Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

Julio 2024

ESCUELA POLITÉCNICA DE MIERES
UNIVERSIDAD DE OVIEDO

MEMORIA

LEANDRO SANTOS GONZÁLEZ

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Universidad de Oviedo

EPM ESCUELA POLITÉCNICA DE MIERES

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Mención en Recursos Energéticos

Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

Autor: **Leandro Santos González**

Tutor: Javier Ribas Bueno

Julio, 2024

ÍNDICE

Contenido

Declaración de Originalidad del Trabajo Fin de Grado

D. Leandro Santos González, con DNI 76946674J estudiante del Grado en Ingeniería de los Recursos Mineros y Energéticos de la Escuela Politécnica de Mieres de la Universidad de Oviedo, declaro bajo mi responsabilidad que:

El Trabajo de Fin de Grado aquí presentado con título ‘Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías’, ha sido realizado bajo mi autoría, es original y que todas las fuentes utilizadas para su realización han sido debidamente citadas en el mismo.

Para que así conste, firmo la presente declaración.

En Mieres, a 2 de Julio de 2024.

Leandro Santos González
76946674J

OBJETIVO GENERAL

El objetivo general del proyecto es evaluar la posibilidad de detectar fugas en conducciones mediante un dispositivo de pequeñas dimensiones que registre y analice las aceleraciones al desplazarse dentro del fluido.
La idea es intentar caracterizar los perfiles de movimiento de un flujo y hacerlo con la suficiente precisión que permita captar las anomalías que puedan ser inducidas en este flujo en las zonas de fugas. Este dispositivo, además, podría servir para conocer el trazado de las tuberías enterradas con una precisión que facilitaría las posibles operaciones de reparación.
El objetivo primario del proyecto es el desarrollo de un prototipo para introducir en las conducciones con la idea de que sea arrastrado por el fluido moviéndose en el seno de este como una partícula más. Este dispositivo contendrá en su interior un acelerómetro y giroscopio (en adelante nos referiremos a este equipo como acelerómetro por sencillez) de 6 ejes para registrar las aceleraciones y los giros y una memoria para para guardar los datos. Con este equipo pretendemos, una vez depurados y tratados los datos, conocer las fuerzas a las que es sometido el fluido y ver si el equipo es capaz de detectar perturbaciones en las zonas en las que se pierda caudal a causa de las fugas.
Está idea surge de la necesidad, real, de un sistema que pueda detectar fugas cuando otros sistemas no pueden o bien porque las dimensiones de las redes de tubería no permiten su aplicación (video cámaras de inspección o detectores de gas trazador) o bien porque no se conoce el trazado exacto o puede existir elementos externos que interfieran con ellos (geófonos o correladores).
En este proyecto se desarrollará, por tanto, un equipo autónomo que sea capaz de detectar y registrar los movimientos que experimenta como resultado de ser arrastrado por un flujo de agua. Se desarrollará dicho equipo desde cero, seleccionando, en primera instancia, los componentes electrónicos para la toma de datos y registro de estos, así como la batería de alimentación para que funcione de forma autónoma. Se desarrollará el software que haga posible la comunicación entre estos componentes electrónicos y con el PC para su posterior tratamiento de datos. Y por último se realizará una envolvente ayudándonos de la tecnología de impresión 3D, para proteger el hardware y facilitar su desplazamiento por el interior de los conductos que se deseen inspeccionar.
Por ello este proyecto será multidisciplinar y de naturaleza empírica y requerirá de bastantes correcciones en fase de ejecución. Por todo ello se propone un tipo de proyecto basado en metodologías de prototipado y desarrollo ágiles.
En este proyecto no se pretende desarrollar un único dispositivo de medida, si no un prototipo que pueda servir como base para que, aprovechando las últimas tecnologías, se puedan realizar dispositivos que se adapten a las necesidades de cada caso particular.
Para el desarrollo de este proyecto se aplicará tanto la información referente a los sistemas de conducción de fluidos y los sistemas para detectar anomalías en ellos como la propia experiencia del autor en este campo.

NECESIDADES

> El agua está en el epicentro del desarrollo sostenible y es fundamental para el desarrollo socioeconómico, la energía, la producción de alimentos, los ecosistemas y para la supervivencia de los seres humanos. El agua también forma parte crucial de la adaptación al cambio climático, y es un decisivo vínculo entre la sociedad y el medioambiente.
>
> Organización Naciones Unidas

Las redes de conducción de fluidos están presentes en multitud de acciones de nuestro día a día sin que apenas nos demos cuenta de ello siempre y cuando funcionen correctamente. Sería impensable y un engorro no poder lavarnos cuando nos levantamos o usar el lavabo cuando lo necesitamos o beber cuando tenemos sed. Pero realmente van mucho más allá, estas redes están presentes en casi cualquier proceso industrial, no podríamos tener acero de calidad si no pudiéramos refrigerarlo a nuestro antojo en varias fases de su producción. No podríamos tener alimento ni bebida sin las redes de vapor para diversos tratamientos en las líneas de producción o limpieza y desinfección de las factorías. Las redes de fluidos son críticas en muchos de los procesos para generar y distribuir energía tanto eléctrica como combustibles. Las ciudades no serían posibles tal como las conocemos sin las redes de saneamiento.
Parece que las redes de conducción de agua son la solución a infinidad de problema debido a la capacidad de este para acumular calor (vector energético) y para disolver infinidad de compuestos químicos (limpieza) y la función más importante de todas, es fundamental para la hidratación pudiendo causar enfermedades incluso la muerte de cualquier ser vivo que no tenga a su disposición agua de calidad. Además, se trata de un elemento barato y que somos capaces de manejar con cierta facilidad pero que no es ilimitado. En la actualidad, tanto la expansión de la población como los crecientes hábitos de consumo hacen que la gestión del agua sea un reto al que enfrentarse.

Existen, en la actualidad, proyectos y aplicaciones para hacer más eficientes los procesos en los que se involucra el agua. Nosotros en este caso nos centraremos en los problemas que pueden surgir de su conducción. También se verán de manera somera las peculiaridades que podría tener la conducción de otros fluidos.

REDES DE AGUA POTABLE EN ESPAÑA

Para hacernos una idea de la importancia de las conducciones de agua basta con ver la situación en la que actualmente nos encontramos en España. Para poner en contexto la situación hídrica en la que nos encontramos tendremos que fijarnos en algunos puntos.

- Las cuencas hidrográficas:

En España debido a la variedad de climatologías y orografía existen grandes diferencias entre cuencas hidrográficas.

Para entender la problemática que puede existir en las diferentes cuencas deberemos tener en cuenta el estrés hídrico al que pueden estar sometidas. Este término hace referencia a la relación entre los recursos hídricos de una cuenca y la demanda de estos, a esta ratio se la conoce como coeficiente de explotación (Water Exploitation Index, WEI). Para determinar el tipo de cuenca en función de su estrés utilizaremos la clasificación de Raskin [1], esta clasificación establece 3 valores clave, hasta el 20% se considera ausencia de estrés, entre el 20 y el 40% se considera estrés y más allá del 40% estrés severo. Se considera cuando sobrepasamos el 100% que la cuenca está sobreexplotada. En el siguiente gráfico (Figura 1) podemos ver la situación de las cuencas hidrográficas en España (a excepción de Las Islas Canarias).

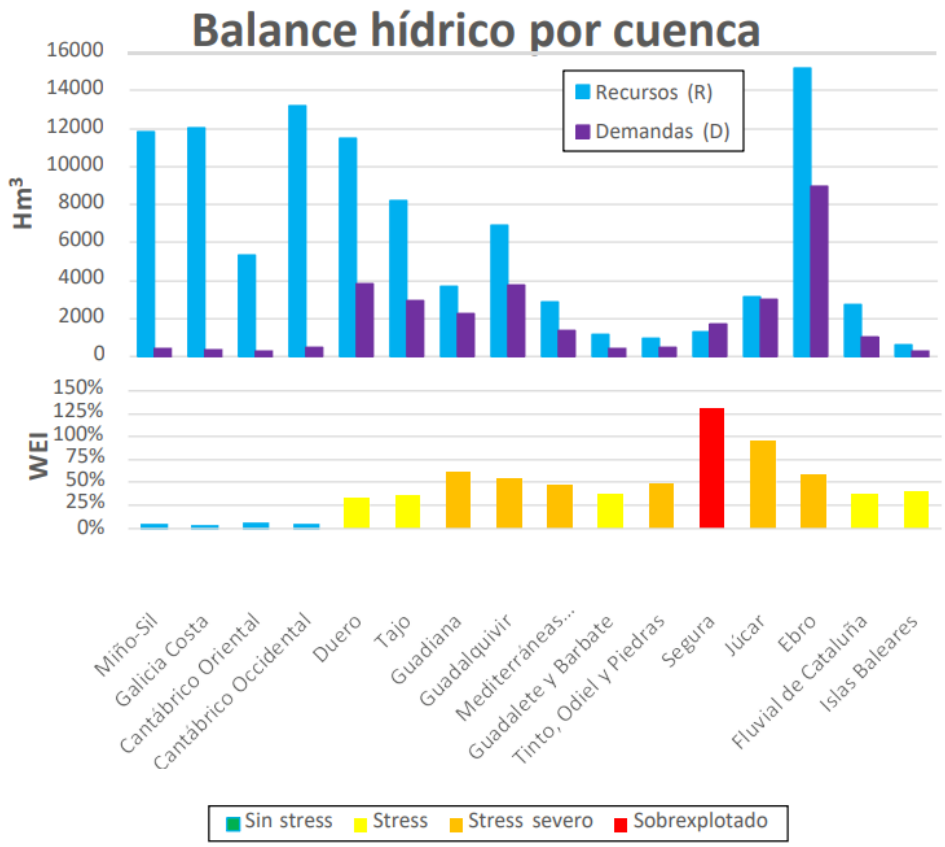


Figura 1.- Demandas, recursos y ratio recursos/demandas para DH de España (excepto Canarias) [2] (Velázquez et al., 2020)

- Las redes de distribución de agua potable en España:

Desde el año 1987 y con bienal la Asociación Española de Abastecimientos de Agua y Saneamiento (AEAS) y la Asociación Española de Empresas Gestoras de los Servicios de Agua a Poblaciones (AGA), con la colaboración de los miembros de

las Comisiones Técnicas de AEAS, elaboran de manera conjunta el "Estudio sobre el Suministro de agua potable y saneamiento en España" en el cuál recoge importantes indicadores relacionados con el ciclo integral del agua. Para el caso de estudio cabría indicar:

- El consumo de Agua No Registrada (ANR) en el año 2020 en España ascendía al 23,5%. Se define el ANR como la diferencia entre el agua suministrado al sistema y el agua registrado en los contadores. Se puede dividir el ANR en dos partes bien diferenciadas. El agua que se pierde en las redes de distribución y el que se consume en puntos no autorizados.
- Las redes, tanto de distribución como las de alcantarillado, están especialmente envejecidas en España. Según el XVII Estudio Nacional de Suministro de Agua Potable y Saneamiento en España, el 26% de las redes de distribución y el 44% de las de alcantarillado tienen más de 40 años, además la tasa de renovación de estás redes están muy por debajo del 2% ideal en ambos casos, siendo del 0,25% en el primer caso y del 0,42% en el segundo.

Si analizamos en conjunto estos datos, nos encontramos con un hecho alarmante. En la actualidad la mayoría de las cuencas hidrográficas en España sufren un estrés hídrico que va de moderado a severo, además, las tendencias climatológicas amenazan con empeorar estos datos. Por otra parte, nos encontramos con que las redes del ciclo del agua están envejecidas y se pierden grandes cantidades de agua agravando la delicada situación en la que nos encontramos. Añadido a esto, la tasa de renovación de redes de agua indica que esta estará cada vez más envejecida lo que hace que se tenga poco margen de maniobra para aliviar la sobreexplotación de las cuencas hidrográficas.
Por todo ello entendemos que puede ser importante el desarrollo de un método de detección de fugas en redes de fluidos que permita saber lo que sucede dentro de la red sin tener que vaciarla y en funcionamiento dándole la versatilidad necesaria para poder realizar medidas periódicas que puedan incluirse dentro de un programa preventivo.

ESTADO DEL ARTE MÉTODOS DE DETECCIÓN DE FUGAS EN TUBERÍAS

IMPORTANCIA DE LA DETECCIÓN DE FUGAS

Antes de estudiar los métodos que actualmente existen para la detección de fugas en redes de tubería se estudiará de forma somera las características más relevantes que pueden determinar cómo se genera una fuga y como esta podría ir evolucionando. Para ello tendremos en cuenta los siguientes puntos:

- Material del que está compuesto las tuberías. Tradicionalmente las tuberías podían ser metálicas (Acero al carbono, Inox, cobre y algunas aleaciones) normalmente para conducciones a presión, de fibrocemento para grandes conducciones y plásticos para desagües y conducciones a baja presión. Actualmente con el desarrollo de los plásticos existe una gran variedad de compuestos que nos permiten aplicaciones con presión y temperatura con polivinilos, multicapa, polipropileno y otros.
  Dependiendo de las características del material, las fugas podrían darse de forma súbita por solicitaciones mecánicas excesivas o podrían darse como resultado de

una reacción entre el material del que se compone la tubería y las características fisicoquímicas, permanentes o puntuales, del fluido que contiene.
Tendremos en cuenta también los posibles cambios de materiales que se puedan dar en juntas o elementos singulares de la instalación y las posibles reacciones entre estos. Por ejemplo, ciertos aceites pueden volverse agresivos con las juntas en presencia de humedad.

