incrementando el costo del sistema.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un prototipo de un sistema automatizado para el control de nivel y mezcla de refrigerante para maquinado, de un equipo RoboDrill D14-α utilizando control de nivel y control de mezcla de refrigerante.

1.4.2 Objetivos Específicos

- Diseñar el sistema electrónico del prototipo que se pretende utilizar con dispositivos de fácil adquisición.
- Diseñar la aplicación de software para realizar el monitoreo, mezcla y control de nivel de líquido refrigerante.

1.5 Alcances y Limitaciones

El prototipo que se pretende realizar tiene como principales objetivos realizar una mezcla correcta y mantener el nivel de refrigerante del equipo CNC. Pero su alcance esta limitado a monitorear niveles y proporcionar la manera de activar y desactivar actuadores externos para proveer los líquidos ya sea por bombeo, por gravedad o por cualquier otro medio disponible. El dispositivo tendrá la capacidad de ser configurado para establecer un máximo de líquido dentro del contenedor, y dos proporciones de líquido, uno para el refrigerante y otro para el agua.

En base al funcionamiento del equipo FANUC de la serie RoboDrill Alpha-DiA5 se determinan las siguientes limitaciones para el diseño:

- El prototipo será diseñado para ser instalado en el contenedor de refrigerante del equipo.
- El centro de control del prototipo y el posicionamiento de sensores no deberán interferir con los movimientos de bancada, husillo, el cambiador de herramienta o el sistema de bombeo.
- El prototipo funcionará de forma independiente al centro de maquinado, no existirá comunicación ni transferencia de datos entre el centro de maquinado y el prototipo.

Por limitaciones en presupuesto para este proyecto se determinan las siguientes limitaciones:

- El equipo electrónico deberá ser diseñado utilizando una placa Arduino UNO y shields para Arduino.
- El control de niveles y mezcla deberá ser controlado por un solo sensor, dos actuadores y software.

II. FUNDAMENTO TEÓRICO

Máquinas de estado finito.

Definición de máquina de estado finito

Una máquina de estado finito (FSM) es un modelo matemático que describe el comportamiento de un sistema reactivo mediante un número determinado de estados y un número determinado de transiciones entre dichos estados.

Elementos de una máquina de estado finito

Una máquina de estado finito se compone de los siguientes elementos:

- Estados: Los estados representan las condiciones que puede tomar la máquina en el tiempo.
- Transiciones: Las transiciones representan los cambios de estado que puede realizar la máquina.
- Entradas: Las entradas representan los estímulos que puede recibir la máquina.
- Salidas: Las salidas representan las respuestas que puede generar la máquina.
- Funcionamiento de una máquina de estado finito

El funcionamiento de una máquina de estado finito se puede describir de la siguiente manera:

- La máquina comienza en un estado inicial.
- La máquina recibe una entrada.
- La máquina se transfiere al siguiente estado en función de la entrada recibida.
- La máquina genera una salida en función del estado actual.
- El proceso se repite hasta que la máquina recibe una entrada que la lleva a un estado final.

Máquinas de estado finito para diseño de menús de aplicaciones informáticas.

Una máquina de estado finito para menús puede tener los siguientes estados:

- Estado inicial: El estado inicial es el estado en el que se encuentra la máquina cuando el menú se muestra por primera vez.
- Estado de selección de opción: El estado de selección de opción es el estado en el que se encuentra la máquina cuando el usuario está seleccionando una opción del menú.

- Estado de confirmación de selección: El estado de confirmación de selección es el estado en el que se encuentra la máquina cuando el usuario confirma la selección de una opción.
- Estado de finalización: El estado de finalización es el estado en el que se encuentra la máquina cuando el usuario cierra el menú.

La máquina de estado finito puede tener las siguientes transiciones:

- Transición de entrada: La transición de entrada se produce cuando el usuario selecciona una opción del menú.
- Transición de confirmación: La transición de confirmación se produce cuando el usuario confirma la selección de una opción.
- Transición de salida: La transición de salida se produce cuando el usuario cierra el menú.

Estructuras en lenguaje de Arduino IDE.

Definición de estructura

Una estructura es un tipo de dato compuesto que permite agrupar datos de diferentes tipos en una sola variable.

Uso de estructuras

Las estructuras se utilizan en Arduino para organizar datos de manera eficiente y para simplificar el código.

Por ejemplo, se pueden utilizar estructuras para almacenar información sobre un sensor, un objeto o un usuario.

Declaración de estructuras

Para declarar una estructura en Arduino, se utiliza la siguiente sintaxis:

```
struct nombre_estructura {
  tipo_dato variable1;
  tipo_dato variable2;
  ...
};
```

Por ejemplo, la siguiente declaración declara una estructura llamada persona que contiene dos variables:

```
struct persona {
  char nombre[20];
  int edad;
};
```

Creación de variables de estructura

Para crear una variable de tipo estructura, se utiliza la siguiente sintaxis:

```
nombre_estructura variable;
```

Por ejemplo, la siguiente declaración crea una variable de tipo persona llamada juan:

```
persona juan;
```

Acceso a variables de estructura

Para acceder a las variables de una estructura, se utiliza la siguiente sintaxis:

```
variable.variable;
```

Por ejemplo, la siguiente declaración imprime el nombre de la persona juan:

```
Serial.println(juan.nombre);
```

Ventajas de usar estructuras

Las estructuras ofrecen las siguientes ventajas:

- Organización de datos: Las estructuras permiten organizar datos de diferentes tipos en una sola variable. Esto facilita la lectura y el manejo de los datos.
- Simplificación del código: Las estructuras pueden simplificar el código al permitir agrupar datos relacionados.
- Eficiencia: Las estructuras pueden mejorar la eficiencia del código al reducir el número de variables necesarias.

Desventajas de usar estructuras

Las estructuras ofrecen las siguientes desventajas:

- Complejidad: Las estructuras pueden aumentar la complejidad del código si no se usan correctamente.
- Requisitos de memoria: Las estructuras requieren más memoria que las variables simples.

Módulo de sensor ultrasónico HC-SR04

Éste módulo (Figura 8) cuenta con un emisor y un receptor ultrasónico, así como un circuito de control integrado en el mismo paquete. Tiene un rango de funcionamiento de 2 cm hasta 400 cm con una precisión de 3mm. Emite un ultrasonido en el rango de los 40 Khz. Cuenta con 4 pines que son tierra, voltaje (+5v), emisor y receptor. El emisor y receptor manejan valores digitales lo que facilita la interacción con las placas de Arduino. (elecfreaks.com)



Figura 8 - Módulo HC-SR04

A continuación, podemos observar los diagramas del comportamiento de los pulsos de nuestro módulo (Figura 9). Es necesario enviar un pulso por el pin de emisor con una duración de 10µs. Lo que ocasionará que se envía una señal que viajará a la velocidad del sonido, ésta al encontrarse con un objeto rebotará y será recibida por nuestro módulo. El tiempo transcurrido entre la emisión de la señal, su recepción y el valor de la velocidad del sonido, nos ayudarán a determinar la distancia. (howtomechatronics.com, n.d.)

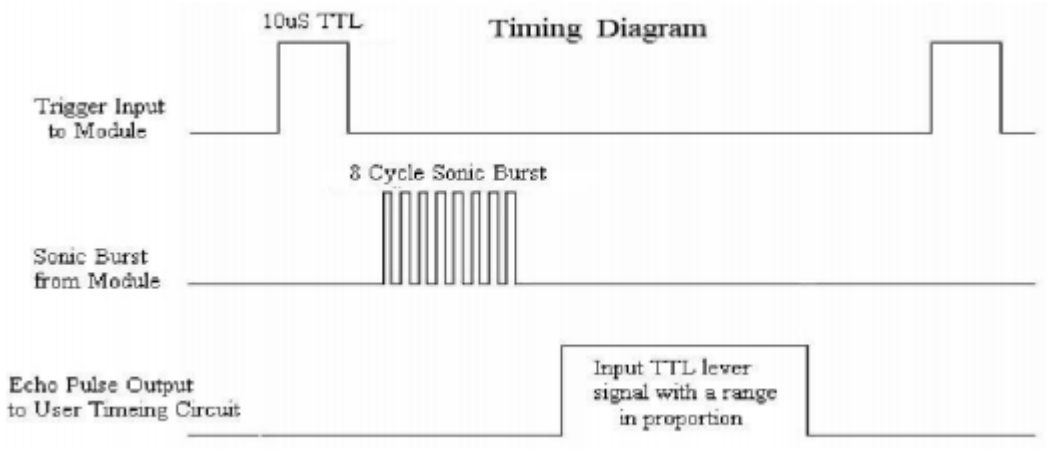


Figura 9 - Diagrama de comportamiento de pulsos

Velocidad del sonido

La velocidad del sonido es la velocidad en la que una vibración es transmitida a través de un medio. Para diferentes gases, la velocidad del sonido es considerada como una constante (Figura 10). (Zuckerwar, 2002)

Gas	Velocidad del sonido (m/s)
Argón	319
Helio	1007
Kriptón	221
Xenón	178
Hidrógeno	1270
Nitrógeno	349
Oxígeno	326
Dióxido de Carbono	267
Metano	446
Propano	258

Figura 10 - Tabla de velocidades del sonido en diferentes medios

$$La\ velocidad\ del\ sonido\ en\ el\ aire\ es\colon 343\ \frac{m}{s}$$

Figura 11 - Velocidad de sonido en el aire

Cálculo de la distancia

Para el cálculo de la distancia es necesario considerar que la onda de sonido debe viajar de ida y de vuelta. Por lo tanto, consideramos la siguiente fórmula.

$D = \frac{t_p * v_s}{2}$ donde:

D = distancia a calcular

tp = tiempo transcurrido entre el pulso generado del emisor y el pulso recibido.

vs = velocidad del sonido

Ultrasonido.

"El ultrasonido es una onda sonora cuya frecuencia supera el límite perceptivo del oído humano (es decir, el sonido no puede ser captado por las personas ya que se ubica en torno al espectro de 20.000 Hz)." (Pérez Porto & Gardey, 2013)

Encoder rotativo

Un encoder rotativo es un tipo de sensor de posición utilizado para determinar la posición angular de un eje rotativo. Genera una señal eléctrica, ya sea análoga o digital, de acuerdo al movimiento de rotación (Figura 12).

El encoder tiene un disco con contactos espaciados uniformemente que están conectados al pin común C, y dos contactos separador pin A y pin B.

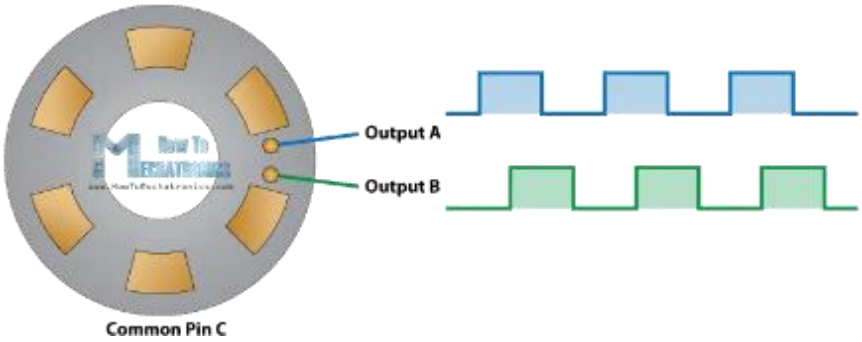


Figura 12 - Diagrama de un encoder rotativo

Cuando el disco gira paso por paso, los pines A y B harán contacto con el pin común C y se obtendrán dos señales de onda rectangular.

Cualquiera de ambas salidas puede ser utilizada para determinar la posición de rotación contando los números de pulso de la señal. Sin embargo, para determinar el sentido de rotación, es necesario considerar ambas señales al mismo tiempo.

Podemos notar que las señales de salida están desfasadas 90 grados con respecto a cada una (Figura 13). Si la rotación en sentido de las manecillas del reloj, la salida del pin A estará por delante de la salida del pin B.

