UNIVERSITAT JAUME I

ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS

MÁSTER EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

ESTUDIO Y OPTIMIZACIÓN DE LA EFICIENCIA EN LA TRANSFERENCIA DE OXÍGENO DE PARRILLAS DE DIFUSORES PARA REACTORES BIOLÓGICOS EN ESTACIONES DEPURADORAS DE AGUAS RESIDUALES MEDIANTE EL USO DE ENSAYOS EXPERIMENTALES

TRABAJO DE FIN DE MÁSTER

AUTOR: JOSE MANUEL APARICI MOLINA

DIRECTORES: RAÚL MARTINEZ CUENCA Y SERGIO CHIVA

FECHA: JULIO DE 2019

Contenido

Listado de figuras

I. MEMORIA

1. Introducción

El presente trabajo se enmarca en el contexto de un convenio de investigación FACSA-UJI con el objetivo de estudiar la transferencia de oxígeno en fluidos bifásicos y determinar la disposición óptima de los difusores en el seno de un fluido de modo que la transferencia mencionada sea óptima. El principal interés de FACSA en este proyecto está relacionado con su aplicación directa en tanques en donde se utilizan procesos de aireación, fundamentalmente en reactores biológicos para tratamiento secundario, de modo que una disposición óptima de los difusores conllevaría una reducción del gasto energético ya que no haría falta introducir tanto caudal de aire para tratar la misma cantidad de agua.

Esta etapa cobra gran importancia en su diseño debido al gasto energético asociado. El compresor encargado de suministras el aire a través de los difusores representa un 60% del consumo eléctrico total de la planta por lo que podemos hacernos una idea de la importancia de la optimización del diseño.

2. Objetivo

El objetivo del proyecto será el diseño de un tanque de laboratorio que permita determinar la disposición óptima de los difusores en el seno de un depósito, es decir, la disposición en que la transferencia de oxígeno es máxima y por lo tanto se produce un ahorro de energía. Se necesita, por tanto, que la colocación de los distintos difusores a testear pueda hacerse fácilmente, sin necesidad de vaciar el tanque ni sumergirse para su montaje.

3. Alcance

Este estudio se limita al diseño, construcción y puesta a punto de una instalación compuesta por un tanque cilíndrico vertical al que se le incorporen distintos elementos que permitan llevar a cabo la caracterización hidrodinámica y de transferencia de oxígeno para distintas configuraciones de difusores. El tanque ha de estar preparado para alojar 8000 litros de agua, incorporar una estructura interna que permita el rápido intercambio de las parrillas de difusores, así como permitir el visionado directo de su interior a través de uno de sus laterales. Además, se ha de incluir tanto el hardware como el software necesario para realizar el escaneado de sondas de medida por el interior del tanque.

Tras el diseño de la instalación se verificará su funcionamiento Tras el diseño de la instalación se verificará su funcionamiento realizando un muestreo en condiciones similares a las de uso habitual y con los equipos de medida en funcionamiento.

4. Metodología

A continuación, se explicará el conjunto de procedimientos utilizados para alcanzar el objetivo de esta investigación científica.

La primera etapa consiste en el diseño y dimensionamiento del depósito que alberga el fluido, los difusores y sobre el cual se realizan las diferentes medidas gracias a la acción de dos motores que incorporan las sondas de medida y el Vectrino. Estos motores, a su vez, son controlados por un microcontrolador, un Arduino en nuestro caso, por lo que ha sido necesario diseñar el cuadro eléctrico donde se ubicaría el controlador y los demás componentes del cuadro. Más adelante, en el apartado 6.4, se explica el diseño del hardware relacionado con el proyecto.

Para obtener los resultados ha sido necesario realizar un programa para el Arduino con el código necesario para que, desde una pantalla externa al depósito, un usuario pueda mover los motores, tomar medidas y obtener los resultados adecuados de una manera sencilla. El código se presenta en el apartado **IV ANEXOS.**

El código se realizará con el software MATLAB y se le transmitirá a un microcontrolador ARDUINO para que gobierne el sistema. El Arduino es capaz de controlar el movimiento de los motores enviando la información necesaria a los drivers de los mismos. Uno de los problemas que nos encontramos a continuación es el número de tarjeta controladora, ya que disponíamos de una tarjeta controladora para los dos motores. La solución a este problema fue la incorporación de un relé cuádruple cuya conmutación es controlada por el Arduino y de ese modo, conmutado o no, nos permitía actuar sobre un motor u otro.

La obtención de los resultados, es decir, el movimiento de las sondas y la toma de medidas, se realiza a través de una interfaz gráfica de modo que su uso sea intuitivo y cualquier usuario ajeno o no al proyecto pueda hacer uso de la instalación. Este ha sido el motivo que nos ha llevado a realizar la programación del microcontrolador con MATLAB ya que permite crear una interfaz gráfica gracias al GUIDE. En la figura 1 se puede ver la interfaz gráfica mediante la cual se controla la instalación.

Finalmente, con la interfaz operativa y habiendo calibrado los motores se demostrará el funcionamiento de la instalación tomando resultados de concentración de oxígeno y velocidad con los difusores en funcionamiento. Estos resultados se muestran en el apartado **7. RESULTADOS EXPERIMENTALES**.

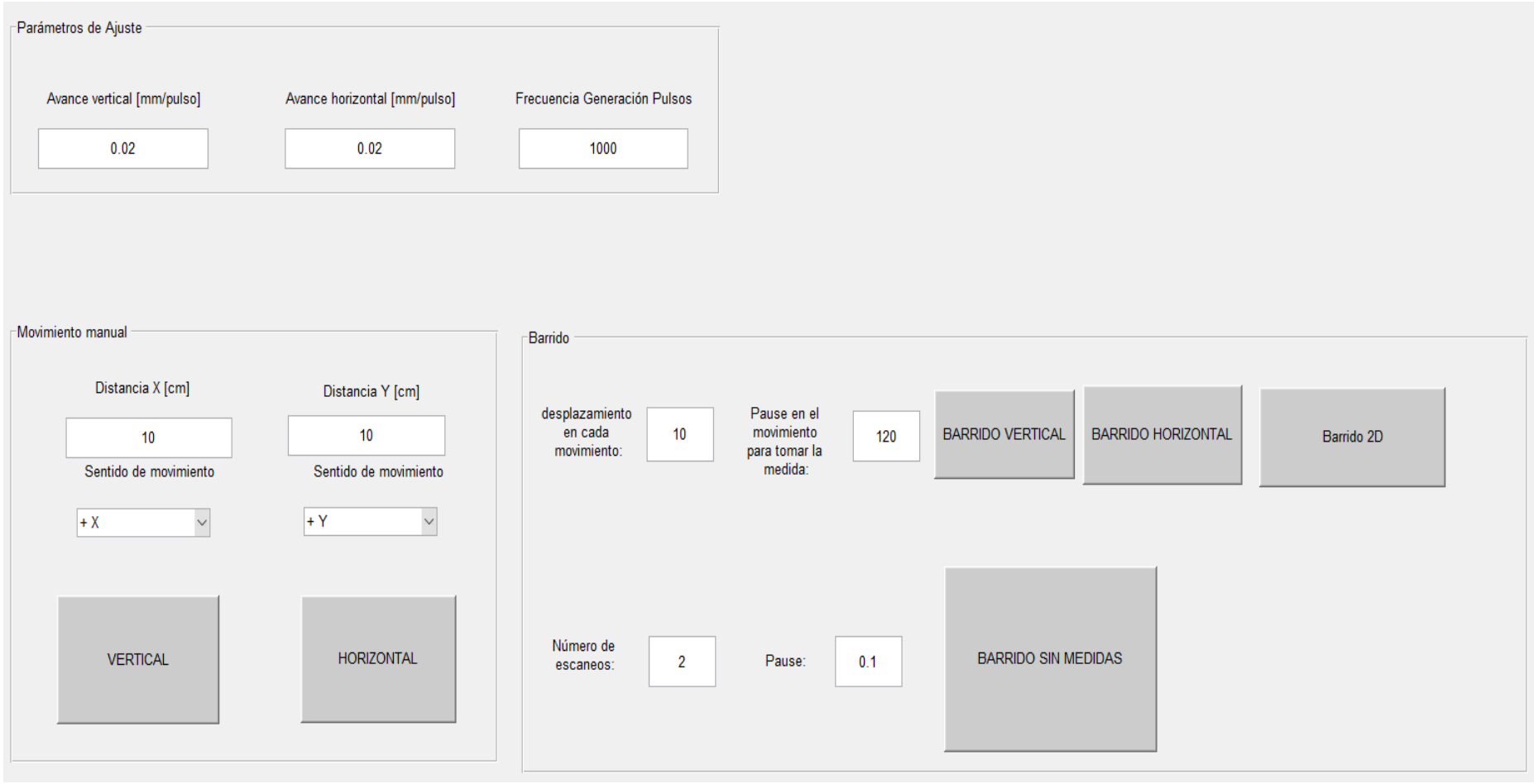


Figura 1. Interfaz gráfica de control

5. Estado del arte

Se denomina flujo multifásico a aquel formado por dos o más fases las cuales fluyen de manera simultánea. La superficie entre ambas fases se denomina interfase. Un flujo bifásico, el cual se encuentra formado por dos fases, está formado por una fase continua y otra dispersa, la cual puede representarse en forma de burbujas de distintos tamaños. La topología de las fases es una característica clave de los fluidos bifásicos ya que influye en la interacción entre las distintas fases. Esta topología influirá en la masa, el momento y las ratios de transferencia de energía. De este modo podemos encontrarnos con diferentes topologías o regímenes de flujo en función de las propiedades del fluido, las propiedades del conducto, los caudales de cada fase y de la configuración de la entrada.

El fluido bifásico líquido-gas es una de las combinaciones más complejas y de las más estudiadas debido a la multiplicidad de aplicaciones industriales de este flujo, tales como torres de destilación en refinerías, centrales térmicas en sus ciclos Rankine y Bryton, centrales nucleares para accionar las turbinas, centrales solares y también en las estaciones depuradoras de aguas residuales en la etapa de aireación.

En las EDAR, como se verá en el siguiente apartado, se llevan a cabo unos procesos de depuración de aguas que provienen de distintos sectores. Estos procesos de depuración tienen diferente naturaleza de modo que son diseñados de manera independiente. En el siguiente apartado se hablará de las diferentes etapas que forman el proceso, pero en concreto se profundizará en la etapa de tratamiento biológico, ya que es en la cual se emplean los difusores para alimentar los microorganismos disueltos en la piscina.

Se hablará también de los procedimientos actuales de diseño de sistemas de aireación, de la importancia del estudio de los fluidos bifásicos y de los distintos elementos de medición utilizados para obtener los resultados del proyecto, los cuales nos llevan a proponer una distribución óptima de los difusores.

5.1. EDAR

El objetivo de las EDAR es el de tratar las aguas negras (aguas tras ser utilizadas en hogares comercios, agricultura…) para transformarlas en aguas con unas mejores características y situar el caudal de salida bajo el cumplimiento de unas determinadas especificaciones marcadas por la ley.

Esas aguas son recogidas mediante sistemas de alcantarillado y conducidas hacia las EDAR, que tras procesaras las enviará el medio natural en las mejores condiciones.

5.1.1. Etapas de depuración

En las depuradoras el agua pasa por un tratamiento formado por varias etapas:

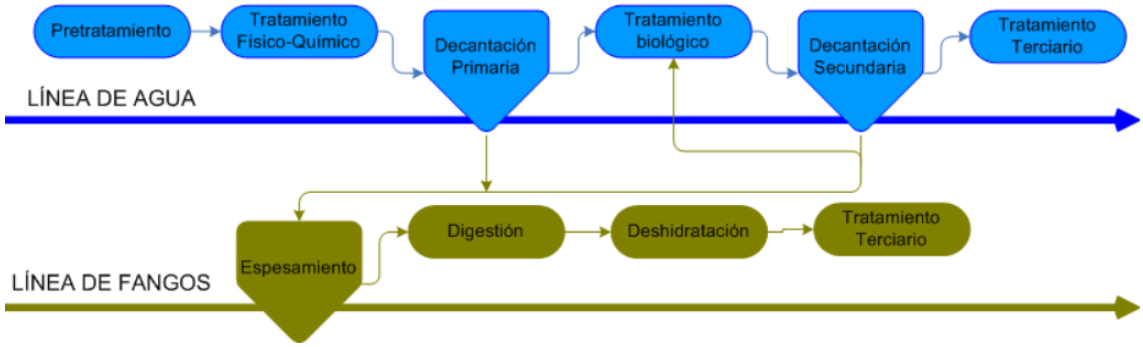


Figura 2. Proceso EDAR

5.1.2. Pretratamiento

El pretratamiento es el proceso en el cual se eliminan los sólidos más grandes y pesados, las arenas y las grasas.

5.1.2.1. Desbaste y tamizado

En esta etapa se procede a la eliminación de las materias grasas, cuerpos gruesos y arenosos con la ayuda de unas rejillas de desbaste

5.1.2.2. Desarenado y desengrasado

Tras esta primera etapa el agua pasa a un depósito donde se introduce aire con el objetivo de eliminar las arenas que al ser más pesadas van al fondo y las grasas que debido a su menor densidad salen a la superficie.

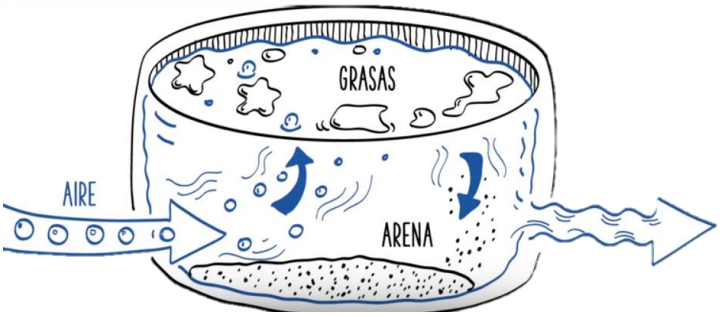


Figura 3. Desarenado y desengrasado

5.1.3. Tratamiento primario

Un tanque de sedimentación donde el agua queda en reposo durante un tiempo, y al ser más pesada, una parte de la suciedad se deposita en el fondo para posteriormente ser eliminada.

5.1.4. Tratamiento secundario o biológico

Tras la decantación primaria el agua es conducida a una piscina burbujeante a la que se le añade oxígeno para desarrollar los microorganismos que se encargaran de eliminar biológicamente la suciedad que aún quede en el agua sin recurrir a productos químicos. Estos microorganismos llevan a cabo una reducción de la carga contaminante y desarrollan nuevos fangos que se separan del agua mediante el proceso de clarificación.

Par lograrlo es necesario mantener una condiciones adecuadas y controladas de modo que se favorezca la vida de dichos microorganismos.

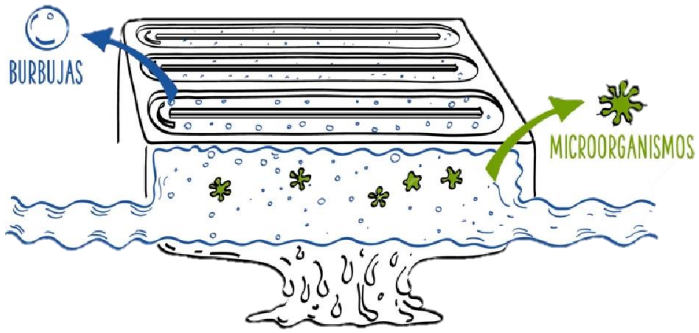


Figura 4. Tratamiento biológico

Esta etapa cabe desarrollarla un poco más debido a que es en la cual se introduce aire mediante difusores para la posterior transferencia de oxígeno, estudio que se va a abordar en el presente proyecto.

Los organismos aeróbicos presentes en esta piscina son los encargados, como se ha comentado, de producir los fangos que posteriormente serán eliminados, pero estos organismos son capaces de realizar sus funciones y de desarrollarse en presencia de oxígeno diatómico. Es por este motivo que surge la necesidad de airear la piscina en la cual se encuentran. Esta aireación se lleva a cabo mediante difusores por lo que resulta necesario conocer los principales parámetros que caracterizarán el proceso de aireación, los cuales se explicarán más adelanta.

5.1.4.1. Tratamiento terciario

Finalmente se realiza un tratamiento terciario en las EDAR que vierten a zonas protegidas con el objetivo de reducir la cantidad de microorganismos patógenos en el agua (ultravioleta y dosificación de un desinfectante).

5.2. Sistemas de medida

Con el objetivo de caracterizar el comportamiento de las unidades de aireación, la instalación está preparada para cambiar de manera fácil y rápida de elemento de medida. La transferencia de oxígeno es un valor que se obtiene indirectamente por lo que es necesario medir con diferentes dispositivos varios parámetros necesarios para su cálculo. Los diferentes elementos de medida de los cuales se puede hacer uso para calcular la transferencia de oxígeno se explican a continuación.

5.2.1. Sondas de conductividad (puntual)

Se trata de sensores electrónicos basados en la distinta conductividad eléctrica de las fases bajo análisis. Así, el uso de dos agujas aislantes paralelas (pero desplazadas axialmente) con puntas conductoras permite medir distinguir qué fase hay entre las dos puntas (cuando hay agua entre ambas circula corriente por el circuito, cuando hay gas el circuito está abierto y no circula corriente). Esta técnica también es utilizada para la medida de las propiedades de la fase dispersa, y permite obtener componentes de la velocidad de las burbujas, así como la fracción

de huecos y el tamaño de burbuja mediante técnicas de reconstrucción inversa. Todo esto se aplica a cada punto de medida, por lo que es necesario desplazar las sondas a fin de obtener perfiles o mapas de fracción de huecos, velocidad de gas, etc. Con este útil podemos determinar el área interfacial, y de ese modo despejar la contante de concentración local de la ecuación 16 más adelante mostrada. Actualmente no se encuentra instalada, pero en un futuro se incorporará en la instalación y se moverá junto con el Vectrino.

5.2.2. Oxímetro o sonda de oxígeno

La sonda de oxígeno es un elemento que nos permitirá determinar la concentración de oxígeno y la evolución de la misma en función del tiempo. En los oxímetros más comunes el sensor del oxígeno está formado por un ánodo o electrodo de referencia (normalmente de plata) y un cátodo, o electrodo de trabajo (de oro o platino) situados en el interior de un cilindro vacío que contiene una disolución electrolítica de cloruro de potasio, KCl, y está cerrado por una membrana de teflón permeable al oxígeno.

5.2.3. Anemometría Doppler Vectorial (VECTRINO)

Se trata de una técnica moderna que permite medir la velocidad de flujos mediante el uso de ultrasonidos. De esta forma, es posible medir la velocidad en fluidos opacos. A fin de poder utilizar esta técnica correctamente, se estudiará la influencia de la presencia de burbujas sobre la señal tomada por estos dispositivos. Se tratará por tanto de utilizar estos dispositivos tanto en aquellas zonas donde sólo haya flujo acuosos como en aquellas en donde haya burbujas.