- ***Condiciones internas.*** Tendremos en cuenta las condiciones a que está expuesta una tubería debido a su funcionamiento, para ello debemos de tener en cuenta que estas condiciones pueden variar debido a funcionamientos de la instalación que no se contemplaran en la fase de diseño y que en algunas ocasiones debido a la falta de información tanto de la instalación como de su funcionamiento puedan estar dándose sin que seamos conscientes, por ejemplo, algún punto de la instalación podría estar sometido a presiones excesivas, golpes de arietes, cambios bruscos de temperatura sin que esto se refleje en el resto de la instalación o se haga de manera somera.
- ***Condiciones externas.*** La tubería podría estar sometida, por el ambiente en el que se encuentra, a vibraciones o tensiones mecánicas, agentes que la agentes como la luz solar, la humedad o cabios bruscos de temperatura u otras condiciones que aceleren su deterioro.

Estás condiciones adversas pueden potenciarse entre ellas incrementando el daño que puede causar en la instalación y además pueden darse de forma intermitente de forma que sea muy difícil su detección a tiempo. Esta es otra de las razones por la que un sistema que pueda detectar las fugas en estado temprano podría resultar muy ventajoso.

MÉTODOS ACTUALES PARA LA DETECCIÓN DE FUGAS

En la actualidad existen numerosos métodos para la detección de fugas en tuberías todos ellos con sus virtudes y defectos. Los métodos de detección de fugas son de observación directa y de inferencia [3]:

- Métodos directos: Un método directo para la detección de fugas es aquel en el que veríamos directamente la tubería y buscaríamos posible roturas o grietas. Podríamos ver las tuberías desde dentro con cámaras o directamente desde afuera en el caso de que las tuberías estuvieran accesibles o haciendo calicatas en el caso de estar ya seguros de donde se encuentra la avería.
  - Ventajas: En el momento que encontramos la avería estaremos ya en disposición de repararla.
  - Inconvenientes: Rara vez las tuberías son accesibles para encontrar las averías y en el caso de tener que desenterrarlas o realizar calicatas tendremos que asegurarnos previamente de donde se encuentra la avería. En el caso de las inspecciones interiores con cámaras tendremos que asegurarnos de que las tuberías no están atascadas ni contienen agua porque en este caso las cámaras no serán efectivas.
- Métodos por inferencia: En este caso se buscarán los efectos secundarios de una fuga de agua. Para este tipo de detección se buscan con geófonos o correladores

los sonidos y vibraciones que generaría una fuga de agua. En el caso de contar con agua a temperatura diferente a la del ambiente se podrá buscar anomalías térmicas con cámaras termográficas. En ocasiones, también se pueden utilizar gases 'trazadores' que se introducen en la red que se quiere comprobar y posteriormente se buscan fugas de dicho gas a lo largo de la red.

- Ventajas: No será necesario conocer el trazo exacto de las tuberías ni desenterrarlas.
- Desventajas: Estos métodos no determinan el punto exacto de las fugas. Pueden tener interferencias en el caso del sonido o las anomalías térmicas. En el caso de los trazadores no sirven en tuberías verticales y podrían no ser efectivas en tuberías que estén enterradas.

Debido a la importancia de las fugas en tuberías, y al desarrollo de la ciencia de datos, es común el desarrollo de nuevos softwares para el control de las redes de tuberías.

REDES DE DISTRIBUCIÓN DE FLUIDOS

Como ya hemos comentado las redes de conducción de fluidos pueden ser de diferente naturaleza y cada una de ellas tienes sus singularidades, pero realmente todas ellas tienen que cumplir unos criterios básicos.
Las redes deben de contener el fluido y no interferir con el químicamente, debe de soportar los cambios de presión, debido a los desacoples entre demanda y consumo, y deben de evitar las fugas del fluido al exterior o entradas desde el exterior si se trabaja en vacío. En ocasiones las solicitaciones mecánicas pueden ser exteriores, la tubería podría tener que soportar tensiones por el tráfico en el caso de estar enterradas en carreteras. Podría tener que soportar su propio peso entre durmientes en el caso de tuberías en altura. O, incluso, tener que soportar otros elementos, como pueden ser bombas de aspiración en pozos.
El hecho de que una tubería no sea capaz a contener la totalidad del fluido que contiene es un problema que depende de la naturaleza del fluido:

- Tuberías de aguas residuales, estas pueden contaminar los acuíferos o causar destrozos por corrosión a instalaciones circundantes.
- Redes de calor, el fluido en estas instalaciones tiene valor energético y podría estar tratado para evitar corrosión o depósitos de cal, entre otros, en intercambiadores, asientos de válvulas, etc.

- Otras: La fuga de los diferentes fluidos en redes de distribución conlleva un gasto, tanto de reposición del fluido en cuestión como de bombeo de este, posibles interferencias con el terreno u otros elementos que podemos encontrarnos en este.

ALCANCE DEL PROYECTO

El presente proyecto se centrará en el desarrollo de un equipo autónomo capaz de registrar aceleraciones y velocidades angulares a que será sometido como resultado de estar sumergido en un fluido que lo arrastre. Este equipo será capaz de guardar en una memoria interna los datos registrados para posteriormente volcarlos a un ordenador para su análisis. Se creará una envolvente para dicho equipo que haga a la vez de sostén y protección de la electrónica y favorezca el movimiento del equipo en el seno del fluido. Para todo ello en el presente proyecto se realizarán los siguientes trabajos:

- Elección y ensamblado de hardware:
  - Se seleccionarán los componentes electrónicos necesarios para llevar a cabo la toma de datos, así como su posterior guardado y volcado al PC.
  - Se ensamblarán dichos componentes realizando conexión soldadas entre sí que eviten fallos de comunicación en campo. Se alojarán en la envolvente de forma que se eviten daños por impactos o contactos accidentales.

- Desarrollo de la envolvente:
  - Se realizarán los diseños para la fabricación de la envolvente en programas de desarrollo 3D.
  - Se tratarán dichos diseños para su posterior impresión 3D con programas de comprobación de mallas y rebanado.
  - Se seleccionará el material más conveniente para la impresión 3D de la envolvente y se llevará a cabo esta impresión.

- Desarrollo del software:
  - Se llevará a cabo la programación del Hardware para que este nos permita registrar y guardar los datos que necesitamos en campo. Así como recuperarlos para su posterior análisis.
- Tratamiento básico de datos:
  - Se realizará tratamiento de los datos con programas de análisis, Excel o similar.

Cabe mencionar que, para la selección de equipos y materiales, así como la realización general del proyecto, se busca realizar un prototipado ágil. Por ello, se seleccionarán equipos y materiales que nos den ciertas facilidades en la fabricación del dispositivo.

DESARROLLO DEL PROYECTO

CONCEPTO GENERAL

Con el presente proyecto se intentará caracterizar el flujo de tal forma que podamos registrar las anomalías experimentadas por este al pasar por zonas de fugas. Para ello se analizarán dos tipos de mediciones:

- En régimen laminar: introduciremos un cuerpo que, siendo arrastrado por el fluido en régimen laminar, vaya registrando su posición relativa al espacio (acelerómetro) y a si mismo (giroscopio).
- Régimen de trabajo: En esta segunda opción, se caracterizará el perfil de movimiento del flujo en condiciones de uso estándar y se comparará con el perfil que se registraría en caso de una fuga para comprobar si se captan las anomalías inducidas por dicha fuga.

Para registrar los datos de movimiento del equipo en el seno del fluido se utilizarán sensores inerciales IMUs (Inertial Measurement Units) cuyos datos quedaran registrados en una memoria EEPROM, todo ello será controlado por un Arduino UNO que integra el microcontrolador ATmega328. El equipo irá alimentado con una pila tipo 6LR61 de 9V. Este equipo será dispuesto en el interior de un recipiente estanco que será introducido en las tuberías a analizar.
En un primer prototipo no se tendrá en cuenta la posición relativa del acelerómetro dentro del equipo. Se utilizará un sensor de 3 ejes tanto en el acelerómetro como en el giroscopio con la idea de registrar y caracterizar el flujo con la mayor precisión posible para intentar registrar, también, el mayor número posible de anomalías. Se hará hincapié en que el dispositivo no amplie pequeñas perturbaciones, en la fase de pruebas se estudiará la posibilidad de colocar diferentes apéndices que eviten el giro incontrolado del mismo y se colocarán ‘plomos’ que fuercen al dispositivo a moverse en el seno del fluido sin una flotación excesiva que lo intente expulsar, ni se hunda demasiado arrastrándose por el fondo.
La cápsula se realizará en plástico mediante tecnología de impresión 3D, se realizará el cierre de la misma mediante tornillería colocada en los labios de cierre, colocando juntas entre ambas semiesferas para asegurar la estanqueidad del dispositivo.

ENVOLVENTE

Para el desarrollo de la envolvente se realizará un encapsulado en plástico PETg mediante la técnica de impresión 3D. Se realizarán los bocetos correspondientes mediante programas de diseño 3D y se realizará el paso del archivo tipo Cad a archivo para la impresión 3d mediante los correspondientes programas en los cuales se comprobará el mallado y se definirán los parámetros de impresión.

IMPRESIÓN 3D

Antes de hablar del modelado de la envolvente intentaremos definir la tecnología de impresión 3D que vamos a utilizar para realizarlo. Debido al gran desarrollo que ha tenido esta tecnología en los últimos años y la variedad de técnicas que utiliza es difícil realizar una definición precisa de lo que actualmente significa impresión 3D. Según la ISO/ASTM 52900-2015, la manufactura aditiva es el término general para todas las tecnologías que se basan en una representación geométrica que crea objetos físicos por la adición sucesiva de material.

Podemos encontrar una gran variedad de máquinas que generan piezas en 3 dimensiones con multitud de técnicas, se habla de impresión 3D siempre que se añada material. Este

material puede añadirse fundido y que se adhiera al material ya depositado, o la cama de impresión si es la primera capa, o añadirlo, por ejemplo, en polvo y aplicarle la energía para su fusión de la forma precisa que genere el sólido que queremos. Los materiales pueden añadirse en forma de sólido continuo, en polvo o líquido. Se puede hacer que el material se fusione mediante calor, mediante químicos como el pegamento o forzar el sinterizado, por ejemplo.

AUTODESK FUSION 360

El programa en el que se desarrollarán los bocetos será el Autodesk Fusión 360. Este programa permite un desarrollo rápido y sencillo de piezas en 3D mediante extrusión de bocetos lo que añade mucha flexibilidad y rapidez a la hora de realizar cambios en el diseño.

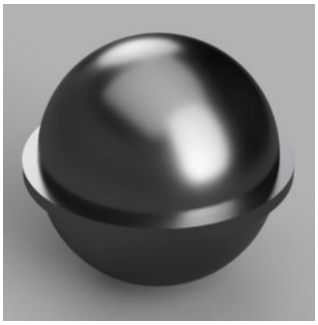


Figura 2 Envolvente Exterior diseñada en 3D

La envolvente se diseña en forma esférica (Figura 2) para favorecer el movimiento continuo a través de la tubería, se le añade un labio a cada semiesfera sobre los cuales se alojarán los cierres que nos permitirán realizar el cierre, cuando de introduzca en las canalizaciones de agua, y su posterior apertura para la descarga de datos. El diámetro de la envolvente vendrá dado por la placa de Arduino Uno que es, en este caso, el componente de mayor dimensión.

Figura 3 Detalle de la envolvente 1/2

En la Figura 3se puede ver una de las dos semiesferas que componen la envolvente. En este caso se observan los soportes sobre los que se alojará la placa de Arduino. Además de que en el diseño actual la placa de Arduino es el componente más grande y es, por tanto, el que nos indica las dimensiones mínimas que ha de tener la envolvente, tendremos presente en el desarrollo que necesitaremos conectar, mediante cable, la placa al ordenador y lo haremos si desmontar la placa de la envolvente, como podemos observar en la Figura 4.