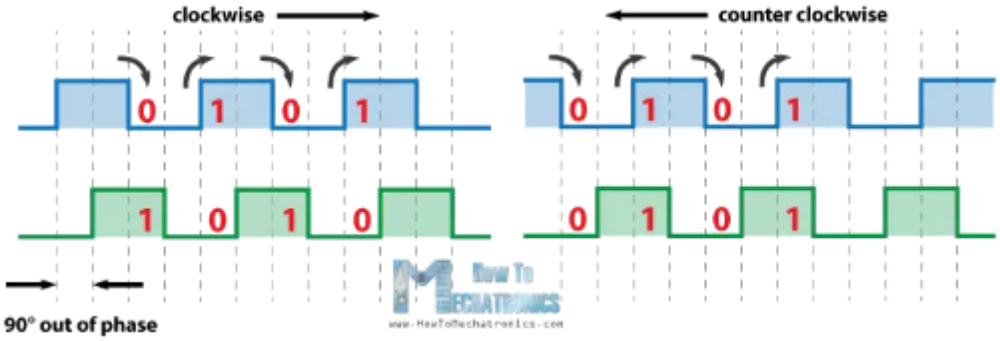


Figura 13 - Señales de salida de un encoder según sentido de giro

Arduino

"Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing." (Arduino, 2017)

Volumen de un líquido

El volumen de un líquido siempre será constante bajo condiciones de temperatura y presión constantes. (Pérez Montiel, 2011)

Volumen de un ortoedro

Para poder calcular el volumen de este objeto es necesario conocer su longitud, latitud y altura. Conociendo estos valores podemos utilizar su fórmula para calcular el volumen. (Mykhailo, 2011-2017)

$$v = a\ x\ b\ x\ h$$

Donde:

$$a = longitud$$
$$b = latitud$$
$$h = altura$$

Considerando (Pérez Montiel, 2011) y (Mykhailo, 2011-2017) podemos determinar que si un contenedor tiene las siguientes dimensiones, a = 12.5, b = 3 y h = 1.5

$$v = 12.5\ x\ 3\ x\ 1.5$$
$$v = 56.25\ unidades\ cúbicas$$

Principio de Arquímedes

El principio de Arquímedes establece que todo cuerpo sumergido en un fluido experimenta una fuerza ascendente igual al peso del fluido desalojado.

También se puede aplicar para demostrar el desplazamiento de un fluido por acción de un fluido más denso.

El refractómetro

El refractómetro de mano es un dispositivo óptico de pequeño tamaño que, mediante el principio de la refracción, es capaz de determinar la concentración de un sólido disuelto en un líquido (Figura 14).

Es decir, gracias a este aparato, podemos saber, por ejemplo, la cantidad de azúcar en una bebida o la cantidad de alcohol que contiene un líquido.

Medición de la concentración de los lubricantes refrigerantes

Para ello, se necesita un refractómetro bien manual o electrónico (Figura 14), un dispositivo que es capaz de detectar fácilmente variaciones en la concentración, anticipándose si existe problemas de calidad.

A modo de ejemplo, para medir la concentración de una emulsión primero hay que calibrar el refractómetro. Para ello, echamos unas gotas de agua en el dispositivo y girando el tornillo que hay en la parte superior pondremos la línea de corte en el 0 de la escala.

A continuación, consultamos la ficha técnica del refrigerante para asegurarnos que estamos usando del factor o índice de refracción correcto. Por lo tanto, si estamos usando un lubricante refrigerante que tiene de índice o factor de refracción, por ejemplo, 2 (este dato tiene que aportarlo el fabricante del producto en la ficha técnica del mismo) y en el refractómetro estamos viendo que el corte del producto está en la escala en el número 5,5, esto quiere decir que la concentración del producto es del 11% (2 x 5,5).

En el caso de los refractómetros manuales, la refracción tiene que compensarse respecto de la temperatura, porque los valores de medición refractométricos dependen considerablemente de la temperatura. Los valores de corrección de la temperatura tienen que computarse con el resultado de medición.

Sin embargo, con el refractómetro electrónico simplemente hace falta poner 1 – 2 gotas de muestra en el prisma de medición y pulsar un botón. En apenas unos segundos, se muestra digitalmente el resultado de medición ya con compensación de temperatura.



Figura 14 - Tipos de refractómetros (análogo y digital)

III. DESARROLLO

En la siguiente sección se describe a detalle la metodología utilizada para el desarrollo de este proyecto (Figura 15), donde podemos encontrar los pasos que se siguieron para llegar a nuestro resultado y los recursos utilizados para el desarrollo de este.

3.1 Determinar dispositivos electrónicos y software a utilizar.

El principal factor para determinar los dispositivos a utilizar fue en base a la disponibilidad de estos y su costo, ya que no se cuenta con presupuesto por parte de la empresa para la realización de este proyecto.

Se procedió a la toma de decisión tomado en consideración el costo, el mantenimiento y la precisión de los sistemas con este orden de relevancia.

- Costo.
- Mantenimiento.
- Precisión.

3.2 Bosquejos de elementos de control.

Una vez determinado el sistema de control que se pretende aplicar en el sistema, se realizaron los bosquejos de los circuitos de control por separado, primeramente, para verificar el funcionamiento correcto de los mismos y por último para verificar que la codificación sea la correcta.

3.3 Desarrollo del prototipo de los elementos electrónico y de software.

Durante esta etapa se procedió a la adquisición de los elementos electrónicos, su conexión, el desarrollo del software de control y el diseño de un contenedor para el dispositivo.

3.4 Pruebas de funcionamiento y mejoras.

Se realizaron pruebas de funcionamiento para evaluar la capacidad de detección y control de activación de los relevadores.

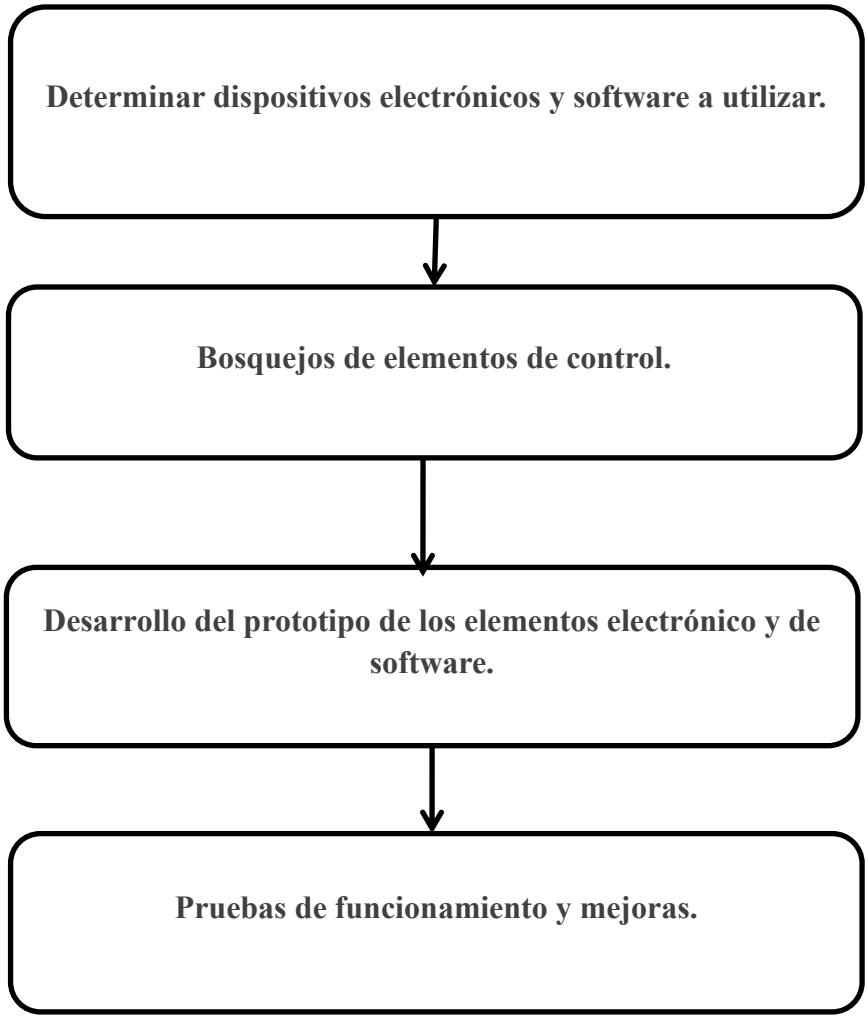


Figura 15 - Metodología para el desarrollo del proyecto

IV. RESULTADO

4.1 Determinar dispositivos electrónicos y software a utilizar.

Pantalla LCD 16x2.

Seleccionamos la pantalla LCD 16x2 (Figura 17) que viene incluido en los paquetes de placas Arduino. Será necesario visualizar menús y datos en pantalla, y la pantalla LCD cumple con el requisito de costo con un precio rondando los $150, son fáciles de conseguir y relativamente sencillos de reemplazar en un circuito. Se seleccionó el modelo que no cuenta con la interfaz I2C (Figura 16).

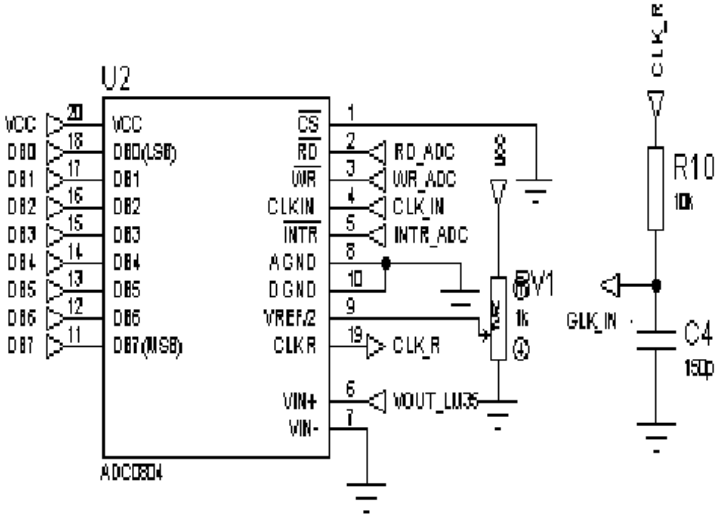


Figura 16 - Esquemático de una pantalla LCD 16x2

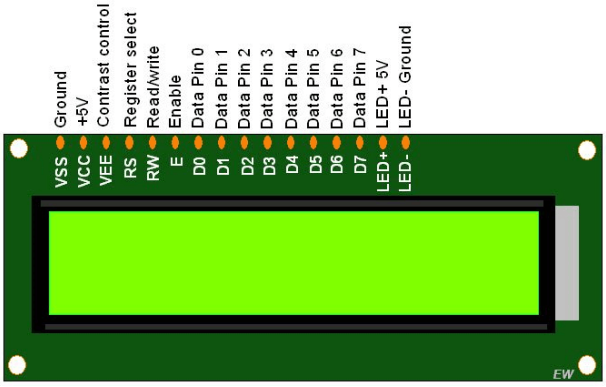


Figura 17 - Pin layout de una pantalla LCD 16x2

Sensor ultrasónico HC-SR04

Este sensor será el encargado de monitorear el nivel de líquido dentro del contenedor (Figura 18, Figura 19). Se seleccionó este sensor por recomendación de varias personas que lo han utilizado para sensar niveles de líquidos con sensores ultrasónicos, el único inconveniente es que no un sensor a prueba de agua. Estos sensores rondan los $50 por unidad, y podemos encontrar paquetes de hasta 10 unidades por alrededor de $300.