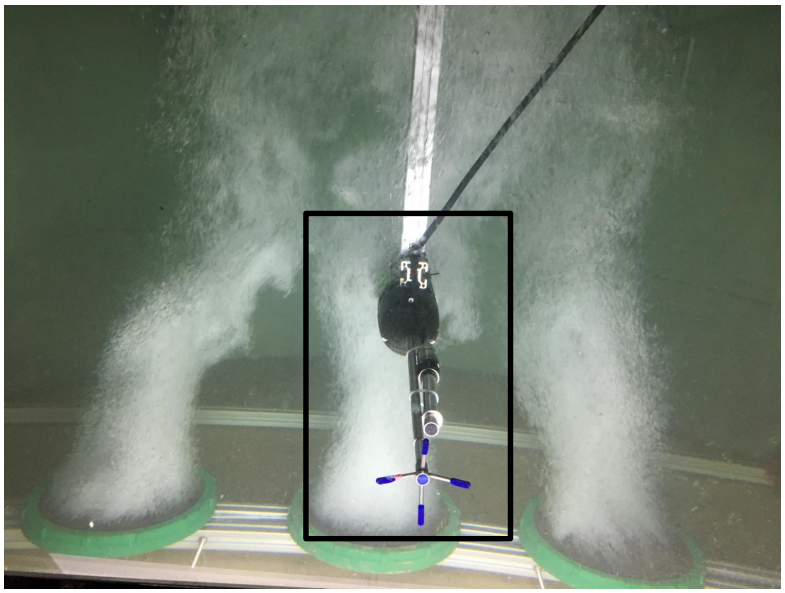


Figura 5. Vectrino

5.3. Componentes del sistema de aireación, transferencia de O_2 y diseño

El sistema de aireación es el encargado de proporcionar el oxígeno necesario en la instalación para alimentar los microorganismos y que la etapa de tratamiento biológico se desempeñe con éxito y de la manera más óptima posible.

Actualmente, la elección del sistema de aireación para el tratamiento secundario se realiza atendiendo a la profundidad del agua, la concentración de lodo o la demanda de oxígeno.

Este sistema de aireación está formado por difusores, equipos de conducción del aire y compresores. Todos estos elementos son diseñados para que la aireación se produzca con éxito teniendo en cuenta las principales características de cada uno y sus parámetros de diseño. En la parte del diseño y dimensionado de los componentes no vamos a adentrarnos ya que no forma parte del alcance de este proyecto, pero cabe mencionar los diferentes elementos que forman el sistema.

5.3.1. Compresores

Estas máquinas rotativas son necesarias en la instalación para impulsar el gas de un punto a otro aumentando a su vez su presión.

Estos soplantes se diseñan de modo que sean capaces de trasegar distintos caudales de aire dentro de un rango de presiones y bajo unas condiciones ambientales de un determinado proceso. Para la correcta selección del mismo es necesario conocer determinados parámetros de la planta tales como los caudales máximos y DBO (demanda bioquímica de oxígeno) máxima de las aguas a tratar, es decir la cantidad de oxígeno que consumen los microorganismos.

5.3.2. Conductos

Son los elementos que unen los compresores con los difusores. Estos elementos contienen el gas durante su desplazamiento de modo que han de estar correctamente diseñados para su correcto transporte. Las presiones con las que se trabajan en este tipo de instalaciones no son excesivamente elevadas, de modo que las tuberías pueden ser ligeras y su diseño no está sujeto a muchas restricciones y consideraciones. Este diseño ha de hacerse de modo que las pérdidas de cargas sean mínimas y de ese modo evitar un coste energético extra asociado a esa pérdida de presión.

5.3.3. Difusores y tipos de aireadores

Uno de los sistemas de difusión de aire, utilizado para el tratamiento de lodos activados son los difusores porosos, que permiten la homogenización de los lodos en el agua. Los difusores porosos son los elementos que introducen el aire en el reactor por medio de burbujas, las cuales pueden ser finas o gruesas. Las burbujas finas permiten una mayor transferencia de oxígeno que las gruesas, pero requieren de mayor cuidado, ya que el montaje de estos difusores requiere de un filtro ubicado antes del compresor, para evitar incrustaciones y posterior taponamiento de los inyectores por parte de las impurezas presentes en el aire (Kister et al., 2008).

Figura 6. Difusor de aireación

En el mercado podemos encontrarnos una gran gama de diferentes tipos de difusores, cada uno de ellos con sus pros y sus contras asociados a su uso. La selección del elemento correcto depende de diversos parámetros tales como la profundidad del agua, la concentración de lodo o la transferencia de oxígeno deseada para rentabilizar la planta.

El objetivo de estas máquinas es en de inyectar aire a presión en el seno de una masa líquida con el objetivo de desarrollar los microorganismos encargados de la depuración biológica. Las unidades de difusión suelen ser soportes porosos, placas o tubos, se construyen de granos de sílice u óxido de aluminio que se incluyen en una masa porosa con cemento cerámico.

La eficiencia en la transferencia de O_2 depende de diversos factores que influyen al rendimiento del difusor (SOTE) entre los cuales destaca la dimensión del propio difusor, su geometría, la profundidad del agua, la geometría del propio tanque, la ubicación de los mismos; factor que se pretende cuantificar con la instalación diseñada en este proyecto y la densidad de difusores ($\frac{A_t}{A_d}$).

De manera más visual se pueden ver estas figuras donde se muestra bajo qué parámetros se produce un aumento del rendimiento.

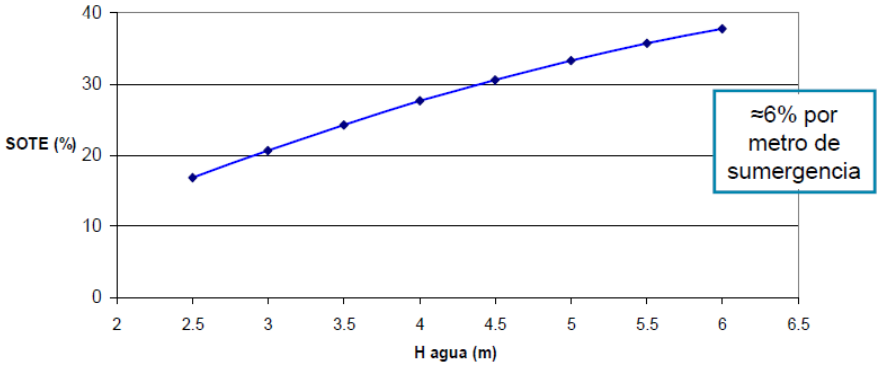


Figura 7. Aumento del rendimiento con la profundidad

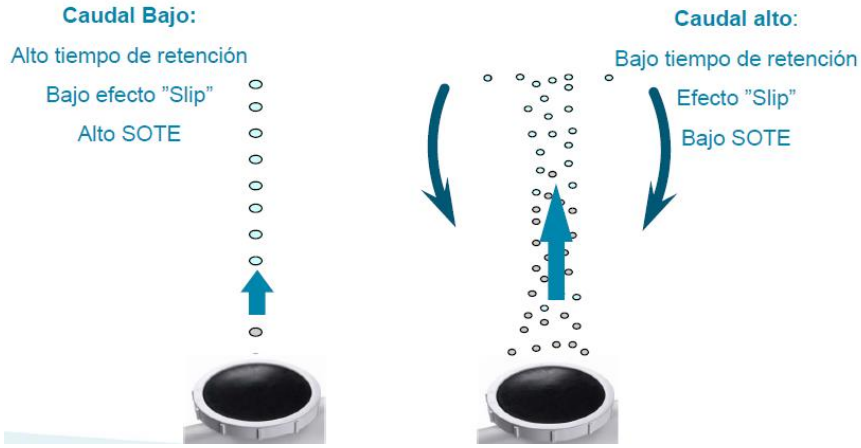


Figura 8. Disminución del rendimiento con el caudal

Esta disminución del rendimiento se debe a:

- Mayor recirculación
- Menor tiempo de residencia de las burbujas
- Mayor diámetro de cada burbuja
- Contra-presión causada por el material poroso aumenta

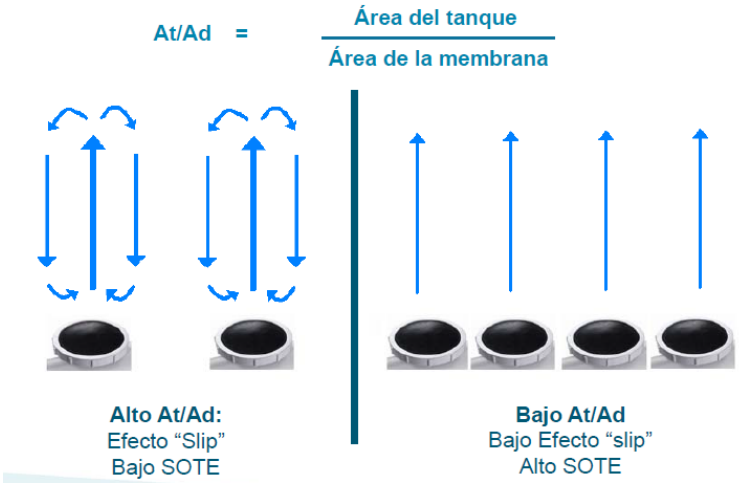


Figura 9. Aumento del rendimiento con el aumento de los difusores

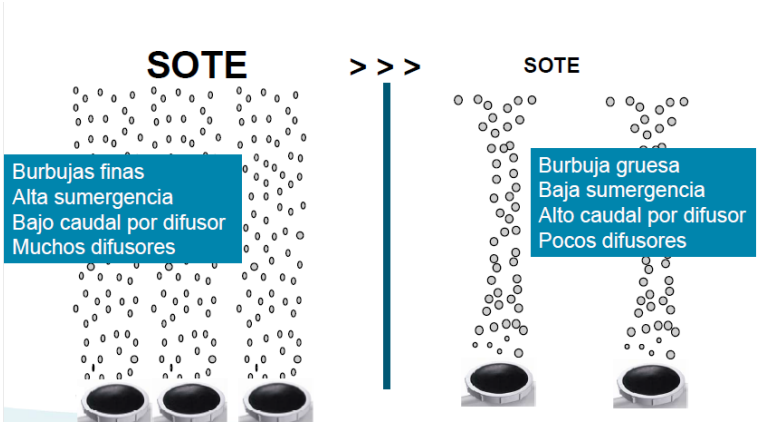


Figura 10. Resultado de combinar los parámetros que maximizan el rendimiento

Para el correcto diseño hay que llegar a un compromiso entre inversión inicial y coste de operación realizando un estudio de viabilidad económica donde se tenga en cuenta el coste diario de manera que se pueda cuantificar el periodo de retorno y la tasa de amortización del proyecto. De manera resumida se puede decir lo siguiente:

POCOS DIFUSORES	MUCHOS DIFUSORES
Baja inversión inicial	Alta inversión inicial
Alto caudal de aire por difusor	Bajo caudal por difusor
Bajo SOTE	Alto SOTE
Caudal total requerido alto	caudal total requerido bajo
Alto coste de operación	Bajo coste de operación

Tras esta breve introducción sobre los diferentes parámetros influyentes se procede a explicar y exponer los diferentes tipos de difusores en el mercado y que ventajas y desventajas conlleva su uso.

Para producir las burbujas necesarias podemos utilizar dos tipos de elementos: difusores o aireadores mecánicos.

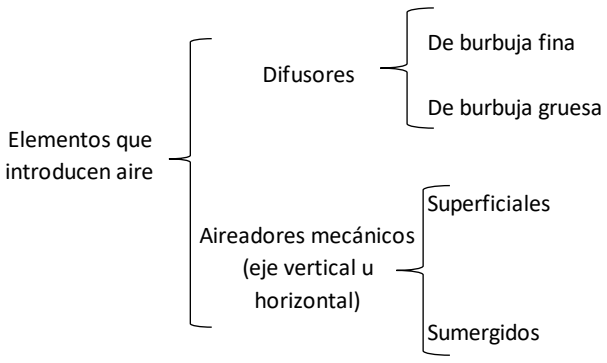


Figura 11. Clasificación de elementos aireadores

Los difusores se dividen en dos grandes grupos atendiendo al tamaño de la burbuja:

5.3.3.1. Difusores de burbuja fina

Los difusores de burbuja fina están compuestos por pequeños poros que crean burbujas de hasta 5 mm de diámetro, por lo que se consigue un alto rendimiento de transferencia de oxígeno (entre$2.5\ y\ 3.6\ \frac{Kg\ de\ oxígeno}{KWh}$) gracias a este reducido tamaño. Como contra, requieren tareas de mantenimiento para prevenir la obturación de los poros o inyectores de aire por lo que es necesario instalar filtros de aire y mantener una atmósfera limpia. También requieren de un compresor de mayor potencia que para los difusores de burbuja gruesa que permita vencer las altas pérdidas de carga introducidas por el medio poroso.

Figura 12. Difusor de burbuja fina

5.3.3.2. Difusores de burbuja gruesa

Este tipo de difusores emplea poros de mayor tamaño de modo que las burbujas producidas son de mayor diámetro, un diámetro de hasta 25 mm. Como se ha comentado previamente, este mayor tamaño de burbuja hace que los rendimientos de transferencia de oxígeno sean menores (entre$1\ y\ 2.5\ \frac{Kg\ de\ oxígeno}{KWh}$) ya que el área interfacial total es considerada inferior. Sin embargo, debido a la mayor dimensión de sus poros no requiere de filtros de aire, lo que supone un menor coste de mantenimiento. Otra ventaja es que, debido a este mayor diámetro poroso, no es necesario aplicar tanta potencia por parte del compresor para trasegar el gas a través de la superficie porosa.

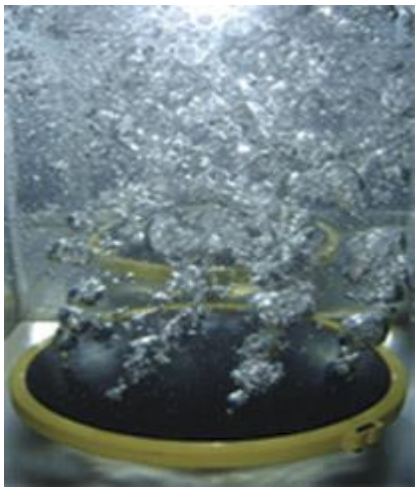


Figura 13. Difusor de burbuja gruesa

Se puede distinguir dos grupos de aireadores mecánicos según las principales características de diseño y funcionamiento de eje vertical y de eje horizontal. A su vez, los dos grupos se dividen en superficiales y sumergidos. En todos los casos, la acción agitadora y de bombeo mantiene el contenido del tanque mezclado. La necesidad energética para mantener un régimen de flujo correcto con aireadores mecánicos depende de su diseño y de la geometría de la planta. Aireadores mecánicos

5.3.3.3. Aireadores mecánicos superficiales

En estos aireadores introducen el aire desde la superficie mediante mecanismos de palas que arrastran el aire de la atmósfera bajo la superficie del líquido. Las eficiencias de estos sistemas oscilan entre$1.2\ y\ 3.2\ \frac{Kg\ de\ oxígeno}{KWh}$. Aunque el coste de operación de este tipo de difusores es bastante reducido, requieren tareas de mantenimiento para mantener limpias las palas y no consiguen airear los reactores en su totalidad, dado que las burbujas introducidas desde la superficie no llegan hasta la base del reactor. Por lo tanto, requieren del uso de agitadores mecánicos. En las siguientes ilustraciones podemos ver dos tipos de aireadores mecánicos superficiales, uno de eje horizontal y otro de eje vertical.

Figura 14. Aireador mecánico superficial de eje horizontal

Figura 15. Aireador mecánico superficial de eje vertical

5.3.3.4. Aireadores mecánicos sumergidos

Se trata de sistemas tipo Venturi que permiten la difusión de oxígeno desde la salida de los mismos. Con este tipo de aireadores se obtienen eficiencias menores que los anteriores, al estar comprendidas $1.2\ y\ 1.8\ \frac{Kg\ de\ oxígeno}{KWh}$. Además, requieren un continuo mantenimiento y una elevada potencia para inyectar el flujo hacia el reactor. Finalmente, su aireación es poco uniforme al airear sólo la zona que rodea la salida del aireador. Dado que estos sistemas basan su funcionamiento en la inyección de líquido a gran velocidad, el uso de varios aireadores de este tipo en paralelo para aumentar la extensión de la zona aireada es bastante complicado.

En la siguiente ilustración podemos observar un aireador mecánico sumergible cuyo funcionamiento se basa en el efecto venturi el cual se detalla a continuación.

Figura 16. Aireador mecánico sumergible por efecto venturi

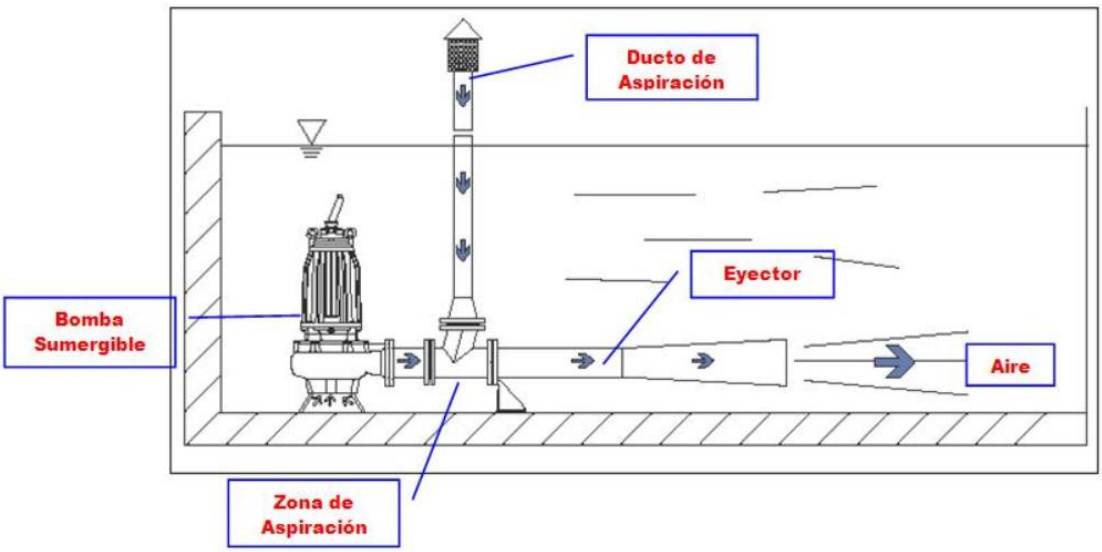


Figura 17. Funcionamiento del aireador sumergible venturi

5.3.4. Transferencia de Oxígeno

El elemento encargado de introducir el aire en la instalación diseñada será un difusor de burbuja fina, ya que es el más utilizado y el que mayor rendimiento ofrece.

Para el diseño de los sistemas de aireación se tiene en cuenta la demanda de aire asociada a esta etapa y el caudal capaz de proporcionar cada difusor. Los difusores se colocan de manera uniforme en la base del reactor de modo que no se tiene en cuenta el comportamiento de las burbujas en el interior del fluido.

Los problemas asociados al régimen bifásico serán el propio choque de las burbujas entre sí, de modo que se produciría coalescencia entre ellas y aumentaría su radio disminuyendo así la transferencia de oxígeno. Otro aspecto que no se tiene en cuenta que el primer difusor debería ubicarse a mayor distancia del siguiente ya que esta primera columna recibirá un mayor

impacto del agua al ser la primera en encontrarse de modo que estará más inclinado. Este factor no se podrá cuantificará en la instalación diseñada ya que el agua en el depósito de ensayos se encuentra estática.

Para el correcto diseño de los sistemas de aireación en las EDAR es necesario conocer los parámetros que caracterizan este proceso.