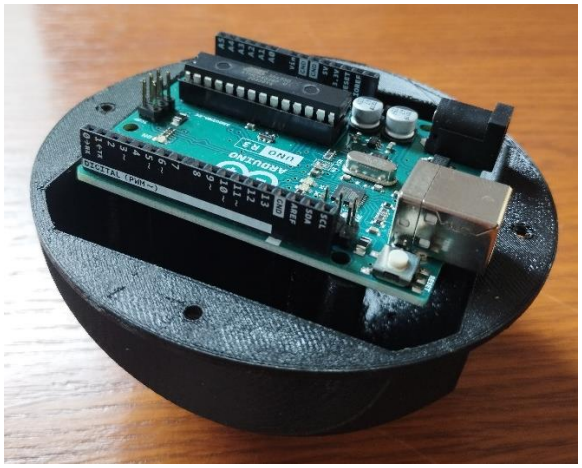


Figura 4.-Detalle del puerto de conexión

Figura 5 Detalle de la envolvente 2/2

En la Figura 5se ven los alojamientos del acelerómetro y la memoria en la otra parte de la envolvente. La fijación de los componentes a la envolvente se realizará utilizando tornillería aprovechando los agujeros que los componentes disponen para tal fin.

La unión de ambas semiesferas se llevará a cabo con tornillería sobre los labios y usando juntas planas, tipo Klingerit o similar, para evitar que algún defecto en las caras internas de contacto entre piezas deje pasar agua al interior.

MESHLAB

Aunque no es totalmente necesario resulta interesante utilizar un programa de procesamiento de mallas para asegurar que no existen defectos en el mallado que se manifiesten como defectos en la pieza una vez impresa. En nuestro caso usaremos el MeshLab, que es un programa gratuito y de código abierto que originalmente se desarrolló como trabajo en un curso de la universidad de Pisa a finales del año 2005. Luego este programa fue ampliado como proyecto en el que participaron tanto estudiantes como el Laboratorio de Computación Visual del ISTI-CNR. Finalmente, el MeshLab es un programa de procesamiento de mallas con gran capacidad de filtrado y reparación de mallas, pero que requiere de cierta experiencia. En nuestro caso solamente utilizaremos este programa para simplificar la malla eliminando vértices y caras

duplicadas, así como triángulos que se entrecruzan y asegurar que la malla no tiene agujeros y comprobar la estanqueidad evitando posibles problemas en el producto final de la impresión.

Para el uso de este programa generaremos archivos STL con cada una de las partes de nuestro prototipo a partir del modelo 3D en el Fusion 360. Dicho archivo lo cargaremos en el MeshLab, simplemente arrastrando al área de trabajo la carpeta donde se haya guardado. Ahora ya está listo para filtrar y reparar:

- En primer lugar, filtraremos los triángulos que se entrecrucen como se observa en la Figura 6.

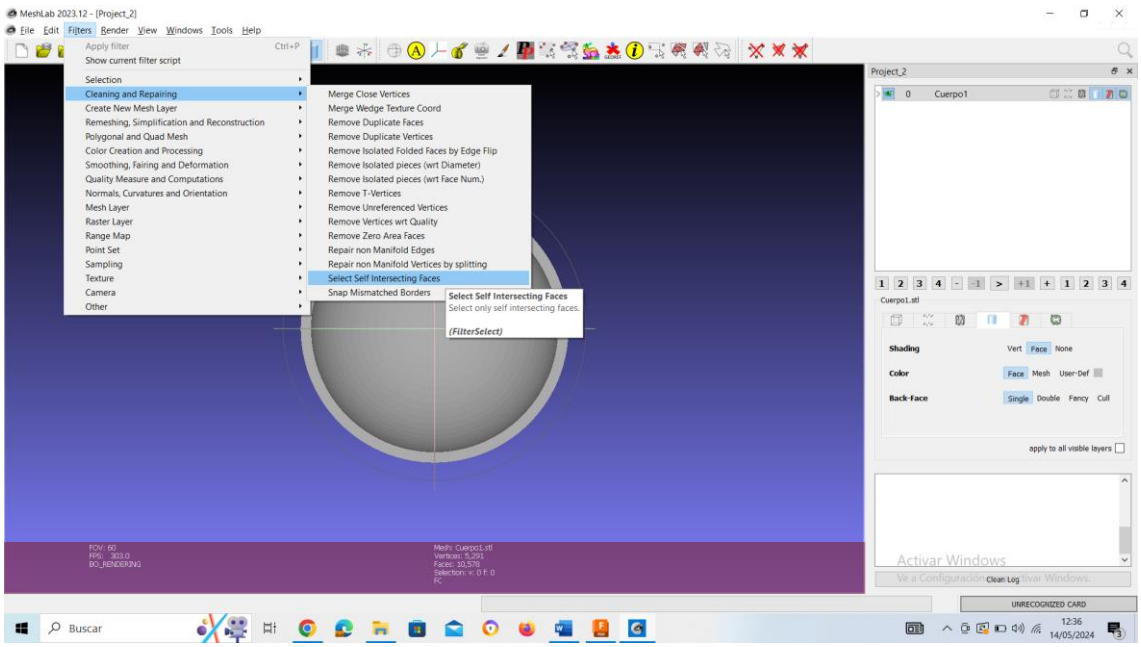


Figura 6.-Filtrado de triángulos duplicados

- Eliminaremos caras y vértices duplicados Figura 7.

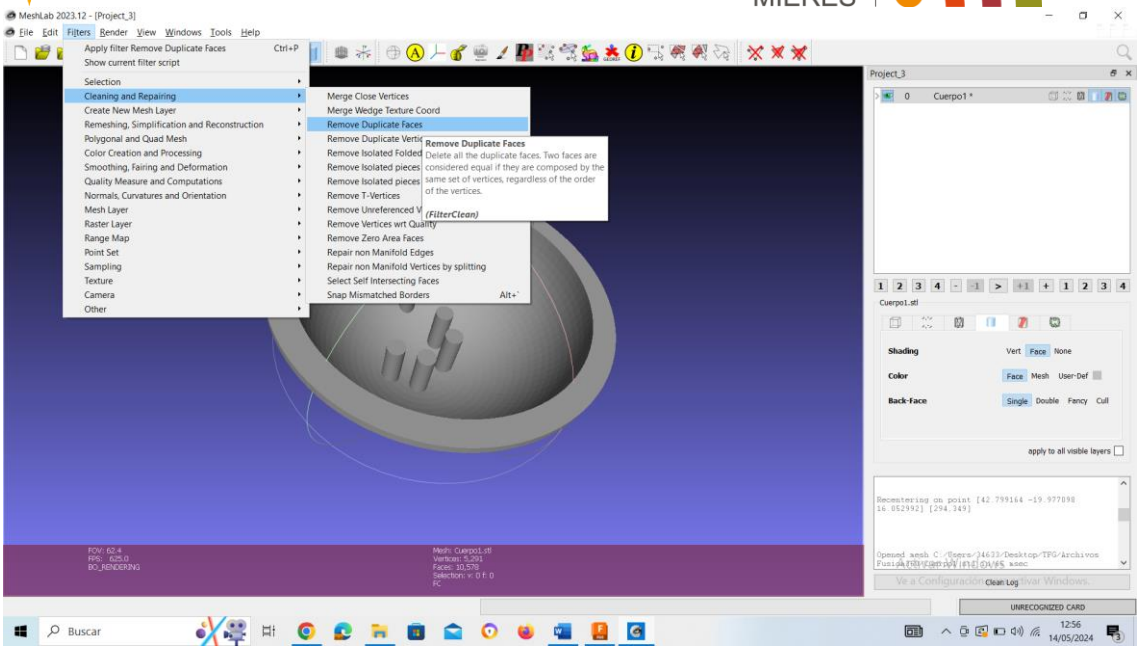


Figura 7.-Eliminación de caras y vértices duplicados

- Por último, cerraremos agujeros en el mallado, Figura 8.

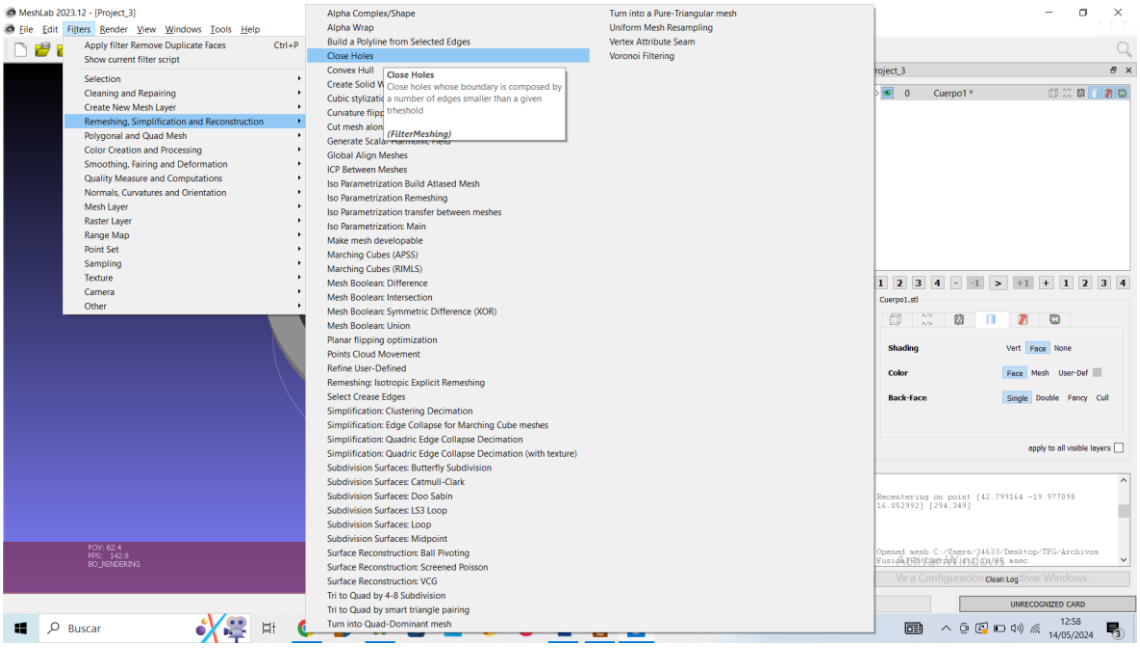


Figura 8.-Cerrado de agujeros en el mallado

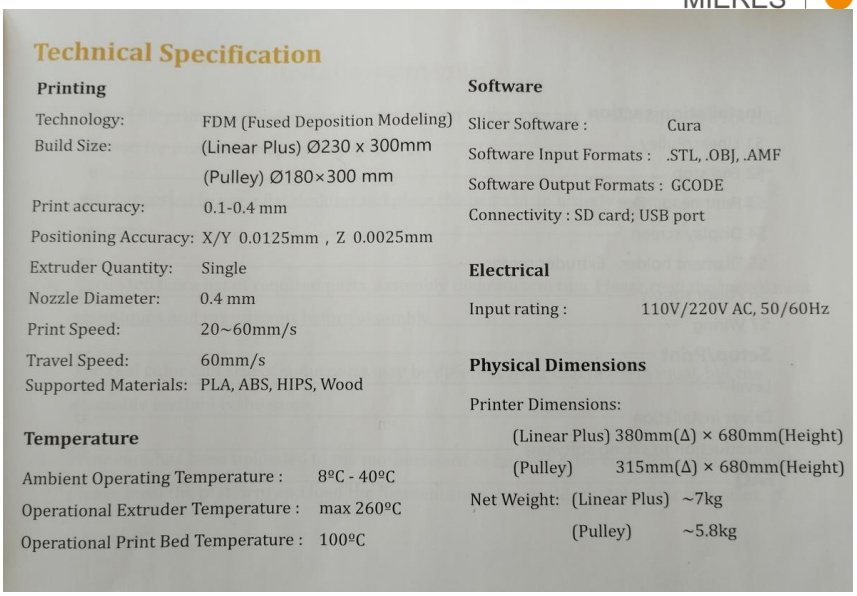
ULTIMAKER CURA SOFTWARE

Para preparar el archivo 3D para la impresión se ha de utilizar un programa de rebanado o Slicing, existen algunos programas de código abierto como son el Skeinforce, Slice3r

y el Ultimaker Cura. Utilizaremos este último que además de ser de código abierto para aplicaciones no profesionales tiene un interfaz de fácil uso. Se les llama programas de rebanado porque estos programas generan las instrucciones para la impresión por capas del modelo 3D. Es este rebanado el que define completamente la calidad de impresión.
La preparación del archivo en 3D para la impresión requiere el análisis de los siguientes parámetros:

1. Ajustes de la impresora:
    - Tipo de impresora y firmware de esta.
    - Tamaño del área de impresión y altura máxima.
    - Número de extrusores y diámetro de las boquillas.
2. Ajuste de los filamentos:
    - Diámetro del filamento.
    - Factor de proporcionalidad, compensación de la expansión del plástico.
    - Temperatura de la cama y de la extrusora.
    - Ventilador de enfriamiento.
3. Ajustes de capas:
    - Altura de la capa, puede ser diferente para la primera capa.
    - Número de conchas. Perímetro o grosor de las capas externas.
    - Porcentaje de relleno.
    - Patrón de relleno.
    - Velocidad de impresión.