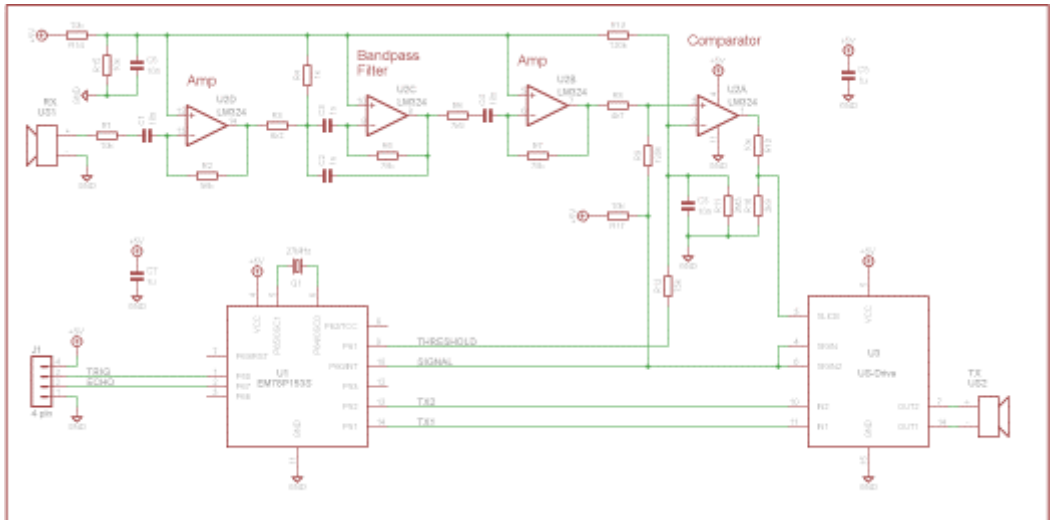


Figura 18 - Esquemático de un sensor HC-SR04



Figura 19 - Módulo HC-SR04

Relevadores.

Serán necesarios dos relevadores para controlar los actuadores para llenado, la mayoría de los actuadores que podrían realizar el trabajo de bombeo o de abrir y cerrar válvulas funcionan con más de 5v, la placa Arduino no podrá proporcionar este voltaje, por lo tanto, será necesario controlar el encendido y apagado por medio de relevadores. Los shields de Arduino con relevadores (Figura 21) se encuentran protegidos por medio de opto-acopladores, de esta

manera nos ayuda a proteger nuestra placa Arduino (Figura 20). Para nuestro proyecto se utilizará un módulo para placa Arduino de 4 relevadores, su costo ronda los $100.

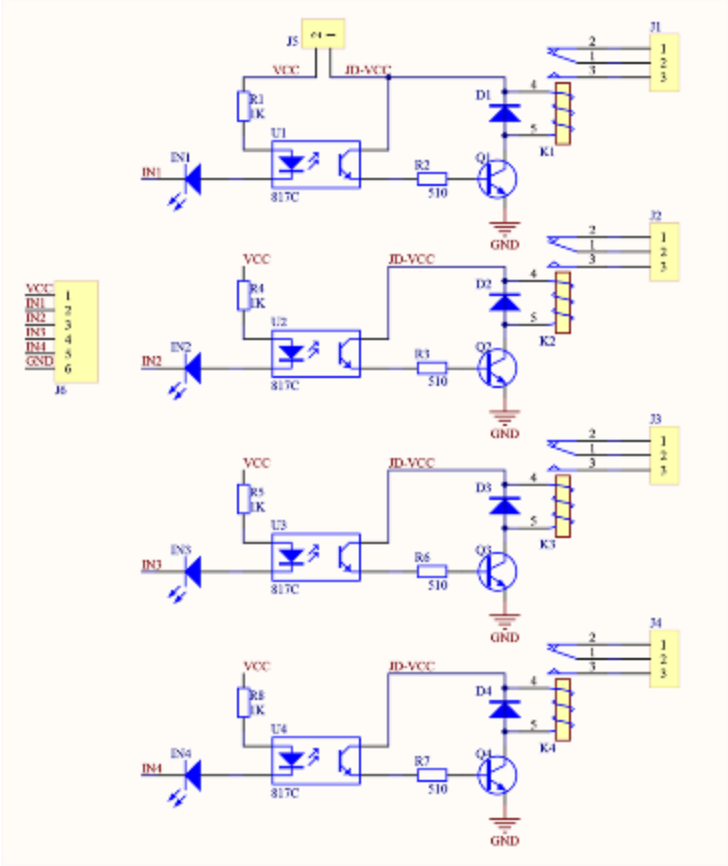


Figura 20 - Esquemático de shield para Arduino de 4 relevadores



Figura 21- Shield para Arduino de 4 relevadores

Encoder KY-040

Al principio se pensó en utilizar botones para realizar la navegación dentro del menú del prototipo, pero después de indagar un poco sobre shields disponibles y opciones de control de menus, se decidió utilizar un encoder rotatorio KY-040 (Figura 22). Con esta opción utilizaremos menor número de entradas digitales de la placa Arduino (Figura 23). Su precio ronda los $30.



Figura 22 - Encoder KY-040

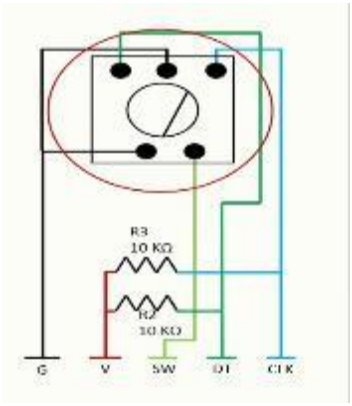


Figura 23 - Esquemático de un encoder KY-040

Microprocesador.

Se realizó un análisis rápido de cuántas entradas y salidas se manejarían para todos los dispositivos necesarios para el prototipo (Figura 24). Dentro de este análisis se determinó si las entradas y las salidas serían digitales o análogas.

Dispositivo	IN/OUT	DIGITAL	ANALOGO
LCD 16x2	6	6	0
Encoder Rotatorio	3	3	0
HC-SR04	2	2	0
Relay x 2	2	2	0
LED x2	2	2	0

Figura 24 - Tabla de entradas digitales y análogas del sistema

El primer dispositivo seleccionado es una placa de Arduino UNO (Figura 25), la que hará el trabajo de controlador principal del sistema. Se había considerado una placa Arduino MEGA ya que la placa Arduino UNO no cuenta con las salidas digitales necesarias para conectar todos los elementos necesarios para el proyecto, pero después de un poco de investigación se encontró que las salidas análogas de la placa pueden ser utilizadas como digitales (Figura 26). EL precio de la placa Arduino varía dependiendo de su fabricante, se pueden encontrar placas entre $200 a $500.



Figura 25 - Placa Arduino UNO R3

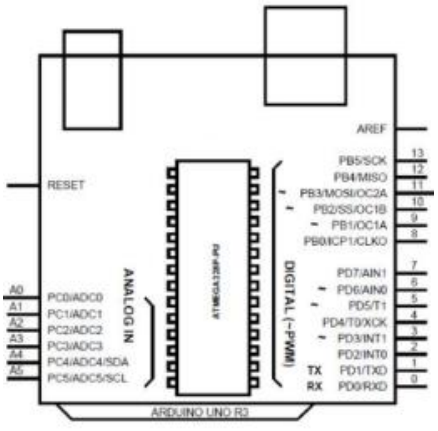


Figura 26 - Pin layout de una placa Arduino UNO R3

4.2 Bosquejos de elementos de control.

Se decidió ir probando cada elemento seleccionado por separado con pequeños programas para probar su funcionamiento y al final incorporar por completo todos los elementos.

4.21 Bosquejos de circuitos eléctricos.

El bosquejo utilizado para conectar nuestra pantalla LCD con la placa Arduino fue uno de los primeros circuitos a evaluar. Como nuestro dispositivo de LCD no contaba con circuito I2C para conectar a la placa Arduino con solo dos pines, fue necesario utilizar la mayor parte de los pines digitales para hacerla funcionar. A continuación, se muestra el bosquejo del circuito de pantalla LCD (Figura 27).

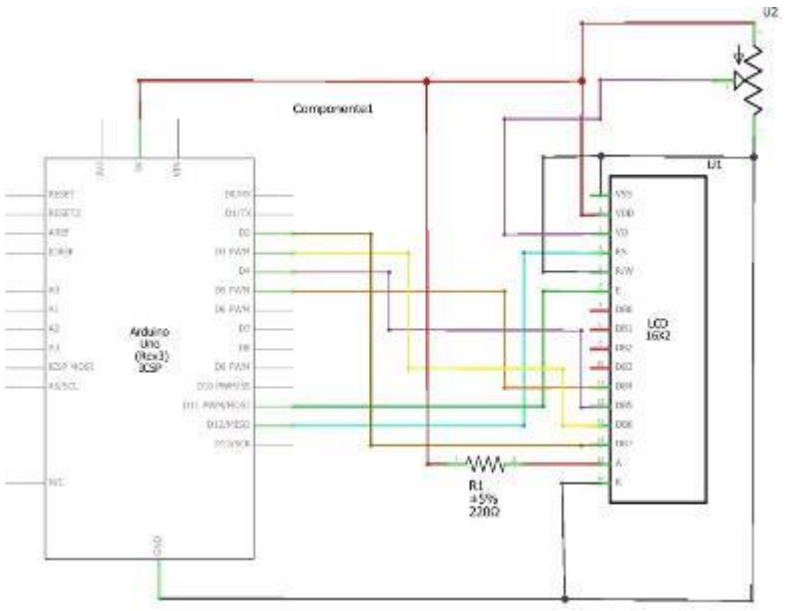


Figura 27 - Esquemático de la conexión de Arduino con LCD

El siguiente circuito a probar fue el de sensor ultrasónico HC-SR04, donde se hicieron pruebas con objetos solidos para comprobar que el circuito funcionara y el código fuera correcto, se utilizó el monitor serial del Arduino IDE para poder verificar los resultados. A continuación, se muestra el circuito para el sensor HC-SR04 (Figura 28).

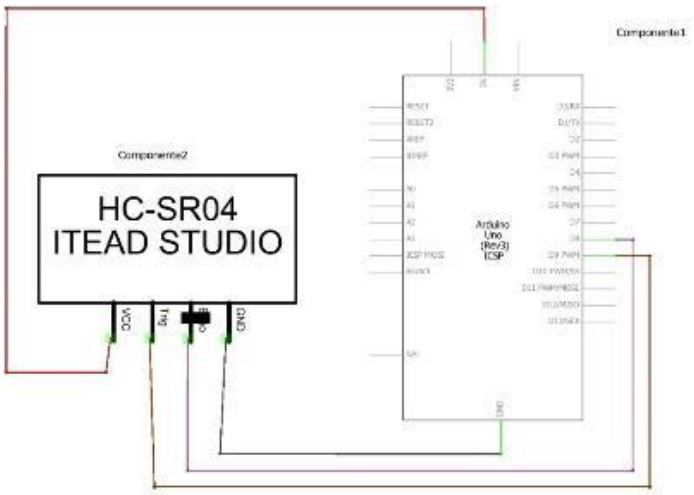


Figura 28 - Esquemático de la conexión de Arduino con HC-SR04

Después llegó el momento de trabajar con el circuito del encoder que utilizaremos para poder navegar fácilmente el menú de operación del prototipo. Utilizar un encoder en lugar de utilizar 6 botones, nos ayudó a liberar pines digitales de la placa Arduino. A continuación, se muestra el circuito del encoder (Figura 29).