Hasta el momento la teoría más empleada para el estudio de la transferencia de O_2 es la de doble película propuesta por Lewis y Whitman en 1942 donde se explica que en la interfase gas-líquido existen dos capas, una en fase gas y otra líquida a través de las cuales se transfiere el gas por difusión molecular. Esa velocidad de difusión es proporcional a la diferencia de concentraciones entre la superficie de la película, de modo que podemos atender a la siguiente ecuación para cuantificar el flujo molar de O_2 a través de la interfase:

$$N^{º} = k_g \cdot (p_G - p_i) = k_L \cdot (C_i - C_L) \qquad \text{Ec. 1}$$

Dónde:

- $N^{º} = flujo\ molar\ de\ O_2\ en\ \frac{mol}{m^2 \cdot s}$
- $k_g y\ k_L\ son\ coeficientes\ locales\ de\ transferencia$
- $p_G\ es\ la\ presión\ parcial\ de\ O_2\ en\ la\ burbuja$
- $p_i\ es\ la\ presión\ parcial\ de\ O_2\ en\ la\ interfase$
- $C_L es\ la\ concentración\ en\ el\ medio\ líquido$
- $C_i es\ la\ concentración\ en\ la\ interfase$

Debido a la dificultad de medir directamente las concentraciones interfaciales, se considera un coeficiente global de transferencia de materia:

$$N^{º} = K_g \cdot (p_G - p^*) = K_L \cdot (C^* - C_L) \qquad \text{Ec. 2}$$

Dónde:

- $K_g y\ K_L\ son\ coeficientes\ globales\ de\ transferencia$
- $p^* es\ la\ presión\ de\ O_2\ en\ la\ burbuja$
- $C^* es\ la\ concentración\ de\ O_2\ de\ saturación\ del\ medio\ en\ quilibrio\ con\ el\ gas$

Por último se aplica la transferencia de O_2 a un volumen de fluido V con un área interfacial, A, y se obtiene la siguiente expresión de velocidad de transferencia de O_2 por volumen unitario:

$$W_{O_2} = \frac{A}{V} \cdot N^{º} = k_L \cdot \frac{A}{V} \cdot (C^* - C_L) = (k_L \cdot a) \cdot (C^* - C_L) \qquad \text{Ec. 3}$$

Atendiendo a la ecuación 4 podemos ver como la velocidad de la transferencia de oxígeno es directamente proporcional a la diferencia de las concentraciones de la burbuja y del líquido multiplicado por una constante de proporcionalidad $(k_L \cdot a)$ donde el coeficiente a es el cociente del área interfacial y el volumen, también llamada superficie específica.

5.3.5. Diseño de sistemas de aireación

Actualmente el método de difusión más utilizado en las EDAR es el de parrillas de difusores. El diseño actual se lleva a cabo de manera simplificada y sin considerar la influencia que unos difusores pueden ejercer sobre otros de modo que se produzca la coalescencia de las burbujas aumentando su tamaño y de ese modo reduciendo la transferencia de oxígeno, ya que, como se ha comentado en el punto anterior, simplemente se tiene en cuenta la demanda total y la capacidad de cada difusor a instalar.

Para el dimensionado de la parrilla de difusores se necesita conocer unos datos de partida:

- Demanda real de oxígeno $\left(AOR, \frac{Kg\ O_2}{d}\right)$ necesario para mantener las reacciones químicas.
- Índice de transferencia de O_2 estándar $(SOTR, \frac{Kg\ O_2}{h})$

A partir de estos valores se calcula la cantidad de aire que es necesario inyectar para aportar la cantidad de oxígeno necesaria suponiendo que el aire se distribuye de manera uniforme en todo el reactor y en cualquier instante de tiempo. Con estos datos se obtiene el caudal de aire necesario a aportar al sistema.

1. Cálculo del caudal de aire necesario

$$q_a = \frac{SOTR}{c_i \cdot e} \qquad \text{Ec. 4}$$

$$q_a = \frac{1}{3} \cdot \frac{SOTR}{k_L \cdot \Delta CC_{O_2}} \cdot \frac{R_b}{T_b} = \frac{1}{3} \cdot \frac{SOTR}{k_L \cdot \Delta CC_{O_2}} \cdot \frac{1}{T_b} \cdot \frac{1}{k_L \cdot a} \qquad \text{Ec. 5}$$

Donde c_i es el contenido de oxígeno del aire $c_i = 0.280\ \frac{Kg\ O_2}{m^3}$ y e es la eficiencia de transferencia de oxígeno estándar (SOTE (%)) que se obtiene de las curvas de los difusores.

En la ecuación 5 intervienen una serie de magnitudes de carácter químico, como la constante de transferencia, k_L, la concentración de O_2 en el reactor, C_{O2}, o la diferencia de concentraciones de O_2 en las burbujas y en el agua, ΔC. Como parámetros mecánicos intervienen el radio medio de las burbujas, R_b, y el tiempo de contacto entre las burbujas y el agua, T_b. El mayor rendimiento de transferencia de oxígeno se da cuando mayor es la profundidad de inmersión de los difusores, debido a que se produce un mayor tiempo de contacto entre el agua residual y las burbujas de aire, así como a la mayor concentración de saturación de oxígeno por ser la presión mayor. También es evidente que la cantidad de aire necesaria será menor si se utilizan burbujas de menor tamaño.

Otro parámetro que afecta el rendimiento del sistema es el caudal unitario, ya que con un aumento de este se produce un decrecimiento de la eficiencia debido a que aumenta la recirculación del agua en el depósito, se reduce el tiempo de retención de las burbujas, las

burbujas serán de un mayor radio y la contra presión causada por el material poroso del difuso crecerá.

2. Número de difusores necesarios. Este cálculo se llevará a cabo dividiendo el caudal previamente obtenido entre el caudal unitario de cada difusor.

Este paso final se hace actualmente en base a la experiencia del personal técnico de la planta, que ya ha trabajado con difusores de distintos tipos (al estar a cargo de varias plantas), y tiene ciertas nociones del caudal máximo que puede aportar cada tipo de difusor. Sin embargo, estas nociones provienen de la observación en planta y están sujetas a múltiples factores mecánicos que son obviados en el esquema de diseño actual, basado en el concepto de reactor completamente agitado.

$$N' = \frac{q_a}{q_{a_{unitario}}} \qquad \text{Ec. 6}$$

Donde:

- q_a será el caudal necesario calculado
- N' será el número de difusores a colocar
- $q_{a_{unitario}}$ será el caudal de aire por difusor

3. Densidad superficial de los difusores. Finalmente se obtendrá la densidad de los difusores, es decir, la proporción de espacio que ocupan respecto a la superficie del reactor.

$$Densidad_{difusores} = \frac{N'}{l \cdot w} \qquad \text{Ec. 7}$$

Donde:

- $Densidad_{difusores}\ en\ \frac{1}{m^2}$
- $l\ es\ la\ longitud\ del\ depósito$
- $w\ es\ la\ profundidad\ del\ depósito$

Estos son los parámetros necesarios para el diseño de la ubicación de los difusores teniendo en cuenta la geometría del depósito y espaciando suficiente los difusores entre sí y respecto a las paredes. Sin embargo, no se tiene en cuenta la distribución local del gas de modo que se produzca coalescencia entre las burbujas como ya se ha comentado. Por lo tanto, a continuación, se va a proceder a caracterizar la distribución local del gas.

En primera aproximación, se puede suponer un caso de un difusor que mediante un caudal Q produce una columna de gas. La fracción de vacío α del gas será:

$$\alpha = \frac{V_{gas}}{V_{total}} \qquad \text{Ec. 8}$$

Donde el volumen total será el producto del área del difusor y la altura de la columna de gas H.

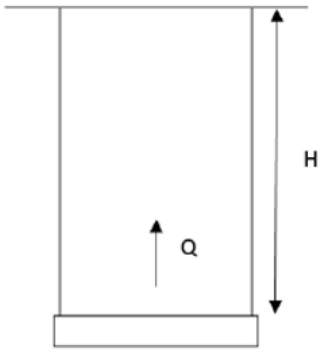


Figura 18. Columna de gas producida por un difuuusor

Por otra parte, el caudal de gas que sale del difusor será el volumen del gas por unidad de tiempo, y la velocidad de ascenso de las burbujas será la altura de la columna dividido por el tiempo de residencia de las burbujas.

$$Q_{gas} = \frac{V_{gas}}{tiempo} \qquad \text{Ec. 9}$$

$$Velocidad = \frac{H}{tiempo} \qquad \text{Ec. 10}$$

Por tanto, despejando el caudal de gas obtenemos la siguiente ecuación:

$$Q_{gas} = \frac{\alpha \cdot V_{gas} \cdot Velocidad}{H} = \alpha \cdot A_{difusor} \cdot Velocidad \qquad \text{Ec. 11}$$

Pero esto es una aproximación, ya que la columna de gas varia de área en función de la altura, de modo que el radio de la columna aumenta con el ascenso del gas.

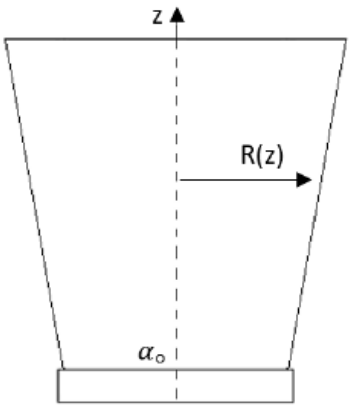


Figura 19. Evolución de la columna de gas

Si el flujo asciende a una velocidad determinada, el caudal será constante en cada sección perpendicular:

$$Q_{gas} = \alpha\,(z) \cdot A_{difusor}(z) \cdot Velocidad \qquad \text{Ec. 12}$$

Con esta fórmula se puede obtener la fracción de vacío en función de la altura del gas.

$$\alpha\,(z) = \frac{Q_{gas}}{A_{difusor}(z) * Velocidad} = \frac{Velocidad \cdot A_0 \cdot \alpha_0}{Velocidad \cdot A_{difusor}(z)} = \frac{\pi \cdot R^2 \cdot \alpha_0}{\pi \cdot R^2(z)} = \alpha_0 \cdot \frac{R^2}{R^2(z)} \qquad \text{Ec. 13}$$

Por otro lado, en el punto donde chocan dos columnas, se cumple que la distancia entre difusores L es:

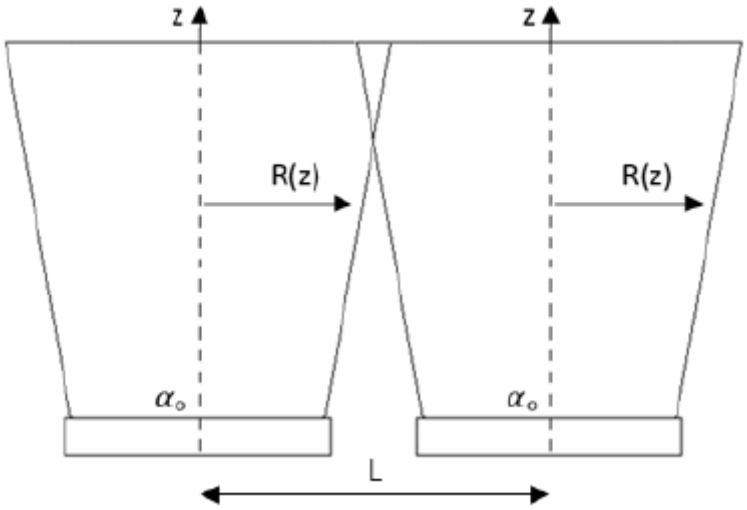


Figura 20. Intersección de dos columnas de gas

$$L = 2 \cdot R(z) \qquad \text{Ec. 13}$$

Con la combinación de estas dos últimas expresiones, se obtiene la fracción de gas en el choque para un difusor. La condición que debe cumplir para que no se produzca la pérdida de eficiencia es que la suma de la fracción de vacío de los difusores en el punto de choque sea menor que la fracción de vacío crítica:

$$\alpha\,(z) = 4 \cdot\ \alpha_0 \cdot \frac{R_0^2}{L^2} \qquad \text{Ec. 14}$$

$$2 \cdot \alpha\,(z) < \alpha_c \qquad \text{Ec. 15}$$

El único valor que faltaría conocer es la fracción de vacío crítica, y junto con los parámetros del difusor se podrá calcular la distancia a la que hay que colocar los difusores.

La obtención de esta distancia entre difusores para conseguir una eficiencia máxima se puede obtener de manera numérica o de manera experimental. En el presente proyecto nos centraremos en la posibilidad de la obtención experimental, de modo que en el siguiente apartado se detalla el proceso de diseño y construcción del depósito de ensayos y de todos los equipos auxiliares necesarios para la obtención de resultados y caracterización del fluido bifásico.

6. Diseño del banco de ensayos

En el siguiente apartado se va a explicar el proceso de diseño de la instalación, desde el depósito que albergará el fluido en el cual se van a tomar las medidas, hasta el código informático que nos permitirá controlar toda la instalación.

6.1. Diseño del depósito

Para realizar el estudio ha sido necesario el diseño y construcción de un depósito de almacenamiento de agua. Para realizar el diseño se ha empleado el software ANSYS un programa que abarca la simulación del comportamiento de dinámica de fluidos, electromagnetismo, resistencia de materiales, etc y se ha realizado el análisis estático del mismo.

El diseño del depósito ha de responder a las exigencias del proyecto y a las condiciones de su emplazamiento que se resumen a continuación:

- Dimensiones del equipo: El depósito contendrá 8.000 Litros de agua, un fluido no presurizado, por lo que las exigencias hidrostáticas no serán muy elevadas
- Temperatura interior máxima: 40ºC
- Temperatura interior mínima: 10ºC
- Requisitos higiénicos
- Presión de trabajo
- Las condiciones de contorno del depósito son adecuadas ya que su emplazamiento va a ser el interior del taller de la Universidad Jaume I

- Permitir la visibilidad de las burbujas de gas ascendiendo
- Soportar el peso de un sistema de guías las cuales contendrán los motores necesarios para tomar las medidas realizando los barridos

El material empleado para el diseño ha sido acero inoxidable, ya que el depósito va a estar en contacto directo con agua por lo que será necesario evitar su corrosión y no contaminar su contenido.

Tras varios diseños se ha optado por una estructura cilíndrica vertical de un espesor de 5 cm ya que los resultados de la simulación dieron un resultado favorable, proporcionando una deformación mínima y soportando unos valores de tensión admisibles. En la siguiente figura se muestra las cargas aplicadas sobre el depósito y los resultados de las simulaciones por elementos finitos mediante ANSYS.

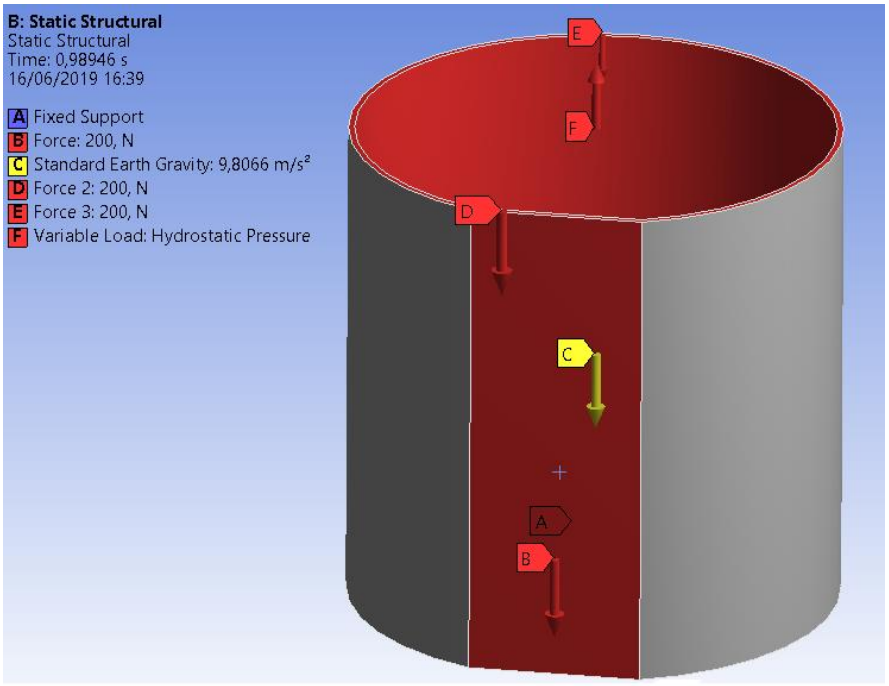


Figura 21. Cargas aplicadas en el depósito

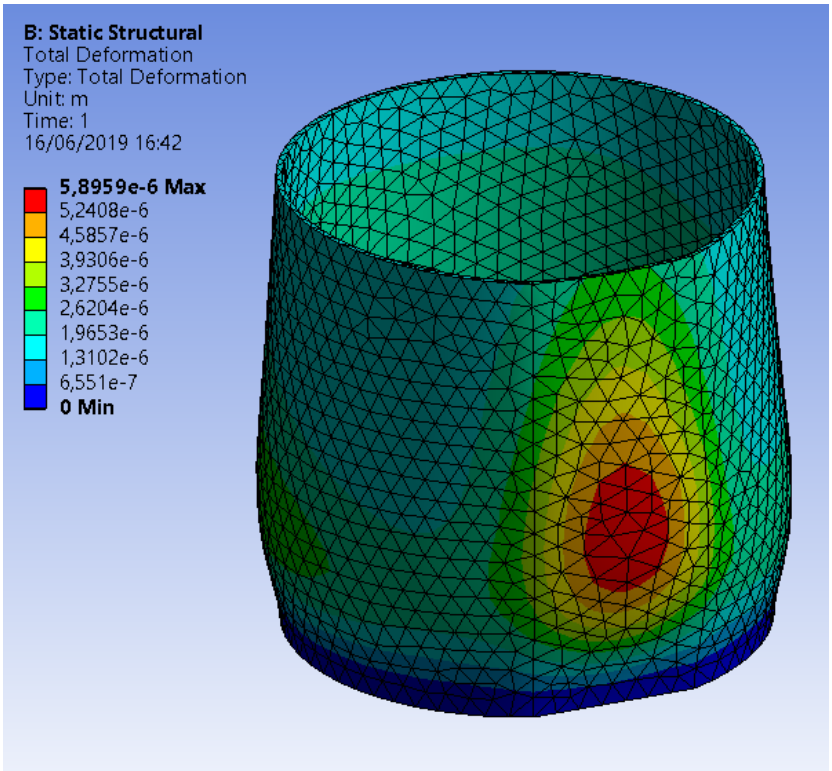


Figura 22. Deformación del depósito

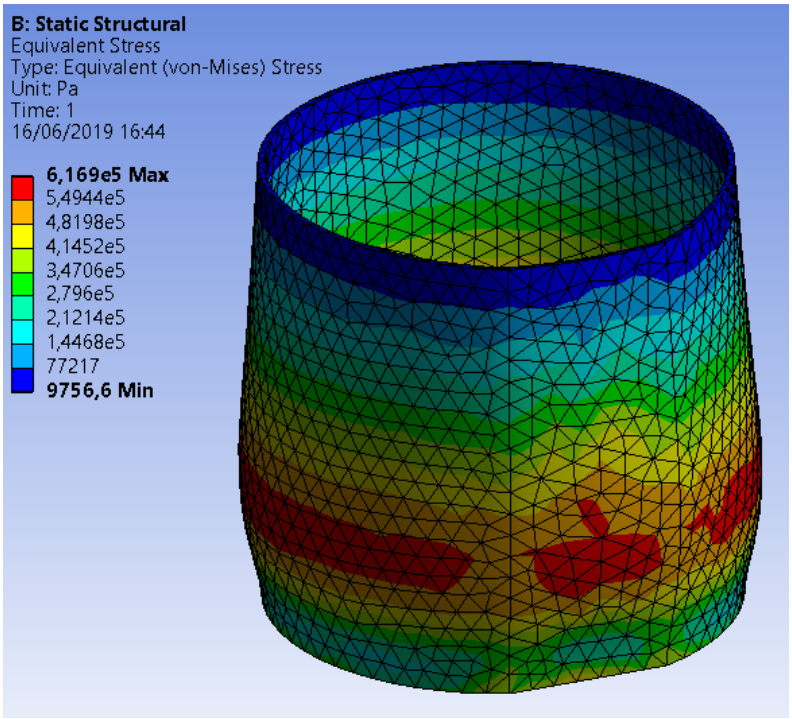


Figura 23. Tensiones equivalentes en el depósito

Como podemos observar, el valor máximo de la tensión que va a soportar la estructura tras la aplicación de las cargas es de $6.169 \cdot 10^5$ Pascales, valor inferior a límite de tensión del acero inoxidable.

