



# Proyecto: Dosificadora de Líquidos

TUTOR:

❖ Cristian Leandro Lukaszewicz

Alumno:

❖ Pedro Ovidio Coronel

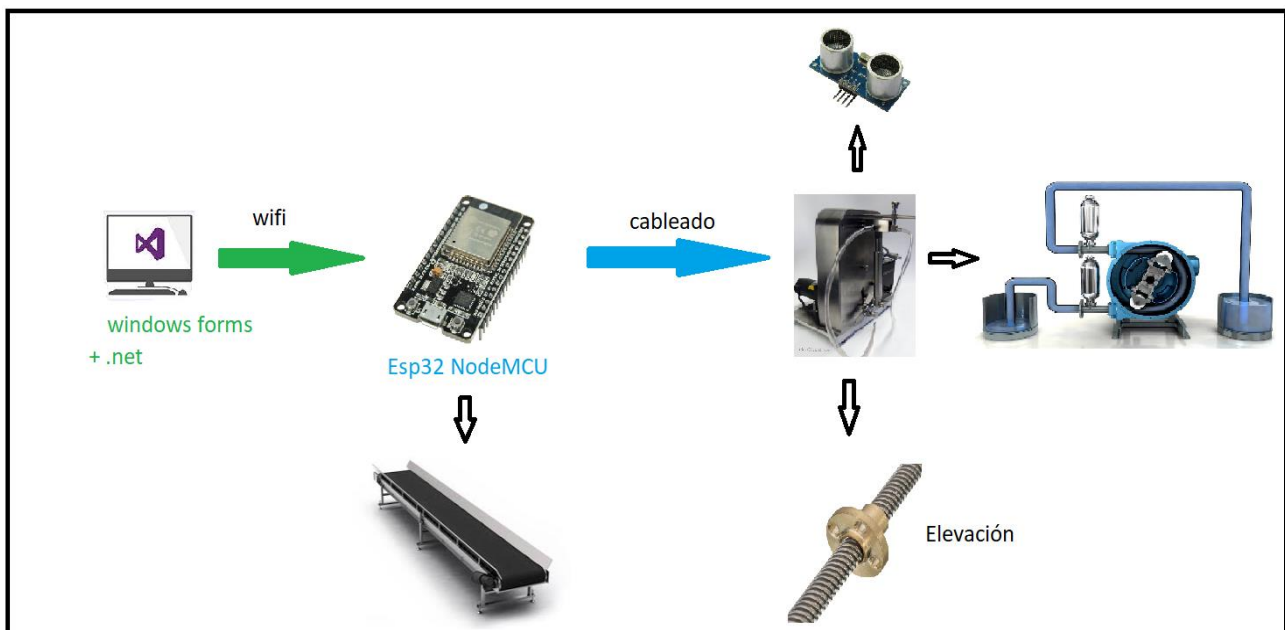
## Necesidad

Se requiere automatizar el sector de dosificación de líquidos dentro de una línea de producción, para poder realizar una mezcla de dos sustancias según la cantidad requerida. La modificación se realizará para de esta manera no solo automatizar la dosificación del líquido, sino también llevar un seguimiento de la productividad diaria, eficiencia y análisis de datos.

El objetivo de este proyecto es el diseño y elaboración de una dosificadora de líquidos para relleno de envases, la cual se pueda insertar dentro de una línea de producción, y un software para el control y monitoreo de la misma.

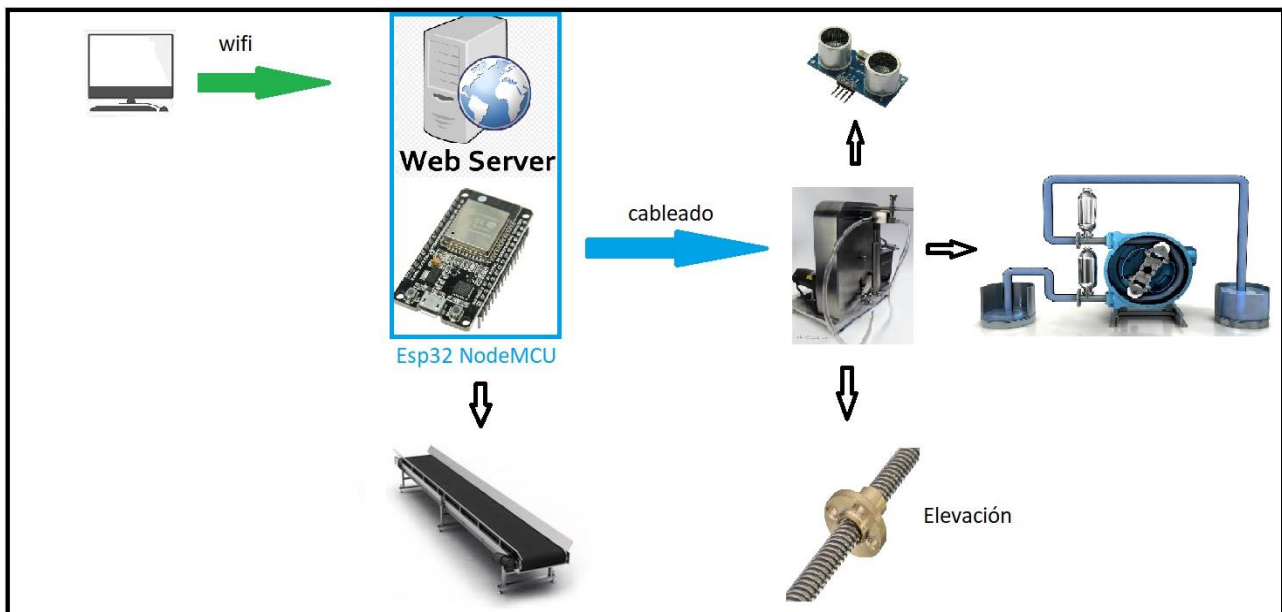
## Alcance:

En el presente trabajo se realizó el diseño de una dosificadora de líquidos, se hizo el prototipo mecánico, eléctrico y de software para el manejo de la misma, la dosificadora tiene la capacidad de mezclar dos líquidos a la vez, los cuales son parametrizados en el software diseñado por nosotros.

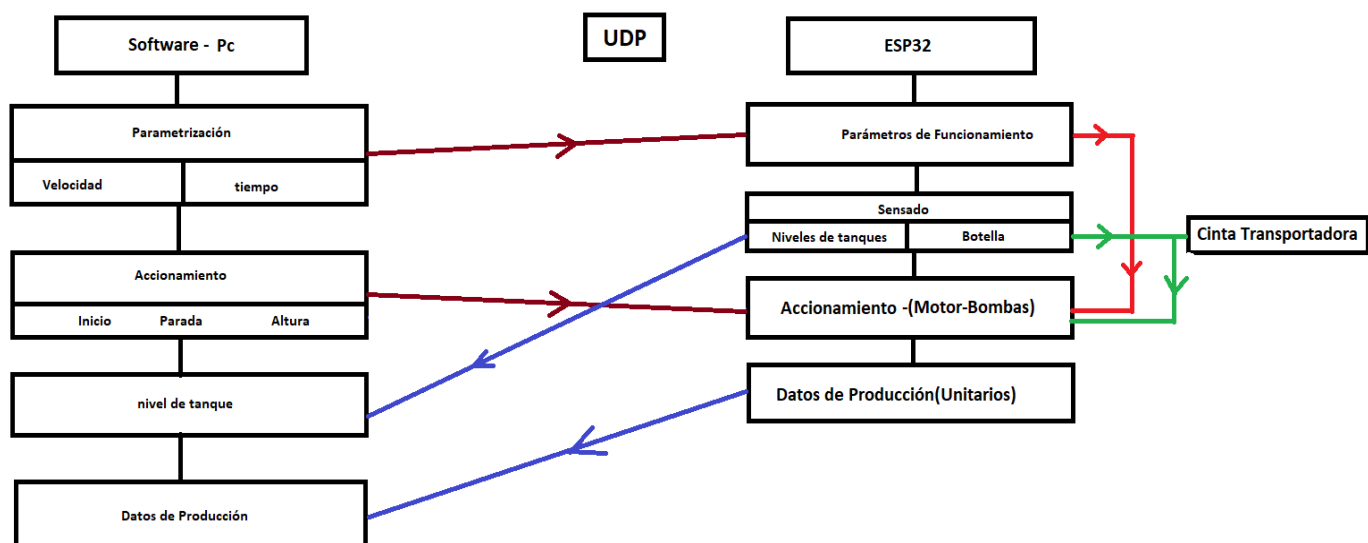


La dosificadora se puede integrar a una línea de producción en la que los envases llegan mediante una cinta transportadora, y cuando el sensor registra la presencia de la botella en posición de llenado, éste enviará una señal al esp32 el cual le enviará una señal a la cinta indicando que debe frenar, hasta que le llegue la señal de que la botella ya está llena y vuelva a moverse

Como opcional se podrá utilizar el servidor web incluido en el Esp32, Programando una interfaz gráfica con HTML para poder hacer la función que realizaría la app.



### Diseño funcional a nivel macro:





Para explicar el diseño funcional de la Máquina dosificadora, utilizamos la gráfica vista anteriormente, en donde se detallan las funciones que serán realizadas del lado Pc (app) y del lado Esp-32, y el método de comunicación entre los mismos, el cual se hará vía UDP (User Datagram Protocol).