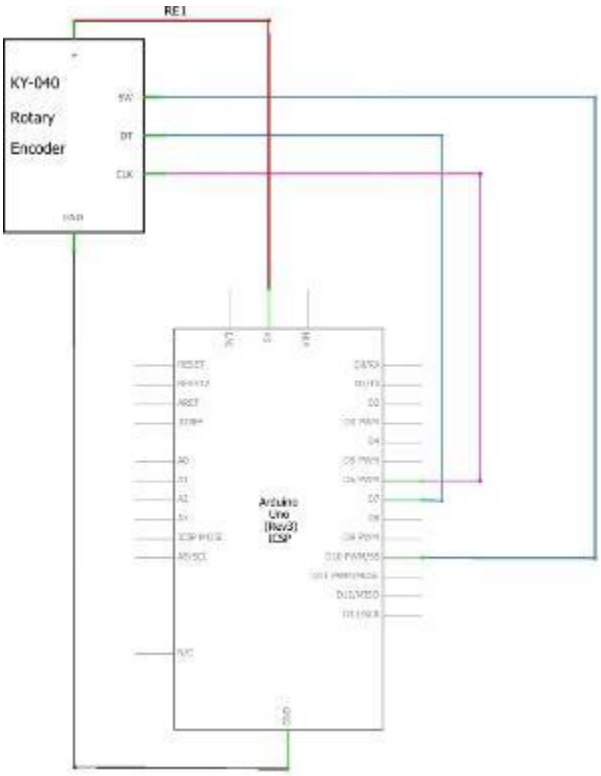


Figura 29 - Esquemático de la conexión de Arduino con KY-040

4.3 Desarrollo del prototipo de los elementos electrónico y de software.

4.3.1 Desarrollo del sistema lógico del proyecto

Una vez habiendo probado cada elemento por separado se inició con el desarrollo del prototipo incorporando todos los elementos. Lo primero fue realizar la conexión de todos los dispositivos a la placa Arduino para verificar su correcto funcionamiento (Anexo 1). El siguiente paso fue iniciar con el desarrollo del software de control, iniciamos con el diseño de una máquina de estado finito para representar el menú de operación del prototipo. A continuación, se muestra la tabla de estados de la máquina de estados finitos (Figura 30).

Estados	**Izquierda**	**Derecha**	**Clic**	**Menú**
1	0	0	1	Inicio
2	1	2	-1	Mezclar
3	1	4	-2	Llenar
4	3	9	-3	Nivel Máximo
5	2	5	3	Configuración
6	4	5	0	Salir
7	9	7	-4	Coolant
8	6	8	-5	Agua
9	7	8	1	Regresar
10	3	6	-6	Nivel Vacío
Salidas	**Proceso**			
-1	Realizar el proceso de mezcla entre coolant y agua.			
-2	Llenar con agua hasta el nivel máximo de capacidad.			
-3	Configurar nivel máximo de capacidad.			
-4	Configurar proporción de coolant.			
-5	Configurar proporción de agua.			
-6	Configurar distancia de nivel vacío			

Figura 30 - Tabla de estados para FSM

Para incorporarlo en la aplicación de Arduino fue necesario generar una estructura de datos especial para contener la tabla de estados en una matriz (Figura 31), esta matriz nos ayudará a navegar el menú y ejecutar los procesos de salida.

```
//Estructura para almacenar estados
typedef struct{
  int izquierda;
  int derecha;
  int dok;
  String texto;
} elemento;
//Array con tabla de estados
elemento A[]={
  /*0*/{0,0,1,"Presiona boton"},
  /*1*/{1,2,-1,"Mezclar"},
  /*2*/{1,4,-2,"Llenar"},
  /*3*/{3,9,-3,"Nivel maximo"},
  /*4*/{2,5,3,"Configuracion"},
  /*5*/{4,5,0,"Salir"},
  /*6*/{9,7,-4,"P. Coolant"},
  /*7*/{6,8,-5,"P. Agua"},
  /*8*/{7,8,1,"Regresar"},
  /*9*/{3,6,-6,"Nivel Vacio"}
};
elemento edo = A[0]; //Inicializar variable estado de fsm
```

Figura 31 - Código de matriz de estados

El siguiente paso en el desarrollo del prototipo fue el desarrollo de las diferentes funciones para los procesos de salida de nuestra máquina de estado finito y funciones auxiliares recursivas para facilitar el trabajo de codificación y tener un código más limpio y ordenado.

La primera función creada fue la función para escribir a la pantalla LCD (Figura 32). Los parámetros de entrada de esta función son un valor booleano para determinar si la pantalla requiere hacer un proceso de limpieza de pantalla, dos valores de tipo entero para determinar la columna y el renglón donde deberá escribir y por último un valor de tipo string para enviar el mensaje a mostrar.

```
//Funcion para escribir en LCD
void writeLCD(bool clr,int col, int ren, String txt)//clr limpiar o no el lcd
{
  if(clr)
  lcd.clear();
  lcd.setCursor(col,ren);
  lcd.print(txt);
  delayMicroseconds(dly);
}
```

Figura 32 - Función para escribir al LCD

La siguiente función creada fue la función para leer el movimiento del encoder (Figura 33). Esta función fue diseñada específicamente para detectar el movimiento del encoder y regresar el valor de DERECHA o IZQUIERDA para determinar la entrada para la maquina de estado finito, y así poder navegar el menú. Fue necesario agregar un contador para poder contar las dos interrupciones del encoder por salto como un solo movimiento, de lo contrario nos daría

dos entradas con el mismo valor y en lugar de interpretarlo como un solo movimiento, lo detectaría como un movimiento doble.

```
int readEncoder()
{
  int r=-100;
  aState = digitalRead(outputA); //Leer estado del econder
  if (aState != aLastState){
    //Movimiento hacia la derecha
    if (digitalRead(outputB) != aState) {
      c++;    //El contador c nos ayuda a contar dos interrupciones del encoder por cada salto
      if(c==2)
      {
       c=0;;
       r = DERECHA;
      }
    } else { //Movimiento hacia la izquierda
      c++;
      if(c==2)
      {
       c=0;
       r = IZQUIERDA;
      }
    }
  }
  aLastState = aState; // Actualizar estado de outputA
  //delay(5);
  return r;
}
```

Figura 33 - Función para lectura de encoder KY-040

Otra función mas que fue desarrollada para poder realizar los demás procesos de salida fue la función de leer la distancia (Figura 34). Esta función recibe un solo parámetro de entrada, que es un valor booleano para indicarle a la función si el valor regresado será la distancia en centímetros o un valor mapeado dependiendo del valor mínimo y el valor máximo de capacidad del contenedor indicando el nivel de llenado en porcentaje.

```
float leerDist(bool m)
{
  long t; //tiempo que demora en llegar el eco
  float d=0.0; //distancia en centimetros

  for(int i=0;i<norm;i++)
  {
    digitalWrite(Trigger, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger, LOW);
    t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
    //d = t/59;             //escalamos el tiempo a una distancia en cm
    //d=d+(t/29.0/2.0);
    d = (t*.0343)/2;
  }
  d = d/norm;

  //d=EMPTY-d;
  if(m)
    d = map(d,nivelMax,vacio,100,0);

  return d;
}
```

Figura 34 - Código de lectura de sensor y cálculo de distancia

Como Arduino IDE no cuenta con herramienta de debugeo de código, fue necesario desarrollar otra función auxiliar para poder ver el estado de la maquina de estados finito y la condición de ciertas variables del programa (Figura 35). Esta función fue cambiando dependiendo el elemento que se requería monitorear, la forma mas sencilla de realizar este monitoreo era escribiendo directamente por el puerto serial y mostrándolo en el monitor serial de Arduino IDE.

```
void monitor()
{
  Serial.print("Izquierda: ");
      Serial.println(edo.izquierda);
      Serial.print("Derecha: ");
      Serial.println(edo.derecha);
      Serial.print("OK: ");
      Serial.println(edo.dok);
      Serial.print("Texto: ");
      Serial.println(edo.texto);
      Serial.println("");
}
```

Figura 35 - Código para monitorear variables de la FSM

Una vez desarrolladas las funciones auxiliares, iniciamos con el desarrollo de la lógica de la máquina de estados finito. Utilizamos la función incorporada de loop de Arduino como un ciclo infinito donde estaremos haciendo un monitoreo constante de entradas. El primer problema con el cual nos enfrentamos fue al momento de usar el botón que viene incorporado en el encoder para aceptar la opción seleccionada en el menú. Al inicio el botón ejecutaba el proceso al momento de detectar que el botón era presionado, ocasionando que el programa detectara ese clic como una entrada doble incluso si se le agregaba un delay para darle tiempo al usuario de soltar el botón. Después de un poco de investigación, encontramos que el método indicado para este tipo de dispositivos era ejecutar los procesos al detectar que soltábamos el botón. Se incorporó el proceso de detección de liberación del botón para poder ejecutar las diferentes opciones del menú.

La primera función de proceso de salida desarrollada fue la función de llenado del contenedor con agua (Figura 36). Esta función estaba pensada para hacer el llenado del agua perdida por evaporación, es la mas sencillas de las funciones que incluyen activar algún actuador y monitorear el sensor dependiendo del nivel. Se considera que el contenedor ya contaba en algún momento con coolant y agua en la proporción determinada por el usuario, por lo tanto el proceso solo busca que el contenedor llegue al 100%.

```
void llenar()
{
  bool activo = true;
  float n=leerDist(true);
  Serial.println("Llenando");
  Serial.println(String(n));
  //Activar actuador
  digitalWrite(P_AGUA,HIGH);
  digitalWrite(L_AGUA,HIGH);
```

Figura 36 - Extracto de código de llenado

Podemos notar durante la lectura de la distancia del líquido utilizando la función de leerDist(bool), que al pasarle el parámetro booleano como true, le estamos pidiendo a la función que regrese un valor mapeado de 0 a 100% para determinar el nivel de líquido del contenedor.

```
n = leerDist(true);
```

Para poder continuar con el desarrollo del programa fue necesario considerar como configurar los valores de los niveles mínimo y máximo de capacidad y las proporciones de coolant y agua, para que de esta manera la codificación de los procesos quedara con variables en lugar de constantes. Por tal motivo se desarrollaron las demás funciones para configurar estos valores.

La función de configuración de nivel vacío es la principal (Figura 37), pues este nivel variará dependiendo de la instalación del sensor con respecto al contenedor. Por ende, es necesario una función que relacione la distancia del fondo del contendor como nivel mínimo, o bien nivel cero. La función nVacio() lee la distancia a la que se encuentra el fondo del contenedor utilizando la función leerDist(bool) con un parámetro booleano con valor falso para indicar que necesitamos la distancia en centímetros. Este valor en centímetros será almacenado para poder referenciarlo mas adelante cuando sea necesario.

```
void nVacio()
{
  float n = vacio;
  bool activo = true;
  Serial.println("Nivel Vacio");
  writeLCD(false,0,1,String(leerDist(false)) + "cm");
```

Figura 37 - Extracto de código para configuración de nivel mínimo

La función de configuración de nivel máximo (Figura 38) es la siguiente función de configuración de nivel desarrollada. Esta función tiene la tarea de determinar el nivel máximo al que se espera que llegue el liquido dentro de nuestro contenedor. A diferencia de la función nVacio(), la función nMax() no mide la distancia a la que se encuentra el líquido, si no que el usuario debe de determinar la distancia del sensor hasta donde debe de llegar el liquido al

encontrase en su nivel máximo dentro del contenedor. Una vez que seleccionamos el valor en centímetros, lo almacenaremos para poder hacer referencia a el cuando sea necesario.

```
void nMax()
{
  float n = nivelMax;
  bool activo = true;
  Serial.println("Nivel Maximo");
  writeLCD(false,0,1,String(n) + "cm");
```

Figura 38 - Extracto de código para configuración de nivel máximo

La selección de la distancia la hacemos utilizando el encoder para aumentar o disminuir la distancia mostrada en la pantalla LCD. Para lo cual utilizamos la función de readEncoder() para determinar la dirección de giro y aumentar o disminuir la distancia. (Figura 33)

Las dos siguientes funciones de configuración de nivel (Figura 39) tienen la misma lógica, que es la de asignar un valor entero a la proporción de coolant o agua que se requiere para la mezcla. Las funciones nCoolant() y nAgua() funcionan similar a la función nMax(), no mide distancias si no mas bien asignamos un valor utilizando la función de readEncoder() para aumentar o disminuir el valor mostrado en pantalla y después almacenarlo para su uso posterior.

```
void nCoolant()
{
  float n = nivelCoolant;
  bool activo = true;
  Serial.println("Proporción de coolant");
  writeLCD(false,0,1,String(n));

void nAgua()
{
  float n = nivelAgua;
  bool activo = true;
  Serial.println("Proporción de agua");
  writeLCD(false,0,1,String(n));
```

Figura 39 - Funciones para configurar las proporciones de refrigerante y agua

Por último, se desarrolló la función de mezclar coolant y agua en las proporciones configuradas en las funciones anteriores (Figura 40). La función mezclar() está diseñada para realizar la mezcla midiendo la distancia a la que se encuentra el nivel de líquido relacionándolo a la proporción configurada anteriormente. Se determinó primeramente agregar la proporción correspondiente al agua, ya que el coolant es un poco mas espeso y corremos el riesgo de que se añada mas coolant del requerido si este no es detectado a tiempo por el sensor de distancia. Por lo tanto pensando en el principio de Arquímedes la mejor

opción es llenar con agua y después añadir el coolant para que al momento de ser añadido, este desplace el agua y podamos tener una lectura correcta.

```
void mezclar()
{
  bool activo = true;
  float n=leerDist(true);//Depende del algoritmo
  Serial.println("Mezclando");
  Serial.println(String(n));
  //Activar actuador
  digitalWrite(P_AGUA,HIGH);
  digitalWrite(L_AGUA,HIGH);
  while(activo)

if(n>=nivelAgua*((100/(nivelAgua+nivelCoolant)))//Llenado con agua
{
```

Figura 40 - Extractos de código para calcular el llenado y mezcal de refrigerante y agua

4.3.2 Desarrollo del contenedor del sistema electrónico

Una vez verificado el funcionamiento lógico del proyecto (Anexo 2), se diseñaron piezas para contener el sistema electrónico del proyecto (Figura 41). Dichas piezas fueron manufacturadas por impresión 3D en material PLA. Se fabricó una tapa frontal (Anexo 4) y una tapa posterior (Anexo 3), con la idea de que sea fácil poder ensamblar y desensamblar el dispositivo para mantenimiento (Anexo 7).