6.2. Estructura interna sumergible

Por comodidad en los cambios de difusores y para darle funcionalidad a la instalación en la cual se tomarán las medidas, se ha decidido construir una estructura interna sumergible y deslizante en la cual se ensamblarán los difusores de modo que su cambio sea cómodo y sencillo.

Esta estructura interna estará formada por dos subestructuras. Una de ellas (deslizante) deslizará sobre las guías de la otra (fija) la cual se encontrará estática dentro del depósito mediante un adhesivo resistente al agua. Como elemento elevador se ha empleado un polipasto mecánico.

El perfil empleado para la estructura ha sido un perfil de aluminio cuadrado de 30x30. Esta estructura no se ha analizado mediante software de elementos finitos ya que la carga que soporta es su propio peso. A continuación, se muestran unas imágenes de ambas estructuras, la fija y la deslizante.

Figura 24. Estructura sumergible fija

Figura 25. Estructura sumergible deslizante

Figura 26. Conjunto estructura interna

6.3. Elementos auxiliares

Debido a necesidades del proyecto, ha sido necesario dotar a la instalación de varios componentes auxiliares que permiten optimizar su funcionamiento. Estos circuitos auxiliares se detallan en los siguientes apartados.

6.3.1. Filtro

Para el saneado del agua ha sido necesarios incorporar a la instalación un filtro de arena a través del cual se bombea el agua con el objetivo de eliminar las partículas de óxido disueltas en el agua. En la siguiente imagen se puede observar el filtro utilizado para el filtrado del agua.

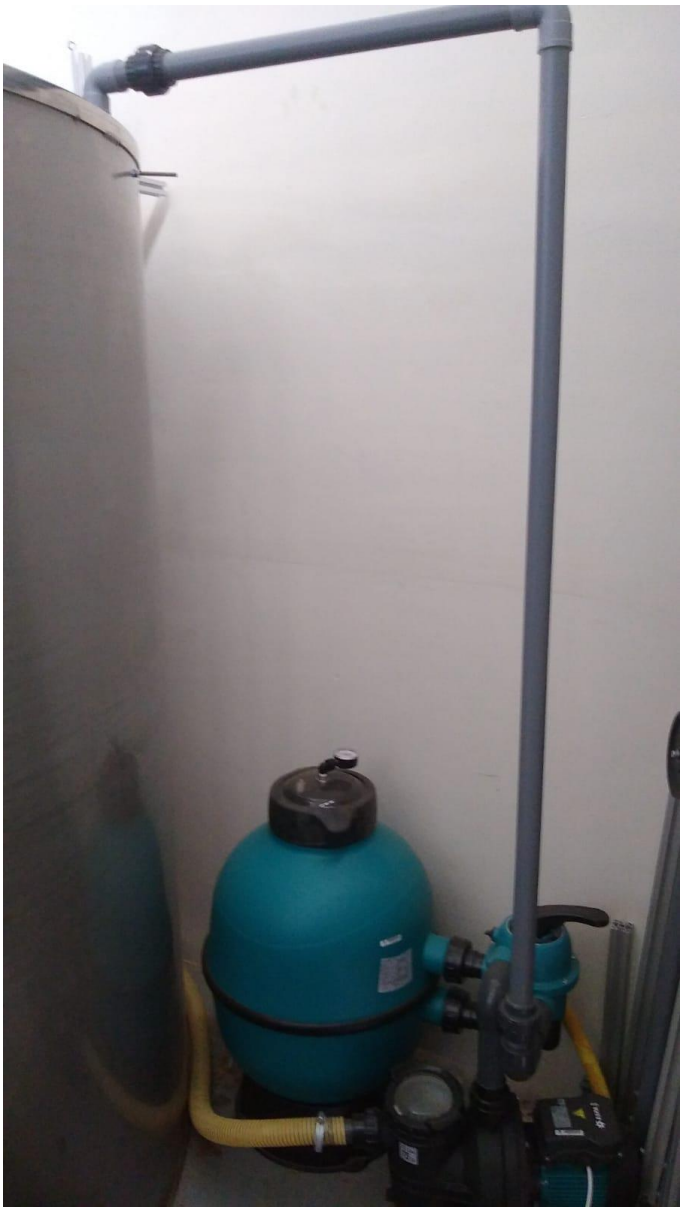


Figura 27. Filtro de arena

Las especificaciones técnicas del filtro empleado se muestran en la siguiente tabla:

Diámetro del filtro	600 mm	Presión de prueba máxima	7 bares
Velocidad de filtración	60 m3/h/m2	Arena	86 Kg
Presión máxima de servicio	3.6 bares	Granulometría	0.4 – 0.8 mm
Superficie de filtración	0.196 m2	Caudal	10 m3/h

6.3.2. Circuito de gas

Debido a la necesidad de introducir y monitorear el caudal de gas que entra en el sistema, ha sido necesario incorporar a la instalación de un sistema de aire a presión por el cual se introduce el gas a los difusores.

Este sistema auxiliar se compone de un caudalímetro-controlador que lleva incorporado una electroválvula que ajusta el caudal a la consigna que ordenas desde el programa. El gas sale de la botella de gas a presión con un regulador de presión, se dirige al caudalímetro-controlador y la electroválvula dará paso del gas hacia los difusores. En la siguiente imagen podemos ver el elemento que controla el paso de gas a los difusores.

Figura 28. Caudalímetro controlador

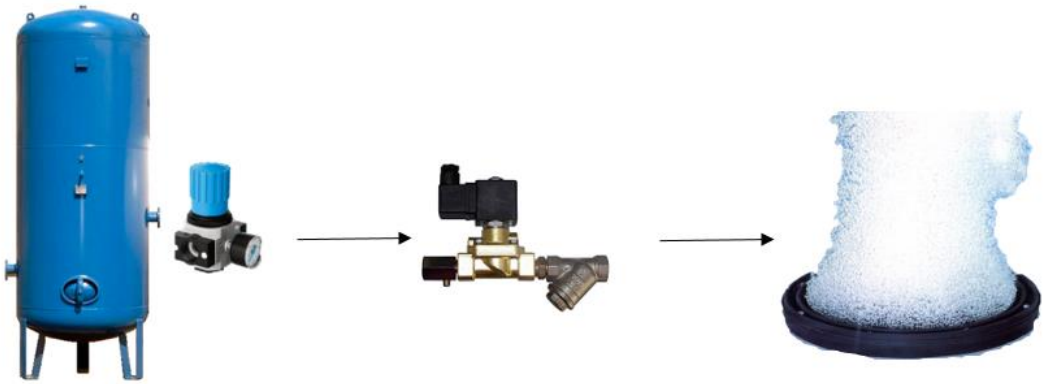


Figura 29. Circuito de gas.

El set point del caudal se introduce en un ordenador que recibe una señal con el valor actual del caudal, de modo que, realizando una comparación de ese valor actual con el valor establecido por el usuario, actua sobre una electroválvula mandándole una señal para abrirla o cerrarla. El control del caudal funciona con un controlador PID que viene incorporado en el caudalímetro, el cual compara el valor actual con el valor de consigna y actua sobre la electroválcula en base al error entre ambos valores. En la siguiente figura podemos ver el software que nos permite controlar el flujo de gas que introducimos en el sistema.

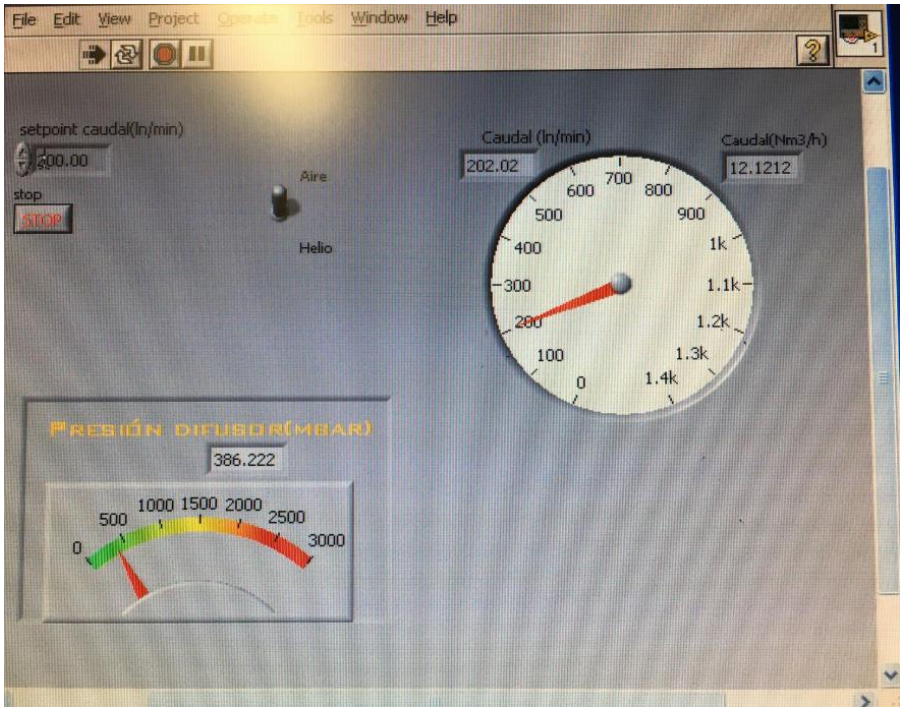


Figura 30. Interfaz de control de gas

6.4. Hardware

Para la correcta utilización de la interfaz gráfica de MATLAB la cual controla el movimiento de los motores es necesario establecer las conexiones entre los diferentes dispositivos empleados para el diseño. Estos dispositivos son:

- Arduino UNO
- Driver de motor
- Fuente de alimentación
- Relé
- Motor desplazamiento horizontal
- Motor desplazamiento vertical

Debido a que por razones económicas solo se disponía de una tarjeta para controlar el motor, fue necesario realizar la conmutación para la selección de un motor u otro mediante un relé cuádruple. El relé conmuta o no en función del motor seleccionado.

A continuación, podemos ver una breve descripción del funcionamiento de los elementos utilizados para la elaboración del cuadro eléctrico.

6.4.1. Arduino UNO

Arduino es una plataforma de creación de electrónica de código abierto la cual permite implementar microcontroladores. La utilidad de este dispositivo en el proyecto será la de controlar tanto el desplazamiento y la dirección de los motores como la toma de medidas de los diferentes dispositivos conectados a él gracias a la posibilidad de utilizar sus pines como entradas o salidas. La programación del mismo se realiza mediante MATLAB estableciendo previamente la comunicación entre ambos dispositivos.

Figura 31. Arduino

6.4.2. Motores para el desplazamiento horizontal y vertical

Los motores empleados para el sistema son motores paso a paso, motores que convierten una serie de pulsos eléctricos en desplazamientos angulares, es decir, el rotor gira al recibir pulsos eléctricos. El dispositivo encargado de acondicionar la información que recibirán los diferentes motores son los drivers.

Los motores encargados de realizar los desplazamiento horizontal y vertical estarán ubicados en una guía con una deslizadera, de modo que la deslizadera se desplazará de forma horizontal o vertical cuando el usuario a se lo ordene través de la interfaz gráfica. El usuario seleccionará el movimiento deseado de modo que el Arduino le mandará una señal al pin STEP de la tarjeta lo que propiciará el movimiento del motor.

En la siguiente figura se pueden observar los motores empleados para la instalación ensamblados entre sí, de manera que se controlan ambos ejes de coordenadas y se permite tomar barridos en dos direcciones.

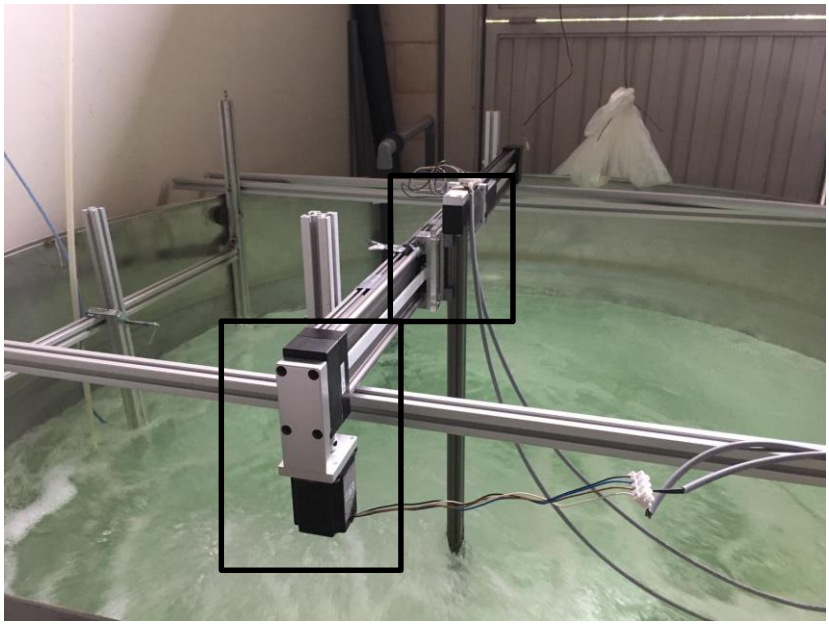


Figura 32. Motores

6.4.3. Driver de motor (G201X)

Para el control de los diferentes motores se empleará una tarjeta controladora o driver, la cual es comandada por el Arduino. La tarjeta recibirá del Arduino los pulsos de STEP para producir el movimiento de los motores y también recibirá en la entrada DIRECTION una señal binaria la cual indicará la dirección a la cual debe girar el motor (1 derecha y 0 izquierda). Esta tarjeta será alimentada por la fuente de alimentación mediante los pines de entrada 1 (POWER GND) y 2 (18 TO 80VDC).

Figura 33. Driver motor

6.4.4. Fuente de alimentación

Elemento necesario para alimentar el sistema electrónico. Esta fuente proporciona una tensión de 12 voltios en continua.

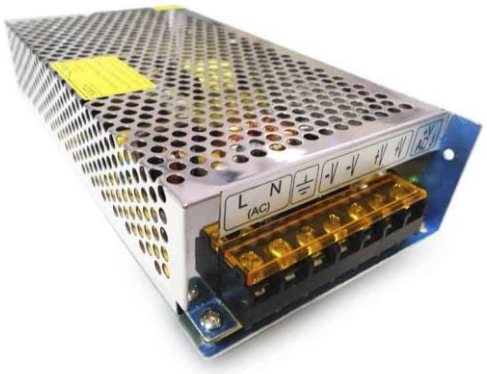


Figura 34. Fuente de alimentación

6.4.5. Relé cuádruple

Para el correcto control de los dos motores necesarios en este proyecto, ha sido necesario incorporar en la instalación un relé debido a que solo disponemos de una tarjeta para controlar ambos motores. El relé se conecta entre la tarjeta y los motores de modo que cuando es necesario realizar el movimiento de uno de los motores, el Arduino le manda una señal digital al relé y le ordena que conmute o no en función del motor que se desee mover.

Por defecto está activado el motor que realiza el desplazamiento horizontal, de modo que, si se desea realizar el movimiento vertical, el Arduino le enviará una señal de “1” al relé para que realice la conmutación y de ese modo que la tarjeta pueda comunicarse con dicho motor.

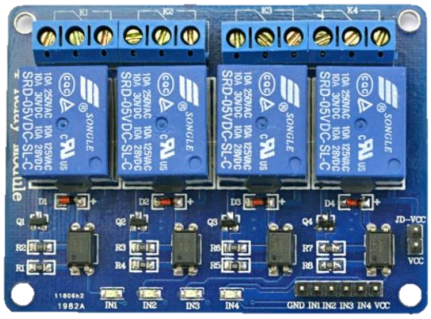


Figura 35. Relé cuádruple

En la siguiente figura se puede observar el cuadro ensamblado al completo. El emplazamiento del mismo será el lateral del depósito, de manera que quede accesible para el usuario para poder dar y quitar alimentación fácilmente.