En este caso vamos a utilizar la impresora Kossel Linear Plus de la marca ANYCUBIC, podemos ver en el propio manual (Figura 9) de la impresora que utiliza la tecnología FDM (Fused Deposition Modeling), tiene unas dimensiones de construcción máximas de ⌀ 230 x 300 mm, este dato nos indica que la cama de impresión es redonda, la impresora es de tipo delta. Dentro de los datos que indica el manual es de especial interés el diámetro del Nozzle (boquilla), es un parámetro que habremos de tener en cuenta a la hora de configurar la altura de las capas de impresión, así como la anchura de los bordes.



Technical Specification

Printing

Technology:	FDM (Fused Deposition Modeling)
Build Size:	(Linear Plus) Ø230 x 300mm
	(Pulley) Ø180×300 mm
Print accuracy:	0.1-0.4 mm
Positioning Accuracy:	X/Y 0.0125mm , Z 0.0025mm
Extruder Quantity:	Single
Nozzle Diameter:	0.4 mm
Print Speed:	20~60mm/s
Travel Speed:	60mm/s
Supported Materials:	PLA, ABS, HIPS, Wood

Temperature

Ambient Operating Temperature :	8ºC - 40ºC
Operational Extruder Temperature :	max 260ºC
Operational Print Bed Temperature :	100ºC

Software

Slicer Software :	Cura
Software Input Formats :	.STL, .OBJ, .AMF
Software Output Formats :	GCODE
Connectivity :	SD card; USB port

Electrical

Input rating :	110V/220V AC, 50/60Hz

Physical Dimensions

Printer Dimensions:

(Linear Plus) 380mm(Δ) × 680mm(Height)

(Pulley) 315mm(Δ) × 680mm(Height)

Net Weight:	(Linear Plus)	~7kg
	(Pulley)	~5.8kg

Figura 9.-Especificaciones Técnicas Impresora 3D

Para la configuración de la impresora en el Ultimaker Cura podríamos introducir los parámetros de esta a mano, lo cual es muy útil cuando modificamos las impresoras, cabe recordar que son equipos muy utilizados para el DIY (Do It Yourself), por lo que no es de extrañar que nos encontremos con máquinas modificadas o, incluso, creadas desde cero. Por el momento, usaremos el equipo tal cuál viene de fábrica por lo que buscaremos y seleccionaremos la impresora sin más.

Después de configurar la impresora configuraremos el material que vamos a utilizar para la impresión. Para las impresoras de modelado por material fundido los filamentos más típicos son el PLA, el ABS y desde hace algún tiempo el PETg. Vamos a ver, de forma superficial, las características de estos tres materiales.

- PLA (PoliÁcido Láctico). Es un polímero termoplástico biodegradable, es un producto derivado del almidón de maíz, por lo que no resulta tóxico. Es un material de fácil impresión ya que no requiere temperaturas muy altas, en torno a los 200 ˚C. Y tiene una baja contracción térmica por lo que no requiere que la cama de impresión esté caliente. Como contraprestaciones podríamos señalar que no es un material muy duro y es químicamente bastante inestable.

- ABS (Acrilonitrilo Butadieno Estireno). Es un material que todos conocemos por ser el plástico que la marca LEGO usa para sus ladrillos. Es un derivado del petróleo por lo que emite gases tóxicos durante la impresión, tiene muy buenas propiedades mecánicas lo que lo hace muy recomendable para aplicaciones finales, carcasas, piezas de coches y otros. Como contraprestaciones tendremos en cuenta que es un material difícil de manejar a la hora de imprimir, requiere altas temperaturas, en torno a 250 ˚C y tiene una gran contracción térmica por lo que requiere de cama caliente y debemos de tener cuidado con su enfriamiento durante la impresión.

- PETg (Polietileno Teraflato de Glicol) El PET es un plástico de uso común ya que es el material a partir del cual se fabrican las botellas de plástico desechables. En este caso se le añade glicol para aumentar su resistencia. Se suele decir que el PETg, tiene las facilidades de impresión del PLA y las propiedades mecánicas parecidas al ABS. Por lo que utilizaremos este material en nuestro proyecto.

El Cura Software ya tiene una base de materiales precargados con parámetros predeterminados para cada caso como podemos ver en la siguiente imagen (Figura 10). En todo caso estos programas nos permiten la modificación de todos los parámetros, pero por el momento, trabajaremos con los parámetros predeterminados.

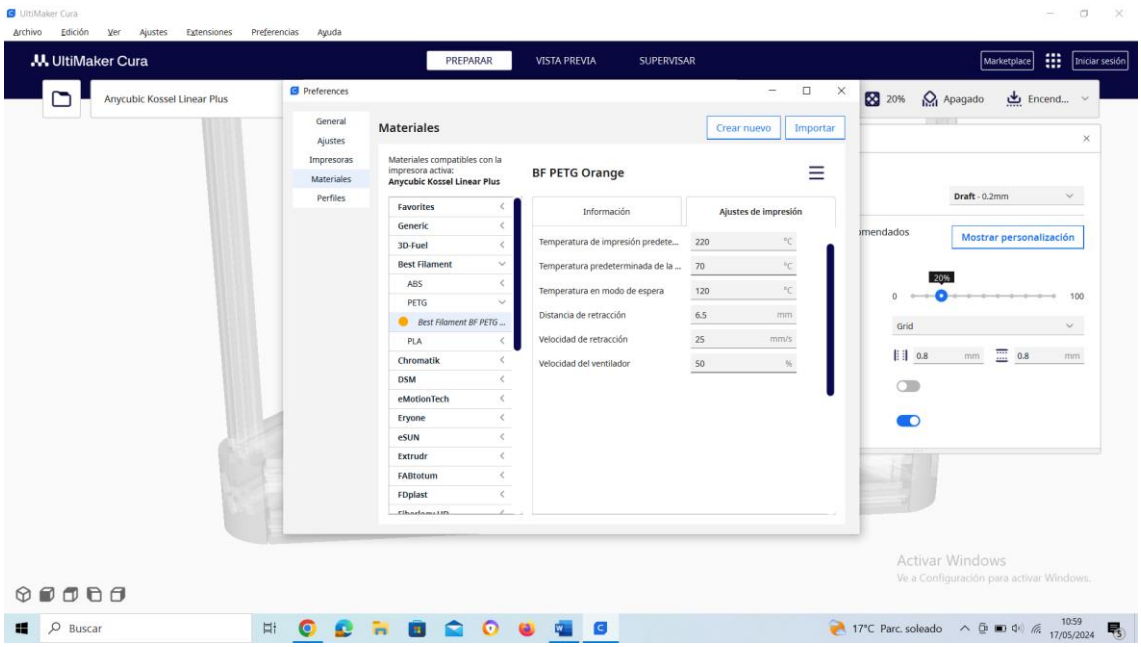


Figura 10.- Material de impresión Ultimaker Cura

Para introducir nuestro modelo, previamente tratado en el procesador de mallas, basta con arrastrar el archivo al área de impresión, como ya hicimos con el MeshLab, y el propio software coloca el prototipo de la mejor manera calculando los recursos invertidos, tiempo de impresión y el material requerido. En las siguientes imágenes podemos ver la diferencia entre la ubicación seleccionada por el programa (Figura 11)y una realizada al azar (Figura 12).

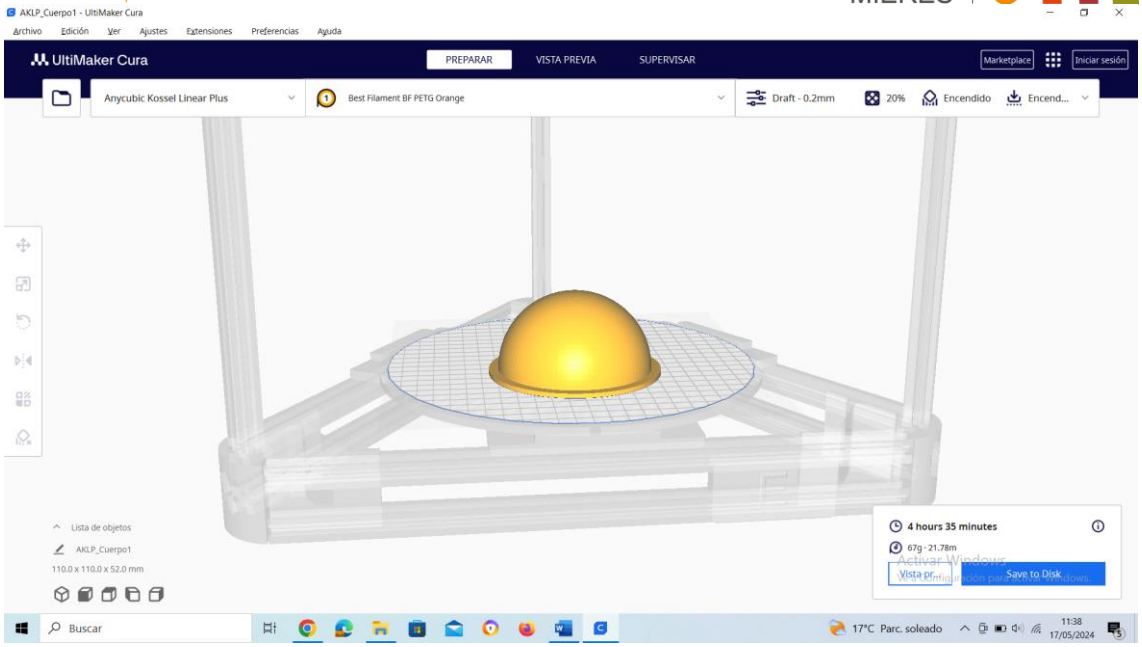


Figura 11.- Ubicación optimizada por software.

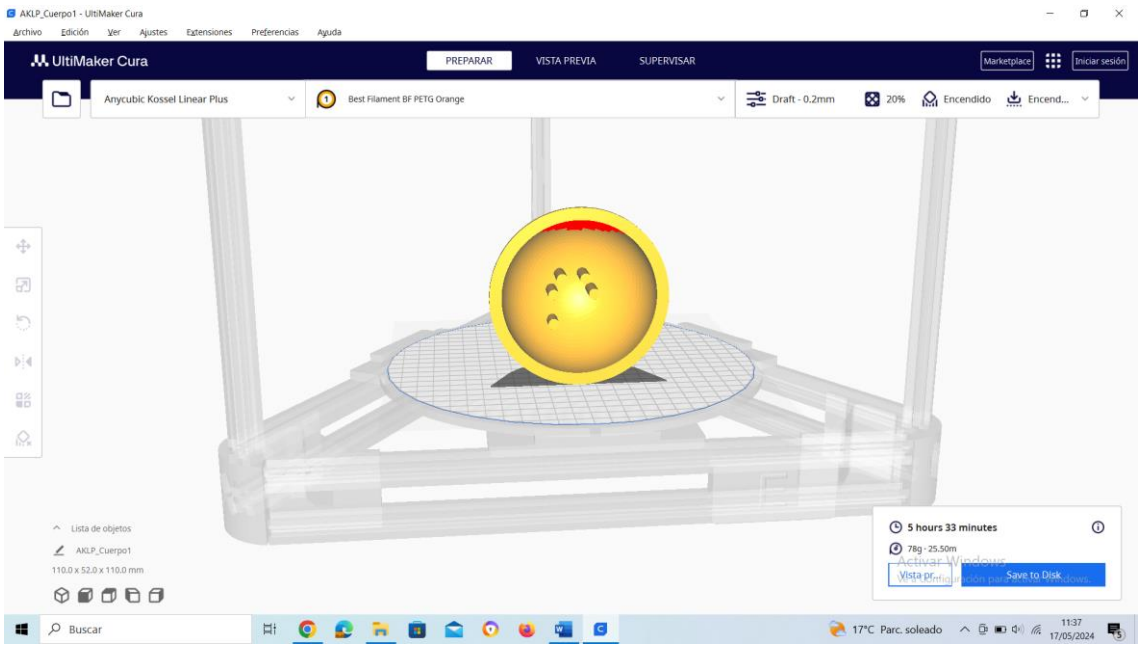


Figura 12.- Ubicación realizada al azar.