## Diagrama de Gantt

|                      |                                      |             |            | Fecha estimada |            | Fecha real |            |             |      |      |      |
|----------------------|--------------------------------------|-------------|------------|----------------|------------|------------|------------|-------------|------|------|------|
| TAREA                |                                      | Responsable | Estado [%] | Inicio         | Fin        | Inicio     | Fin        | Predecesora | est. | real | dif. |
| Fase 1: Analisis     |                                      |             | 100        | 15/8/2021      | 9/10/2021  | 15/8/2021  | 29/10/2021 |             | 82   | 82   | 0    |
| 1                    | Elección final de proyecto           | Todos       | 100        | 15/8/2021      | 5/9/2021   | 15/8/2021  | 5/9/2021   |             | 8    | 8    | 0    |
| 2                    | Estudio de Mercado                   | Todos       | 100        | 5/9/2021       | 17/9/2021  | 5/9/2021   | 15/9/2021  | 1           | 16   | 10   | 6    |
| 3                    | Definición de Funcionamirto          | Todos       | 100        | 1/9/2021       | 21/9/2021  | 1/9/2021   | 19/9/2021  | 2           | 20   | 20   | 0    |
| 4                    | Especificaciones Técnicas            | Todos       | 100        | 21/9/2021      | 30/9/2021  | 20/9/2021  | 29/10/2021 | 3           | 30   | 32   | -2   |
| 5                    | Primer presupuesto                   | Ne          | 100        | 28/9/2021      | 9/10/2021  | 26/9/2021  | 9/10/2021  | 4           | 8    | 12   | -4   |
| Fase 2: Diseño       |                                      |             | 100        | 28/9/2021      | 6/11/2021  | 26/9/2021  | 15/12/2021 |             | 80   | 320  | -240 |
| 6                    | Eléctrico / Electrónico              | Ne          | 100        | 8/10/2021      | 16/10/2021 | 10/10/2021 | 6/12/2021  | 4           | 20   | 55   | -35  |
| 7                    | Estructural / Mecánico               | SI          | 100        | 8/10/2021      | 23/10/2021 | 13/10/2021 | 15/12/2021 | 4           | 25   | 115  | -90  |
| 8                    | Softwre                              | Co          | 100        | 28/9/2021      | 16/10/2021 | 26/9/2021  | 12/12/2021 | 3           | 25   | 100  | -75  |
| 9                    | Análisis del sistema Integrado       | Todos       | 100        | 30/9/2021      | 6/11/2021  | 1/10/2021  | 15/12/2021 | 6/7/8       | 10   | 50   | -40  |
| Fase 3: Construcción |                                      |             | 100        | 17/10/2021     | 18/11/2021 | 15/10/2021 | 28/5/2024  |             | 115  | 445  | -330 |
| 10                   | Desarrollo Eléctrico                 | Ne          | 100        | 19/10/2021     | 30/10/2021 | 16/10/2021 | 28/5/2024  | 6           | 20   | 80   | -60  |
| 11                   | Desarrollo Mecánico                  | SI          | 100        | 17/10/2021     | 30/10/2021 | 16/10/2021 | 28/5/2024  | 7           | 30   | 100  | -70  |
| 12                   | Desarrollo de Software               | Co          | 100        | 17/10/2021     | 6/11/2021  | 15/10/2021 | 3/6/2024   | 8           | 30   | 90   | -60  |
| 13                   | Esquema de Comunicación y Control    | Ne / Co     | 100        | 25/10/2021     | 13/11/2021 | 15/10/2021 | 19/5/2024  | 9           | 15   | 25   | -10  |
| 14                   | Ensamble de prototipo                | Todos       | 100        | 8/11/2021      | 18/11/2021 | 15/10/2021 | 28/5/2024  | 10/11/12/13 | 20   | 150  | -130 |
| Fase 4: Testeos      |                                      |             | 100        | 1/11/2021      | 10/12/2021 |            |            |             | 130  | 272  | -142 |
| 15                   | Funcionamiento Eléctrico             | Na          | 100        | 1/11/2021      | 6/12/2021  | 16/10/2021 | 3/6/2024   | 10          | 20   | 30   | -10  |
| 16                   | Funcionamiento Mecánico              | SI          | 100        | 1/11/2021      | 6/12/2021  | 16/10/2021 | 3/6/2024   | 11          | 30   | 40   | -10  |
| 17                   | Software                             | Co          | 100        | 1/11/2021      | 6/12/2021  | 15/10/2021 | 3/6/2024   | 12          | 50   | 70   | -20  |
| 18                   | Pruebas en conjunto                  | Todos       | 100        | 15/11/2021     | 8/12/2021  | 20/11/2021 | 3/6/2024   | 15/16/17    | 20   | 50   | -30  |
| 19                   | Primer Entrega                       | Todos       | 0          | 30/09/2024     | 30/09/2024 | 30/09/2024 | 30/09/2024 | 18          | 2    | 2    | 0    |
| 20                   | Primer Ajuste                        | Todos       | 0          | 30/09/2024     | 28/10/2024 | 30/09/2024 | 28/10/2024 | 19          | 20   | 25   | -5   |
| 21                   | Segunda Entrega                      | Todos       | 0          | 28/10/2024     | 28/10/2024 | 28/10/2024 | 28/10/2024 | 20          | 2    | 2    | 0    |
| 22                   | Segundo Ajuste                       | Todos       | 0          | 28/10/2024     | 25/11/2024 | 28/10/2024 | 25/11/2024 | 21          | 20   | 20   | 0    |
| 23                   | Entrega Final                        | Todos       | 0          | 25/11/2024     | 25/11/2024 | 25/11/2024 | 25/11/2024 | 22          | 2    | 2    | 0    |
| 24                   | Ajustes Finales                      | Todos       | 0          | 25/11/2024     | 09/12/2024 | 25/11/2024 | 09/12/2024 | 23          | 20   | 30   | -10  |
| 25                   | Entrega y Defensa del Proyecto Final | Todos       | 0          | 09/12/2024     | 09/12/2024 | 09/12/2024 | 09/12/2024 | 24          | 1    | 1    | 0    |

## Análisis de Desvíos:

Observando el diagrama de Gantt a detalle, podemos ver que tenemos, en algunos casos, grandes diferencias entre el tiempo planeado para la ejecución de una tarea y el tiempo real.

Observando el Gantt vemos que la fase de Diseño y construcción fueron las que más horas extras de trabajo llevaron y las que mayor diferencia podemos encontrar entre lo planeado y lo real.

Entre las principales ocasiones podemos destacar:

- Mejoras en los diseños:
  - Inicialmente utilizamos un eje Guía y a prueba y error pudimos verificar que dos ejes Guías realizaban un mejor movimiento.
  - Inicialmente utilizamos madera para las bases, cuyo material luego se cambió por aluminio debido a la mayor precisión que pudimos obtener.
  - Dificultades a la hora de obtener un mejor rendimiento en las bombas peristálticas.
  - Muy alto consumo en bombas peristálticas (5A), esto provocó dificultades a la hora de disminuir la velocidad por PWM.
  - Falsos contactos en ubicaciones esporádicas. Ej: soldadura en LM7805, soldadura en IRF320, en pines y cables varios, etc.
  - Baja resolución en Sensores Ultrasónicos.
  - Dificultades a la hora de entregar corriente a Bombas y ESP con la misma fuente.
  - Dificultades a la hora del cambio de metodología a la hora de dosificar por medio de volumen.
  - Dificultades a la hora de crear una vinculación por medio de WIFI entre PC y ESP.

## Etapas de diseño:

El proceso de diseño es la etapa fundamental para la elaboración de cualquier producto. Comienza con el análisis de las necesidades del consumidor o usuario para planificar un bien final que se pueda comercializar y que de respuesta a las necesidades estudiadas.

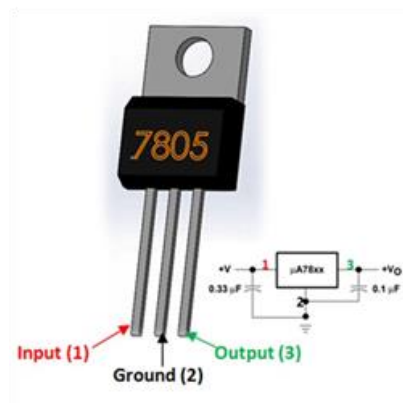
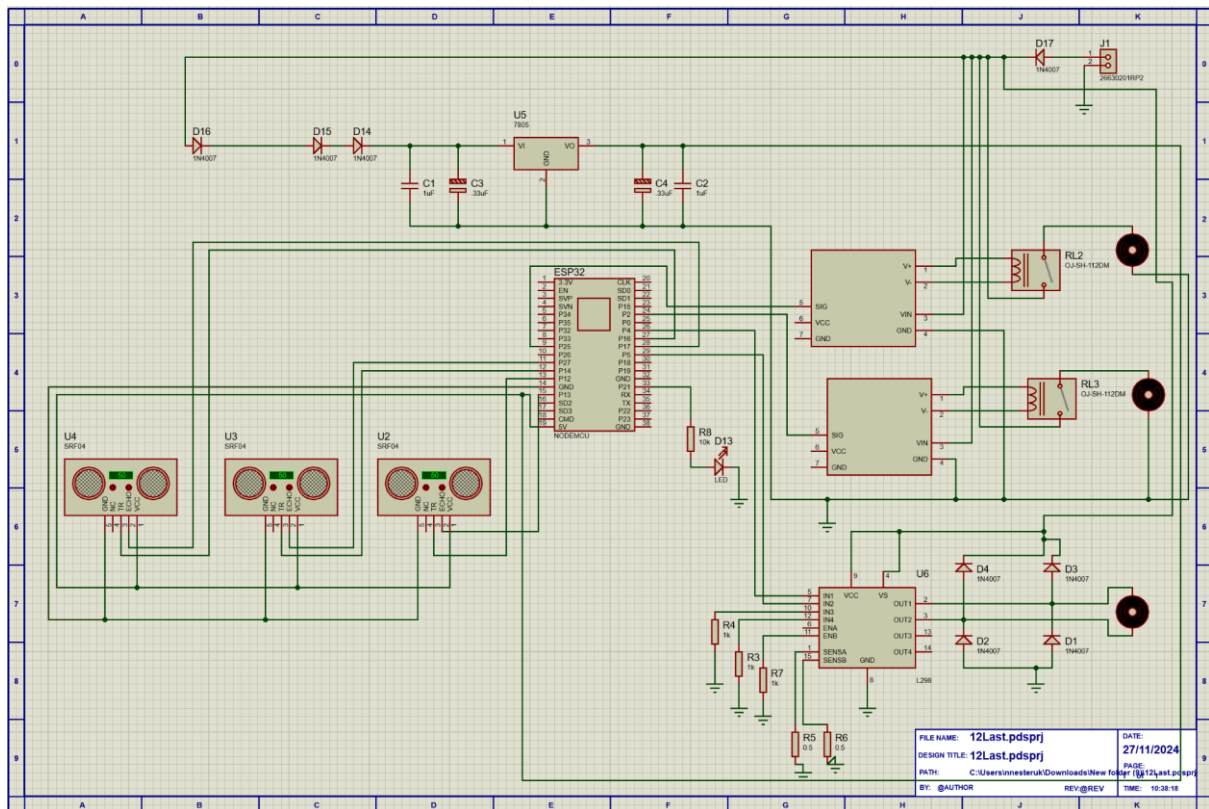
Primero se realizó un análisis del mercado para conocer el tamaño del mercado al que se desea introducir el producto, detectar qué otras empresas ofrecen el mismo producto y qué características buscan los clientes. Se pudo determinar si es viable la inversión en dicho producto de acuerdo a las necesidades de los potenciales clientes y la posible competencia. Algunos de los requerimientos fueron:

- Que tenga una interfaz de control mediante dispositivo PC. Será un programa desarrollado para el dispositivo seleccionado, en el cual el operador configure la cantidad de líquido a dosificar y pueda monitorear los parámetros de productividad.
- Conexión entre dispositivo y controlador mediante wifi. Para cumplir con este ítem se seleccionó como controlador una placa esp-32 con protocolo de comunicación UDP, haciendo el software de Servidor y el esp32 de Cliente.
- Detección del recipiente. El Esp-32 se encarga de la detección de recipientes y llenado mediante un sensor ultrasónico, a su vez controla las bombas de llenado, el motor encargado de la regulación de altura. y dará la señal de stop-Avance a la cinta transportadora.
- Detección de recipientes. Para ello se utiliza un sensor ultrasónico.
- Información sobre el estado de llenado de los tanques. Para eso se utilizan dos sensores ultrasónicos.
- Sincronización de la dosificadora con la línea de producción. Se realiza mediante el sensor de detección de botella, el cual al detectar la existencia de la misma prende una señal lumínica que representa la parada de la cinta transportadora.
- Dosificado:
  - El rango de dosificado entre ( 100ml - 1000ml )
  - La capacidad de llenado es de 1L / min.
  - Rango de densidades a manejar entre (680 Kg/m<sup>3</sup> - 1410 Kg/m<sup>3</sup>)
  - Capacidad tanques de 4 Litros.
- Parada de emergencia en programa y en el sistema eléctrico en caso de inconvenientes.
- Plantillas para poder configurar los siguientes parámetros:
  - Tamaño de envases, con tamaños preestablecidos para seleccionar según requiera el usuario.
  - Regulación de la altura. Esto se logra mediante un motor según el tamaño de envase seleccionado.
  - Porcentaje de mezcla
  - Cantidad de líquido, de acuerdo al tamaño de envase seleccionado.
  - La velocidad de llenado.