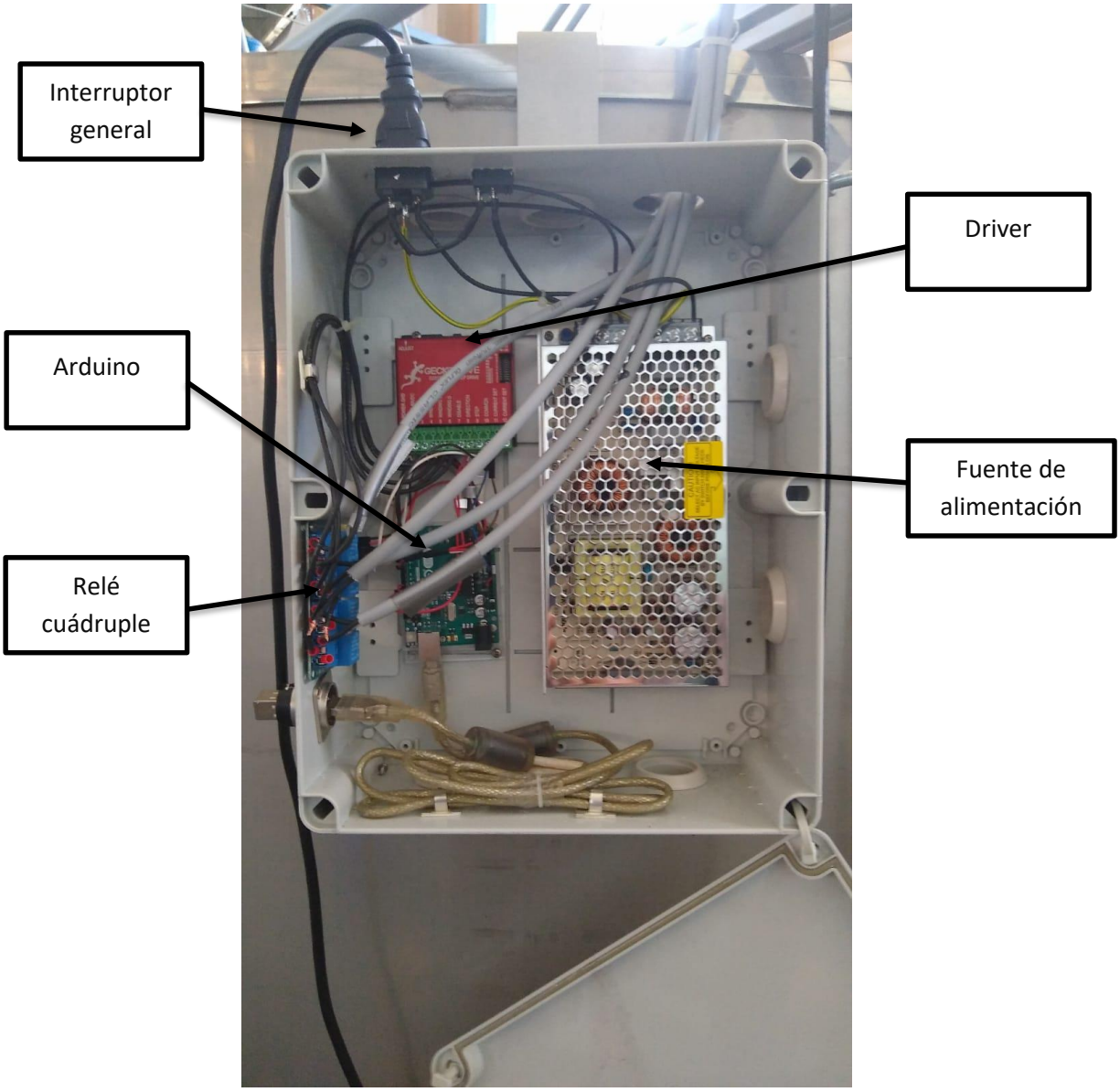


Figura 36. Cuadro eléctrico

6.4.6. Esquema del conexionado eléctrico

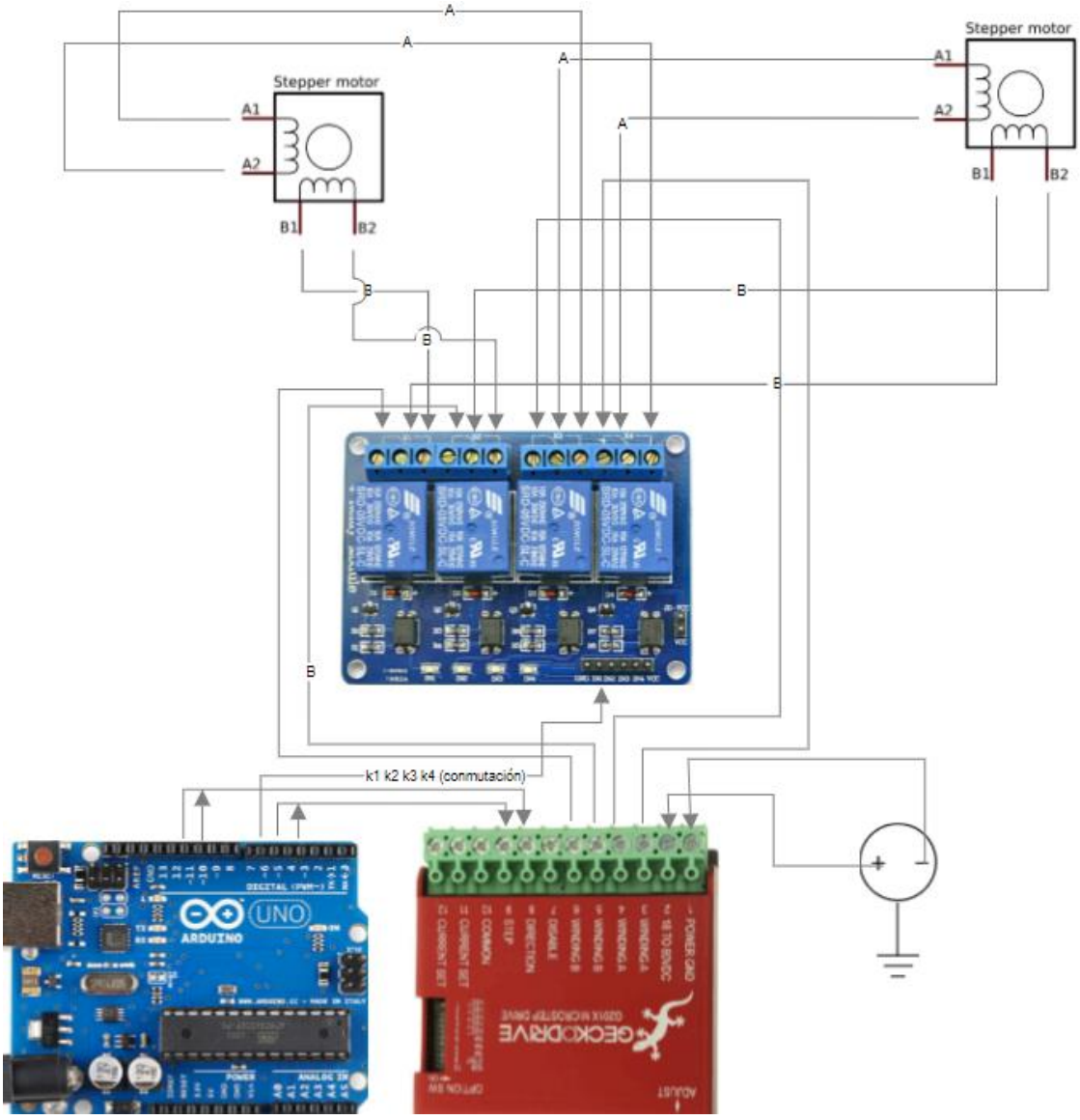


Figura 37. Circuito eléctrico

6.5. Software

Para el correcto funcionamiento de la instalación en la cual se van a llevar a cabo las pruebas experimentales ha sido necesario programar un código en MATLAB.

MATLAB (MAtrix LABoratory) es un programa comercial usado en ciencia e ingeniería, que contiene numerosas funciones para resolver problemas numéricos. El usuario puede construir sus propias funciones, en un lenguaje parecido a C.

Este código ha sido programado mediante el GUIDE de MATLAB, una interfaz gráfica de usuario que permite controlar la instalación de una manera más intuitiva evitando de ese modo que un usuario que desee tomar medidas tenga que adentrarse en un código ajeno a él, lo cual sería más tedioso. Gracias a la interfaz se puede controlar fácilmente la totalidad de los movimientos de los motores necesarios para ubicar la sonda de medida en la posición deseada, tratar los datos tomados por la sonda, y mostrarlos de una manera gráfica y representativa que facilita el análisis de los mismos.

El movimiento de los motores es controlado por Arduino UNO, al cual se le carga el código de MATLAB tras haber permitido la comunicación entre ambos.

El código implementado se presenta en el apartado **IV. ANEXOS.**

6.6. Instalación final

Tras el diseño del hardware y del software asociado al proyecto, se procede al ensamblaje del mismo y a la verificación de la instalación, comprobando que los motores realizan los barridos deseados y están bien calibrados. Las comprobaciones que se llevaron a cabo previas a realizar los barridos fueron las siguientes:

1. Calibración de los motores. Comprobar que el motor se movía la misma distancia que se le asignaba por pantalla. Este valor se puede corregir en caso de cambiar de motores gracias a un parámetro mostrado en la interfaz ($^{mm}/_{pulso}$).
2. Rango máximo de movimiento. Es importante conocer los límites de la instalación ya que al realizar los barridos no nos podemos salir de los mismos para evitar colisiones.
3. Lectura adecuada de los sentidos de movimiento y de la matriz de barrido que Matlab leerá de Excel.

Finalmente, en las siguientes figuras se puede ver la instalación al completo.

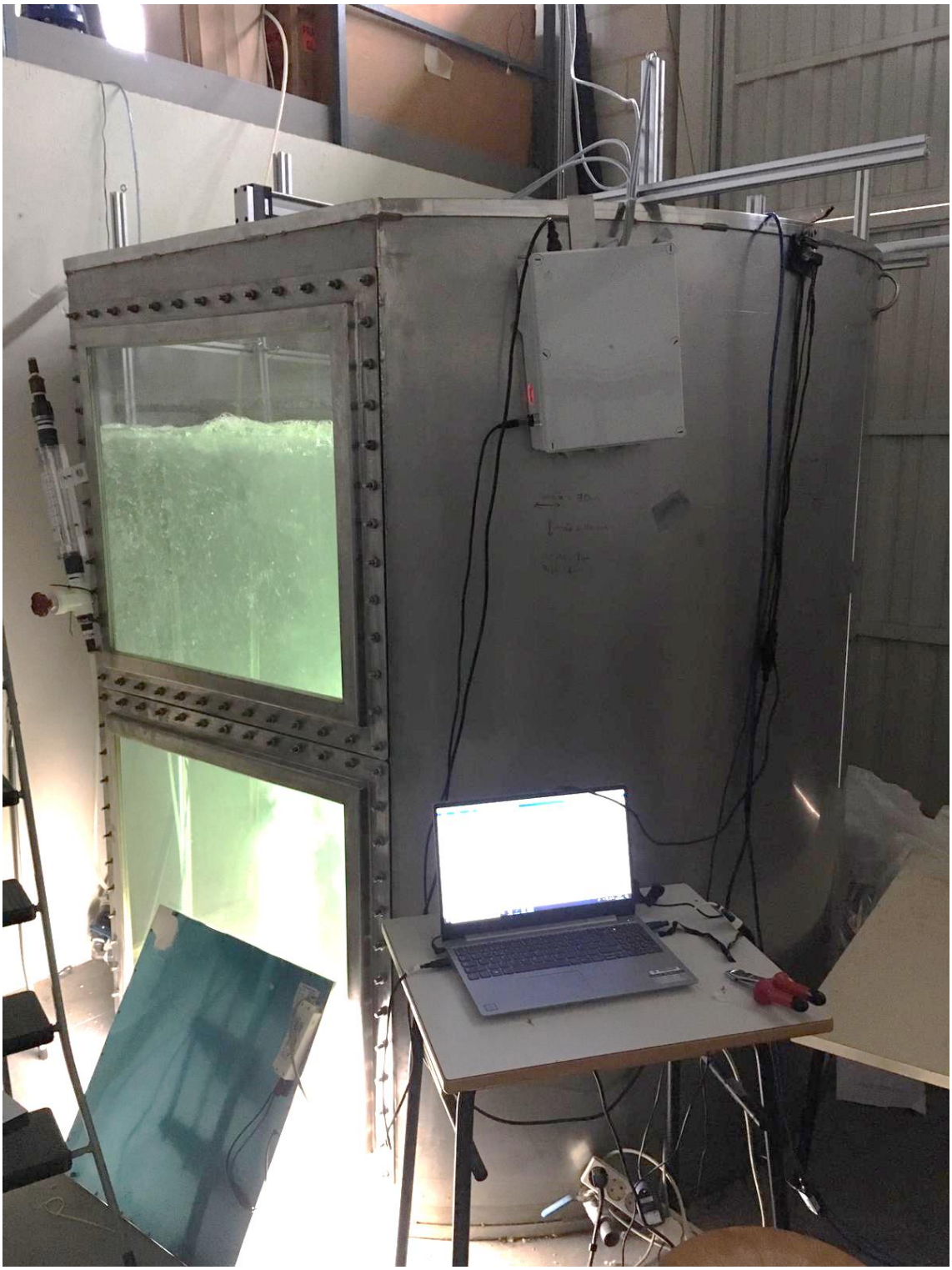


Figura 38. Instalación final

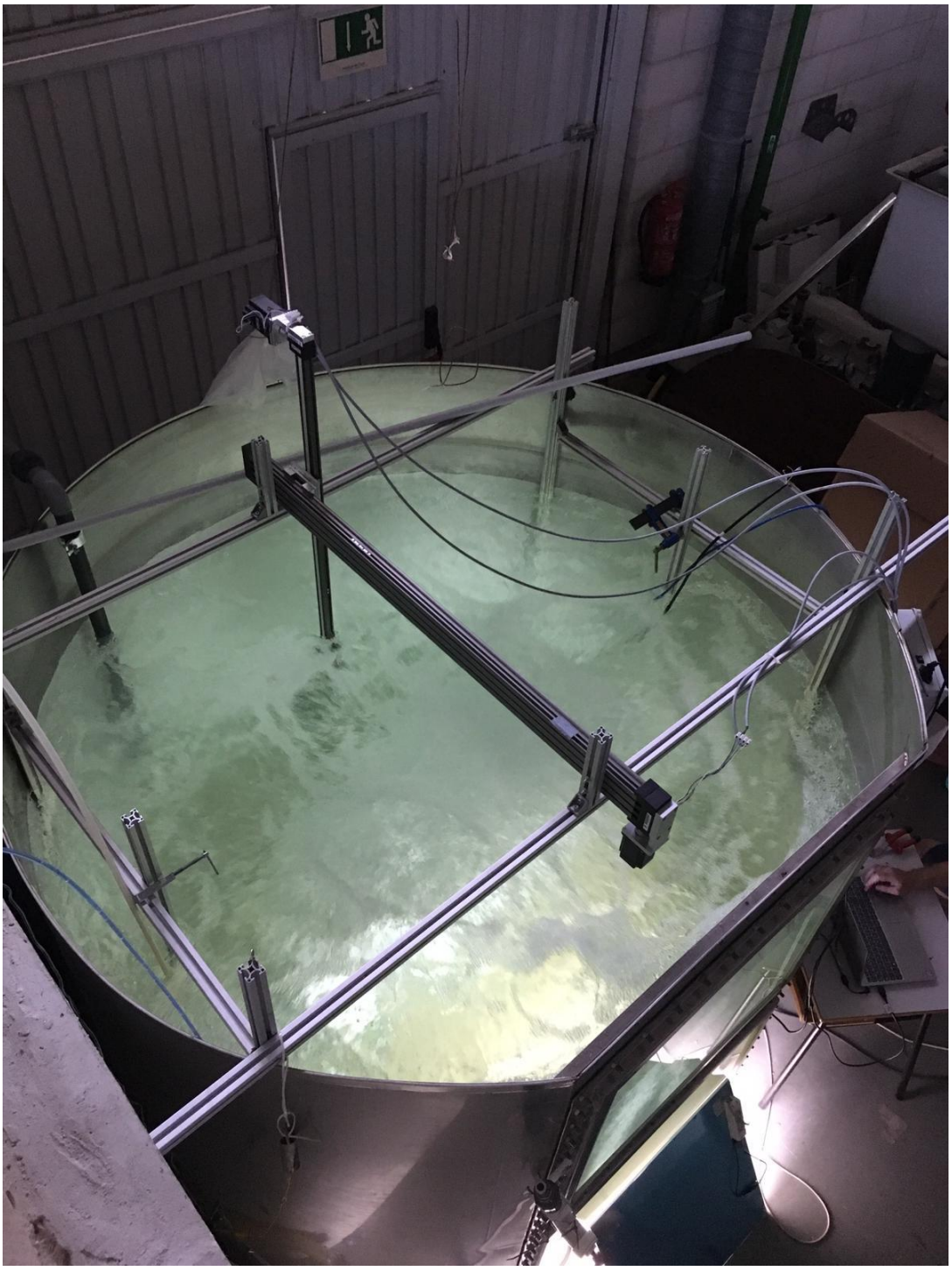


Figura 39. Vista superior de la instalación

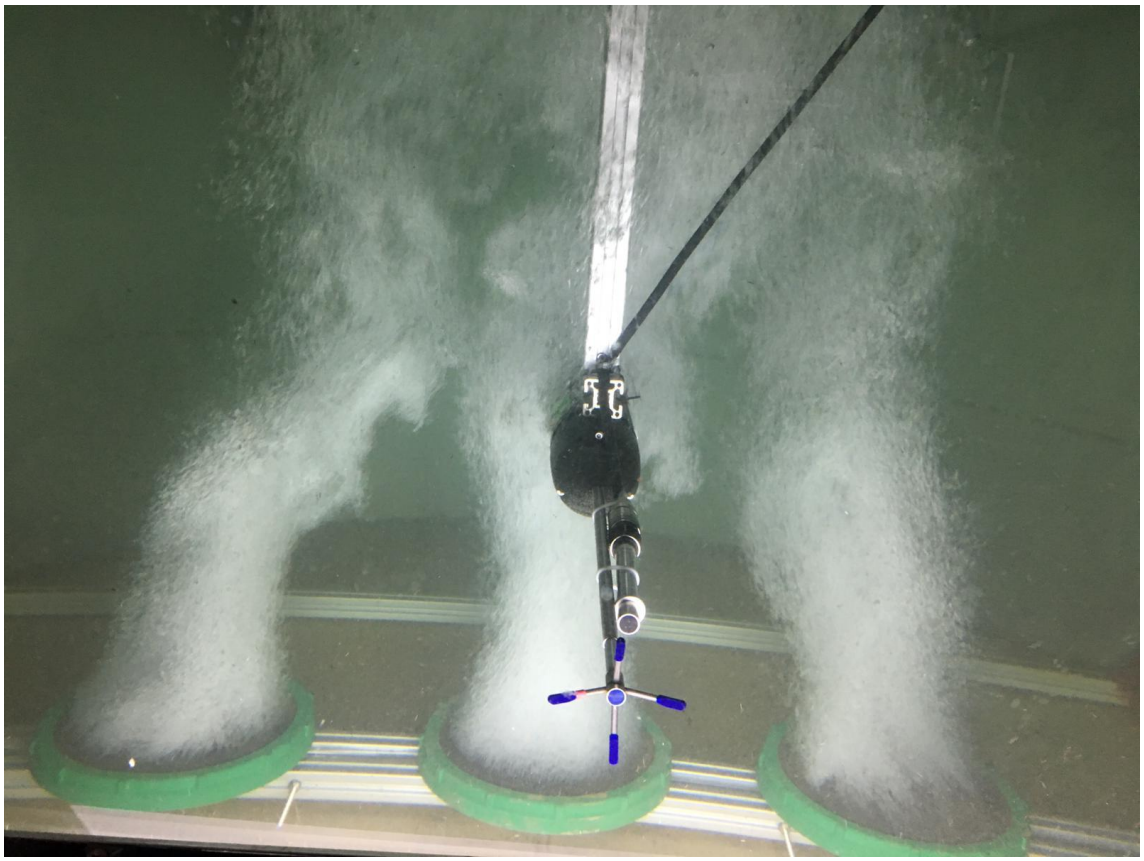


Figura 40. Difusores y Vectrino durante el escaneo

7. Resultados experimentales

Para validar la funcionalidad de la instalación se realizarán unos barridos mediante los cuales se obtendrán resultados tanto de concentración de oxígeno como de velocidad de ascenso de las burbujas y turbulencia. Los resultados de la concentración de oxígeno se obtendrán mediante la sonda de oxígeno, y nos determinarán la evolución de la concentración de O_2respecto del tiempo, parámetro necesario para calcular la tasa de oxígeno. El otro parámetro a medir será la turbulencia, parámetro del que depende también el coeficiente de transferencia local K_L. Los resultados obtenidos se presentan a continuación y verifican el correcto funcionamiento de la instalación.

Para saber la condenada correspondiente a cada dato de medida, crearemos mediante MATLAB un archivo temporal gracias al cual podemos asociar cada medida de la sonda o del Vectrino a un punto de la instalación, de manera que se puedan identificar zonas críticas para la ubicación de los difusores.

7.1. Determinación de la concentración de oxígeno

La concentración de oxígeno en función del tiempo la obtendremos con la sonda de O_2. El sensor de oxígeno se encarga de recoger datos de la cantidad de oxígeno que se encuentra en el flujo, de modo que podemos medir la variación de oxígeno frente al tiempo y de ese modo obtener la tasa de oxígeno mediante la siguiente formula:

$$\frac{dC(t)}{dt} = k_L a_i \left(C(t) - C_{sat}\right) \qquad \text{Ec. 16}$$

Los resultados de la evolución de la concentración de oxígeno obtenidos se muestran en la siguiente figura.