El programa nos informa del tiempo de impresión, es estimado y suele quedarse un poco corto, y vemos que la situación escogida por el propio software da lugar a una impresión 1 hora más rápida, en nuestro caso no tendremos en cuenta los tiempos de impresión para este proyecto, pero no es difícil de imaginar la importancia que tendría en una producción en cadena. También nos informa del material que requerirá para realizar el modelo, aunque podría ser intuitivo pensar que crear una pieza requiere el mismo material independientemente de la posición que tenga en el espacio, en las impresoras de deposición de material fundido no es así ya que cada nueva capa requiere ser depositada sobre una capa anterior, o sobre la cama en la primera capa por lo que

tendrá que realizar apoyos para las partes que no tengan un apoyo previo. Aunque no vamos a entrar a fondo sobre este tema, es importante para la impresión 3D el estudio de los voladizos y el material de soporte que requieren. En general, aunque podríamos hacer pruebas para cada impresora y material, los voladizos que tienen más de 45 grados de inclinación y los puentes que tienen más de 5 mm entre apoyos requieren de soportes adicionales, esto no es importante para nuestro modelo, pero podría serlo si se necesitasen acabados muy finos. En nuestro caso simplemente retiraremos el material de soporte una vez terminada la impresión. Una vez determinada la posición del modelo respecto de la impresora y realizada la segmentación podemos realizar una vista preliminar para ver las capas de impresión (Figura 13) y ver, también como realizará los soportes. Es interesante para ver que no exista algo que se nos pase y verlo después durante la impresión teniendo que desechar el material y empezar la impresión de cero.

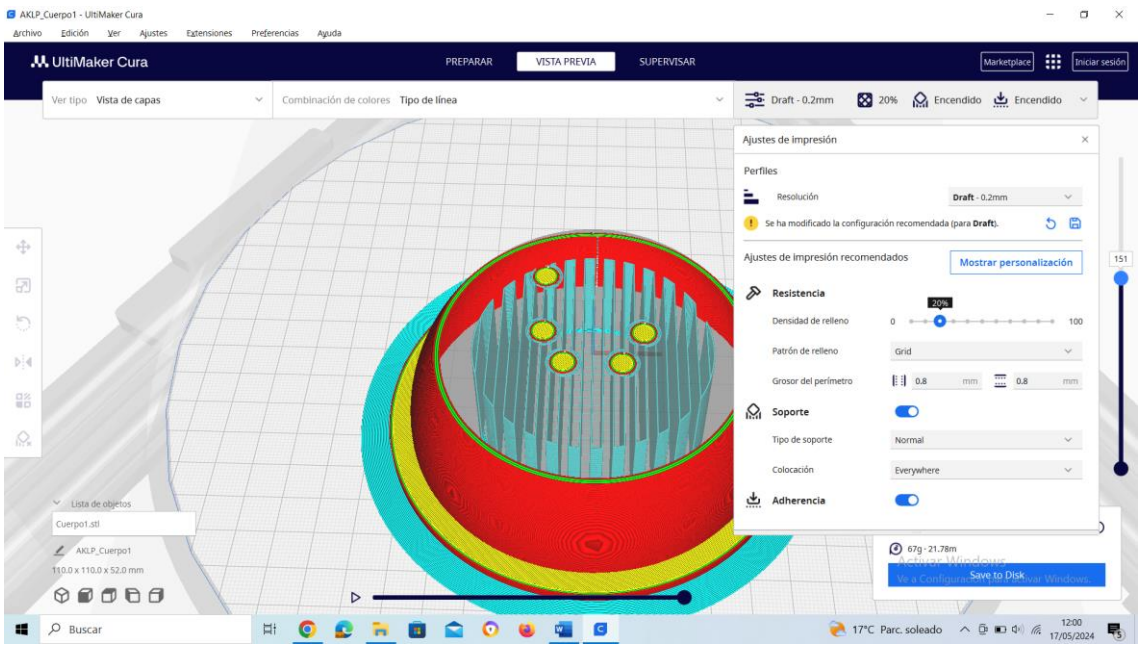


Figura 13.- Vista preliminar de la impresión.

Una vez determinados los parámetros de velocidad y temperatura de la impresión, dejaremos los que por defecto selecciona el software en función del material de impresión, pasaremos a definir la suportación, en este caso simplemente seleccionamos la opción para que se añadan sin cambiar ninguna opción porque no es importante para nuestro modelo. Seleccionaremos los parámetros de calidad y relleno de la impresión para darle un acabado robusto y de calidad para las pruebas.

- Altura de capa, de 0,2 mm es la mitad de la medida de la boquilla del extrusor.
- Grosor de la pared y grosor superior e inferior. Escogemos 1,2 mm para estos parámetros. Teniendo en cuenta que el extrusor es de 0,4 mm este grosor hará que las capas exteriores tengan un grosor de 3 capas antes de empezar con el mallado de relleno.
- Relleno. Aplicaremos un 25 % a la densidad de relleno, los valores típicos están en torno al 10 – 20 %, lo aumentamos un poco para darle mayor rigidez ya que una rotura durante una prueba haría que se estropease todo el material e incluso

que pudiera quedar material dentro de una tubería. Para el patrón de relleno utilizaremos la rejilla, valor que viene por defecto.

Con estos parámetros seleccionados ya solo queda guardar el archivo en extensión *.STL y enviarlo a la impresora. Podríamos conectar el ordenador directamente, pero es más recomendable utilizar una memoria externa (microSD) y cargar el archivo en la impresora desde ahí.

IMPRESIÓN 3D

Para este último paso solamente tendremos que enviarle la información a la impresora en archivo de extensión *. GCODE y comenzar la impresión. Es recomendable supervisar los primeros pasos que realiza la impresora para ver que todo está bien. Cabe recordar que la impresión nos llevará unas cuantas horas y cualquier error al principio de la impresión puede hacer que nos encontremos con una maraña de plástico al cabo de este tiempo de impresión. Los puntos más importantes para tener en cuenta, y que podremos observas al principio de la impresión, son:

- La temperatura de la boquilla y de la cama. La propia impresora nos indicará en su pantalla las temperaturas de ambos (Figura 14). Si las temperaturas no fueran las correctas podríamos, en el caso de la boquilla, quemar el material de impresión (por exceso) u obstruir la propia boquilla (por defecto). En el caso de que sea la cama la que tiene una temperatura inadecuada podría dificultar el pegado del material sobre esta, dejando una base irregular.

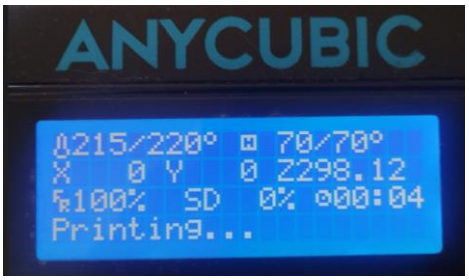


Figura 14.-Pantalla Impresora 3D.

- Inyección de plástico: En el caso de que la inyección del plástico no sea adecuada veremos que el extrusor patina sobre el plástico, podría deberse a que la boquilla está obstruida.
- Movimiento del cabezal: Al empezar la impresión el cabezal de la impresora realiza un movimiento hacia el lado más alejado de la cama con el fin de buscar los ceros de máquina y desde ahí baja hasta la cama de impresión. Este es un momento especialmente delicado ya que en caso de que las referencias tengan un error, bien porque se haya movido algún final de carrera, bien porque el archivo este dañado y envíe mal las coordenadas a los motores, u otras razones, la boquilla podría estrellarse contra la cama provocando daños en la propia impresora. En el caso contrario, en el que el cabezal no llegue a bajar lo

suficiente, el plástico no se depositará sobre la cama correctamente y la impresión será defectuosa.

Todas estas operaciones se pueden observar en un par de minutos, por lo que nos merecerá la pena realizar estas inspecciones previas. Una vez comenzada la impresión solo nos quedará esperar. Al acabar la impresión tanto la cama como la boquilla podrían estar calientes por lo que esperaremos un tiempo prudencial para retirar la pieza. Si la impresión ha sido correcta la pieza podría quedarse pegada a la cama de impresión por lo que la retiraremos utilizando una espátula.

CÁLCULOS

Como comentamos con anterioridad, la idea es introducir el dispositivo en una tubería y que este, impulsado por la corriente de fluido, recorra la tubería tomando datos de aceleraciones que nos den información de las posibles anomalías en dicho flujo y por tanto de la tubería que lo canaliza. Para ello necesitamos que el dispositivo se mueva en el seno del fluido como una partícula más de este. Si nuestro dispositivo se arrastrase por el fondo por exceso de peso o por la parte más alta por exceso de flotación iría detectando los golpes y el frenado de su propio arrastre por lo que captaríamos una cantidad elevada de señal de ruido que dificultaría el análisis de los datos recogidos. Por ello, realizaremos cálculos para una correcta flotación, medidos en bácula de precisión, y el volumen de la esfera, con el valor el propio programa de desarrollo nos indica.

Cálculo preliminar de flotación	
Volumen	
Esfera	509,87 cm^3
Peso	
Esfera	92 g
Arduino UNO	27 g
MPU-6050	2 g
DS3231	8 g
Batería SD-9VCV-TYPEC-800	23 g
Total	152 g

Tabla 1.- Pesos medidos.

Aplicaremos el principio de Arquímedes de forma básica para el cálculo de la flotación y según la ecuación:

$$E = \rho \, x \, V \, x \, g$$

Que despejando nos quedaría:

$$E = 1 \, \frac{g}{cm^3} \, x \, 509.87 \, cm^3 \, x \, 9.81 \, \frac{m}{s^2} = 5.001{,}82 \, gf$$

Por lo que necesitaríamos un lastre aproximado de

$$Masa_{lastre} = \frac{5.001{,}82\ gf}{9{,}81\ \frac{m}{s^2}} - 152\ g = 357{,}82\ g$$

Según los cálculos el lastre que necesitaríamos será de unos 350 g, como sabemos que se pueden haber producido errores de medida y redondeo, se prepararan lastres en forma de arandelas que se puedan añadir a los espárragos de cierre de la envolvente de forma que se facilite el ajuste de pesos en campo.

HARDWARE

SENSORES INERCIALES

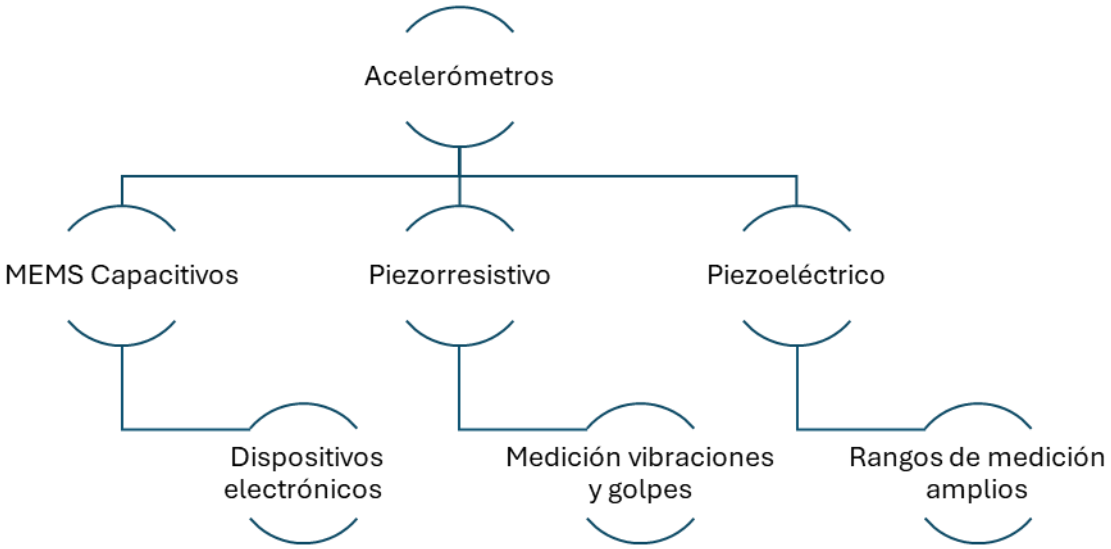
Los sensores inerciales son dispositivos capaces de medir aceleración lineal (acelerómetros) y velocidad angular (giroscopios). Lo que consiguen estos dispositivos es determinar los cambios de posición en el espacio y generar una señal eléctrica proporcional a esos movimientos. Transformar una magnitud física en una señal eléctrica proporcional es la labor típica de los transductores, por lo que podríamos definir a los sensores inerciales como transductores de fuerzas de inercia.

ACELEROMETROS

El uso de estos dispositivos inició su auge a principios de los años 20´s principalmente en aplicaciones de navegación, dirección y control en aeronaves, barcos y dispositivos guiados automáticamente como misiles por ejemplo [4].
La necesidad de producir cada vez máquinas más precisas hace que en el campo de la electrónica se hayan desarrollado equipos también más precisos para la medición de todos los parámetros que puedan ayudar o influir en la precisión de mediciones y control. Dentro de este desarrollo y aplicación de los sistemas de medición y control cabría destacar los sensores Micro-Electro-Mecánicos, en adelante MEMS por sus siglas en inglés. Estos sensores están presentes hoy en día en infinidad de máquinas y dispositivos que utilizamos como son los teléfonos móviles, los ordenadores, los vehículos y muchos otros.
El primer sensor inercial micro-maquinado electromecánico o MEMS (micro-electro-mechanical-systems) fue presentado en 1979 por la universidad de Stanford [5]
Dependiendo del tipo del mecanismo que se utilice para la medida de aceleración o giro podemos encontrar diversos tipos de acelerómetros o giroscopios.
Aunque existen en el mercado gran variedad de acelerómetros vamos a analizar de manera somera los más habituales:
Los acelerómetros capacitivos de tecnología MEMS, son los más baratos y pequeños. Están fabricados a partir de una placa con un resorte que se mueve en función de las aceleraciones a las que se la someta. Estos dispositivos funcionan como condensadores variables de placas generando una impedancia al paso de la corriente diferente en función de la distancia entre las placas. Para ello la placa resorte estará formada por varias placas de regulación que estarán colocadas entre placas fijas en relación con el propio dispositivo. Son utilizados en dispositivos electrónicos debido a sus prestaciones y su relación calidad precio.