Una vez obtenidos los requerimientos se pudo realizar la definición del funcionamiento, donde se establece cómo cumplirá éste producto con tales y cómo es su funcionamiento en general.

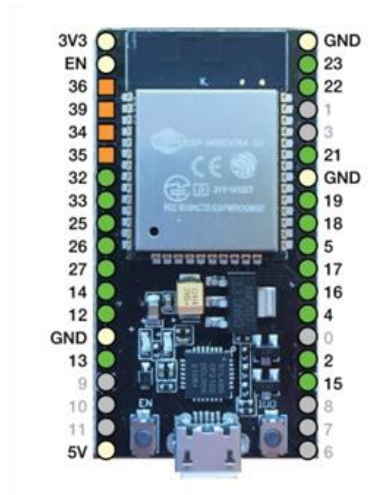
El funcionamiento de la dosificadora consiste en un sensor ultrasónico que detecta la presencia de un envase en la cinta transportadora, le envía una señal a ésta para que frene y así poder llenar el envase con líquido. El llenado ocurre a través de bombas peristálticas que van dosificando el líquido de acuerdo a los parámetros seleccionados por el usuario en la aplicación. El software también permite verificar el estado de llenado de los tanques mediante sensores ultrasónicos que miden la altura del líquido en los mismos. El controlador utilizado para comunicar todos los componentes es un ESP32.

## Diseño sistema electrónico





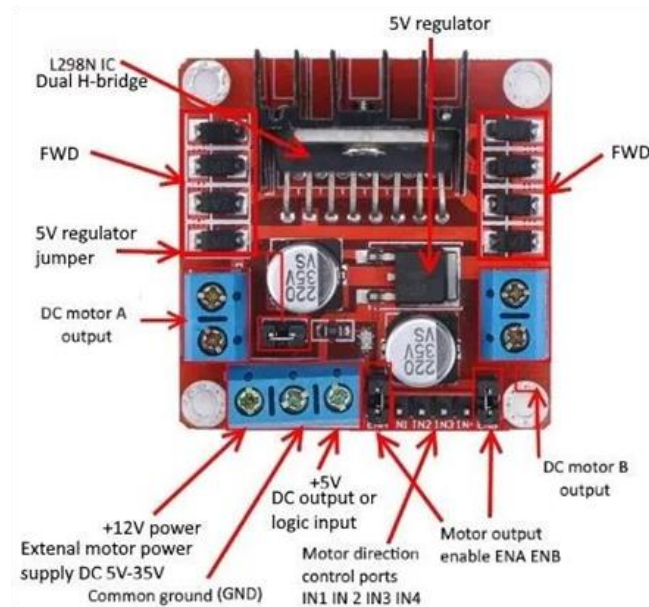
El regulador de tensión LM7805 tolera hasta 35V de entrada y genera una tensión de salida de 5V y hasta 1A. En nuestro caso utilizamos una fuente de 12V, ya que el fabricante indica que para un correcto funcionamiento se debe tener una diferencia mínima de 5V y de esta manera podemos energizar toda la placa con una sola fuente.



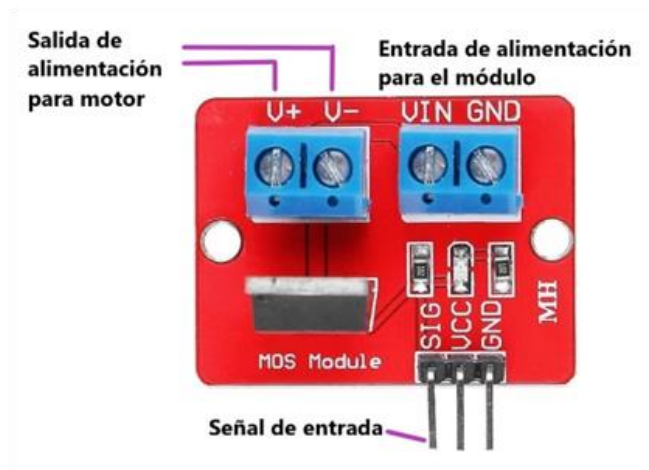
El ESP32 trabaja a 5V y 70mA



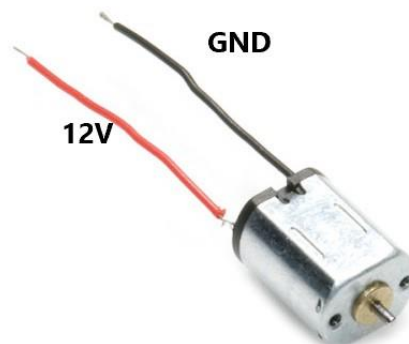
El sensor ultrasónico HC-SR04 trabaja con un voltaje de 5V y una corriente de 15mA



El controlador de motor L298N puede transmitir un voltaje de hasta 35V al motor



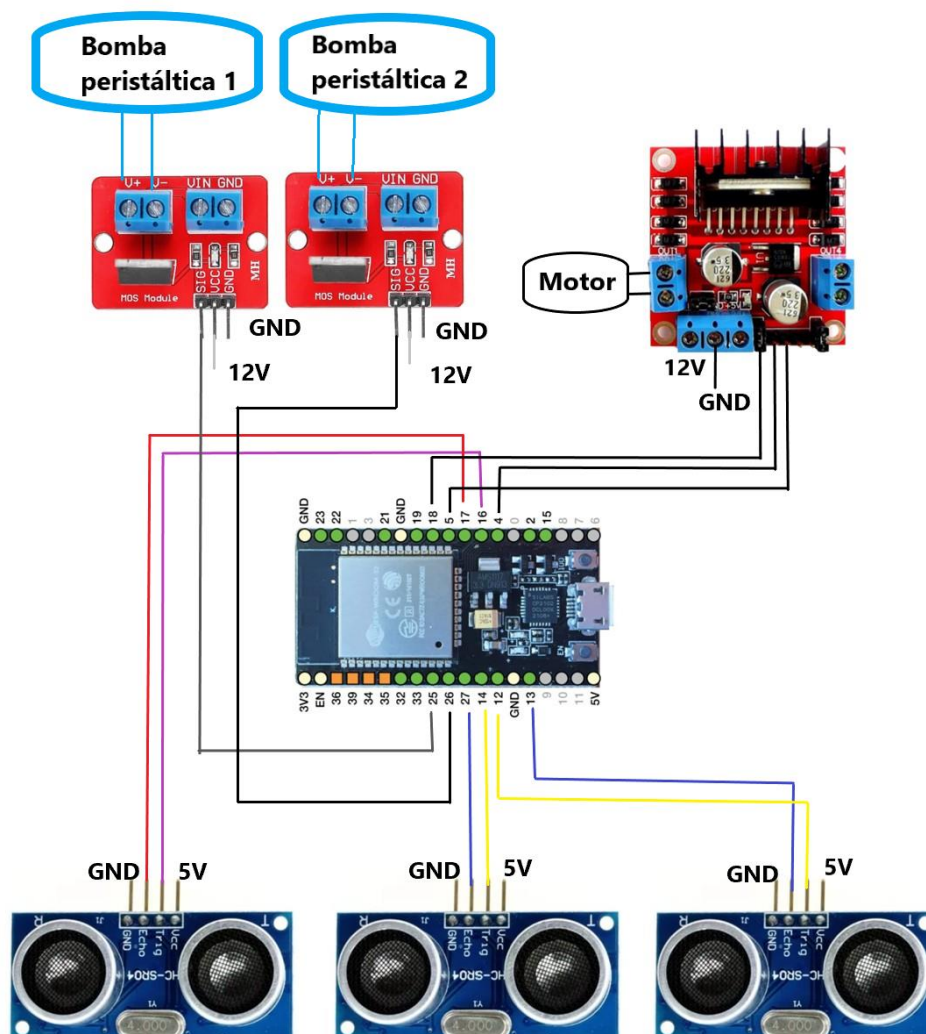
El módulo controlador IRF520 puede transmitir al motor un voltaje de salida de hasta 24V y una corriente de 5A.



Los motores utilizados para las bombas peristálticas consumen 5A de corriente y trabajan a una tensión de 12V, por lo tanto para poder tener en funcionamiento ambas bombas a la vez, se requiere una fuente capaz de entregar 10 A.

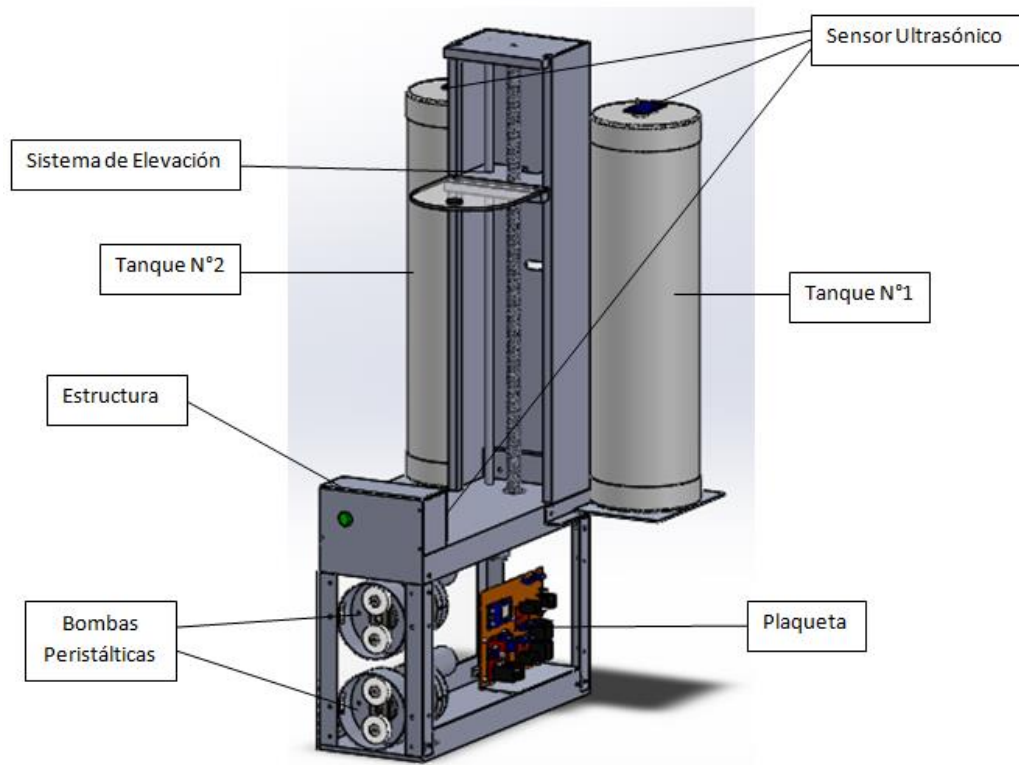
El motor utilizado para la elevación de las mangueras trabaja a 12V y consume 0,2A, por lo que no genera un consumo representativo.