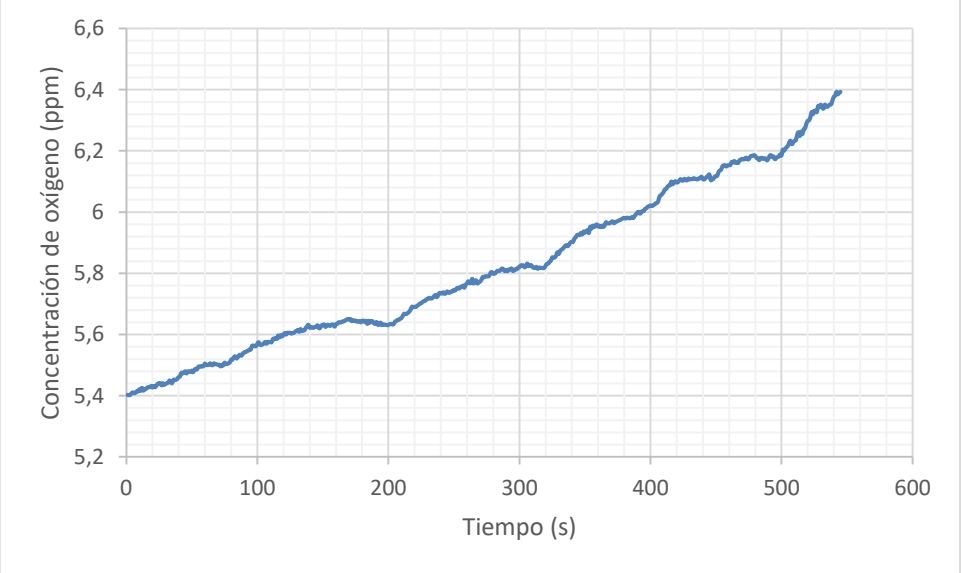


Figura 41. Evolución de la transferencia de oxígeno

Cabe mencionar que las irregularidades en la pendiente de la gráfica se deben a que los datos han sido medidos con la sonda en movimiento. Si la sonda hubiera permanecido en una posición fija, el incremento de la concentración de oxígeno hubiera sido monótono, ralentizándose a medida que nos aproximamos a la concentración de saturación (C_{sat}).

Con esta medición obtenemos el valor de la concentración de O2, C(t), por lo que despejando de la ecuación 16, obtenemos el valor del coeficiente de transferencia local K_L.

De la ecuación 16 se puede despejar k_l ya que:

- $\frac{dC(t)}{dt}$ se calcula a partir de la curva (es la pendiente)
- $C(t)$ es lo que da el oxímetro
- C_{sat} depende de la composición del agua y la temperatura. Se mide para un agua en concreto introduciendo aire mucho tiempo y viendo cuál es el valor máximo que se alcanza. Como depende de la temperatura, la sonda de oxígeno tiene un sensor de temperatura incorporado.
- a_i en el futuro se medirá con localmente con sondas de conductividad. Ahora se saca globalmente midiendo el tamaño de burbuja con fotografías, con su tiempo de contacto, el caudal de gas y el volumen de agua total.

a)

b)

c)

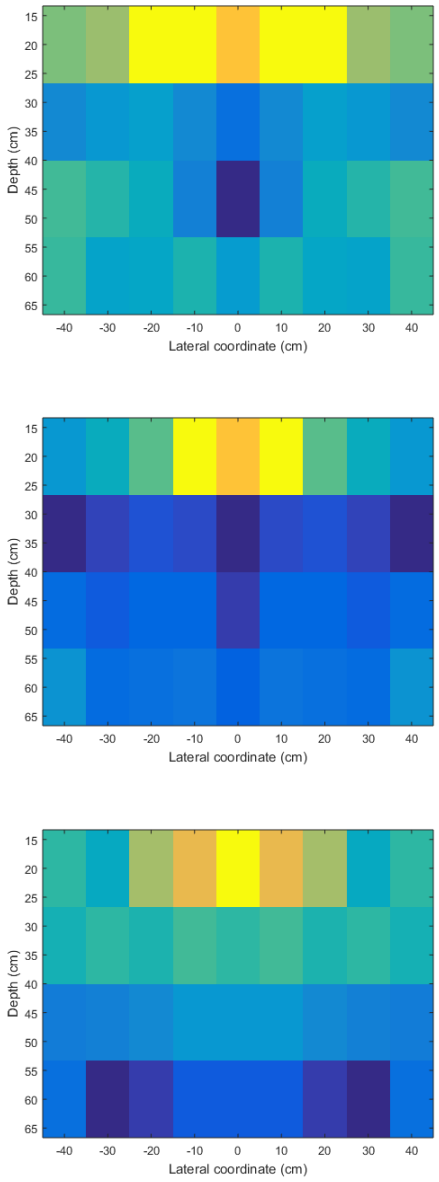


Figura 42. Secciones meridionales de concentración de oxígeno para distintos caudales de aire inyectados, a) 25 lpm, b) 50 lpm y c) 75 lpm

En la figura 42 se puede ver la concentración de oxígeno en la sección de medida. Las zonas representadas con colores cálidos muestran valores de concentración elevados, mientras que las zonas con concentración de oxigeno más baja se muestra con colores azulados.

7.2. Mediciones de la turbulencia y la velocidad

En las siguientes figuras se puede apreciar la variación de la velocidad del flujo en función del tiempo, valor que nos servirá en un futuro para calibrar el análisis por códigos de simulación numérica de mecánica de fluidos (CFD) en un futuro. En cada una de las gráficas podemos apreciar la variación de la velocidad en cada una de las direcciones de manera que queda perfectamente cuantificado este parámetro.

Observando los resultados podemos ver que en los ejes U y V prácticamente no hay variación de la velocidad, pero en el eje W sí que la hay ya que es el eje que representa el ascenso del flujo de gas. El valor medio en este eje se sitúa en torno a 0.1 m/s y representa la velocidad del agua.

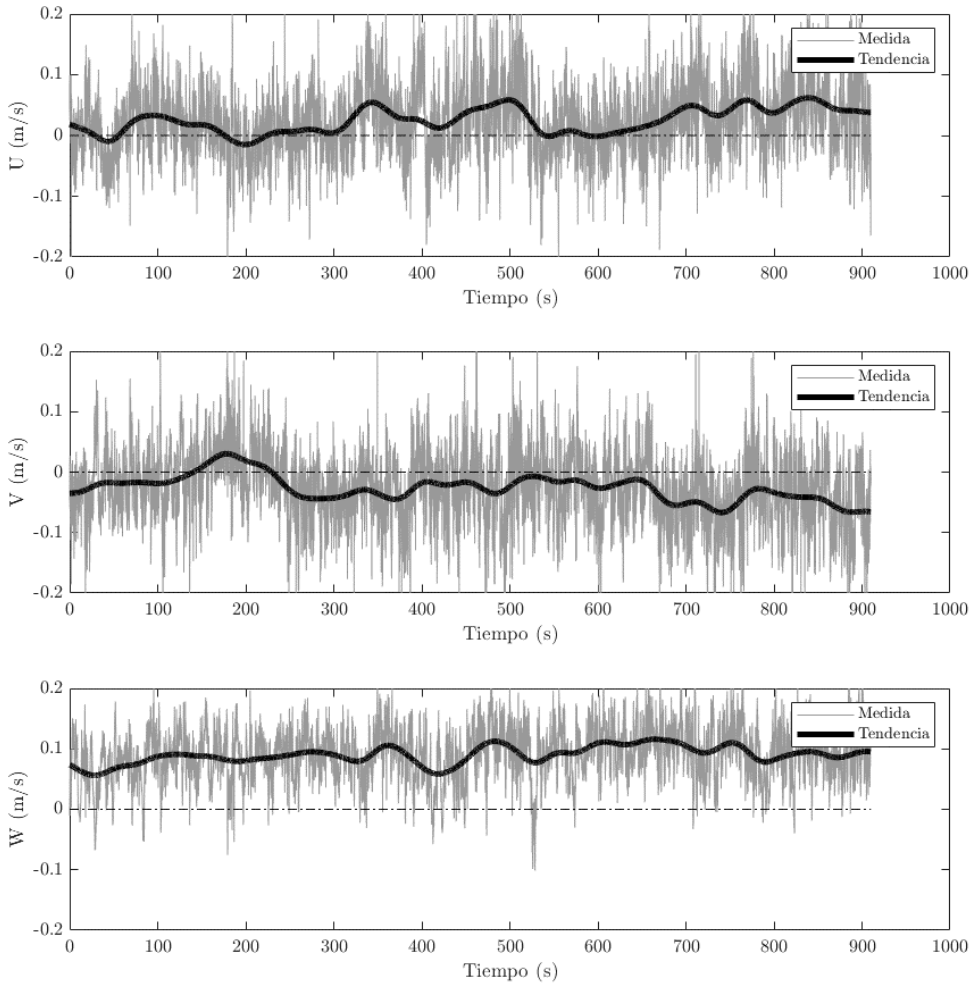


Figura 43. Medida directa de las componentes de la velocidad con el Vectrino y procesado de la señal para obtener velocidades medias instantáneas

Otra gráfica que se ha podido obtener y que será necesaria para la calibración futura del análisis por CFD es la del campo de velocidades. Figura que podemos observar a continuación.

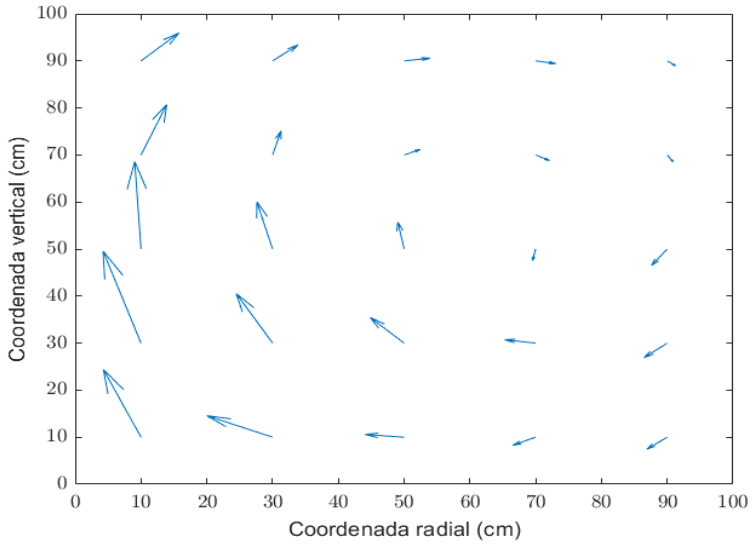


Figura 44. Señal procesada del Vectrino para obtener el campo de velocidades

La figura 43 muestra el flujo que se forma en el interior del fluido.

8. Análisis e interpretación de los resultados

A la vista de los resultados, se puede observar como la instalación es capaz de realizar barridos tanto horizontales como verticales, de modo que queda validada la funcionalidad de la instalación. También podemos apreciar atendiendo a las gráficas, como los resultados que se muestra son coherentes, ya que tanto los niveles de oxígeno como la turbulencia medida han sido los esperados.

Con esto queda finalizada la validación del proyecto, cuya transcendencia futura se explica en el siguiente punto.

9. Conclusiones y transcendencia futura

Se ha podido comprobar el correcto funcionamiento de la instalación para la distribución de difusores mencionada, la cual ha sido con tres difusores alineados en el centro del depósito. Por lo tanto, se ha diseñado y construido un banco de ensayos fiable y robusto el cual es capaz de caracterizar diferentes distribuciones de difusores.

La transcendencia futura de esta instalación será la obtención de la distribución que optimice dicha transferencia, de modo que se proporcione un ahorro energético haciendo más eficientes y sostenibles las EDAR.

10. Referencias

https://www.google.es/search?q=organismos+aerobicos&rlz=1C1CHBF_esES821ES821&oq=organismos+aerobicos&aqs=chrome..69i57j0l5.5672j1j7&sourceid=chrome&ie=UTF-8

https://es.slideshare.net/zarethitha/patrones-de-flujo

http://depa.fquim.unam.mx/amyd/archivero/TM2013-11-0517a_25720.pdf

http://www.scielo.org.co/pdf/biote/v15n2/v15n2a13.pdf

http://www.lis.edu.es/uploads/967d742f_455b_4bd0_a29f_438968130ea1.pdf

https://www.Arduino.cc/

https://programarfacil.com/blog/Arduino-blog/curso-de-Arduino/

https://es.wikipedia.org/wiki/Ox%C3%ADmetro

http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1665-27382012000200006

http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59442015000100001

https://pdfs.semanticscholar.org/ff49/05461b98b3ce29c9bd1a047810f5d75cf012.pdf

https://openlanuza.com/utilizando-un-rele-en-Arduino/

http://diwo.bq.com/utilizar-rele-Arduino-zum-core/

https://www.mouser.es/Electromechanical/Motors-Drives/Stepper-Motors/_/N-u8a8

https://en.wikipedia.org/wiki/Stepper_motor

https://earchivo.uc3m.es/bitstream/handle/10016/10234/PFC_Daniel_Penalba_Sanchez.pdf?sequence=2

https://es.scribd.com/document/285863006/Resolucion-de-tensiones-en-deposito-a-presion-mediante-Ansys-Workbench

http://oa.upm.es/50284/1/TFG_FRANCISCO_MURILLO_MARTINEZ.pdf

II. PLIEGO DE CONDICIONES

1. Introducción

En esta sección se describen las condiciones de trabajo; objetivo y emplazamiento del proyecto, personal de trabajo para la instalación y los ensayos experimentales, condiciones legales y técnicas y todos los datos iniciales. Seguidamente se exponen las cláusulas administrativas; especificaciones burocráticas y económicas que deben cumplirse por las partes implicadas en el proyecto. A continuación, se describen las exigencias técnicas y particulares. En esta sección se describen los materiales y los equipos necesarios, así como las condiciones de ejecución para llevar a cabo el proyecto. Por último, se exponen las condiciones finales describiendo las medidas de seguridad y como se debe implementar el proyecto.

2. Condiciones generales

2.1. Objetivo y emplazamiento

Con este proyecto se pretende diseñar una instalación capaz de caracterizar el comportamiento de los difusores empleados en las estaciones depuradoras de aguas residuales. El desarrollo y redacción del proyecto ha sido llevado a cabo en el edificio de talleres (TT) del Campus Riu Sec de la Universidad Jaume I bajo unas condiciones climatológicas y de salubridad adecuadas.

2.2. Personal involucrado

El diseño del depósito, el cuadro eléctrico y la programación del microcontrolador ha sido realizada por personal relacionado directamente con la consecución del proyecto de investigación. Todo el trabajo ha sido supervisado por un Director Técnico quien ha evaluado y guiado al equipo en la consecución del objetivo.

Todo técnico que intervenga de algún modo el desarrollo del proyecto debe estar lo suficientemente cualificado para la tarea que realiza, sea mediante formación o experiencia. Además, la formación de los técnicos debe ser llevada a cabo de acuerdo con un plan de desarrollo y siempre de forma previa a las actividades del proyecto.

La construcción del proyecto se ha subcontratado a una empresa externa con la cual se ha mantenido un contacto a medida que iban avanzando en la construcción del depósito. Los elementos defectuosos serán desechados y se reclamará su devolución al proveedor.

2.3. Fecha de inicio

Para el desarrollo de la instalación, el director dejará la fecha de inicio especificada en el proyecto, tanto en la memoria como en el presupuesto. Cualquier cambio que suponga alguna diferencia respecto al mencionado proyecto será notificado y aprobado por la dirección técnica.

3. Cláusulas administrativas

3.1. Documentación

El ingeniero proyectista entregará una copia del proyecto al departamento de Ingeniería Mecánica y Construcción de la Universidad Jaume I y en ese momento pasará a ser propiedad de esta. El proyectista mantendrá en completa confidencialidad el proceso experimental, las pruebas y el funcionamiento de la instalación. El documento constará de las siguientes partes:

a) Memoria

En la memoria se detalla el contenido del proyecto, desde los objetivos hasta la obtención de los resultados, y se detalla los pasos que se deben seguir para la realización del mismo. La memoria se inicia con una introducción donde se enmarca el proyecto y se presenta la necesidad que motiva a la realización del mismo. Se continua con una descripción clara y concisa de los objetivos y el alcance. Finalmente se concluye con los resultados, las conclusiones y las propuestas de futuro.

b) Pliego de condiciones

En este apartado se presentan las condiciones bajo las cuales se encuentra el proyecto, el personal involucrado y las necesidades técnicas necesarias para su realización.

c) Planos

Se incluirán los planos relacionados con la instalación.

d) Anexos

Se incluirá el código de programación del Arduino mediante el cual se controla el funcionamiento de la instalación.

e) Presupuesto

Esta parte refleja el coste económico del proyecto. El coste total se dividirá en costes de trabajo y desarrollo, costes de material y costes del equipo informático.

4. Requisitos técnicos

Los equipos y elementos empleados deberán ser nuevos o presentar una buena calidad, y se ajustarán a las especificaciones que para los mismos hayan sido requeridos por las normas y disposiciones oficiales y por el presente Pliego de Condiciones.

4.1. Depósito

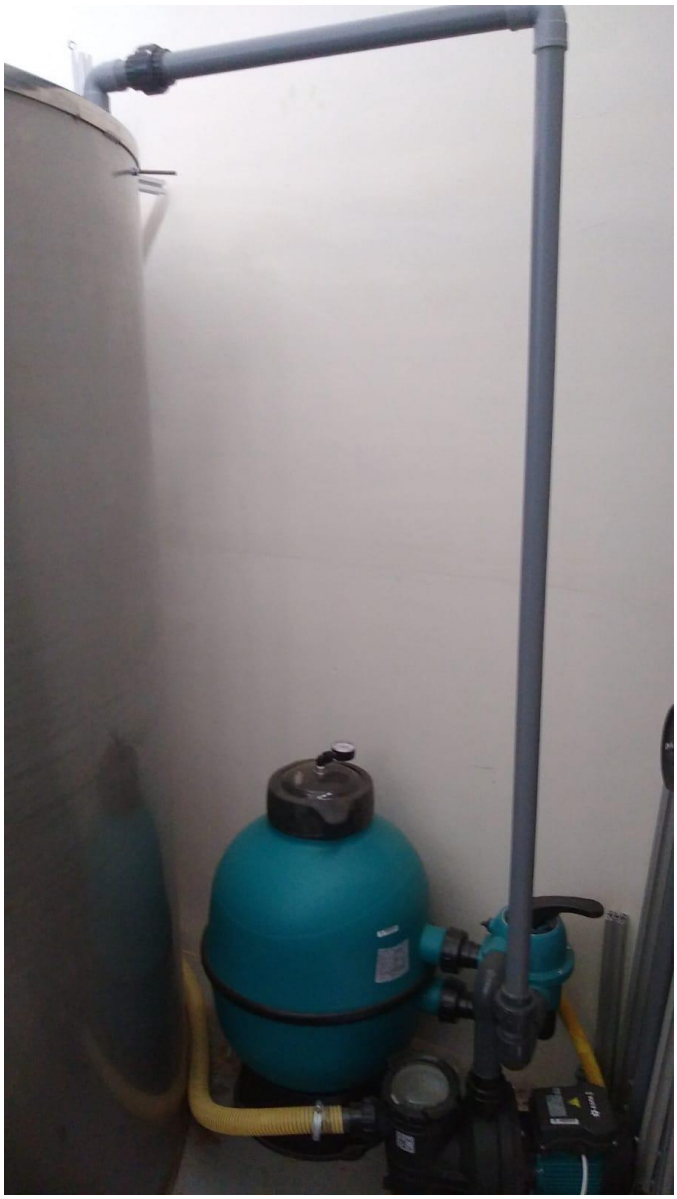
El depósito ha de ser capaz de almacenar aproximadamente 8.000 litros de agua sin presentar fugas y sin contaminar su contenido. Su instalación se deberá realizar de una manera adecuada con la ayuda de una empresa externa que será la encargada del transporte y manejo del torito monta cargas. La instalación en el taller se hará de modo que no interrumpa la salida y no impida el trabajo de los docentes.