Los acelerómetros de tipo piezorresistivo están formados por materiales que cambian su resistencia cuando se someten a esfuerzos mecánicos. En este medidor se une una carga, que será la que se verá sometida a las aceleraciones, a un material cuya deformación se traducirá en un cambio en las propiedades de la corriente eléctrica que lo atraviesa. Son comunes en las aplicaciones de medición de vibraciones y golpes.

Los acelerómetros de tipo piezoeléctrico son similares a los de tipo piezorresistivos con la peculiaridad de que, en este caso, cuando se somete el material piezoeléctrico a una tensión mecánica es el propio material el que genera una diferencia de potencial entre sus extremos por lo que no necesita ser conectado a una fuente de energía externa. Tienen rangos de medición muy amplios con buena precisión.

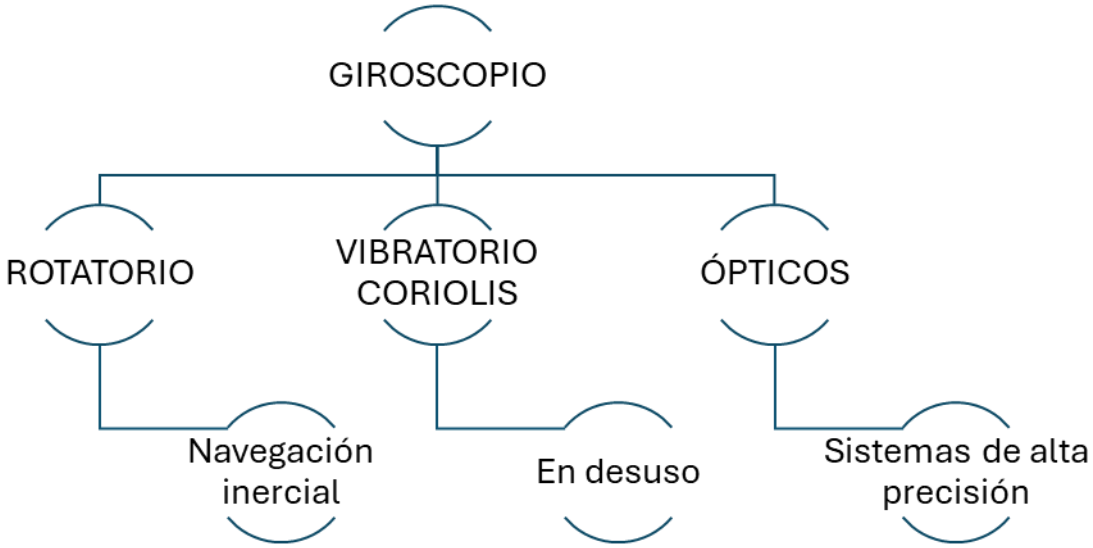


Cabría indicar que existen una gran variedad de estos dispositivos y que puede haber diferentes clasificaciones de ellos dependiendo de la capacidad para medir, por ejemplo, cargas estáticas y dinámicas. Estos dispositivos son comúnmente empleados en industria para la medición de vibraciones en ambientes agresivos por lo que existen equipos muy robustos y complejos para mediciones muy exactas. Pero no es objeto del presente documento entrar tan a fondo en estos equipos y por el momento se escogerá un dispositivo sencillo y versátil que permita una implementación ágil.

GIROSCOPIO

Los giroscopios, por su parte, sirven para medir o mantener la posición y la velocidad angular del sistema en el que se haya montado. Estos aparatos se categorizan de acuerdo a su principio físico de operación, entre ellos se encuentran los ópticos, vibratorios y mecánicos [6]:
Al igual que en el caso de los acelerómetros se trata de medir la variación que una fuerza ejercía sobre un objeto. En este caso todos los sistemas tratan de medir la

variación que un giro ejerce sobre un cuerpo, en el caso de los mecánicos y vibratorios, o sobre una partícula (fotón) en el caso del giroscopio óptico.
Más concretamente en el caso de los giroscopios rotatorios el efecto del giro se detecta en el efecto de este sobre un disco de inercia. En el caso del giroscopio de vibración es el efecto de Coriolis sobre una estructura vibrante Para el giroscopio óptico se envían 2 haces de luz en sentido opuesto, anotando los tiempos que tardan en llegar a su receptor podremos saber si están sometidos a un giro y la magnitud de este, ya que el haz que viaja a favor de este giro tardará menos en llegar, al contrario que el haz que se mueve en sentido contrario. Con el cálculo apropiado podremos saber la velocidad angular a la que está sometido dicho giroscopio. Aunque está es una clasificación de los giroscopios no todos ellos son equipos de medida, como es el caso del giroscopio mecánico que se utiliza sobre todo en la navegación inercial.



Los giroscopios también utilizan la tecnología MEMS para poder realizar medidas de velocidad angular en equipos electrónicos de pequeñas dimensiones como teléfonos móviles, cámaras y otros. Será este el tipo de equipo que utilizaremos en el presente proyecto, sin entrar en detalles de cuál es la tecnología final de medición de giro.

MPU-6050

Para el presente proyecto se selecciona el MPU-6050, que incorpora en un mismo chip un acelerómetro de 3 ejes junto con un giroscopio también de 3 ejes, con lo que tendremos datos de aceleración y velocidad angular en las tres dimensiones. Este chip irá montado sobre una placa GY-521, que además del chip incorpora componentes para la conexión eléctrica, regulador de tensión, y la transferencia de datos, I2C.

Los datos de este módulo son [7]:

- Rapidez de respuesta: La velocidad o rapidez de respuesta del módulo GY-521 puede variar dependiendo de la programación que se realice de los parámetros de este pero el limitante en la transmisión de datos y por tanto la velocidad de

toma de datos de la que disponemos estará marcada como límite superior por el protocolo de comunicación de datos serie que utilizaremos. En este caso el I2C a 400 kHz (400000 muestras por segundo).

- Rango de acción: Realiza mediciones de aceleración y velocidad de giro en los 3 ejes. Ambos medidores tienen diferentes escalas de medición programables.

Giroscopio		
Rango de escala completa	Condición FS_SEL = 0 FS_SEL = 1 FS_SEL = 2 FS_SEL = 3	 ±250 °/s ±500°/s ±1000°/s ±2000°/s
Factor de sensibilidad de escala	Condición FS_SEL = 0 FS_SEL = 1 FS_SEL = 2 FS_SEL = 3	 131 LSB(°/s) 65.5 LSB(°/s) 32.8 LSB(°/s) 16.4 LSB(°/s)

Tabla 2.-Giroscopio.

Acelerómetro		
Rango de escala completa	Condición AFS_SEL = 0 AFS_SEL = 1 AFS_SEL = 2 AFS_SEL = 3	 $\pm 2g$ $\pm 4g$ $\pm 8g$ $\pm 16g$

Factor de sensibilidad de escala	Condición	
	AFS_SEL = 0	16.384 LSB/g
	AFS_SEL = 1	8.192 LSB/g
	AFS_SEL = 2	4.096 LSB/g
	AFS_SEL = 3	2.048 LSB/g

Tabla 3.-Acelerómetro.

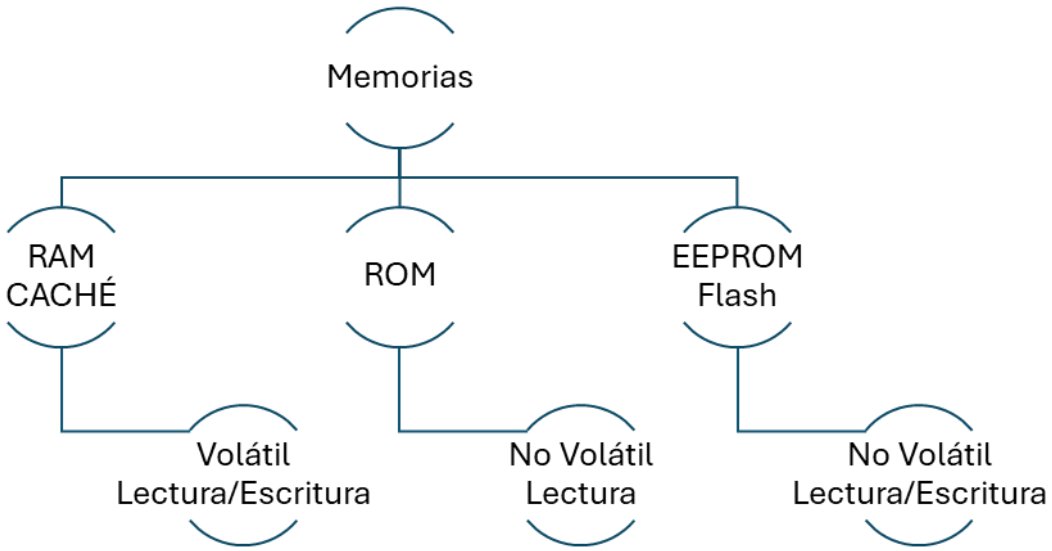
- Resolución: El MPU-6050 tiene una resolución de 16 bits, lo que significa que podría presentar valores de aceleración o velocidad angular de1/2^16 es decir 0.000061 und/und de medida.
- Las medidas del MPU-6050 son de gran calidad debido a que tienen bajos niveles de ruido e incorporan filtros programables de paso bajo para mayor claridad de la señal, aunque esperamos no tener que utilizar estos filtros ya que no el presente proyecto no busca detectar señales tan finas.
- La alimentación del MPU-6050 se realizará a través del Arduino al que también estará conectado para la transmisión de datos hacia la memoria. La tensión de alimentación permite un rango VDD de 2.375 V – 3.46 V. El consumo dependerá lógicamente del uso pero este dispositivo posee un búfer FIFO de 1024 bytes que ayuda a reducir el consumo realizando la transmisión de datos por ráfagas. La corriente de funcionamiento con el acelerómetro y el giroscopio en funcionamiento será de unos 3.8 mA.
- Los rangos de temperatura de funcionamiento oscilan, según el fabricante, entre -40˚C y +105˚C, se observan algunas oscilaciones en el reloj interno del chip con las variaciones de temperatura, pero no se tendrá en cuenta en el presente proyecto ya que el dispositivo irá encapsulado dentro de una envolvente de plástico que hará las veces de aislante y no se espera utilizar realizar mediciones en fluidos con temperaturas extremas.

Por todas estas razones el MPU-6050, parece ser un dispositivo óptimo para el presente proyecto.

MEMORIAS

En la actualidad existen en el mercado una gran variedad de tipos de memorias, que al igual que una gran cantidad de dispositivos tecnológicos nos acompañan en el día a día. Para imaginar la necesidad de las memorias en los dispositivos electrónicos y por ello en todas las máquinas que ensamblen algún tipo de sistema de control electrónico, podemos imaginar cualquier máquina que decide realizar una acción en función de unos parámetros externos. Lo que realmente hace es comparar valores externos con los valores precargados en su memoria y realizar la acción que el algoritmo encuentra en otra parte de la memoria.
Para hacernos una idea del uso de las memorias en una máquina pondremos un ejemplo. Podemos imaginar una máquina de venta de café, en primer lugar, evalúa que se introduzca el importe que el programa determina que es el necesario para empezar el

proceso de la venta del café, este valor estará precargado en algún sitio, a medida que vamos introduciendo monedas el equipo las detecta y va sumando su valor. Tendrá que guardar este dato, el valor de las monedas que se le introducen, durante algún tiempo. Para ello podrá usar una memoria de lectura / escriturar, ya que no necesita este valor una vez se haya realizado la transacción y podrá rescribirlo cada vez. Además, esta memoria podría ser volátil, una vez que la máquina se queda sin alimentación podría no guardar los datos de los importes de las monedas que se le han introducido. Cuando el sistema detecta que el importe introducido es el correcto, permite al usuario la elección del producto, al pulsar el botón lo que realmente le decimos a la máquina es cual algoritmo ha de realizar, todo el proceso de producción y servicio del producto estará guardado en otra memoria. En este caso la memoria podría ser de solo escritura, y tener guardados los procesos sin posibilidad de cambio, parece más lógico utilizar una memoria de lectura / escritura que permita realizar cambios en los productos que puede servir la máquina. En todo caso sería una memoria no volátil, porque de lo contrario perdería los datos siempre que perdiera la alimentación lo que, a priori, no parece muy lógico. Podemos clasificar cualquier memoria con esta clasificación, volátiles, no volátiles y de estas de solo escritura o lectura / escritura. También se suelen definir por la forma de guardar los datos y de leerlos, pero no es objeto del presente proyecto entrar tan a fondo en el tema. Veremos en el siguiente organigrama esta clasificación con algunos ejemplos de memorias.