### Esquema de conexionado



### Diseño mecánico

El diseño mecánico abarca la estructura de soporte de los componentes (esp32, bombas, sensores botones) para el funcionamiento de la máquina y el mecanismo de regulación de altura, el cual es realizado por un sistema de cremallera o tornillo sin fin. En el siguiente esquema se pueden ver todas las partes del sistema mecánico:



Esquema Sistema mecánico finalizado

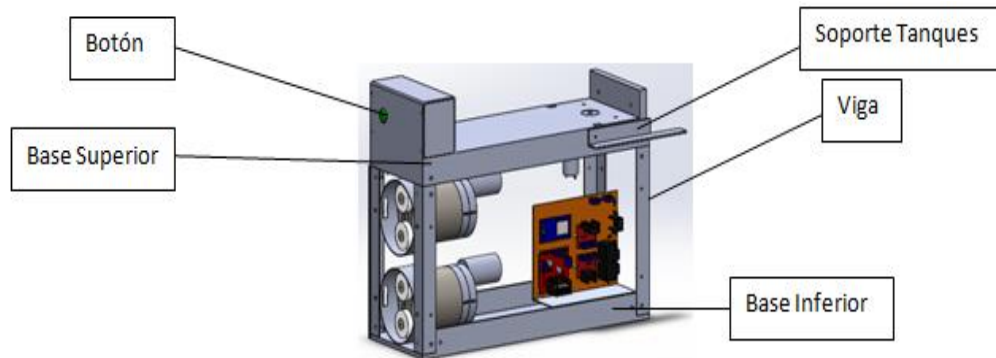
### # Estructura

En primera instancia, cabe destacar que en todo el diseño de la estructura optamos por utilizar Aluminio, ya que posee una fácil maniobrabilidad tanto para doblarlo como para maquinarlo según nuestras necesidades. Además de ser un material de fácil acceso, tanto en el mercado como en la vida diaria, es un material dentro de toda la gama de productos, barato.

Dicho diseño consta de dos partes principales, una que contiene todo el sistema de elevación y otra que contiene las bombas peristálticas y la electrónica.

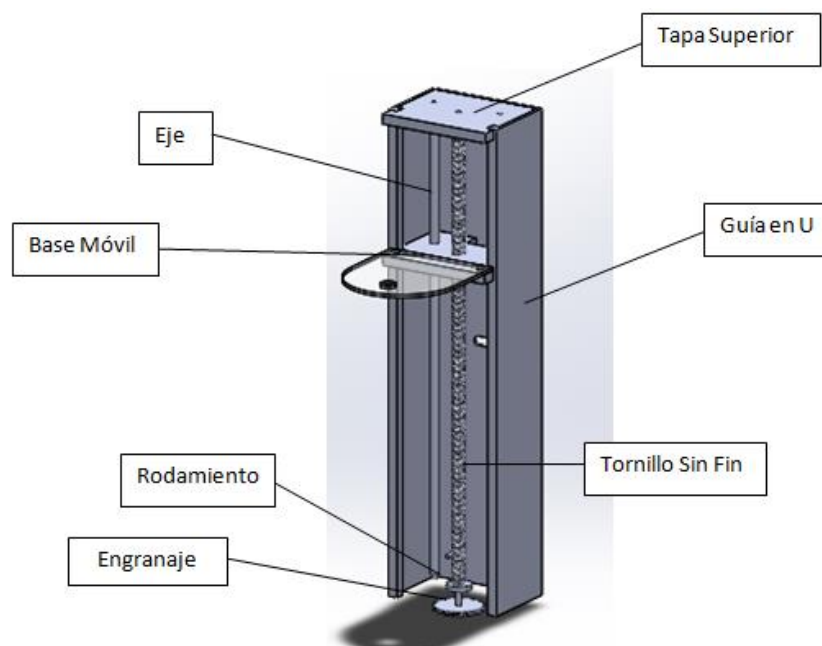
Por tal motivo, la parte inferior del sistema tiene las medidas necesarias para poder mantener en su lugar y proteger tanto la electrónica como las bombas. Esta estructura se pensó como dos bases unidas por cuatro vigas y encajonadas con plástico transparente resistente, de manera tal que se pueda observar el interior de las mismas.

y apreciar tanto el movimiento de las bombas, como el diseño electrónico, en la imagen se pueden visualizar las partes estructurales mencionadas. La base inferior también contiene el sensor de llenado y el control de energía.



(Imagen 1)

EL segundo componente Estructural es una especie de columna (la misma la podemos visualizar en la *imagen 2*), la misma se encarga de contener tres ejes, los cuales, en conjunto con la columna, se encargan de guiar la base móvil restringiéndole dos grados de libertad y de esta manera permitirle solo movimiento vertical. Dicho movimiento vertical se controla por medio de un sistema de tornillo sin fin que se detalla en el Diseño del sistema de elevación.



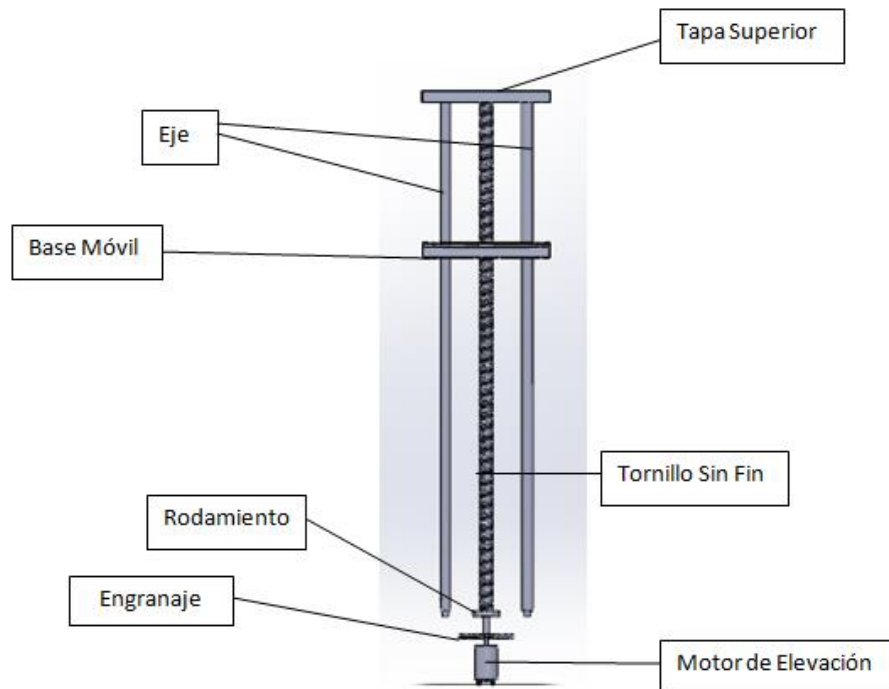
(Imagen 2)

## # Sistema de elevación

Una vez realizado el diseño estructural, nos centramos en el diseño del sistema de elevación. Dicho diseño consta de las siguientes partes:

- Base superior y la Guía en U (Las cuales son parte del diseño estructural)
- Tornillo sin fin
- Motor
- Engranajes
- Ejes guía
- Base Móvil

Dichos componentes en conjunto forman toda la estructura del sistema de elevación, los componentes principales de este sistema pueden visualizarse en la siguiente imagen (*Imagen 3*) y serán descriptos a continuación.

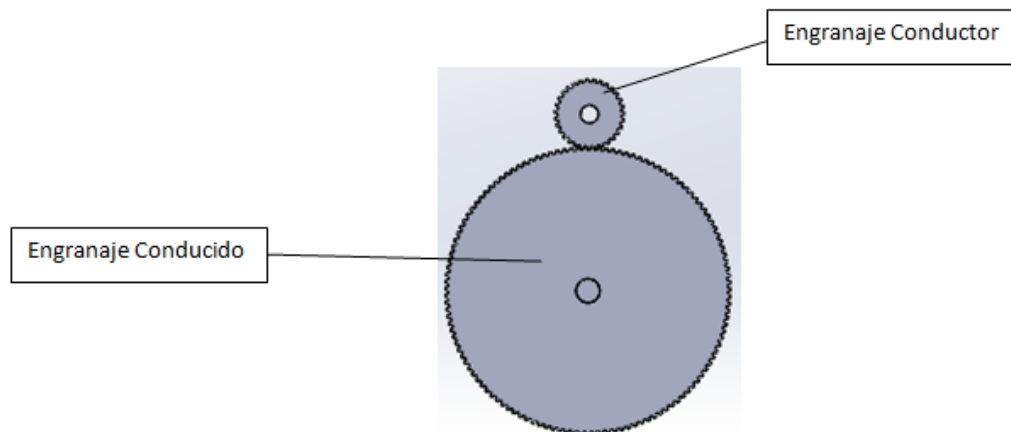


(Imagen 3)

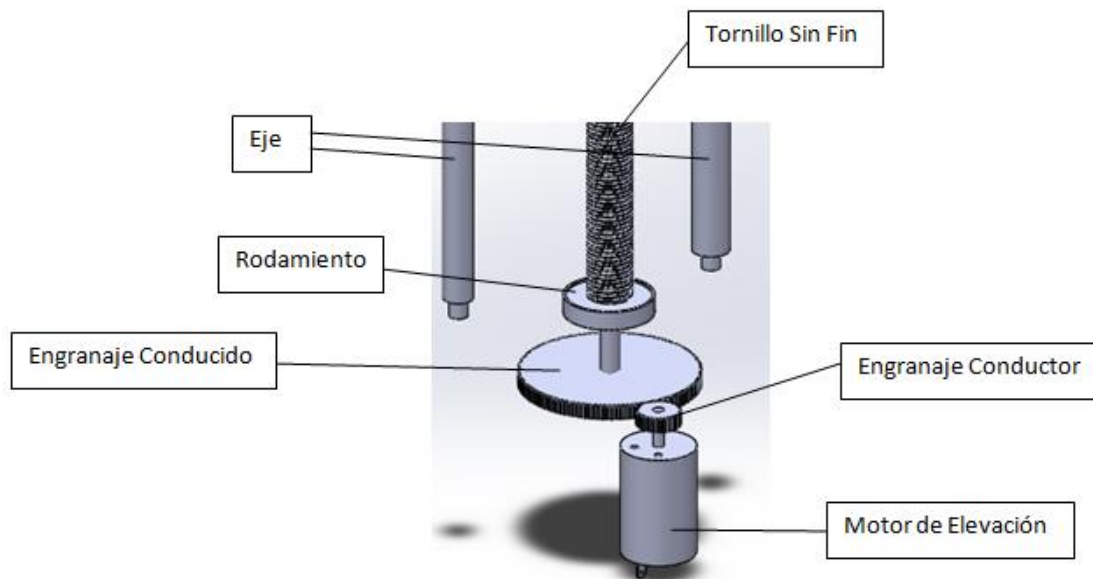
Este diseño basa su movimiento en un tornillo sin fin, el cual es controlado por medio de un motor con una unión de engranajes que los vincula. Los engranajes (*Imagen 4*) están diseñados de manera tal que haya una reducción de 6:1, para de esta manera poder reducir la velocidad y aumentar la fuerza requerida para subir la base guía y las mangueras.