4.2. Cuadro eléctrico

El cableado del cuadro eléctrico se realizará adecuadamente de modo que los cables queden perfectamente unidos a los borneros y no se produzcan corto circuitos ni derivaciones que puedan poner en peligro al usuario.

4.3. Circuito auxiliar

El circuito encargado de filtrar las partículas dispersas en el fluido ha de ser capaz de realizar su función, de modo que la bomba incorporada será capaz de vencer la diferencia de altura desde la base del depósito a la parte de arriba y también de vencer la resistencia aplicada por el filtro de arena.



El circuito encargado de trasegar el aire a presión desde el compresor hasta el difusor ha de ser capaz de trabajar bajo condiciones de presurización.

4.4. Estructura interna

La estructura interna ha de ser capaz de sustentar los difusores y de resistir la corrosión ya que se encontrará totalmente sumergida. También será capaz de deslizar por unas guías y facilitar las extracción e intercambio de los difusores.

4.5. Condiciones de funcionamiento

El barrido de los motores estará siempre supervisado por el técnico o usuario de la instalación, de modo que, en caso de error, se pueda reaccionar rápido desactivando la fuente de alimentación desde el interruptor o bien desenchufando la fuente de la red general.

4.6. Elementos de medida

Los elementos de medidas deberán ser fiables y estar en perfecto estado de calibración, de modo que, previo a su uso, se realizarán ensayos de modo que se valide su funcionamiento.

4.7. Requisitos humanos

La instalación completa de los elementos del reactor debe ser realizada por personal cualificado.

5. Condiciones finales

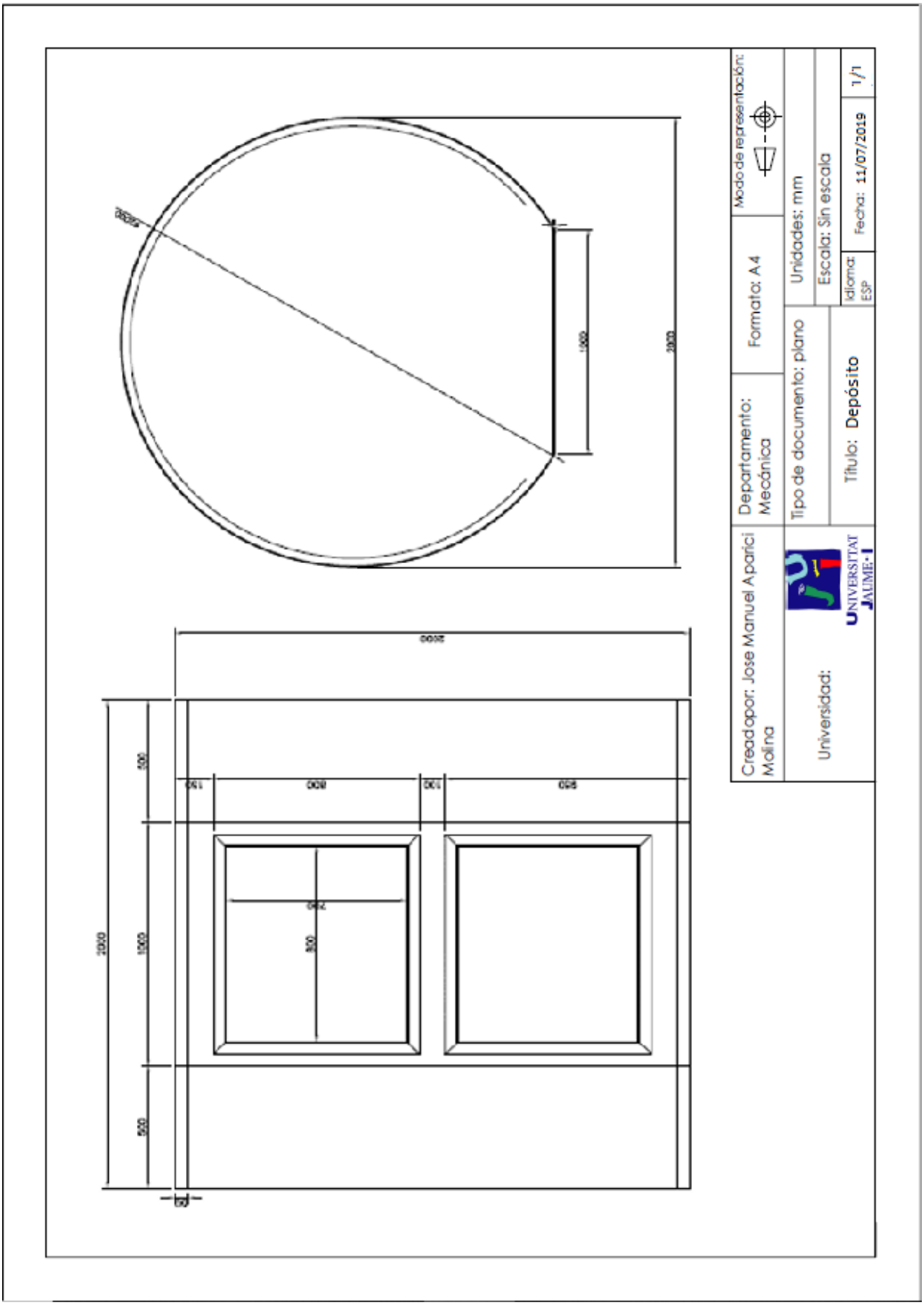
5.1. Medidas de seguridad

Los administradores están obligados a señalizar la instalación utilizando las señales apropiadas y notificándolo al personal, tomando las medidas adecuadas para prevenir accidentes.

5.2. Tiempo de ejecución

El plazo de realización de este proyecto es de seis meses, tiempo en el cual se ha diseñado, construido y se han realizados las medidas que verifiquen el correcto funcionamiento de la instalación.

III. PLANOS


Creado por: Jose Manuel Aparici Molina
Universidad:
UNIVERSITAT JAUME·I
Departamento: Mecánica
Formato: A4
Modo de representación:
Tipo de documento: plano
Unidades: mm
Escala: Sin escala
Título: Depósito
Idioma: ESP
Fecha: 11/07/2019
1/1

IV. ANEXOS

Anexo 1. Código de MATLAB

1. Código principal del GUIDE

```
function varargout = JOSEMANUEL_GUIDE_v11(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton,
...
                   'gui_OpeningFcn',
@JOSEMANUEL_GUIDE_v11_OpeningFcn, ...
                   'gui_OutputFcn',
@JOSEMANUEL_GUIDE_v11_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before JOSEMANUEL_GUIDE_v11 is
made visible.
function JOSEMANUEL_GUIDE_v11_OpeningFcn(hObject,
eventdata, handles, varargin)
handles.output = hObject; %inicializacion Arduino
clc
clear ard1
clear handles.ard1
for i=1:10
    delete(instrfind({'Port'},{['COM'
num2str(i)]}));%deja libre conexion al COM
end
% Update handles structure
i=1;
isArduino=0;
while and(isArduino==0,i<10)
```

```
    i=i+1;
    try
    handles.ard1=Arduino(['COM' num2str(i)]);
    isArduino=1;
    end
end
%se crea la variable debugmode y así el código se
salta los pasos que
%necesite conexión con el ard
handles.debugmode=1-isArduino;

if handles.debugmode==0
pinMode(handles.ard1,3,'OUTPUT'); %se conecta a STEP
de la tarjeta (OK)
pinMode(handles.ard1,2,'OUTPUT');
pinMode(handles.ard1,4,'OUTPUT');
pinMode(handles.ard1,6,'OUTPUT'); %pin 6 del Arduino
al relé (OK)
pinMode(handles.ard1,10,'OUTPUT'); %se conectan a la
DIRECCIÓN de la tarjeta (OK)
% pinMode(handles.ard1,12,'INPUT'); %se conecta a
finales de carrera
% pinMode(handles.ard1,13,'INPUT'); %se conecta a
finales de carrera
end

guidata(hObject, handles);
%Default values
set(handles.pxp,'string','0.02');
set(handles.axp,'string','0.02');
set(handles.freq,'string','1000');
set(handles.distanciaX,'string','10');
set(handles.distanciaY,'string','10');
set(handles.incr,'string','10');
set(handles.tiempo,'string','120');
set(handles.nscan,'string','2');
set(handles.pause,'string','0.1');
handles.sentidoX=1;
handles.sentidoY=1;

%+++NO ENTIENDO

handles.x_range=[0,20]; %mm
handles.y_range=[0,20]; %mm


%Pines entradas/salidas Arduino
```

```
% handles.PIN_VectrinoX = 0; %VECTRINO DIGITAL INPUT
% handles.PIN_VectrinoY = 1; %VECTRINO DIGITAL INPUT
% handles.PIN_VectrinoZ1 = 2; %VECTRINO DIGITAL INPUT
% handles.PIN_VectrinoZ2 = 3; %VECTRINO DIGITAL INPUT
handles.PIN_Valor_sonda = 1; %al pin 1 del ard
conectaré la sonda
handles.PWM_motor = 3; %señales de accionamiento de
los motores
handles.PIN_sentido_motor= 10; %señal de dirección
handles.PIN_rele = 6;
% handles.PIN_FinalCarrera_X = 12;  %entrada
% handles.PIN_FinalCarrera_Y = 13;  %entrada
% handles.PIN_ParadaEmergencia_MotorX = 2;
% handles.PIN_ParadaEmergencia_MotorY = 4;

global x y fila
x = 0;
y = 0;
fila=1;

handles.output = hObject;
guidata(hObject, handles);

% --- Outputs from this function are returned to the
command line.
function varargout =
JOSEMANUEL_GUIDE_v11_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;

% --- Executes when figure1 is resized.
function figure1_SizeChangedFcn(hObject, eventdata,
handles)

function pxp_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting
all properties.
function pxp_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function axp_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting
all properties.
function axp_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function freq_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting
all properties.
function freq_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function distanciaX_Callback(hObject, eventdata,
handles)

% --- Executes during object creation, after setting
all properties.
function distanciaX_CreateFcn(hObject, eventdata,
handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function distanciaY_Callback(hObject, eventdata,
handles)

% --- Executes during object creation, after setting
all properties.
function distanciaY_CreateFcn(hObject, eventdata,
handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in mover_x.
function mover_x_Callback(hObject, eventdata, handles)
```

```
% if get(handles.sentido_x,'Value')==1 %si es +X
%     sentido=1
% elseif get(handles.sentido_x,'Value')==2 %si es -X
%     sentido=0
% end


if handles.debugmode==0
    digitalWrite(handles.ard1,handles.PIN_rele,0);
%escribimos para que el relé se quede como está
    disp('el relé está desactivado')
end



        set(handles.mover_x, 'BackgroundColor',[0 1
0])  %color verde cuando esta en movimiento
        pause(0.2); %el pause es necesario para que
el fondo se coloree de verde
        npasos =
round(str2double(get(handles.distanciaX,'string'))/str
2double(get(handles.pxp,'string'))*7.5);
        if handles.debugmode == 0

digitalWrite(handles.ard1,handles.PIN_sentido_motor,ha
ndles.sentidoX);  %le decimos al motor el sentido
        sentidoX = handles.sentidoX
%           a =
digitalRead(handles.ard1,handles.PIN_sentido_motor_X)
        end
        pause(0.2)
        T=1/str2double(get(handles.freq,'string'));
        ipaso=0;
% Inicio del ciclo TTL
        tic
% Inicio del contador
        pinValue=1;
        while ipaso < npasos  %npasos vale 3750 por
defecto

            while  toc<(T/2) %Abrimos bucle para
indicar que se ponga a 1 la señal TTL mientras el
contador se apague durante la mitad del período
            end

                pinValue=1-pinValue;
```

```
                if handles.debugmode==0

digitalWrite(handles.ard1,handles.PWM_motor,pinValue);
%pin 3 del ard al pin 9 de la tarjeta
                end
                tic
                ipaso = ipaso + 1; %Duración ciclo TTL
        end

    pause(0.2)

j = 0.8;
for i = 1:2
    j = 1-j;
    set(handles.mover_x, 'BackgroundColor',[1 1 j])
    pause(0.5)
end
set(handles.mover_x, 'BackgroundColor',[0.8 0.8 0.8])
guidata(hObject, handles);

% --- Executes on button press in mover_y.
function mover_y_Callback(hObject, eventdata, handles)

% if get(handles.sentido_y,'Value')==1
%     sentido=1
% elseif get(handles.sentido_y,'Value')==2
%     sentido=0
% end

if handles.debugmode==0
    digitalWrite(handles.ard1,handles.PIN_rele,1);
%escribimos para que el relé conmute
end
        set(handles.mover_y, 'BackgroundColor',[0 1
0])
        pause(0.2); %el pause es necesario para que
el fondo se coloree de verde


npasos=round(str2double(get(handles.distanciaY,'string
'))/str2double(get(handles.axp,'string'))*7.5);
        %divide los mm por pulso entre los mm para
obtener los pulsos
        %necesarios
        %disp(npasos) -- por defecto valdrá 3750
        if handles.debugmode==0
```

```
digitalWrite(handles.ard1,handles.PIN_sentido_motor,ha
ndles.sentidoY);
          sentidoY = handles.sentidoY
%           b =
digitalRead(handles.ard1,handles.PIN_sentido_motor_Y)
          end
          pause(0.2)
          %global x y
          T=1/str2double(get(handles.freq,'string'));
          ipaso=0; %Inicio del ciclo TTL
          tic  %Inicio del contador
          pinValue=1;
         while ipaso<npasos
              while  toc<(T/2) %Abrimos bucle para
indicar que se ponga a 1 la señal TTL mientras el
contador se apague durante la mitad del período
              end
                  pinValue=1-pinValue; %Cuando el Pin
de salida 3 reciba una señal '1' de encendido,
ordenamos que vuelva a '0'
                  if handles.debugmode==0

digitalWrite(handles.ard1,handles.PWM_motor,pinValue);
%Damos a la salida del Pin nº 3 un determinado valor
(1, encendido)
                  end
                  tic
                  ipaso=ipaso+1; %Duración ciclo TTL
set(handles.mover_y, 'BackgroundColor',[0 0.8 0])
        end

%set(handles.mover_y, 'BackgroundColor',[0 0 1])
%color azul cuando esta parado
%pause(1)
%set(handles.mover_y, 'BackgroundColor',[0.8 0.8 0.8])
%vuelve al color por defecto tras parar 1 segundo
j = 0.8;
for i = 1:2
    j = 1-j;
    set(handles.mover_y, 'BackgroundColor',[1 1 j])
    pause(0.5)
end
set(handles.mover_y, 'BackgroundColor',[0.8 0.8 0.8])
guidata(hObject, handles);
```

```
% --- Executes on button press in
guardarValor_conductividad.
function guardarValor_conductividad_Callback(hObject,
eventdata, handles)
t=7;
fila=1;
tic
while 1
        %tomar datos sonda
        if handles.debugmode == 0 %se salta el trozo
si el ard no está
        [volt,hora] = medir_conductividad(handles,t);
        datos{1,fila}=volt;
        datos{2,fila}=hora;
        fila=fila+1;
        %ya tenemos una tabla con valores de
mediciones y tiempo
        save('CONDUCTIVIDAD_RESULTS.mat','datos')
%guardamos la variable 'datos' en un archivo .MAT
        %VECTRINO%VECTRINO%VECTRINO
        %VECTRINO%VECTRINO%VECTRINO
        %VECTRINO%VECTRINO%VECTRINO
        %VECTRINO%VECTRINO%VECTRINO
        [medidaX,medidaY,medidaZ1,medidaZ2,hora] =
medir_velocidad(handles,t);
        datosVectrino{1,fila}=medidaX;
        datosVectrino{2,fila}=medidaY;
        datosVectrino{3,fila}=medidaZ1;
        datosVectrino{4,fila}=medidaZ2;
        datosVectrino{5,fila}=hora;
        fila=fila+1;
        %ya tenemos una tabla con valores de
mediciones y tiempo
        save('VECTRINO_RESULTS.mat','datosVectrino')
%guardamos la variable 'datosVectrino' en un archivo
.MAT
        end
end
  guidata(hObject, handles);

% --- Executes on button press in barrido_1DX.
function barrido_1DX_Callback(hObject, eventdata,
handles)
%% BARRIDO 1DX
    set(handles.barrido_1DX, 'BackgroundColor',[0 1
0])  %color verde cuando esta en movimiento
```

```
    pause(0.2); %el pause es necesario para que el
fondo se coloree de verde

    incr = str2double(get(handles.incr,'string'));
    t = str2double(get(handles.tiempo,'string'));
    sentido=[1 0 1 0 1];
    nscan=length(sentido);
    for s=1:nscan
        s
        %for i=1:n
            %tomar datos
            if handles.debugmode == 0

            %MEDIR CONDUCTIVIDAD-MEDIR CONDUCTIVIDAD-
MEDIR CONDUCTIVIDAD-MEDIR CONDUCTIVIDAD
    %
[dat,hora]=medir_conductividad(handles,t);
    %          l=length(dat);
    %          volt{1,i}=dat;
    %          volt{2,i}=hora;

            %MEDIR VELOCIDAD-MEDIR VELOCIDAD-MEDIR
VELOCIDAD-MEDIR VELOCIDAD-MEDIR VELOCIDAD
    %
[medidaX,medidaY,medidaZ1,medidaZ2,hora] =
medir_velocidad(handles,t);
    %          datosVectrino{1,i}=medidaX;
    %          datosVectrino{2,i}=medidaY;
    %          datosVectrino{3,i}=medidaZ1;
    %          datosVectrino{4,i}=medidaZ2;
    %          datosVectrino{5,i}=hora;
            mover_MX(handles,incr,sentido(s))
            pause(t)

            end
    end
%     if handles.debugmode == 0
%          voltage{s}=volt;
%          save('RESULTS4.mat','voltage')
%          datosVectrino1D{1,s}=datosVectrino;
%
save('VECTRINO1D_RESULTS.mat','datosVectrino1D')
%     end
%
% %NO ENTIENDO POR QUE ESTÁ OTRA VEZ EL SAVE
% if handles.debugmode == 0
%   save('RESULTS4.mat','voltage')
```

```
%    save('VECTRINO1D_RESULTS.mat','datosVectrino1D')
% end
j = 0.8;
for i = 1:3
    j = 1-j;
    set(handles.barrido_1DX, 'BackgroundColor',[1 1
j])
    pause(0.5)
end
set(handles.barrido_1DX, 'BackgroundColor',[0.8 0.8
0.8])
set(handles.mover_x, 'BackgroundColor',[0.8 0.8 0.8])
digitalWrite(handles.ard1,handles.PIN_sentido_motor,0)
%escribo un 0 en el PIN del la dirección
disp('FIN DEL SCAN 1D')
guidata(hObject, handles);

% --- Executes on button press in barrido_1DY.
function barrido_1DY_Callback(hObject, eventdata,
handles)
%% BARRIDO 1DY
set(handles.barrido_1DY, 'BackgroundColor',[0 1 0])
%color verde cuando esta en movimiento
pause(0.2); %el pause es necesario para que el fondo
se coloree de verde
incr = str2double(get(handles.incr,'string'));
t = str2double(get(handles.tiempo,'string'));
sentido=[1 0 1 0 1];
nscan=length(sentido);
    for s=1:nscan
        s
        %for i=1:n
            %tomar datos
            if handles.debugmode == 0

            %MEDIR CONDUCTIVIDAD-MEDIR CONDUCTIVIDAD-
MEDIR CONDUCTIVIDAD-MEDIR CONDUCTIVIDAD
    %
[dat,hora]=medir_conductividad(handles,t);
    %          l=length(dat);
    %          volt{1,i}=dat;
    %          volt{2,i}=hora;

            %MEDIR VELOCIDAD-MEDIR VELOCIDAD-MEDIR
VELOCIDAD-MEDIR VELOCIDAD-MEDIR VELOCIDAD
```

```
    %
[medidaX,medidaY,medidaZ1,medidaZ2,hora] =
medir_velocidad(handles,t);
    %           datosVectrino{1,i}=medidaX;
    %           datosVectrino{2,i}=medidaY;
    %           datosVectrino{3,i}=medidaZ1;
    %           datosVectrino{4,i}=medidaZ2;
    %           datosVectrino{5,i}=hora;

              %MOVER EJE-MOVER EJE-MOVER EJE-MOVER EJE-
MOVER EJE-MOVER EJE-MOVER EJE-MOVER EJE
               mover_MY(handles,incr,sentido(s))
               pause(t)
              end
    end
    if handles.debugmode == 0
%           voltage{s}=volt;
%           save('RESULTS4.mat','voltage')
%           datosVectrino1D{1,s}=datosVectrino;
%
save('VECTRINO1D_RESULTS.mat','datosVectrino1D')
    end

%NO ENTIENDO POR QUE ESTÁ OTRA VEZ EL SAVE
% if handles.debugmode == 0
%   save('RESULTS4.mat','voltage')
%   save('VECTRINO1D_RESULTS.mat','datosVectrino1D')
% end
j = 0.8;
for i = 1:3
    j = 1-j;
    set(handles.barrido_1DY, 'BackgroundColor',[1 1
j])
    pause(0.1)
end
set(handles.barrido_1DY, 'BackgroundColor',[0.8 0.8
0.8])
set(handles.mover_y, 'BackgroundColor',[0.8 0.8 0.8])
digitalWrite(handles.ard1,handles.PIN_sentido_motor,0)
%escribo un 0 en el PIN del la dirección
disp('FIN DEL SCAN 1D')
guidata(hObject, handles);


% --- Executes on button press in barrido_2D.
function barrido_2D_Callback(hObject, eventdata,
handles)
```

```
%% CARGAR DATOS DE POSICIONES
%filename = 'Posiciones_SCAN_2D.xlsx';
handles.Posiciones_auto =
xlsread('Posiciones_SCAN_2D_2.xlsx','Hoja2','B3:C22');
t=7;  % tiempo de medida
%% REALIZAR SCAN 2D
for nscan=1:50
    nscan
    for j=1:size(handles.Posiciones_auto,1) %size =
[20 2], es decir, entra en el bucle 20 veces

        distancia_a_moverX =
handles.Posiciones_auto(j,1);
        distancia_a_moverY =
handles.Posiciones_auto(j,2);

        %MEDIR CONDUCTIVIDAD-MEDIR CONDUCTIVIDAD-MEDIR
CONDUCTIVIDAD-MEDIR CONDUCTIVIDA
        [volt,hora] = medir_conductividad(handles,t);
        %l=length(dat);
        datos{1,j}=volt;
        datos{2,j}=hora;

        %MEDIR VELOCIDAD-MEDIR VELOCIDAD-MEDIR
VELOCIDAD-MEDIR VELOCIDAD-MEDIR VELOCIDAD
        [medidaX,medidaY,medidaZ1,medidaZ2,hora] =
medir_velocidad(handles,t);
        datosVectrino{1,i}=medidaX;
        datosVectrino{2,i}=medidaY;
        datosVectrino{3,i}=medidaZ1;
        datosVectrino{4,i}=medidaZ2;
        datosVectrino{5,i}=hora;

        %Movimiento eje X
        if distancia_a_moverX > 0
            sentidoX = 1;  %+X

mover_MX(handles,distancia_a_moverX,sentidoX)
        elseif distancia_a_moverX < 0
           sentidoX = 0; %-X

mover_MX(handles,abs(distancia_a_moverX),sentidoX)
        else
            disp('ERROR  en movimiento MX')
        end

        %Movmiento eje Y
```

```
        if distancia_a_moverY > 0
            sentidoY=0;  %+Y

mover_MY(handles,distancia_a_moverY,sentidoY)
        elseif distancia_a_moverY<0
            sentidoY=1; %-Y

mover_MY(handles,abs(distancia_a_moverY),sentidoY)
        else
            disp('ERROR en movimiento MY')
        end

    end
     disp('FIN SCAN')
     resultado{nscan} = datos;
     save('RESULTS_SCAN2D_2.mat','resultado')
%save(filename,variables)
     datosVectrino2D{nscan} = datosVectrino;
     save('VECTRINO2D_RESULTS','datosVectrino2D')
%save(filename,variables)

end
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 save('RESULTS_SCAN2D_2.mat','resultado')
 save('Vectrino2D_RESULTS','datosVectrino')
 disp('FIN SCAN 2D')

 function t_muestreo_Callback(hObject, eventdata,
handles)

% --- Executes during object creation, after setting
all properties.
function t_muestreo_CreateFcn(hObject, eventdata,
handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function incrT_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting
all properties.
function incrT_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting
all properties.
function grupoX_CreateFcn(hObject, eventdata, handles)

% --- Executes when selected object is changed in
grupoX.
function grupoX_SelectionChangedFcn(hObject,
eventdata, handles)
    get(hObject,'string');
    %ahora compararemos el string con +X, si coincide,
sentidoX =    1

handles.sentidoX=uint8(strcmp(get(hObject,'string'),'+
X'));
    guidata(hObject, handles);


% --- Executes when selected object is changed in
grupoY.
function grupoY_SelectionChangedFcn(hObject,
eventdata, handles)
    get(hObject,'string');
    handles.sentidoY =
uint8(strcmp(get(hObject,'string'),'+Y')); %esto dará
1 o 0
    guidata(hObject, handles);


% --- Executes on selection change in sentido_x.
function sentido_x_Callback(hObject, eventdata,
handles)
        if get(handles.sentido_x,'Value')==1
            handles.sentidoX=1;
%         sentido=1;
        set(handles.sentido_x, 'BackgroundColor',[0
0.6 0])

    elseif get(handles.sentido_x,'Value')==2
%         sentido=0;
      handles.sentidoX=0;
```

```
        set(handles.sentido_x, 'BackgroundColor',[0.8
0 0])
        end
        guidata(hObject, handles);

% --- Executes during object creation, after setting
all properties.
function sentido_x_CreateFcn(hObject, eventdata,
handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

% --- Executes on selection change in sentido_y.
function sentido_y_Callback(hObject, eventdata,
handles)
    if get(handles.sentido_y,'Value')==1
%         sentido=1;
handles.sentidoY=1;
        set(handles.sentido_y, 'BackgroundColor',[0
0.6 0])

    elseif get(handles.sentido_y,'Value')==2
%         sentido=0;
handles.sentidoY=0;
        set(handles.sentido_y, 'BackgroundColor',[0.8
0 0])

    end
  guidata(hObject, handles);

% --- Executes during object creation, after setting
all properties.
function sentido_y_CreateFcn(hObject, eventdata,
handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end



function incr_Callback(hObject, eventdata, handles)
% hObject    handle to incr (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
```

```
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
incr as text
%        str2double(get(hObject,'String')) returns
contents of incr as a double


% --- Executes during object creation, after setting
all properties.
function incr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to incr (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function BARRIDO_Callback(hObject, eventdata, handles)
%% REALIZAR SCAN 2D
set(handles.BARRIDO, 'BackgroundColor',[0 1 0])
%color verde cuando esta en movimiento
pause(0.2); %el pause es necesario para que el fondo
se coloree de verde
handles.Posiciones_auto =
xlsread('Posiciones_SCAN_2D_2.xlsx','Hoja2','B3:C100')
;
t=7;  % tiempo de medida
handles.imedida = 0;
handles.XY = [];
handles.tstart = [];
handles.tend = [];
handles.tmedida =
str2double(get(handles.pause,'string'));
for nscan=1:str2double(get(handles.nscan,'string'))
    for j=1:size(handles.Posiciones_auto,1) %size =
[20 2], es decir, entra en el bucle 20 veces
        handles.imedida = handles.imedida + 1;
```

```
        handles.XY(handles.imedida) = j;
        distancia_a_moverX =
handles.Posiciones_auto(j,1)
        distancia_a_moverY =
handles.Posiciones_auto(j,2)
        %Movimiento eje X
        if handles.debugmode==0
            if distancia_a_moverX > 0
                sentidoX = 1;

mover_MX(handles,distancia_a_moverX,sentidoX)
            elseif distancia_a_moverX < 0
               sentidoX = 0;

mover_MX(handles,abs(distancia_a_moverX),sentidoX)
            else
                disp('No se mueve MX')
            end

        %Movmiento eje Y
            if distancia_a_moverY > 0
                sentidoY = 1;   %+Y

mover_MY(handles,distancia_a_moverY,sentidoY)
            elseif distancia_a_moverY < 0
                sentidoY = 0; %-Y

mover_MY(handles,abs(distancia_a_moverY),sentidoY)
            else
                disp('No se mueve MY')
            end
        end
        handles.tstart(handles.imedida,:) = clock;
        pause(handles.tmedida)
        handles.tend(handles.imedida,:) = clock;
    end

     XY=handles.XY';
     tstart=handles.tstart;
     tend=handles.tend;
     save('resultscan.mat','XY','tstart','tend');
end
    set(handles.BARRIDO, 'BackgroundColor',[0.8 0.8
0.8])
     disp('FIN SCAN')
```

```
function tiempo_Callback(hObject, eventdata, handles)
% hObject    handle to tiempo (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
tiempo as text
%        str2double(get(hObject,'String')) returns
contents of tiempo as a double


% --- Executes during object creation, after setting
all properties.
function tiempo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tiempo (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pause_Callback(hObject, eventdata, handles)
% hObject    handle to pause (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
pause as text
%        str2double(get(hObject,'String')) returns
contents of pause as a double


% --- Executes during object creation, after setting
all properties.
```