EEPROM

La memoria EEPROM (Electrucally Erasable Programmable Read-Only Memory) se clasifica en ocasiones como una memoria no volátil de solo lectura por ser una evolución de la ROM (Read-Only Memory), pero realmente es una evolución de esta que si permite la escritura y borrado electrónico. Será la seleccionada para el guardado de datos procedentes de la medición de aceleraciones y velocidades de giro ya que tiene algunas ventajas que la hacen interesante para el proyecto.

- Tienen un bajo consumo de energía durante su uso típicamente menor que el de una memoria Flash.
- Son muy duraderas a los ciclos de lectura / escritura, pudiéndose encontrar en el mercado memorias que sobrepasen el 1,000,000 de ciclos antes de mostrar signos de deterioro.
- Tienen una buena relación calidad precio.

Por tanto, aunque la opción de la memoria Flash resulta muy interesante debido a su gran velocidad de escritura, por el momento nos decantamos por la memoria EEPROM y más concretamente por el módulo DS3231mostrado en la Figura 15, que está compuesto por un reloj de alta precisión y una memoria AT24C32 que posee una capacidad de 32768 bits y está dentro de los parámetros de consumo de energía y de transferencia de datos que podemos manejar con el Arduino.

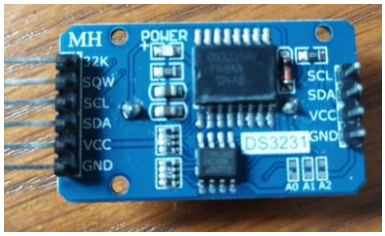


Figura 15.- Módulo DS3231.

Los datos del módulo DS3231 son, según el fabricante [8], los siguientes:

- Capacidad: posee una memoria de 32 Kb (kilobits) y tiene una vida útil de 1*10^6 ciclos de escritura con una retención de datos de 100 años.
- Transmisión de datos mediante I2C, lo que nos dará una velocidad de transmisión de datos que podría llegar a 400 KHz
- Velocidad de escritura de datos. Este parámetro podría ser el talón de Aquiles de este dispositivo ya que las memorias EEPROM son más lentas que las memorias Flash o las memorias RAM. En el caso de la memoria AT24C32 alimentada con una tensión de 5V (este parámetro puede cambiar por otras causas, pero la alimentación es crucial) tiene un tiempo de escritura de 10 ms, podremos hacer una escritura cada 10*10^-3 segundos.
- Alimentación: el módulo DS3231 se alimenta con una tensión entre 1.8V y 5.5V, y tendrá un consumo de 3mA durante una escritura a 100Hz.
- El módulo tiene un rango de operación máximo entre -65˚C y +150˚C. En todo, como ya comentamos anteriormente, no esperamos trabajar en rasgos de temperatura extremos ya que toda la electrónica será embebida en una envolvente plástica que ha de ser estanca.
- Señales de ruido. Posee un disipador de ruido Schmitt para eliminar señales de ruido.

Tendremos muy en cuenta que las velocidades de grabación de datos de la memoria son importantes en este proyecto ya que la velocidad de toma de datos nos ayudará a tener

un perfil más fiable del flujo hidráulico, pero si forzamos a la memoria a valores muy altos de velocidad de escritura podríamos recibir datos erróneos que desvirtúen este perfil.
Debemos tener en cuenta también que la memoria no es infinita por lo que podríamos encontrarnos con que nos quedamos sin capacidad si hacemos una medición demasiado larga en el tiempo y, dado que estamos hablando de un equipo en movimiento, en el espacio. Podríamos determinar, por tanto, si queremos grabar todos los datos del acelerómetro y el giroscopio de cada vez o implementamos algún algoritmo que, en base a algún parámetro, determine si graba todos los parámetros o solo alguno de ellos. Por el momento el presente proyecto seguirá la opción de grabar todos los datos para ver si es capad de detectar anomalías en el flujo. En caso de que este prototipo ejecutado en este proyecto permita detectar dichas anomalías se estudiará la posibilidad de implementar el algoritmo de mejora.

ARDUINO

HISTORIA

El primer Arduino fue diseñado en el instituto de postgrado IVREA, al norte de Italia, por uno de sus alumnos Massimo Banzi. En primera instancia, Banzi creo una palca sencilla formada solamente por un microcontrolador y algunas resistencias con fines puramente académicos. La idea original de Banzi era crear una placa de bajo costo que sus propios compañeros de centro pudieran usar para el aprendizaje práctico de informática y electrónica.
De una placa sencilla que solo podía manipular elementos sencillos como leds y resistencias y que no contaba con el soporte de algún lenguaje de programación para su uso, paso en pocos años a ser uno de los elementos más utilizado en el mundo por la comunidad DIY (Do It Yourself). Es de código abierto (Open Source), nació con la idea de que así fuera y así seguirá siendo, según el propio Banzi [9], ‘tiene efecto IKEA’ y está desarrollada con la idea de que ‘No se necesita el permiso de nadie para fabricar algo grandioso’. Hace algunos años que el proyecto Arduino ya desarrolla placas con todo tipo de utilidades, capacidad de transmisión de datos con diferentes protocolos, soporte de varios leguajes de programación, siendo los más comunes C/C++, Python e incluso su propio IDE (Entorno de Desarrollo Integrado). Cuenta con gran variedad de placas con diferentes capacidades de memoria, conexión WIFI y como además es tan ampliamente utilizada y es de código abierto cuenta con infinidad de dispositivos que se conectan a estas placas y multitud de bibliotecas especialmente desarrolladas para su control. Todo esto hace que sea de mucho interés para un proyecto como el presente. En este caso elegiremos la placa Arduino UNO que es una de las placas más genéricas de la marca, tiene un bajo coste y es de fácil uso, veremos a continuación las peculiaridades de este modelo y otras opciones que podrían ser interesantes en un futuro para la ampliación de este proyecto.

ARDUINO UNO

En la Figura 16vemos una placa de Arduino Uno y sobre ella marcamos algunos de los componentes más importantes para definirlos a continuación. A simple vista ya se puede observar que el modelo desarrollado por Banzi hace menos de dos décadas ha

evolucionado en gran medida y si hacemos una búsqueda de componentes periféricos bibliotecas para el funcionamiento de dichos periféricos nos daremos cuenta de cómo el proyecto en su totalidad a evolucionado aumentando las posibilidades de estos equipos de manera exponencial.

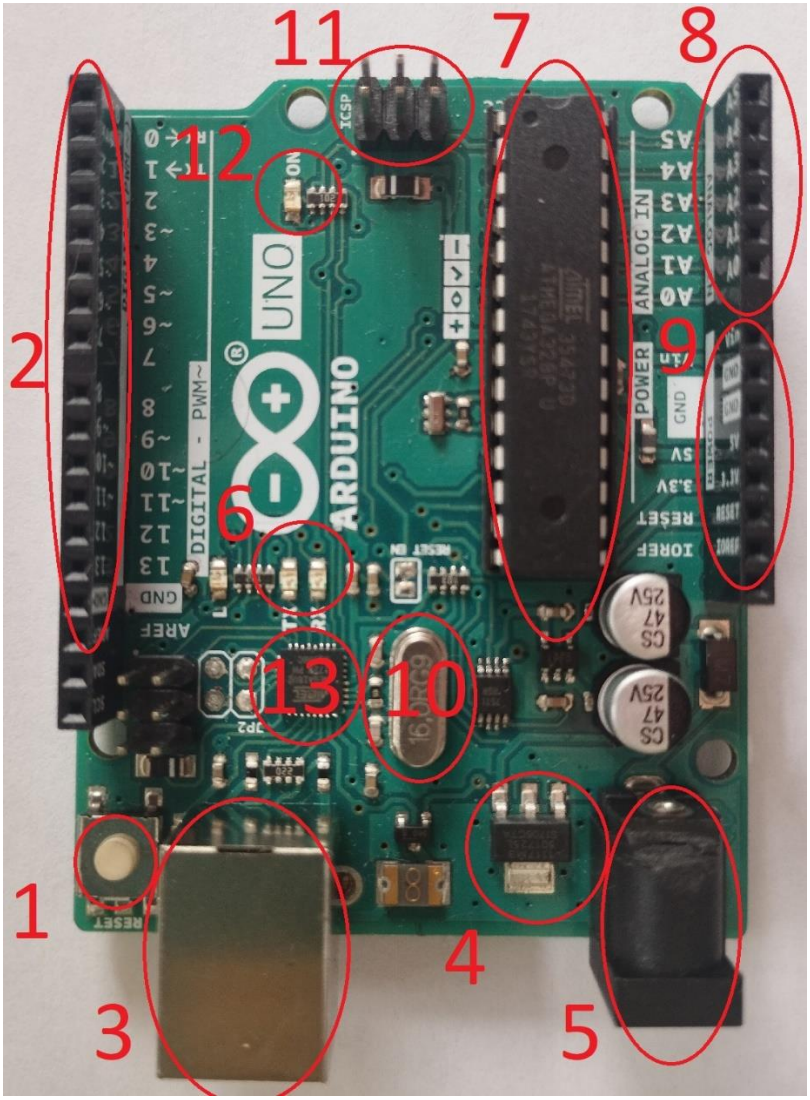


Figura 16.-Placa Arduino UNO.

1. Reset: Este pulsador sirve para reiniciar el programa cargado en el microcontrolador. El reset no corta la alimentación de la placa ni de los periféricos conectados a ella.
2. Puertos de entradas y/o salidas digitales: También conocidos como pines, este zócalo posee un total de 13 tomas que se definirán en él programa para ser utilizadas como entradas o salidas digitales. Estas entradas tienen una tensión predefinida entre 0V (Low) y 5V (High), en caso de necesitar una señal muy precisa o una tensión diferente podremos hacer uso del pin AREF (Analog Reference) que establecería una referencia de voltaje. Los pines 1 y 2, vienen predefinidos para el uso de transferencia de datos en diferentes protocolos, en el caso de este proyecto protocolo el I2C.

3. Interfaz USB: Se utiliza para conectar el ordenador con la placa. Sirve para transmitirle el programa al Arduino, pero también sirve para alimentar a la placa, y los periféricos que tenga conectados, eléctricamente a la vez que hace de puerto de transmisión de datos de forma que podríamos ver, en tiempo real, los parámetros que hayamos definido a tal efecto en el programa o enviarles señales desde el ordenador a la placa.
4. Regulador de tensión: Este chip asegura que la tensión en los componentes de la palca de Arduino es la adecuada para su funcionamiento. En la teoría la alimentación hacia la placa puede realizarse con tensiones entre 5V y 20V, pero una tensión inferior a 7 voltios no asegura una tensión estable en el pin de salida de 5V y una tensión superior a los 12V podría hacer que la placa se recalentase por lo que la tensión a la que alimentamos la placa debería estar entre los 7V y los 12V.
5. Conector de alimentación: Es el puerto por el que comúnmente se alimenta a la placa cuando no se hace desde el puerto USB.
6. Leds Tx y Rx: Estos leds están conectados a los pines 0 y 1 del zócalo de entradas/salidas digitales que, como habíamos comentado, están predefinidos para la comunicación con los periféricos. Estos leds se iluminan cuando la placa está recibiendo datos Rx o transmitiéndolos Tx.
7. Microcontrolador: En este caso la placa lleva integrado un ATmega 328p. Dependiendo del modelo también podría llevar el ATmega 8 o Atmega 168, debido a que este componente es el cerebro de la placa la capacidad del conjunto depende en gran media en el tipo de controlador que posee. En el caso del ATmega 328p le da al equipo una capacidad de procesamiento de datos de 8 bits a 20 MHz, además posee varias memorias internas y capacidad para comunicación UART (Universal Asynchronous Receiver/Transmitter), I2C (Inter-Integrated Circuit) y SPI (Serial Peripheral Interface).
8. Pines de entrada analógicos: Este zócalo permite recibir señales entre 0V y 5V con una resolución de 10 bits, lo que quiere decir que detecta 1024 valores en este rango.
9. Pines de tensión: Los pines en este zócalo sirven para alimentar los periféricos de nuestro circuito, pines 3.3V, 5V y GND. Y para alimentar la propia placa VIN y GND.
10. Reloj oscilador de cristal de 16 MHz: Este elemento marca la frecuencia a la que funciona el microcontrolador, es muy importante sincronizar las frecuencias de funcionamiento de los diferentes elementos del circuito porque en caso de no estar sincronizados el circuito no funcionará correctamente.
11. ISCP (In Chip Serial Programming), este zócalo permite la conexión al ordenador para grabar programas sin necesidad de usar el USB.
12. Led de encendido. Este led estará encendido cuando la placa este encendida lo que nos ayudará a conocer este dato con un solo vistazo de la placa.
13. Chip de comunicación serie: Este chip apoya a la comunicación serie del microcontrolador mejorando su fiabilidad y dotándolo de mayor facilidad de uso.