El diámetro del engranaje Conductor es 10mm y el del Conducido es 60mm. De esta manera obtenemos la reducción deseada de  $i=6:1$ .





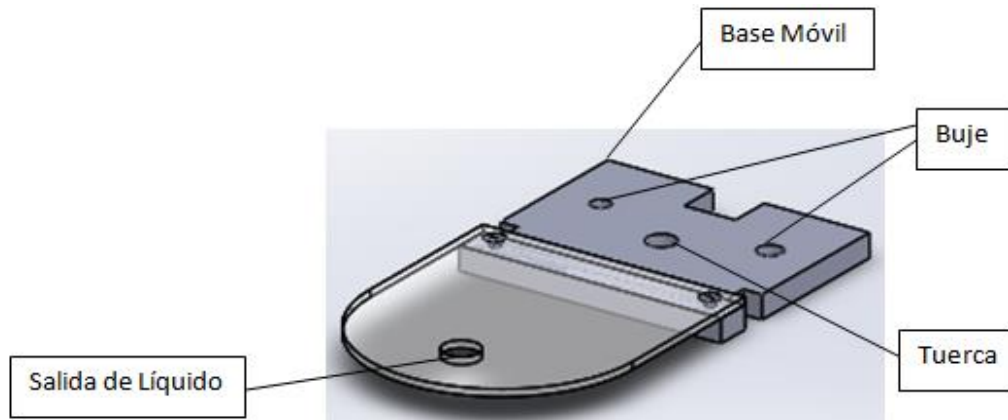
La base inferior y la base superior se encargan de mantener fijo al tornillo sin fin por medio de rulemanes, para de esta manera permitir el giro y restringir cualquier tipo de movimiento en los ejes, además de tener una baja fricción y un movimiento más armónico y libre. Por otro lado también son las encargadas de fijar los ejes guía las cuales tienen restringidos todos los grados de libertad y sirven para conducir la base móvil y restringirle el giro sobre su propio eje. A continuación, podemos visualizar los componentes descritos anteriormente y las dimensiones de los mismos (Imagen 4)



(Imagen 4)

El último componente que forma parte de este sistema es la base móvil. Dicha base es la encargada de contener el pico vertedor y las mangueras. Esta base tiene restringido dos grados de libertad, ya que no puede girar ni desplazarse en sentido horizontal y solo puede moverse en sentido vertical. Este componente tiene las dimensiones necesarias para caber dentro de la estructura, dos bujes de bronce para deslizarse por los ejes guía y una tuerca que la vincula al tornillo sin fin, de manera tal que el giro del tornillo hace subir o bajar dicha base.

A continuación podemos visualizar el diseño y dimensiones de la estructura mencionada.



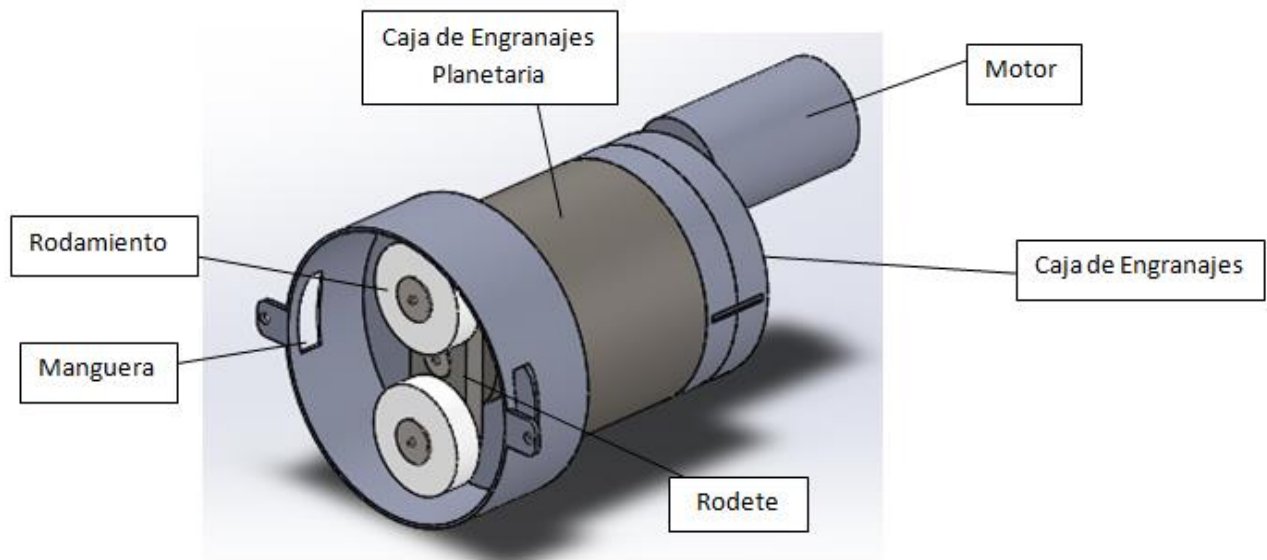
#### # Bombas peristálticas

Luego de realizar una investigación por todo el mercado, vimos que este tipo de bombas para el área industrial tienen un costo elevado. Por tal motivo se decidió por el diseño propio y por otro lado, de esta manera ajustamos las bombas según nuestras necesidades de potencia y tamaño.

El conjunto de componentes que forman parte del diseño de la bomba son los siguientes:

- Motor
- Caja de engranajes
- Caja de engranajes planetaria
- Carcasa
- Rodete
- Rodamientos
- Mangueras





El motor seleccionado es de 12V y con altas revoluciones, por tal motivo se utilizan dos cajas reductoras, las cuales en total hacen una relación de reducción de 36:1. Lo cual reduce la velocidad significativamente y aumenta la fuerza ampliamente.

Debido a la relación de transmisión que tenemos y sabiendo que la velocidad nominal del motor es de 3000 RPM, de esta relación podemos decir que:

$$i = 36:1 \Rightarrow \frac{3000RPM}{36} = 83,3RPM \text{ (Velocidad de rotación del Rodete)}$$

Teniendo esta velocidad podemos calcular los ml por minuto de la siguiente manera:

$$L = \text{Largo Manguera} = \frac{\text{Perímetro}}{2} = \pi \cdot r_{\text{Bomba}} = \pi \cdot 4,1\text{Cm} = 12,88\text{Cm}$$

Diámetro interno manguera = 6mm =>  $r_i = 0,3\text{ Cm}$

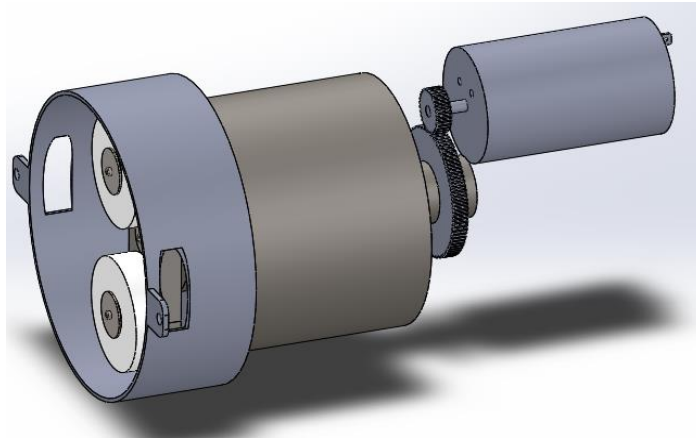
$$\text{Área Manguera} = \text{Área Manguera} = \pi \cdot r_i^2 = \pi \cdot 0,3^2\text{Cm}^2 = 0,28\text{Cm}^2$$

$$\text{Volumen Manguera} = \text{Área Manguera} \cdot L = 0,28\text{Cm}^2 \cdot 12,88\text{Cm} = 3,64\text{Cm}^3$$

De esta manera sabemos que tiene la capacidad de llenar  $3,64\text{Cm}^3$  por Vuelta.

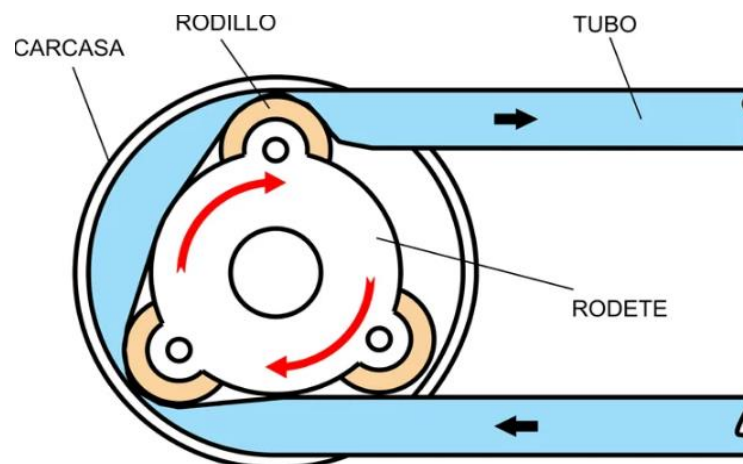
Y de esta manera, podemos determinar que según la relación de transmisión en 83,3 Vueltas llenamos 303,3 ml. Es decir las bombas tienen la capacidad de llenado de 303,3 ml/min.

Nuevamente todos los componentes fueron realizados en Aluminio, salvo la caja de engranajes planetaria la cual es un conjunto prearmado. La primer caja reductora es utilizada principalmente para vincular el motor con la caja de engranajes planetaria y entre ambas forman una reducción suficiente para nuestra necesidad.

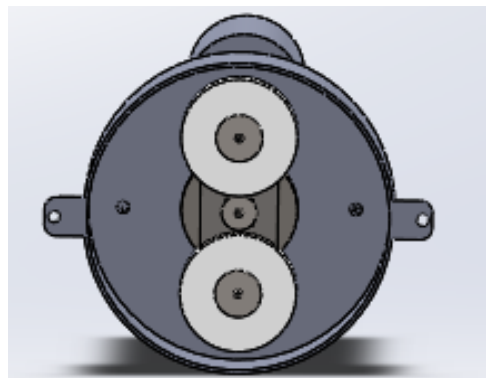


Finalmente la salida de la caja planetaria se une con el rodete. Dicho rodete contiene dos rodillos, los cuales son los encargados de presionar la manguera contra la carcasa y de esta manera generar una circulación de líquido en el sentido de giro.