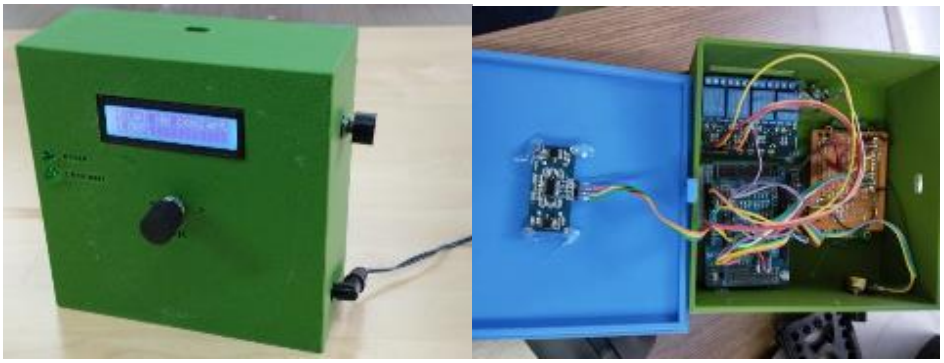


Figura 41- Vistas externas e internas del prototipo

El producto final está diseñado para ser instalado de forma horizontal sobre el contenedor que deseemos trabajar (Figura 42).



Figura 42 - Prototipo instalado en máquina

Los actuadores que deseemos utilizar pueden ser conectados directamente por el lado izquierdo del dispositivo (Figura 43). Dejando al usuario poder seleccionar el tipo de actuador que desee utilizar, como pueden ser bombas o electroválvulas.



Figura 43 - Acceso a conexión para actuadores

4.4 Pruebas de funcionamiento y mejoras

4.4.1 Navegación de menú

El primer elemento a probar con el dispositivo completamente ensamblado fue la navegación del menú, este fue diseñado en base a la tabla de estados de la máquina de estados finitos (Anexo 10).

Al encender el dispositivo automáticamente hace una lectura del nivel actual de refrigerante, presentándolo en porcentaje (Figura 44).



Figura 44 - Pantalla principal

Para poder entrar al menú debemos presionar el botón del encoder. Si hacemos cualquier movimiento, nos mostrara un mensaje de que debemos presionar el botón.

Una vez dentro del menú tenemos las opciones de mezclar, llenar, configurar o salir a la pantalla principal (Figura 45). Podemos navegar entre las opciones usando el giro del encoder.



Figura 45 - Pantallas de opciones principales

Dentro de la opción de Configuración podemos encontrar las opciones para configurar el nivel máximo, el nivel mínimo, la proporción de agua, la proporción de lubricante y una opción para regresar al menú anterior (Figura 46).



Figura 46 - Pantallas de opciones de configuración del sistema

4.4.2 Prueba de sensor de distancia

Una vez comprobado que podemos navegar por el menú y tener acceso a las diferentes funcionalidades del dispositivo, toca el turno de hacer pruebas con el sensor de distancia. Para realizar la prueba fue necesario utilizar los siguientes niveles predefinidos en el dispositivo:

Nivel máximo = 10 cm

Nivel vacío = 40 cm

Proporción de agua = 20 unidades

Proporción de lubricante = 10 unidades

Esto significa que debemos colocar nuestro dispositivo en una posición inicial y a partir de ahí colocar una marca a 10 cm y una mas a 40 cm desde el sensor ultrasónico (Figura 47).



Figura 47 - Marcas a 10 y 40 cm para pruebas preliminares

Colocamos nuestro dispositivo en la marca de inicio y un objeto en la marca de 40 cm, el sensor debe leer 0% (Figura 48).



Figura 48 - Prueba de simulación de contenedor vacío

Después colocamos nuestro objeto en la marca de 10 cm, el sensor debe marcar 100% (Figura 49).



Figura 49 - Prueba de contenedor al máximo

Para probar la relación de proporción entre agua y lubricante, se realiza la misma prueba, pero en esta ocasión debemos marcar una línea más calculando la proporción 20 a 10. El porcentaje calculado es 66.67 % a una distancia aproximada de 20 cm desde el sensor. El cálculo se realiza utilizando la siguiente fórmula:

$$\% = Agua * (\frac{100}{Agua + Coolant})$$

O bien utilizando la siguiente fórmula para calcular la distancia:

$$distancia = \frac{Agua * \left(\frac{100}{Agua + Coolant}\right) * (Agua - Coolant)}{100}$$

Donde:

% = porcentaje de agua necesario para completar el total de líquido en el contenedor

distancia = distancia en cm la cual debe estar el objeto

Agua = proporción de agua configurada en el dispositivo

Coolant = proporción de lubricante configurado en el dispositivo

Colocamos el objeto en la marca de 40 cm y activamos la opción de mezclar, esto nos muestra un porcentaje de 0% en la marca de 40 cm encendiendo el relevador para el actuador y encenderá el LED que nos muestra que se está llenando con agua, mientras movemos el objeto mas cerca del dispositivo el porcentaje aumenta, cuando llega a la marca de 67% cambia el estado de los relevadores, en esta ocasión desactiva el relevador del actuador para el agua y

enciende el actuador para el lubricante al mismo tiempo que apaga el LED de agua y enciende el LED para lubricante (Figura 50).



Figura 50 - Pruebas de cambio de actuador

Otras pruebas fueron realizadas cambiando los valores de nivel mínimo, máximo y las proporciones de agua y lubricante. Después se realizaron pruebas utilizando líquidos sobre un contenedor. Como se había estimado anteriormente, la lectura de líquidos fue un poco menos estable por el movimiento de los mismos, pero no afectó gravemente la funcionalidad del prototipo.

Por último, en la sección de anexos de este documento se pueden encontrar las instrucciones para configurar los niveles y proporciones de agua y lubricante para el dispositivo. (Anexo 11)

4.4.3 Mejoras y propuestas

La mejora principal que se pretende añadir en una próxima versión del prototipo es utilizar la EPPROM o algún módulo externo para almacenar valores de configuración de los diferentes elementos del prototipo, estos incluyen niveles de agua y refrigerante, valores de calibración de nuevos elementos que puedan incluirse al prototipo, etc. Cabe mencionar que utilizar el EPPROM tiene ciertas desventajas, pues la placa Arduino solo permite cierto número de escrituras en esta memoria antes de perder su funcionalidad. Por lo que se contempla utilizar un medio de almacenamiento externo como una memoria SD o quizá añadirle funcionamiento de IoT para guardar datos en la nube.

Otras mejoras que se contemplan para la siguiente versión, es utilizar un sensor a prueba de agua, para evitar cualquier daño que el dispositivo actual pudiera padecer, un sensor de pH para revisar la acidez de la mezcla e incluir un método para cuantificar los sólidos totales contenidos en la mezcla.

Por último seria añadirle comunicación por medio de Bluetooth para poder monitorear y configurar dese algún dispositivo móvil, es por eso que los pines rx y tx de la placa Arduino quedaron disponibles.

V. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Los resultados obtenidos de este proyecto de diseño de un dispositivo para controlar el nivel de refrigerante de un equipo CNC resultaron satisfactorios, sin embargo, en cada avance que se realizaba dentro del proyecto, nuevas ideas surgían de como agregarle nuevas funcionalidades, lo que generaba a manera personal que el proyecto cada vez se viera muy sencillo en cuanto a su alcance.

En cuestiones de presupuesto podemos concluir que el diseño y manufactura del prototipo no tuvo un costo elevado, puesto que la mayoría de los componentes se pudieron procurar por medio de donaciones realizadas por compañeros de trabajo y la manufactura en su totalidad se realizó con el equipo dentro de la universidad.

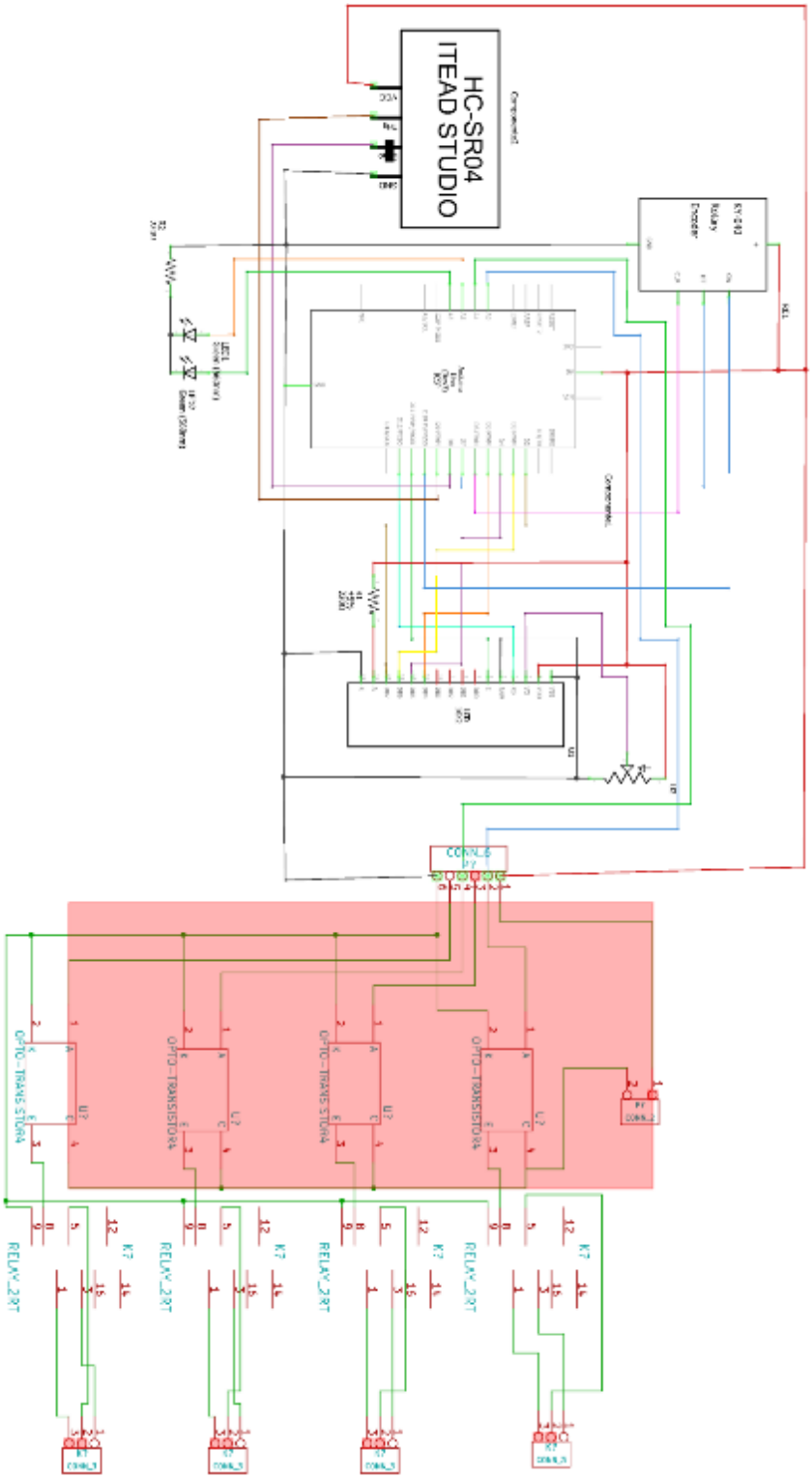
En cuanto a la precisión del prototipo, realmente no era un punto tan crítico pues cuando se realizaba la mezcla y llenado de refrigerante de manera manual, se realizaba al tanteo, y al ser un equipo que no está funcionando 24/7, su mayor problema era la perdida de agua por evaporación y no tanto perdida de lubricante por adhesión a las piezas maquinadas.