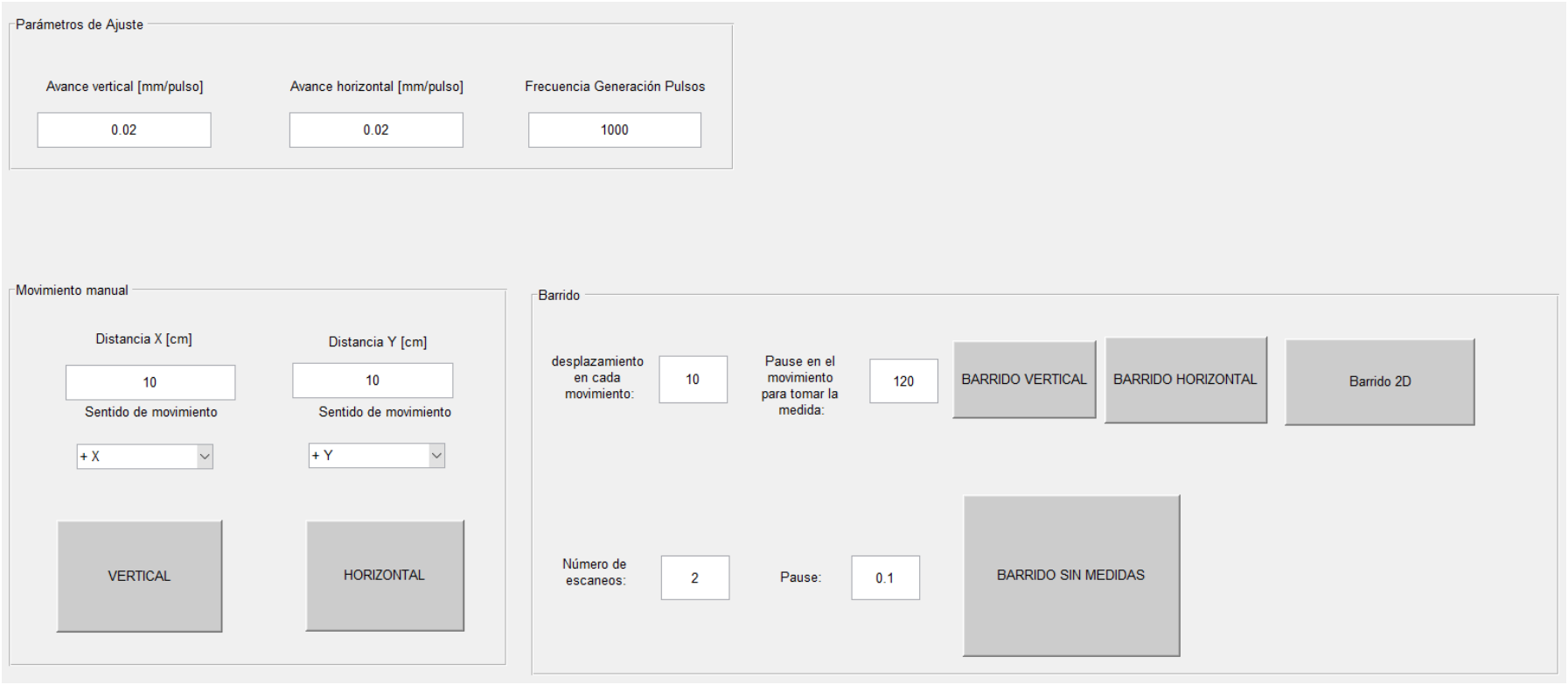
```
function pause_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pause (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function nscan_Callback(hObject, eventdata, handles)
% hObject    handle to nscan (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
nscan as text
%        str2double(get(hObject,'String')) returns
contents of nscan as a double


% --- Executes during object creation, after setting
all properties.
function nscan_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nscan (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

2. Función para mover los motores

```
function [] = mover_MX(handles,incrX,sentido)

digitalWrite(handles.ard1,handles.PIN_rele,0);
         set(handles.mover_x, 'BackgroundColor',[0 1
0])  %color verde cuando esta en movimiento
         pause(0.2);

%npasos=round(str2double(get(handles.incr,'string'))/s
tr2double(get(handles.pxp,'string'))*7.5);

npasos=round((incrX/str2double(get(handles.pxp,'string
')))*7.5);

digitalWrite(handles.ard1,handles.PIN_sentido_motor,se
ntido) %pin 10 del Arduino
         pause(0.5)
         sentido
         T=1/str2double(get(handles.freq,'string'));
         ipaso=0;
% Inicio del ciclo TTL
         tic
% Inicio del contador
         pinValue=1;
        while ipaso < npasos
                 while  toc<(T/2) %Abrimos bucle para
indicar que se ponga a 1 la señal TTL mientras el
contador se apague durante la mitad del período
                 end
                     pinValue=1-pinValue;

digitalWrite(handles.ard1,handles.PWM_motor,pinValue)
                     tic
                     ipaso=ipaso+1;
% Duración ciclo TTL
        end

function []=mover_MY(handles,incrY,sentido)

digitalWrite(handles.ard1,handles.PIN_rele,1);
         set(handles.mover_y, 'BackgroundColor',[0 1
0])
         pause(0.2);

npasos=round((incrY/str2double(get(handles.axp,'string
')))*7.5);
```

```
          sentido

digitalWrite(handles.ard1,handles.PIN_sentido_motor,se
ntido) %pin 10 del Arduino
          pause(0.5)
          T=1/str2double(get(handles.freq,'string'));
          ipaso=0;
% Inicio del ciclo TTL
          tic
% Inicio del contador
          pinValue=1;
         while ipaso<npasos
                   while  toc<(T/2)
% Abrimos bucle para indicar que se ponga a 1 la señal
TTL mientras el contador se apague durante la mitad
del período
                   end
                       pinValue=1-pinValue;
% Cuando el Pin de salida 3 reciba una señal '1' de
encendido, ordenamos que vuelva a '0'

digitalWrite(handles.ard1,handles.PWM_motor,pinValue)
                       tic
                       ipaso=ipaso+1 ;
% Duración ciclo TTL
         end
```

3. Ventana GUIDE



Desde esta ventana, un usuario ajeno a la realización del proyecto podrá mover las sondas ensambladas a los motores y tomar medidas para interés personal. Este ha sido uno de los factores por los que se ha decidido usar MATLAB para la programación del ARDUINO, ya que mediante esta interfaz tan intuitiva no hace falta saber de programación para tomar medidas y usar la instalación diseñada en este proyecto.

V. PRESUPUESTO

1. Presupuesto total

Presupuesto neto	23.688,23 €
Coste del proyecto con IVA	28.662,76 €

2. Desglose del presupuesto

a) Desarrollo y trabajo

Concepto	Cantidad	Coste unitario neto	Coste unitario con IVA	Precio total neto	Precio total con IVA
Documentación	100	20,00 €	24,20 €	2.000,00 €	2.420,00 €
Programación	200	20,00 €	24,20 €	4.000,00 €	4.840,00 €
Mediciones	100	20,00 €	24,20 €	2.000,00 €	2.420,00 €
Técnico de laboratorio	250	25,00 €	30,25 €	6.250,00 €	7.562,50 €
Total					14.822,50 €

b) Materiales

Concepto	Cantidad	Coste unitario neto	Coste unitario con IVA	Precio total neto	Precio total con IVA
Depósito INOX 8.000L	1	9.000,00 €	10.890,00 €	9.000,00 €	10.890,00 €
Filtro de arena	1	150,00 €	181,50 €	150,00 €	181,50 €
Arduino	1	12,00 €	14,52 €	12,00 €	14,52 €
Driver	1	212,00 €	256,52 €	212,00 €	256,52 €
Motores y guías	2		- €	- €	- €
Estructura de aluminio y tornillería	1	342,43 €	414,34 €	342,43 €	414,34 €
Relé	1	13,00 €	15,73 €	13,00 €	15,73 €
Fuente de alimentación	1	165,00 €	199,65 €	165,00 €	199,65 €
Total					11.972,26 €

c) Licencias e informática

Concepto	Cantidad	Coste unitario neto	Coste unitario con IVA	Precio total neto	Precio total con IVA
Ordenador	1	800,00 €	968,00 €	800,00 €	968,00 €
MATLAB	1	743,80 €	900,00 €	743,80 €	900,00 €
ANSYS student	1	- €	- €	- €	- €
Total					1.868,00 €

Por tanto, se puede afirmar que el presupuesto de la realización de este proyecto asciende a la expresada cantidad de veintiocho mil seiscientos sesenta y dos con setenta y seis euros.


UNIVERSITAT
JAUME·I