A continuación podemos ver como es el funcionamiento de la bomba, la cual es representativa (Imagen 6) y nuestro diseño (Imagen 7)



(Imagen 6)

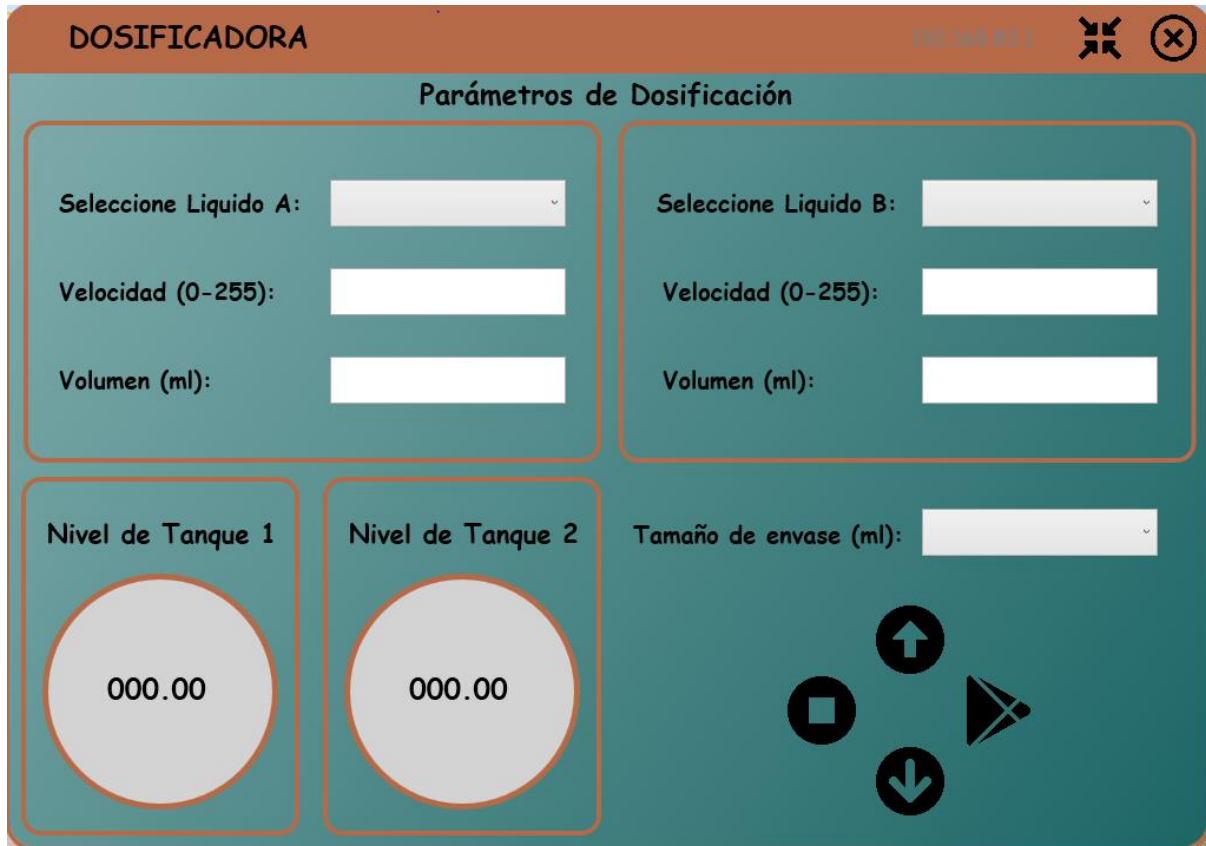


(Imagen 7)

## Diseño del software

Para la creación del software se optó por la herramienta Visual Studio 2019, específicamente la extensión de WPF (Windows Presentation Foundation).

El software se comunica con el controlador mediante el protocolo UDP, haciendo el mismo de servidor y el esp32 de cliente.



### Funcionalidades Aplicación (c#)

La aplicación posee una sencilla interfaz de control, la cual permite parametrizar la dosificación antes de dar inicio a la misma. Además, recibe datos de volúmenes de líquido en los tanques para así mostrarlos al usuario.

Bloques:

**Nombre del Líquido:** Se utiliza para generar un reporte de dosificaciones según cantidad dosificada, en que fecha y hora.

**Velocidad de Dosificación:** El sistema está preparado para poder variar la velocidad, por medio de la técnica de modulación de ancho de pulsos llamada PWM para el caso de este proyecto. no es posible porque a medida que bajamos la velocidad por PWM, aumenta el consumo y supera los niveles admitidos por el módulo IRF520.

**Volumen:** Valor de volumen a dosificar. Estos valores son enviados al ESP32.

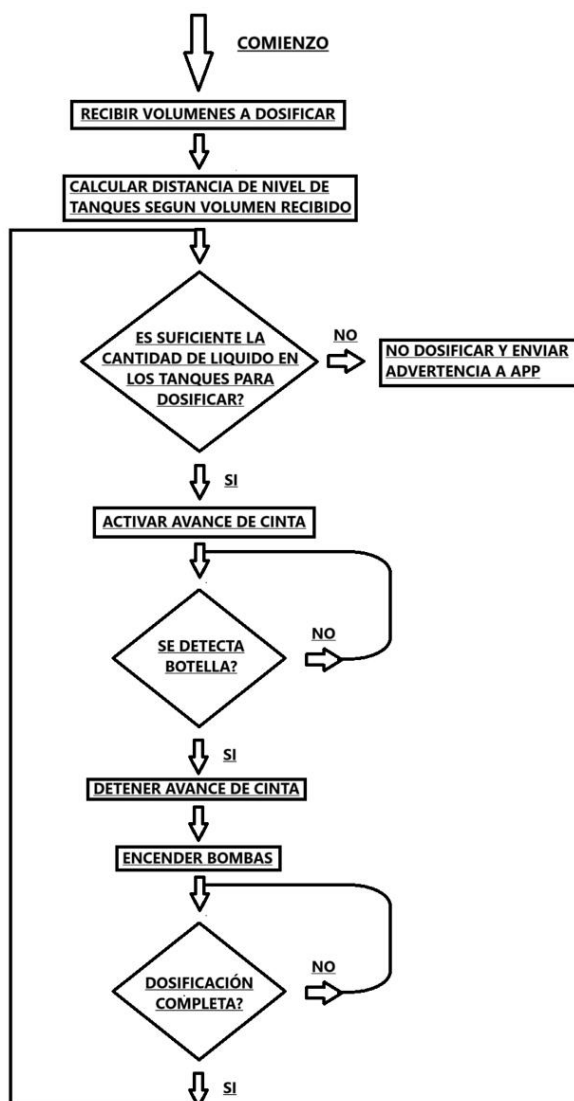
**Nivel de tanque:** cada 1 segundo la APP recibe información sobre la cantidad de líquido contenida en los tanques.

Control de altura: Mediante las flechas Arriba-Abajo el usuario puede ajustar la altura de la salida de líquido, esto permite utilizar distintos tamaños de botella.

Play-Stop: Mediante el botón PLAY se da inicio a la dosificación y se puede detener todo el proceso en cualquier instante con el botón de STOP.

### Funcionalidades ESP32

A continuación se presenta el diagrama de Flujo que explica el funcionamiento ideal del sistema.



Para esta etapa del proyecto aun no esta aplicado el ciclo de funcionamiento, debido a que se necesitaría la cinta transportadora para poder ajustar tiempos de espera para los cuales la cinta transportadora permite cambiar de botella

Funciones y sentencias importantes utilizadas en la programación

Conexión por Wifi: La aplicación procede a conectarse a una red. utilizando la librería Wifi.h para ello necesita las siguientes variables:

- Nombre de la red
- Contraseña de la red
- Ip de destino para comunicación
- protocolo de comunicación UDP

En cada loop del programa, se ejecuta la función `receive_packet()`, esto nos quiere decir que el `esp32` está escuchando en cada ciclo.

La función `receive_packet()` declara un paquete a recibir por `udp`, esta información es enviada en código ASCII, el cual luego es transformado a una cadena de texto.

Luego tiene una serie de sentencias `If`, mediante las cuales realiza las siguientes acciones. (subir base móvil, bajar base móvil, detener toda la actividad, y comenzar dosificación).

Envío de Datos: utilizando un timer el programa envía datos con el nivel de tanques cada 3 segundos, para ello realiza 20 mediciones en cada ultrasonido y realiza un promedio del mismo, esto se realiza para tener una mejor precisión en la medición.

## Código para Aplicación

### Código XAML para interfaz Gráfica

Para definir la interfaz gráfica el código combina controles visuales básicos (Border, Label, Button, ComboBox, TextBox, etc, y un diseño estructurado mediante el uso de Grid con definiciones de filas y columnas para los alineamientos, además, ciertos botones, labels, textboxes, etc. tienen un controlador de eventos definido en el backend que los vincula con una función de acción, como por ejemplo en el caso del botón "subir" que desencadena la función `subir_Click`.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:APPDOSIFICADORA"
    xmlns:Properties="clr-namespace:APPDOSIFICADORA.Properties"
    x:Class="APPDOSIFICADORA.MainWindow"
    mc:Ignorable="d"
    Title="MainWindow" Height="720" Width="1024"
    WindowStyle="None"
    AllowsTransparency="True"
    Background="Transparent"
    WindowStartupLocation="CenterScreen" Icon="gota-de-agua.ico" Initialized="Window_Initialized">

    <Grid>
        <Grid>
            <Border CornerRadius="25" BorderBrush="#FFB66A47" BorderThickness="4">
                <Border.Background>

                    <LinearGradientBrush>
                        <GradientStop Color="#FF82AEAE" Offset="0.0"/>
                        <GradientStop Color="#FF1D6666" Offset="1"/>
                    </LinearGradientBrush>
                </Border.Background>
            </Border>
        </Grid>
    </Grid>
```

```
</Border.Background>

<!--REGION Nivel de Tanques y T/envase -->
<Grid x:Name="Tanques" Height="324" VerticalAlignment="Bottom">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="256"/>
    <ColumnDefinition Width="256"/>
    <ColumnDefinition Width="256"/>
  </Grid.ColumnDefinitions>

  <Ellipse Grid.Column="0" Height="200" Width="200" Fill="LightGray" Stroke="#FFB66A47"
    StrokeThickness="5" Margin="0,60,0,0" />
  <Border Grid.Column="0" CornerRadius="15" BorderBrush="#FFB66A47"
    BorderThickness="4" Width="236" Height="304" />
  <Label Grid.Column="0" Content="Nivel de Tanque 1" Width="210" Height="50"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="22" FontWeight="Bold" Margin="8,-200,0,0" />
  <Label x:Name="Nivel_1" Grid.Column="0" Content="000.00" Width="100" Height="50"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="24" FontWeight="Bold" Margin="0,60,0,0" />
  <Label x:Name="Nivel_1vacio" Grid.Column="0" Content="vacio" Width="100" Height="50"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="24" FontWeight="Bold" Margin="0,150,-40,0"
    Foreground="Transparent"/>

  <Ellipse Grid.Column="1" Height="200" Width="200" Fill="LightGray" Stroke="#FFB66A47"
    StrokeThickness="5" Margin="0,60,0,0"/>
  <Border Grid.Column="1" CornerRadius="15" BorderBrush="#FFB66A47"
    BorderThickness="4" Width="236" Height="304" />
  <Label Grid.Column="1" Content="Nivel de Tanque 2" Width="210" Height="50"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="22" FontWeight="Bold" Margin="8,-200,0,0" />
  <Label x:Name="Nivel_2" Grid.Column="1" Content="000.00" Width="100" Height="50"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="24" FontWeight="Bold" Margin="0,60,0,0" />
  <Label x:Name="Nivel_2vacio" Grid.Column="1" Content="vacio" Width="100" Height="50"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="24" FontWeight="Bold" Margin="0,150,-40,0"
    Foreground="Transparent"/>