5.2 Recomendaciones

Una de las principales recomendaciones es siempre tratar de encontrar la manera de automatizar un proceso involucrando la menor cantidad de elementos, esto garantizará que se saque el mayor provecho a cada uno de ellos dentro de nuestro sistema y será mucho más sencillo encontrar fallas y repararlas. Sin embargo, es necesario analizar de manera profunda nuestro proceso y establecer de manera correcta el alcance de nuestro proyecto y sus requisitos de funcionalidad, sobre todo al llevarlo de nuestra idea abstracta a la realidad.

VI. ANEXOS

Anexo 1 – Esquemático del circuito completo.



Anexo 2 – Código de aplicación para Arduino

```
#include <LiquidCrystal.h>
#define outputA 6 //Direccion de encoder
#define outputB 7 //Direccion de encoder
#define OK_BTN 10 //Identificador de click a boton de encoder
#define DERECHA 4 //Constante para Giro Derecha
#define IZQUIERDA 5 //Constante para Giro Izquierda
#define Trigger 9 //Sensor ultrasonico
#define Echo 8 //Sensor ultrasonico
#define MEZCLA -1
#define LLENA -2
#define MAX -3
#define COOLANT -4
#define AGUA -5
#define EMPTY -6
#define P_AGUA A0
#define P_COOL A1
#define L_AGUA A2
#define L_COOL A3
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2; //Pines de LCD
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); //Configuracion de LCD
int aState; //Estado de encoder
int aLastState; //Ultimo estado de encoder
int c=0; //Contador de clics de encoder
int dly =100; //Delay de escritura
float nivelMax=10.0; //Variable para nivel maximo
float vacio = 40;
int nivelAgua = 20;
int nivelCoolant = 10;
float norm=1.0;
bool btn=false;
int input =0; //Variable de entrada
//Estructura para almacenar estados
typedef struct{
  int izquierda;
  int derecha;
  int dok;
  String texto;
} elemento;
//Array con tabla de estados
elemento A[]={
  /*0*/{0,0,1,"Presiona boton"},
  /*1*/{1,2,-1,"Mezclar"},
```

```
  /*2*/{1,4,-2,"Llenar"},
  /*3*/{3,9,-3,"Nivel maximo"},
  /*4*/{2,5,3,"Configuracion"},
  /*5*/{4,5,0,"Salir"},
  /*6*/{9,7,-4,"P. Coolant"},
  /*7*/{6,8,-5,"P. Agua"},
  /*8*/{7,8,1,"Regresar"},
  /*9*/{3,6,-6,"Nivel Vacio"}
};
elemento edo = A[0]; //Inicializar variable estado de fsm
void setup() {
  Serial.begin(9600);
  Serial.println("");
  lcd.begin(16,2);
  writeLCD(true,0,0,"Inicializando...");
  for(int i=0;i<16;i++)
  {
    writeLCD(false,i,1,"*");
    delay(75);
  }
  writeLCD(true,0,0,"Microcontrolador OK");
  delay(1000);
  //Configuración de pines del encoder
  pinMode (outputA,INPUT_PULLUP); //Configurar pin de encoder
  pinMode (outputB,INPUT_PULLUP); //Configurar pin de encoder
  pinMode(OK_BTN,INPUT_PULLUP);
  pinMode(P_AGUA,OUTPUT);
  pinMode(P_COOL,OUTPUT);
  pinMode(L_AGUA,OUTPUT);
  pinMode(L_COOL,OUTPUT);
  digitalWrite(P_AGUA,LOW);
  digitalWrite(P_COOL,LOW);
  Serial.println("Encoder configurado");
  aLastState = digitalRead(outputA);   //Leer estado inicial de encoder
  //Configuración del sensor ultrsonico
  pinMode(Trigger, OUTPUT); //Configurar pin de sensor ultrasonico como salida
  pinMode(Echo, INPUT);  //Configurar pin de sensor ultrasonico como entrada
  digitalWrite(Trigger, LOW);//Inicializamos el pin con 0
  Serial.println("Sensor configurado");
  Serial.println("LCD configurado");
  writeLCD(true,0,0,"Nivel de coolant");
  writeLCD(false,0,1,String(leerDist(true)) + "%");
  edo = A[0];
}
```

```
void loop() {
  if(digitalRead(OK_BTN)==HIGH)
  {
  switch(readEncoder())
  {
    case DERECHA:
      Serial.println("DERECHA");
      edo = A[edo.derecha];
      writeLCD(true,0,0,edo.texto);
      monitor();
      break;
    case IZQUIERDA:
      Serial.println("IZQUIERDA");
      edo = A[edo.izquierda];
      writeLCD(true,0,0,edo.texto);
      monitor();
      break;
    default:
      break;
  }
  }
  else
  {
    if(!btn)
    {
      btn = true;
      while(true)
      {
        if(digitalRead(OK_BTN)==HIGH)
        {
          btn = false;
          break;
        }
      }
      if(edo.dok>=0)
        edo = A[edo.dok];
      else
        {
      switch(edo.dok)
      {
        case MEZCLA:
          mezclar();
          break;
        case LLENA:
```

```
          llenar();
          break;
        case MAX:
          nMax();
          break;
        case COOLANT:
          nCoolant();
          break;
        case AGUA:
          nAgua();
          break;
        case EMPTY:
          nVacio();
          break;
        default:
          break;
      }
        }
      if(edo.dok ==1)
        {
          writeLCD(true,0,0,"Nivel de coolant");
          writeLCD(false,0,1,String(leerDist(true)) + "%");
        }
        else
          writeLCD(true,0,0,edo.texto);
      monitor();


      delayMicroseconds(dly*10);
    }
  }
  delayMicroseconds(dly);
}

//Función para configurar nivel maximo de agua
void nVacio()
{
  float n = vacio;
  bool activo = true;
  Serial.println("Nivel Vacio");
  writeLCD(false,0,1,String(leerDist(false)) + "cm");
  while(activo)
  {
    if(digitalRead(OK_BTN)==LOW)
```

```
  {
    if(!btn)
    {
      btn = true;
      while(true)
      {
        if(digitalRead(OK_BTN)==HIGH)
        {
          btn = false;
          break;
        }
      }
      n = leerDist(false);
      Serial.println("Guarda");
      vacio = n;
      activo = false;
      writeLCD(true,0,0,"Guardando");
      delay(3000);
      edo=A[9];
      writeLCD(true,0,0,"Nivel de coolant");
      writeLCD(false,0,1,String(leerDist(true)) + "%");
      monitor();
    }
    else
    {
      Serial.println("Nivel Vacio");
      writeLCD(false,0,1,String(leerDist(false)) + "cm");
    }
    delayMicroseconds(dly*10);
  }
  delayMicroseconds(dly);
  }
}
//Funcion de mazcla de tanque
void mezclar()
{
  bool activo = true;
  float n=leerDist(true);
  Serial.println("Mezclando");
  Serial.println(String(n));
  //Activar actuador
  digitalWrite(P_AGUA,HIGH);
  digitalWrite(L_AGUA,HIGH);
  while(activo)
```

```
{
  if(digitalRead(OK_BTN)==HIGH)
  {
    writeLCD(true,0,0,"Mezclando");
    writeLCD(false,0,1,String(n) + "%");
    if(n>=nivelAgua*((100/(nivelAgua+nivelCoolant))))//Llenado con agua
    {
      //Desactivar actuador
      digitalWrite(P_AGUA,LOW);
      digitalWrite(L_AGUA,LOW);
      Serial.println("Lleno");
      activo = false;
      writeLCD(true,0,0,"Lleno");
      delay(2000);
      edo=A[1];
      writeLCD(true,0,0,edo.texto);
      monitor();
    }
    n = leerDist(true);
    Serial.println(String(n));
  }
  else
  {
    if(!btn)
  {
    btn = true;
    while(true)
    {
      if(digitalRead(OK_BTN)==HIGH)
      {
        btn = false;
        break;
      }
    }
    //Desactivar Actuadores
    digitalWrite(P_AGUA,LOW);
    digitalWrite(L_AGUA,LOW);
    digitalWrite(P_COOL,LOW);
    digitalWrite(L_COOL,LOW);
    Serial.println("Cancelado");
    activo = false;
    writeLCD(true,0,0,"Cancelado");
    delay(3000);
    edo=A[1];
```

```
    writeLCD(true,0,0,edo.texto);
    monitor();
  }
  }
}

activo = true;
n=leerDist(true);
Serial.println("Mezclando");
Serial.println(String(n));
//Activar actuador
digitalWrite(P_COOL,HIGH);
digitalWrite(L_COOL,HIGH);
while(activo)
{
  if(digitalRead(OK_BTN)==HIGH)
  {
    writeLCD(true,0,0,"Mezclando");
    writeLCD(false,0,1,String(n) + "%");
    if(n>=100)//Llenado con coolant
    {
      //Desactivar actuador
      digitalWrite(P_COOL,LOW);
      digitalWrite(L_COOL,LOW);
      Serial.println("Lleno");
      activo = false;
      writeLCD(true,0,0,"Lleno");
      delay(2000);
      edo=A[1];
      writeLCD(true,0,0,edo.texto);
      monitor();
    }
    n = leerDist(true);
    Serial.println(String(n));
  }
  else
  {
    if(!btn)
  {
    btn = true;
    while(true)
    {
      if(digitalRead(OK_BTN)==HIGH)
      {
```

```
          btn = false;
          break;
        }
      }
      //Desactivar Actuadores
      digitalWrite(P_AGUA,LOW);
      digitalWrite(L_AGUA,LOW);
      digitalWrite(P_COOL,LOW);
      digitalWrite(L_COOL,LOW);
      Serial.println("Cancelado");
      activo = false;
      writeLCD(true,0,0,"Cancelado");
      delay(2000);
      edo=A[1];
      writeLCD(true,0,0,edo.texto);
      monitor();
    }
    }
  }
}
//Funcion de llenado de tanque
void llenar()
{
  bool activo = true;
  float n=leerDist(true);
  Serial.println("Llenando");
  Serial.println(String(n));
  //Activar actuador
  digitalWrite(P_AGUA,HIGH);
  digitalWrite(L_AGUA,HIGH);
  while(activo)
  {
    if(digitalRead(OK_BTN)==HIGH)
    {
      writeLCD(true,0,0,"Llenando");
      writeLCD(false,0,1,String(n) + "%");
      if(n>=100)
      {
        //Desactivar actuador
        digitalWrite(P_AGUA,LOW);
        digitalWrite(L_AGUA,LOW);
      Serial.println("Lleno");
      activo = false;
      writeLCD(true,0,0,"Lleno");
```

```
      delay(2000);
      edo=A[2];
      writeLCD(true,0,0,edo.texto);
      monitor();
      }
      n = leerDist(true);
      Serial.println(String(n));
    }
    else
    {
      if(!btn)
    {
      btn = true;
      while(true)
      {
        if(digitalRead(OK_BTN)==HIGH)
        {
          btn = false;
          break;
        }
      }
      //Desactivar Actuadores
      digitalWrite(P_AGUA,LOW);
      digitalWrite(L_AGUA,LOW);
      Serial.println("Cancelado");
      activo = false;
      writeLCD(true,0,0,"Cancelado");
      delay(2000);
      edo=A[2];
      writeLCD(true,0,0,edo.texto);
      monitor();
    }
    }
  }
}
void monitor()
{
  Serial.print("Izquierda: ");
      Serial.println(edo.izquierda);
      Serial.print("Derecha: ");
      Serial.println(edo.derecha);
      Serial.print("OK: ");
      Serial.println(edo.dok);
      Serial.print("Texto: ");
```

```
      Serial.println(edo.texto);
      Serial.println("");
}
//Función para configurar nivel de agua
void nCoolant()
{
  float n = nivelCoolant;
  bool activo = true;
  Serial.println("Proporción de coolant");
  writeLCD(false,0,1,String(n));
  while(activo)
  {
    if(digitalRead(OK_BTN)==HIGH)
  {
    switch(readEncoder())
    {
      case DERECHA:
        if(n>99)
          n = 99;
        else
          n++;
        writeLCD(false,0,1,String(n));
        Serial.println("Suma a n");
        break;
      case IZQUIERDA:
        if(n<1)
          n=1;
        else
          n--;
        writeLCD(false,0,1,String(n));
        Serial.println("Resta a n");
        break;
      default:
        break;
    }
  }
  else
  {
    if(!btn)
    {
      btn = true;
      while(true)
      {
        if(digitalRead(OK_BTN)==HIGH)
```

```
        {
          btn = false;
          break;
        }
      }
      Serial.println("Guarda");
      nivelCoolant = n;
      activo = false;
      writeLCD(true,0,0,"Guardando");
      delay(2000);
      edo=A[6];
      writeLCD(true,0,0,edo.texto);
      monitor();
    }
    delayMicroseconds(dly*10);
  }
  delayMicroseconds(dly);
  }
}
//Función para configurar nivel de agua
void nAgua()
{
  float n = nivelAgua;
  bool activo = true;
  Serial.println("Proporción de agua");
  writeLCD(false,0,1,String(n));
  while(activo)
  {
    if(digitalRead(OK_BTN)==HIGH)
  {
    switch(readEncoder())
    {
      case DERECHA:
        if(n>99)
          n = 99;
        else
          n++;
        writeLCD(false,0,1,String(n));
        Serial.println("Suma a n");
        break;
      case IZQUIERDA:
        if(n<1)
          n=1;
        else
```

```
          n--;
        writeLCD(false,0,1,String(n));
        Serial.println("Resta a n");
        break;
      default:
        break;
    }
  }
  else
  {
    if(!btn)
    {
      btn = true;
      while(true)
      {
        if(digitalRead(OK_BTN)==HIGH)
        {
          btn = false;
          break;
        }
      }
      Serial.println("Guarda");
      nivelAgua = n;
      activo = false;
      writeLCD(true,0,0,"Guardando");
      delay(2000);
      edo=A[7];
      writeLCD(true,0,0,edo.texto);
      monitor();
    }
    delayMicroseconds(dly*10);
  }
  delayMicroseconds(dly);
  }
}
//Función para configurar nivel maximo de agua
void nMax()
{
  float n = nivelMax;
  bool activo = true;
  Serial.println("Nivel Maximo");
  writeLCD(false,0,1,String(n) + "cm");
  while(activo)
  {
```

```
  if(digitalRead(OK_BTN)==HIGH)
{
  switch(readEncoder())
  {
    case DERECHA:
      if(n>400)
        n = 400;
      else
        n = n + 0.5;
      writeLCD(false,0,1,String(n) + "cm");
      Serial.println("Suma a n");
      break;
    case IZQUIERDA:
      if(n<0.5)
        n=0.5;
      else
        n = n -0.5;
      writeLCD(false,0,1,String(n) + "cm");
      Serial.println("Resta a n");
      break;
    default:
      break;
  }
}
else
{
  if(!btn)
  {
    btn = true;
    while(true)
    {
      if(digitalRead(OK_BTN)==HIGH)
      {
        btn = false;
        break;
      }
    }
    Serial.println("Guarda");
    nivelMax = n;
    activo = false;
    writeLCD(true,0,0,"Guardando");
    delay(2000);
    edo=A[3];
    writeLCD(true,0,0,"Nivel de coolant");
```