  <Label Grid.Column="2" Content="Tamaño de envase (ml):" Width="240" Height="40"
    Grid.Row="1" FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,-206,240,0"/>
  <ComboBox Grid.Column="2" Width="200" Height="40" Margin="220,-210,0,0"
    FontSize="18">
    <ComboBoxItem Content="250"/>
    <ComboBoxItem Content="500"/>
    <ComboBoxItem Content="1000"/>
    <ComboBoxItem Content="1500"/>
    <ComboBoxItem Content="Otro"/>
  </ComboBox>

</Grid>
<!--#ENDREGION-->
</Border>

<!--Header-->
<Grid x:Name="Header" Height="60" VerticalAlignment="Top">
  <Border CornerRadius="25,25,0,0" Background="#FFB66A47"/>
  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition Width="150"/>
      <ColumnDefinition Width="75"/>
      <ColumnDefinition Width="75"/>
    </Grid.ColumnDefinitions>

    <!--REGION Titulo-->
    <Image Source="/Recursos/IMG/gota-de-agua.ico" Margin="10,8,0,12" Height="40"
      HorizontalAlignment="Left" />
    </--REGION Titulo-->
  </Grid>
</Grid>
</Header-->
```



```
<Label Content="DOSIFICADORA" HorizontalAlignment="Left" Height="40"
Margin="49,5,0,0" VerticalAlignment="Top" Width="302" FontFamily="Comic Sans MS" FontSize="25"
FontWeight="Bold"/>
<!--#ENDREGION-->

<!--REGION Boton Cerrar-->
<Button x:Name="Cerrar" Grid.Column="3" Background="Transparent"
BorderBrush="Transparent" VerticalAlignment="Center" Cursor="Hand" Margin="20,6,10,6" Click="Close"
>
    <Image Source="/Recursos/IMG/times-circle-regular.png" Height="45"/>

</Button>
<!--#ENDREGION-->

<!--REGION Boton Minimizar-->
<Button Grid.Column="2" Background="Transparent" BorderBrush="Transparent"
VerticalAlignment="Center" Cursor="Hand" Margin="33,10,0,10" Click="Min" >

    <Image Source="/Recursos/IMG/compress-arrows-alt-solid.png" Height="36"
HorizontalAlignment="Right" Width="65"/>
</Button>

<Label Grid.Column="1" Name="Label_ip" Content="" Height="60" FontFamily="Comic Sans
MS" FontSize="18" Margin="10,12,35,-12" Foreground="#FF817B7B" Grid.ColumnSpan="2"/>

</Grid>

</Grid>

</Grid>

<Label Content="Parámetros de Dosificación" Width="330" Height="50" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="24" FontWeight="Bold" Margin="0,-560,0,0" />
<Grid HorizontalAlignment="Left" Height="290" Margin="0,100,0,0" VerticalAlignment="Top"
Width="1024">

    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>

    <Border Grid.Column="0" CornerRadius="15" BorderBrush="#FFB66A47" BorderThickness="4"
Width="490" Height="290" Margin="10,0,0,0"/>
    <Label Grid.Column="0" Content="Seleccione Liquido A:" Width="230" Height="40" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,-150,200,0"/>
    <ComboBox x:Name="textname1" Grid.Column="0" Width="200" Height="40"
Margin="276,50,36,200" FontSize="18" />
    <Label Grid.Column="0" Content="Velocidad (0-255):" Width="230" Height="40" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,0,200,0"/>
    <TextBox x:Name="velocidadb1" Grid.Column="0" Width="200" Height="40" Margin="276,0,36,0"
FontSize="18"/>

    <Label Grid.Column="0" Content="Volumen (ml):" Width="230" Height="40" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,+150,200,0"/>
    <TextBox x:Name="volumen_a" Grid.Column="0" Width="200" Height="40"
Margin="276,150,36,0" FontSize="18"/>

    <Border Grid.Column="1" CornerRadius="15" BorderBrush="#FFB66A47" BorderThickness="4"
Width="490" Height="290" Margin="-5,0,0,0"/>
    <Label Grid.Column="1" Content="Seleccione Liquido B:" Width="230" Height="40" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,-150,210,0"/>
    <ComboBox Grid.Column="1" Width="200" Height="40" Margin="266,50,46,200" FontSize="18"
/>
    <Label Grid.Column="1" Content="Velocidad (0-255):" Width="230" Height="40" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,0,200,0"/>
```

```
<TextBox x:Name="velocidadb2" Grid.Column="1" Width="200" Height="40" Margin="266,0,46,0"
FontSize="18"/>
```

```
<Label Grid.Column="1" Content="Volumen (ml):" Width="230" Height="40" Grid.Row="1"
FontFamily="Comic Sans MS" FontSize="20" FontWeight="Bold" Margin="0,+150,200,0"/>
<TextBox x:Name="volumen_b" Grid.Column="1" Width="200" Height="40"
Margin="266,150,46,0" FontSize="18"/>
</Grid>
```

```
<Grid HorizontalAlignment="Left" Height="240" Margin="512,0,0,0" VerticalAlignment="Bottom"
Width="512">
```

```
<!--REGION Boton Subir-->
<Button x:Name="subir" Grid.Column="0" Background="Transparent" BorderBrush="Transparent"
VerticalAlignment="Center" Cursor="Hand" Margin="0,-120,0,0" Width="80" Height="80"
Click="subir_Click" >
```

```
<Image Source="/Recursos/IMG/arrow-alt-circle-up-solid.png" Height="60"
HorizontalAlignment="Right" Width="60"/>
```

```
</Button>
<!--#ENDREGION-->
<!--REGION Boton Bajar-->
<Button x:Name="bajar" Grid.Column="0" Background="Transparent" BorderBrush="Transparent"
VerticalAlignment="Center" Cursor="Hand" Margin="0,120,0,0" Width="80" Height="80"
Click="bajar_Click" >
```

```
<Image Source="/Recursos/IMG/arrow-circle-down-solid.png" Height="60"
HorizontalAlignment="Right" Width="60"/>
```

```
</Button>
<!--#ENDREGION-->
<!--REGION Boton Comenzar-->
<Button x:Name="comenzar" Grid.Column="1" Background="Transparent"
BorderBrush="Transparent" VerticalAlignment="Center" Cursor="Hand" Margin="150,0,0,0" Width="80"
Height="80" Click="comenzar_Click" >
```

```
<Image Source="/Recursos/IMG/google-play-brands.png" Height="60"
HorizontalAlignment="Right" Width="60"/>
```

```
</Button>
<!--#ENDREGION-->
```

```
<!--REGION Boton Detener-->
<Button x:Name="detener" Grid.Column="1" Background="Transparent"
BorderBrush="Transparent" VerticalAlignment="Center" Cursor="Hand" Margin="-150,0,0,0" Width="80"
Height="80" Click="detener_Click" >
```

```
<Image Source="/Recursos/IMG/stop-circle-solid.png" Height="60" HorizontalAlignment="Right"
Width="60"/>
```

```
</Button>
<!--#ENDREGION-->
</Grid>
</Window>
```

### **Codigo C#**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
```



```
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using System.Reflection;

using System.Threading;
using System.Net;
using System.Net.Sockets;

namespace APPDOSIFICADORA
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        String linea_com;
        Boolean UDPC = false;
        Int32 B, C;

        private void Window_Initialized(object sender, EventArgs e)
        {
            string pathToData = AppDomain.CurrentDomain.BaseDirectory;
            #if DEBUG
                pathToData = pathToData.Remove(pathToData.Length - 11);
            #endif

            pathToData = System.IO.Path.Combine(pathToData, "Recursos\\BN\\ConfIP.txt");

            StreamReader leer = new StreamReader(pathToData);
            linea_com = leer.ReadLine();

            Label_ip.Content = linea_com;

            Thread thdUDPServer = new Thread(new ThreadStart(serverThread)); /// abro hilo para Servidor UDP
            thdUDPServer.Start();
        }

        public void serverThread() //SERVER
        {
            UdpClient udpServer = new UdpClient(8080);

            while (UDPC == false)
            {
                IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 8080);
                Byte[] receiveBytes = udpServer.Receive(ref RemoteIpEndPoint);
                string returnData = Encoding.ASCII.GetString(receiveBytes);

                B = returnData.IndexOf("B");
                C = returnData.IndexOf("C");

                this.Dispatcher.Invoke(() => Nivel_1.Content = returnData.Substring(0,B));
                this.Dispatcher.Invoke(() => Nivel_2.Content = returnData.Substring(B+1,C-B-1));
            }
            udpServer.Close();
        }

        private void enviar_data (String e)
        {
            UdpClient udpClient = new UdpClient();
        }
    }
}
```

```
        udpClient.Connect(linea_com, 8080);
        Byte[] senddata = Encoding.ASCII.GetBytes(e);
        udpClient.Send(senddata, senddata.Length);
    }

    // Botones -----

    private void subir_Click(object sender, RoutedEventArgs e)
    {
        enviar_data("subir");
    }

    private void bajar_Click(object sender, RoutedEventArgs e)
    {
        enviar_data("bajar");
    }

    private void detener_Click(object sender, RoutedEventArgs e)
    {
        enviar_data("stop");
    }

    private void comenzar_Click(object sender, RoutedEventArgs e)
    {
        enviar_data("play" + volumen_a.Text + "B" + volumen_b.Text);
    }

    private void Min(object sender, RoutedEventArgs e)
    {
        this.WindowState = WindowState.Minimized;
    }

    private void Close(object sender, RoutedEventArgs e)
    {
        UDPC = true;
        Close();
    }

}
}
```

### **Codigo ESP-32**

```
#include <WiFi.h>
#include <WiFiUdp.h>
#include "esp_system.h"

//----- Configuración de LED, WiFi y variables -----//
#define ON_Board_LED 2
const char* ssid = "CORONEL";
const char* password = "1234asdf";

int volumenadosi = 250;
int volumenadosi2 = 250;
int auxposvol = 0;
```

```
bool godosificado = false;
float distanciavol = 0;
float distanciavol2 = 0;
float distins = 0;
float Volumen1 = 0;
float Volumen2 = 0;
bool doscompleta1 = false;
bool doscompleta2 = false;

//----- Configuración de pines de Ultrasonido -----//
const int trigPin = 16; // US1
const int echoPin = 17;
const int trigPin2 = 32; // US2
const int echoPin2 = 33;
const int trigPin3 = 12; // US3
const int echoPin3 = 13;

#define SOUND_VELOCITY 0.034
long duration;
float distanceCm1, distanceCm2, distanceCm3;

//----- Configuración de pines de motores PWM -----//
const int in1 = 4;
const int in2 = 5;
int en1 = 18;
const int pwmfreq = 5000;
const int canal = 0;
const int resolucion = 8;
String aux_lectura;

int velocidad_altura = 0;
#define velocidad_valor 255

int enb1 = 25;
const int canalb = 1;
int velocidad_bomba1 = 0;

int enb2 = 26;
const int canalc = 2;
int velocidad_bomba2 = 0;

//----- Configuración de Timer -----//
volatile boolean Enviar_data = false;
volatile int contador = 0;
int contadorEnvio = 0;
int UStiempo = 1;
const int wdtTimeout = 1000;
hw_timer_t *timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

//----- Configuración de UDP y WiFi -----//
unsigned int localPort = 8080;
IPAddress DestinationIP(192,168,137,1);
WiFiUDP udp;

char packetBuffer[50];
String inString;
byte SendBuffer[50];

//----- ISR del Timer -----//
void IRAM_ATTR OnTimer() {
    portENTER_CRITICAL_ISR(&timerMux);
    contador += 1;
    if (contador == UStiempo){
        Enviar_data = true;
        contador = 0;
    }
}
```

```
    contadorEnvio += 1;
  }
  portEXIT_CRITICAL_ISR(&timerMux);
}

//----- Setup -----//
void setup() {
  Serial.begin(115200);
  delay(500);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin3, INPUT);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(en1, OUTPUT);

  ledcSetup(canal, pwmfreq, resolucion);
  ledcAttachPin(en1, canal);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

  pinMode(enb1, OUTPUT);
  ledcSetup(canalb, pwmfreq, resolucion);
  ledcAttachPin(enb1, canalb);

  pinMode(enb2, OUTPUT);
  ledcSetup(canalc, pwmfreq, resolucion);
  ledcAttachPin(enb2, canalc);

  WiFi.begin(ssid, password);
  pinMode(ON_Board_LED, OUTPUT);
  digitalWrite(ON_Board_LED, HIGH);

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    digitalWrite(ON_Board_LED, LOW);
    delay(250);
    digitalWrite(ON_Board_LED, HIGH);
    delay(250);
  }

  digitalWrite(ON_Board_LED, HIGH);
  Serial.println("\nConnected to: " + String(ssid));
  Serial.println("IP Address: " + WiFi.localIP().toString());

  udp.begin(localPort);
  Serial.println("Local Port: " + String(localPort));

  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &OnTimer, true);
  timerAlarmWrite(timer, wdtTimeout * 1000, true);
  timerAlarmEnable(timer);
}

//----- Loop -----//
void loop() {
  receive_packet();

  // Medir distancia solo cada 1 segundo
  if (Enviar_data) {
```

```
distanceCm1 = medirDistancia(trigPin, echoPin);
// Serial.println("tk 1:");
//Serial.println(distanceCm1);
distanceCm2 = medirDistancia(trigPin2, echoPin2);
// Serial.println("tk 2:");
// Serial.println(distanceCm2);
distanceCm3 = medirDistancia(trigPin3, echoPin3);

Enviar_data = false;
}

// Enviar datos cada 3 segundos
if (contadorEnvio >= 3) {
    senseUS_and_send_packet();
    contadorEnvio = 0;
}

// Control de dosificación y altura
ledcWrite(canal, velocidad_altura);
ledcWrite(canalb, velocidad_bomba1);
ledcWrite(canalc, velocidad_bomba2);

if (godosificado) {
    if (distanceCm3 < 18 && distanceCm3 > 4) {
        //Serial.println("Cinta detenida");
        //Serial.println("Motores encendidos");
        if ( volumenadosi > 0 && doscompleta1 == false){
            velocidad_bomba1 = 255;
        }
        if ( volumenadosi2 > 0 && doscompleta2 == false){
            velocidad_bomba2 = 255;
        }
    }

    if (distanciavol <= distanceCm1) {
        velocidad_bomba1 = 0;
        Serial.println("Bomba 1 detenida");
        doscompleta1 = true;
    }
    if (distanciavol2 <= distanceCm2) {
        velocidad_bomba2 = 0;
        Serial.println("Bomba 2 detenida");
        doscompleta2 = true;
    }
    if (doscompleta1 && doscompleta2) {
        doscompleta1 = doscompleta2 = godosificado = false;
        Serial.println("Dosificación completa");
    }
}
}
}

//----- Funciones de Ultrasonido y UDP -----//
void receive_packet() {
    int packetSize = udp.parsePacket();
    if (packetSize) {
        int len = udp.read(packetBuffer, 255);
        if (len > 0) packetBuffer[len] = 0;
        aux_lectura = String(packetBuffer);

        if (aux_lectura == "subir") {
            digitalWrite(in1, HIGH);
            digitalWrite(in2, LOW);
            velocidad_altura = velocidad_valor;
        } else if (aux_lectura == "bajar") {
            digitalWrite(in1, LOW);
            digitalWrite(in2, HIGH);
        }
    }
}
```

```

    velocidad_altura = velocidad_valor;
} else if (aux_lectura == "stop") {
    velocidad_altura = velocidad_bomba1 = velocidad_bomba2 = 0;
    doscompleta1 = false;
    doscompleta2 = false;
    godosificado = false;
} else if (aux_lectura.startsWith("play")) {
    auxposvol = aux_lectura.indexOf("B");
    volumenadosi = aux_lectura.substring(4, auxposvol+1).toInt();
    volumenadosi2 = aux_lectura.substring(auxposvol+1).toInt();
    godosificado = true;

    distanciavol = ((volumenadosi / (3.14 * (5.5 * 5.5)))) + distanceCm1;
    distanciavol2 = ((volumenadosi2 / (3.14 * (5.5 * 5.5)))) + distanceCm2;
    Serial.printf("Distancia vol 1 : %f",distanciavol);
    Serial.printf("Distancia vol 2 : %f",distanciavol2);
}
}
}

void senseUS_and_send_packet() {
    Volumen1 = (37.7 - distanceCm1) * (3.14 * (5.5 * 5.5));
    Volumen2 = (37.7 - distanceCm2) * (3.14 * (5.5 * 5.5));
    inString = String(Volumen1) + "B" + String(Volumen2) + "C";
    inString.getBytes(SendBuffer, 50);
    udp.beginPacket(DestinationIP, 8080);
    udp.write(SendBuffer, 50);
    udp.endPacket();
}

float medirDistancia(int trigIndex, int ecoindex) {

    float suma = 0;

    for (int i=0;i<10;i++) {
        digitalWrite(trigIndex, LOW);
        delayMicroseconds(2);
        digitalWrite(trigIndex, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigIndex, LOW);
        duration = pulseIn(ecoindex, HIGH);

        suma = suma + ( duration * SOUND_VELOCITY / 2);
        delay(30);
    }
    return suma/10;
}

```

### Costos:

| CANT | producto                    | precio x unidad(pesos) | Precio total(pesos) |
|------|-----------------------------|------------------------|---------------------|
| 1    | Esp-32/Nodemcu V1.0         | 12400                  | 12400               |
| 1    | lm7805                      | 2000                   | 2000                |
| 3    | Hc-sr04 Sensor De Distancia | 2000                   | 6000                |

|    |                                    |       |       |
|----|------------------------------------|-------|-------|
|    | Ultrasónico                        |       |       |
| 1  | Doble Puente H Driver L298n        | 4000  | 4000  |
| 1  | Modulo Driver Mosfet Irf520 24v 5a | 3000  | 6000  |
| 1  | Varilla roscada 1/4                | 2500  | 2500  |
| 2  | Bujes                              | 600   | 1200  |
| 2  | Bomba Peristáltica                 | 45000 | 90000 |
| 2  | Engranajes                         | 5000  | 10000 |
| 1  | Motor 5v                           | 2000  | 2000  |
| 1  | Chapa 1mm                          | 3000  | 3000  |
| 2  | Manguera de Látex                  | 2500  | 5000  |
| 18 | Diodo 1n4007                       | 250   | 4500  |
| 2  | Capacitor                          | 160   | 320   |
| 2  | Capacitor elect.                   | 170   | 340   |
| 8  | Resistores                         | 170   | 1360  |

**Costo estimado total en dólares: 145,24 USD. TC = 1037 al 11/12/2024**

## Etapa de desarrollo

En ésta etapa se produce la elaboración del producto en todo su conjunto tomando en cuenta el diseño obtenido en la etapa anterior.



Una fuente de 12 v alimenta al puente H que controla el motor del mecanismo de elevación de las mangueras y un I238 que baja el voltaje a 5v que alimenta al resto de los componentes electrónicos del sistema.

El circuito está compuesto por un esp32 que controla al puente h, recibe y envía información sobre el estado del sistema mediante una comunicación por protocolo IP. Se conecta con el puente h con los pines P5, P4 y P18, con los sensores de distancia ultrasónicos que miden la altura del líquido en los tanques por los pines P14, P27, P16 y P17, y con el sensor ultrasónico que detecta la proximidad del recipiente a llenar mediante los pines P13 y P12.

La información enviada es recibida por el usuario a través de una interfaz creada con visual studio que le permite visualizar los datos de llenado y etc y modificar algunos parámetros como la densidad del líquido a dosificar y la velocidad de llenado.

Las bombas encargadas de la dosificación del líquido son controladas por un driver conectado a los pines P25 y P26, que controlan los motores que trabajan de acuerdo a los parámetros seleccionados previamente por el usuario.

Para el bombeo del líquido se optó por dos bombas peristálticas debido a la precisión con la que permite dosificar los fluidos mediante la compresión de los tubos por los que se desplaza el líquido, a su vez, es una ventaja que la bomba esté separada del líquido ya que le brinda una mayor durabilidad y permite cumplir con las normas requeridas para la industria alimenticia. Se eligieron mangueras de silicona curada al platino para la dosificación de líquidos debido a que este material es atóxico, extremadamente flexible, tiene una gran longevidad a la flexión y tiene una superficie interna extremadamente lisa que ayuda a evitar que queden atrapadas partículas durante la transferencia de fluidos sensibles.



Durante el desarrollo del proyecto se llevaron a cabo consultas a distintos profesores para cumplir con la realización del proyecto y avanzar en el mismo. Dichas consultas se encuentran documentadas en la sección Anexo, donde se puede encontrar una digitalización de las mismas.

## Etapa de testeos:

En esta etapa se realiza una evaluación del producto para verificar que todo funcione correctamente y se analicen posibles mejoras.

Los testeos se fueron haciendo de forma progresiva, probando primero el funcionamiento de los sensores, luego del controlamiento del motor para la regulación de altura, el control de los motores de las bombas peristálticas y finalmente del sistema en su conjunto con el software y la estructura mecánica.

De estas pruebas pudimos determinar la velocidad para la dosificación del líquido, se pudieron hacer ajustes en cuanto a las distancias en las que se debe detectar el recipiente a llenar y se logró determinar si la interfaz era amigable para el usuario. Se realizaron calibraciones para tener un mejor error en el volumen de llenado y una mejor lectura del volumen actual en los tanques.

Posibilidades de mejoras:

- Utilizar sensores de mayor confianza en los tanques para lograr una mejor precisión en las medidas del volumen del líquido.
- Implementar el uso de bombas peristálticas de menor consumo de corriente.
- Regular la velocidad de las bombas peristálticas de acuerdo a los requerimientos del líquido a dosificar.
- Tener un sistema automático de llenado, en el cual el usuario pueda poner la cantidad deseada de botellas y con una cinta transportadora movilizar los recipientes.
- Guardado automático del inventario realizado.
- Advertencia del volumen bajo en los tanques.