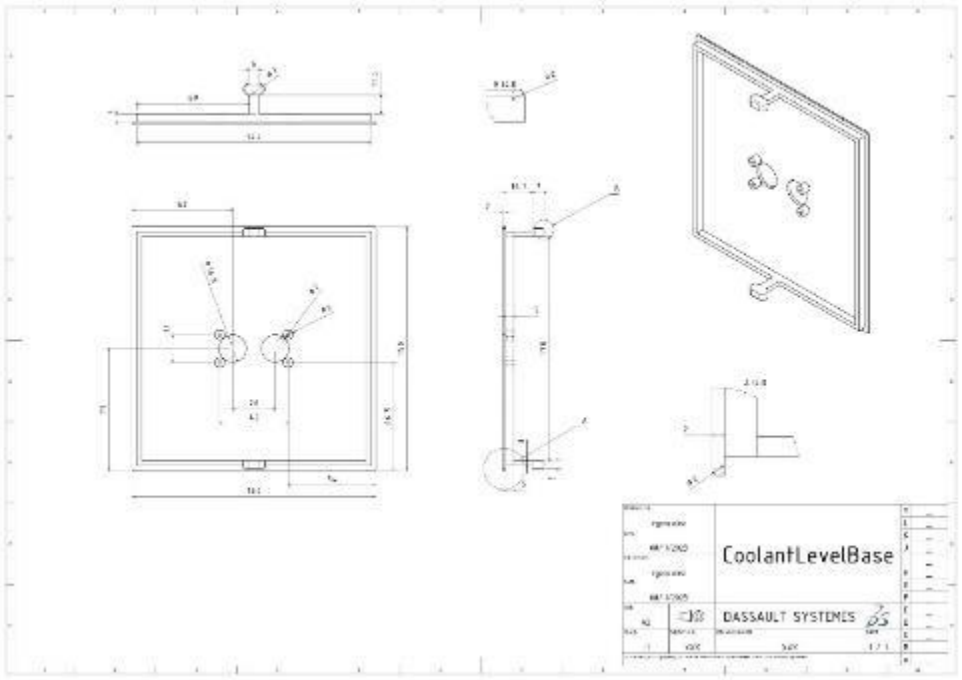
```
      writeLCD(false,0,1,String(leerDist(true)) + "%");
      monitor();
    }
    delayMicroseconds(dly*10);
  }
  delayMicroseconds(dly);
  }
}
//Función para leer giro de encoder, regresa si es DERECHA o IZQUIERDA
int readEncoder()
{
  int r=-100;
  aState = digitalRead(outputA); //Leer estado del econder
  if (aState != aLastState){
     //Movimiento hacia la derecha
     if (digitalRead(outputB) != aState) {
       c++;    //El contador c nos ayuda a contar dos interrupciones del encoder por cada salto
       if(c==2)
       {
        c=0;;
        r = DERECHA;
       }
     } else { //Movimiento hacia la izquierda
       c++;
       if(c==2)
       {
        c=0;
        r = IZQUIERDA;
       }
     }
   }
   aLastState = aState; // Actualizar estado de outputA
   //delay(5);
   return r;
}
//Funcion para escribir en LCD
void writeLCD(bool clr,int col, int ren, String txt)//clr limpiar o no el lcd
{
  if(clr)
  lcd.clear();
  lcd.setCursor(col,ren);
  lcd.print(txt);
  delayMicroseconds(dly);
}
```

```
//Función para leer distancia
float leerDist(bool m)
 {
    long t; //tiempo que demora en llegar el eco
    float d=0.0; //distancia en centimetros

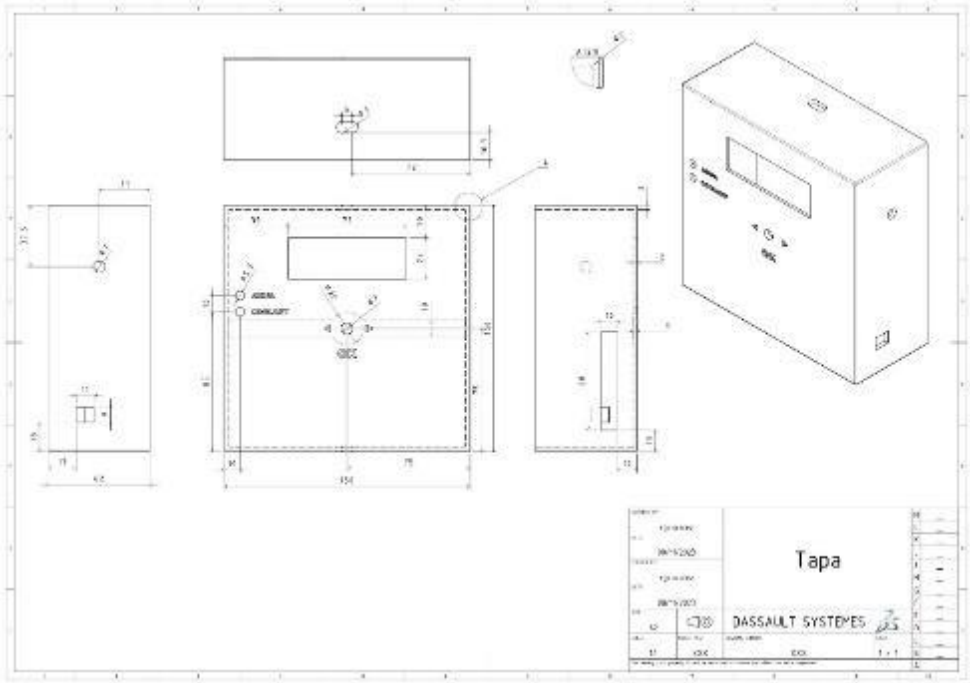
    for(int i=0;i<norm;i++)
    {
      digitalWrite(Trigger, HIGH);
      delayMicroseconds(10);          //Enviamos un pulso de 10us
      digitalWrite(Trigger, LOW);
      t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
      //d = t/59;             //escalamos el tiempo a una distancia en cm
      //d=d+(t/29.0/2.0);
      d = (t*.0343)/2;
    }
    d = d/norm;
    if(m)
      d = map(d,nivelMax,vacio,100,0);

    return d;
 }
```

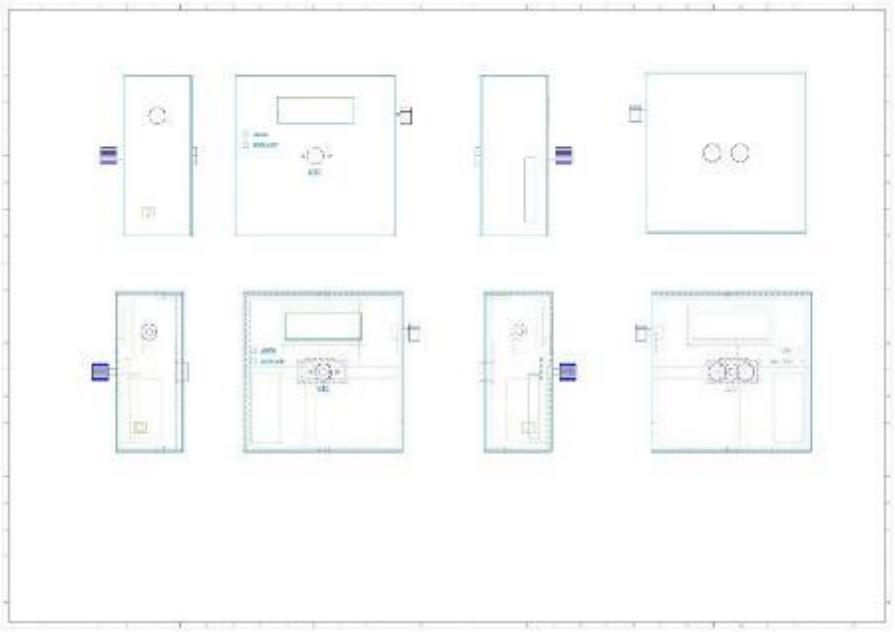
Anexo3 – Tapa Posterior



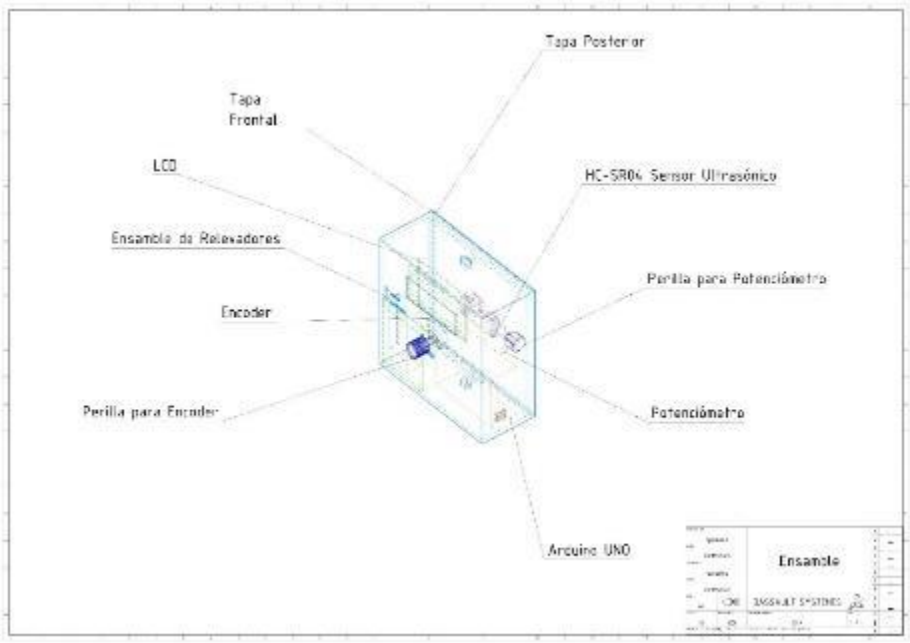
Anexo 4 – Tapa Frontal



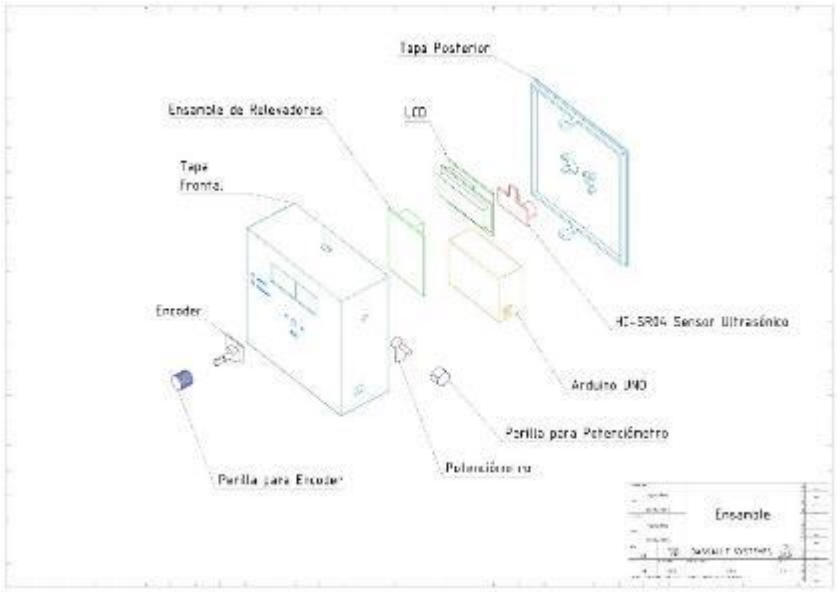
Anexo 5- Varias vistas del dispositivo



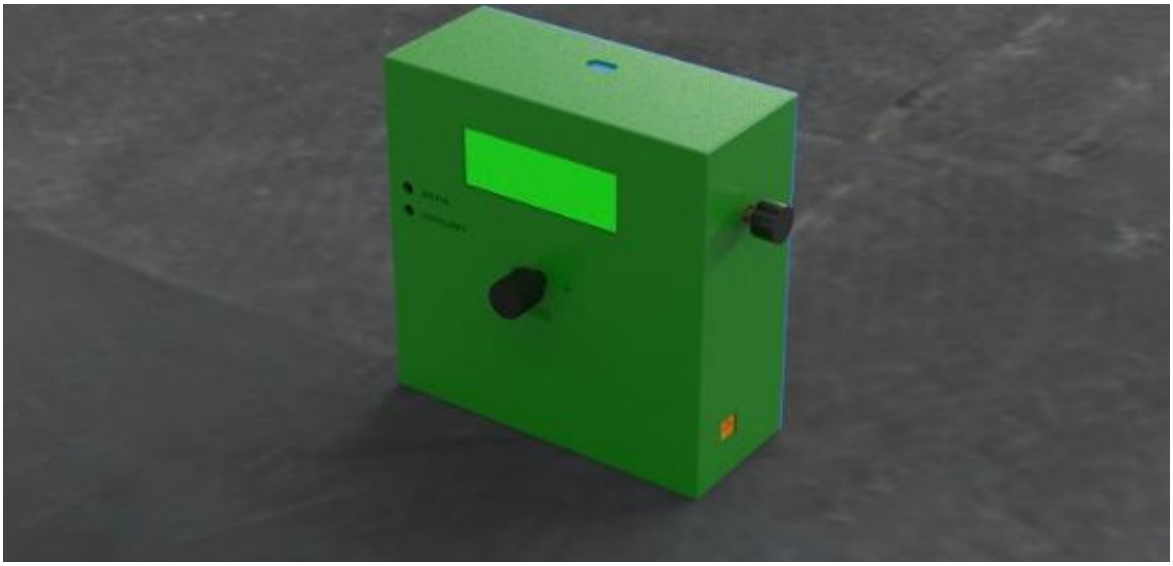
Anexo 6 – Componentes del dispositivo



Anexo 7- Ensamble del dispositivo



Anexo 8- Renderizado el producto



Anexo 9 – Producto terminado



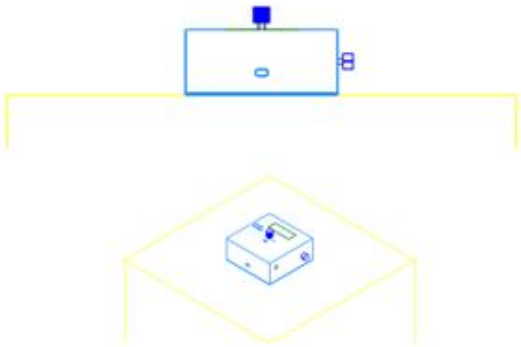
Anexo 10 – Tabla de estados de la máquina de estados finitos.

Estados	**Izquierda**	**Derecha**	**Clic**	**Menú**
1	0	0	1	Inicio
2	1	2	-1	Mezclar
3	1	4	-2	Llenar
4	3	9	-3	Nivel Máximo
5	2	5	3	Configuración
6	4	5	0	Salir
7	9	7	-4	Coolant
8	6	8	-5	Agua
9	7	8	1	Regresar
10	3	6	-6	Nivel Vacío
Salidas	**Proceso**			
-1	Realizar el proceso de mezcla entre coolant y agua.			
-2	Llenar con agua hasta el nivel máximo de capacidad.			
-3	Configurar nivel máximo de capacidad.			
-4	Configurar proporción de coolant.			
-5	Configurar proporción de agua.			
-6	Configurar distancia de nivel vacío			

Anexo 11 – Instrucciones para configuración.

Instrucciones para configurar el dispositivo.

- Instalar el dispositivo de forma horizontal sobre el contenedor completamente vacío.



- Conectar dispositivo a fuente de poder.



- El nivel de coolant mostrará un valor, debemos ahora configurar los niveles para que estos valores sean correctos.



- Presionar el botón para tener acceso al menú y navegar a la opción Configuración.



- Navegar a la opción Nivel Vacío.



- Mostrará una distancia. Presionar el botón para guardar.



- Navegar a la opción Nivel Máximo.



- Mostrará una distancia, utilizando la perilla central seleccionar el valor en centímetros a la que debe encontrarse el sensor contra el nivel máximo de líquido. Presionamos el botón para guardar.



- Navegar a la opción P. Coolant para configurar la proporción de lubricante.
- Seleccionar la proporción de lubricante utilizando la perilla y presionar el botón para guardar.



- Navegar a la opción de P. Agua para configurar la proporción de lubricante.
- Seleccionar la proporción de agua utilizando la perilla y presionar el botón para guardar.



VII. BIBLIOGRAFÍA

Arduino. (2017). Obtenido de www.arduino.cc/.

asconumatics.eu. (s.f.). Obtenido de www.asconumatics.eu: https://www.asconumatics.eu/images/site/upload/_es/pdf1/00005es.pdf

elecfreaks.com. (s.f.). *http://www.micropik.com/PDF/HCSR04.pdf.* Obtenido de www.micropik.com.

Gutiérrez, O. (2022). *Máquinas de Estado Finito.* México: Alfaomega.

How to Mechatronics. (s.f.). Obtenido de How to Mechatronics: https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

howtomechatronics.com. (s.f.). *http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/*. Obtenido de http://howtomechatronics.com/.

Lubricantes y Filtros para Procesos Industriales. (s.f.). *https://www.lufilsur.es/medir-concentracion-lubricantes-refrigerantes/#:~:text=Para%20saber%20el%20porcentaje%20de,lubricante%20refrigerante%20que%20estamos%20consumiendo.* Obtenido de https://www.lufilsur.es/medir-concentracion-lubricantes-refrigerantes/#:~:text=Para%20saber%20el%20porcentaje%20de,lubricante%20refrigerante%20que%20estamos%20consumiendo.: https://www.lufilsur.es/medir-concentracion-lubricantes-refrigerantes/#:~:text=Para%20saber%20el%20porcentaje%20de,lubricante%20refrigerante%20que%20estamos%20consumiendo.

Master Fluids Solutions. (2004). *https://www.masterfluids.com/mx/es-mx/products/show-product-concentrations-coolants.php?pid=E206NDN*. Obtenido de https://www.masterfluids.com/mx/es-mx/products/show-product-concentrations-coolants.php?pid=E206NDN: https://www.masterfluids.com/mx/es-mx/products/show-product-concentrations-coolants.php?pid=E206NDN

Millán, J. A. (2022). *Arduino: Programación y Proyectos.* España: Editorial Marcombo.

Mykhailo, D. (2011-2017). Obtenido de http://es.onlinemschool.com/: http://es.onlinemschool.com/math/formula/volume/

Pérez Montiel, H. (2011). *Física General BAchillerato* (4 ed.). Grupo Editorial Patria.

Pérez Porto, J., & Gardey, A. (2013). *http://definicion.de/ultrasonido/*. Recuperado el 2017, de definicion.de.

Rubin, J. (2019). *Interfaces de Usuario: Diseño y Construcción.* Estados Unidos: Addison-Wesley.

Zebra Skimmers. (s.f.). *https://store.zebraskimmers.com/products/the-dazzle-sump-level-controller*. Obtenido de https://store.zebraskimmers.com/products/the-dazzle-sump-level-controller: https://store.zebraskimmers.com/products/the-dazzle-sump-level-controller

Zuckerwar, A. (2002). *Handbook of the Speed of Sound in Real Gases.* Academic Press.