Se resumen las características más importantes de la placa Arduino Uno en el siguiente cuadro:

Arduino UNO				
Chip	ATmega328P			
Memoria	Flash 32KB	SRAM 2 KB	EEPROM 1 KB	
Interfaces	I2C	SPI	UART	USB
Alimentación	Tensión 7 – 12V		Corriente Máx 410mA	
Pinout	Digital 20	PWM 6	Análogico 6	

Tabla 4 Datos técnicos Arduino UNO [10]

La datasheet de la placa de Arduino UNO nos indica un consumo de corriente en la placa con el microcontrolador ATmega328P de 410mA. Este sería un dato de consumo máximo, pero como no disponemos de un dato medido asumiremos este para los cálculos de consumo totales.

BATERÍA

Para la elección de la batería nos fijaremos en tres parámetros de gran importancia:

- Tensión: El valor de tensión que la fuente ha de mantener lo fija la placa de Arduino UNO, ya que a través de él se alimentarán la memoria y el sensor de inercias. Por tanto, necesitamos una fuente de alimentación entre 7 y 12 Voltios.
- La capacidad de la batería resultará de la suma de las corrientes necesarias para los equipos, teniendo en cuenta, además, el tiempo de uso.

Arduino UNO	410 mA
Memoria EEPROM	3 mA
Sensor MPU 6050	3.8 mA
Total	416.8 mA

Tabla 5.- Consumos eléctricos.

- Las dimensiones y peso de la batería. Teniendo en cuenta que es un equipo que busca tener unas dimensiones moderadas y que pretende moverse en el seno de un fluido buscaremos una batería de pequeñas dimensiones y de poco peso para evitar que se vaya al fondo y se arrastre por la tubería.

Buscando baterías que reúnan estas características nos encontramos con las baterías 6f22 de 9 voltios de litio recargable con una capacidad de 1000 mAh y un peso de 23 gramos. Como podemos observar esta batería nos daría una autonomía de algo más de 2 horas de uso, pero se intentará no cargar/descargar la batería completamente para alargar su vida útil.

SOFTWARE

Arduino es una plataforma de creación sistemas electrónicos de código abierto, tanto su hardware como su software son libres y podremos, por tanto, encontrar todas las

especificaciones para replicar sus placas o sus programas. Esto hace que desde su nacimiento haya existido una extensa comunidad de personas que han hecho crecer las aplicaciones para estos dispositivos. Los lenguajes de programación que se pueden usar para Arduino siguen está misma línea y sería muy discutible cuál es la mejor forma para desarrollar el programa que necesitamos para nuestra aplicación. Pero teniendo en cuenta que el propio Arduino tiene su propio IDE (Entorno de Desarrollo Integrado) optaremos por esta opción para desarrollar el presente proyecto.

IDE ARDUINO

Aunque existen una gran cantidad de modelos de placa Arduino todos ellos pueden programarse utilizando su entrono de desarrollo que, además, como todo en Arduino, es de acceso libre y podremos descargárnoslo de la Internet sin ningún problema. El entorno de desarrollo es muy intuitivo en cuanto a crear, guardar y abrir los programas que vamos a manejar para controlar nuestra placa. A estos programas se les conoce con el nombre de Sketch, y así es como hace referencia a ellos en el propio IDE de Arduino, podemos crear nuestros propios sketches desde cero o descargar proyectos similares e ir modificándolos. Aunque no es objeto de este trabajo la programación en sí misma, veremos los puntos más interesantes de esta a la vez que vemos el desarrollo de nuestro programa.

INTERFAZ DE USUARIO DEL IDE DE ARDUINO

Cuando abrimos la IDE de Arduino nos encontramos con un interfaz de programa como el que podemos ver en la Figura 17. Dependerá de la versión que hayamos descargado pero las funciones más relevantes serán iguales.

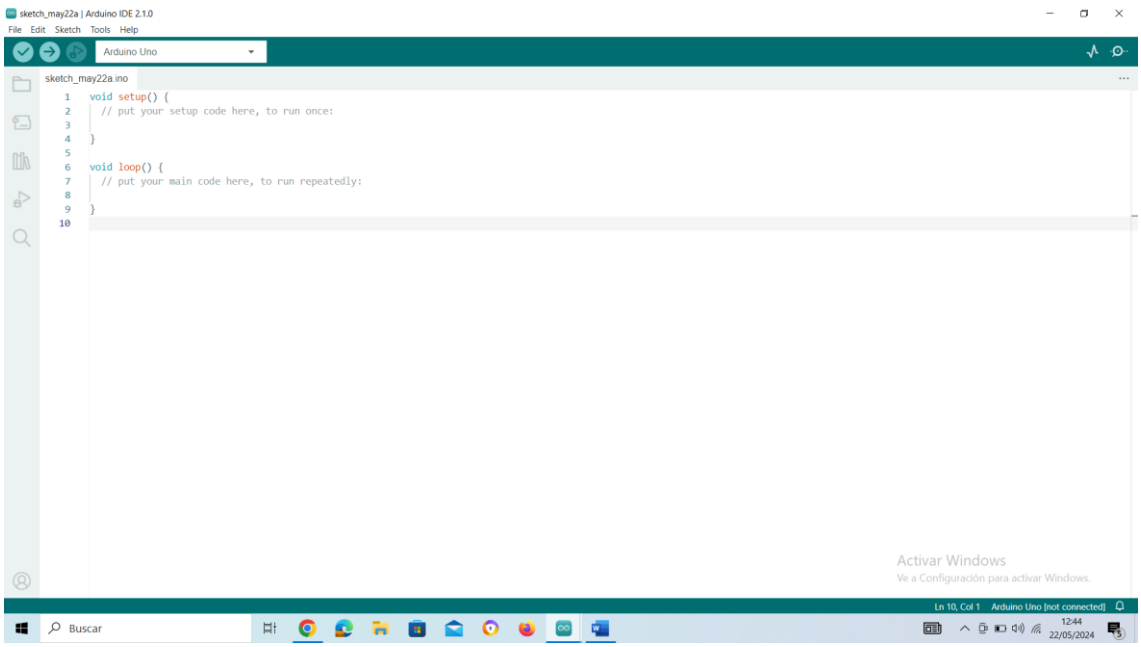


Figura 17.- IDE de Arduino

En esta interfaz nos muestra la estructura básica que habrá de seguir cualquier programa de Arduino. Aunque podría ser más extenso, un programa básico en el IDE de Arduino tendrá 3 partes bien diferenciadas:

- Una primera parte donde se añaden las bibliotecas, se definen las direcciones I2C de los componentes electrónicos, y se definen las variables globales.
- Función Setup. Es donde se realiza la configuración de inicio de la placa. Configuración de pines, inicialización de variables y otros.
- Función Loop. Esta función es la que define el comportamiento del programa y por tanto de la placa. Esta parte del código se ejecuta de forma cíclica mientras la placa tenga alimentación eléctrica.

Aunque podemos usar otras funciones, en este proyecto así lo haremos, estas funciones son básicas y se utilizaran en cualquier programa de Arduino por pequeño que sea.
Vemos que tenemos las cintas típicas de opciones, que son bastante intuitivas. Solo destacaremos, para no extendernos demasiado, algunas opciones que nos serán de gran ayuda para el desarrollo de nuestro proyecto.

- Sobre la cinta verde, en la parte superior de la pantalla, a la izquierda encontramos dos iconos muy importantes. El primero de ellos, un icono con la imagen de un tick, que nos sirve para comprobar si la sintaxis del código es correcta. Nos indicará si el código es correcto, o por el contrario si tiene errores, describiendo en una partición en la zona inferior de la pantalla la naturaleza de estos y el lugar del sketch donde se encuentra. El segundo, una imagen con una flecha hacia la derecha, que sirve para subir el código a nuestra placa de Arduino.
- Sobre la misma cinta verde, esta vez a la derecha, tendremos un icono con una lupa. Esta opción abrirá el monitor serie que se mostrará en una partición de la pantalla en la parte inferior. Nos será de mucha ayuda mientras estemos creando el programa ya que nos mostrará en tiempo real el valor de cualquier valor que le indiquemos, simplemente añadiendo al código el comando *Serial.print ()* con la variable que deseemos ver dentro de los paréntesis.

- En la cinta vertical, a la izquierda de nuestra pantalla, encontraremos un icono, con un pictograma de unos libros, para la administración de las librerías. El uso de librerías hace que la programación sea mucho más sencilla. Las bibliotecas son códigos específicamente desarrollados para hacer funcionar componentes específicos, como por ejemplo un acelerómetro una pantalla LCD o cualquier componente electrónico que esté disponible para Arduino. Podríamos desarrollarlas nosotros mismos, pero la verdad es que es una labor realmente ardua y que requiere de un gran conocimiento de programación, de modo que buscaremos las bibliotecas en el propio IDE de Arduino, ya tiene precargadas una gran cantidad de ellas. Como ya hemos comentado la comunidad de personas que colaboran con Arduino es extensa, por lo que si no encontrásemos la biblioteca que necesitamos en el gestor del IDE podremos descargarlas de internet y cargarlas en la IDE para usarlas en nuestro proyecto.

A continuación, desgranaremos el código de programación con el que funcionará nuestro prototipo e iremos viendo de forma somera las funciones y aplicaciones que utilizamos.
El la Figura 18, vemos que utilizamos las primeras 25 líneas de código para introducir librerias, lína de la1 a la 5 inclusive, las direcciones I2C de los componentes que necesitemos comunicar. Y por último se definen las variables que vamos a utilizar indicando del tipo que son.
Para conocer las direcciones I2C de los componentes que tenemos conectados a la placa, basta con utilizar un sketch específico para ello, podremos encontrar este código en la página de Arduino Spain [11], por ejemplo. En nuestro caso, al conectar ambos dispositivos veremos que solo nos aparecen 2 direcciones y deberían de aparecer 3, es porque 2 componentes tienen la misma dirección. Obviamente no se pueden utilizar 2 componentes con una misma dirección por lo que buscaremos en las especificaciones del fabricante la posibilidad de cambiar las direcciones. Tanto la memoria como el acelerometro pueden cambiar su dirección. Pero en el caso de este último es más sencillo ya que solo necesitamos poner un cable desde el pin de alimantación VCC al pin AD0. Para cambiar la dirección de la memoria deberíamos de hacer soldaduras sobre la propia placa y esto siempre es un riesgo. Despues de realizar esta operación volveremos a utilizar el escaner y vemos que nos muestra una nueva dirección I2C, que será la correspondiente al acelerometro.

Ver_Def_word.ino

```
1   #include "I2Cdev.h"
2   #include "Wire.h"
3   #include "MPU6050.h"
4   #include "AT24Cxx.h"
5   #include "DS3231.h"
6
7   MPU6050 sensor (0x69);
8   AT24Cxx eep (0x57 , 32);
9
10  int address=0;
11
12  int ax, ay, az;
13  int gx, gy, gz;
14
15  char menos = "-";
16  char mas  = "+";
17
18  byte ax1, ax2, ax3;
19  byte ay1, ay2, ay3;
20  byte az1, az2, az3;
21  byte gx1, gx2, gx3;
22  byte gy1, gy2, gy3;
23  byte gz1, gz2, gz3;
24
25  char signo;
26
```

Figura 18.-Inicio de Código.

Es importante definir bien las variables porque en algunos casos, sobre todo en las variables numéricas, porque, aunque puede parecer irrelevante, cada variable requerirá de un espacio específico en la memoria y permitirá realizar diferentes operaciones con los valores que se le asignen.

```
21
22  void setup() {
23    // put your setup code here, to run once:
24  Serial.begin (9600);
25  Wire.begin();
26  sensor.initialize();
27    while (!Serial){
28   ;
29    }
30  }
31
```

Figura 19.- Función Void Setup.

Dentro de la función *void setup()*, Figura 19, tendremos el trozo de código que inicializa el puerto serie, y determina la velocidad de transmisión de este con la función *Serial.begin (9600)*. También se inicializa la conexión I2C, mediante la instrucción *Wire.begin ()* y el sensor MPU-6050, es decir, acelerómetro y giroscopio.
Para mostrar el código de la función *void loop()* veremos la Figura 20, la Figura 21y l la Figura 22. En este caso solo analizaremos como se trata la variable *ax*, el resto de las variables seguirán la misma mecánica y se dispondrá de todo el código en el Anexo II.

```
31
32  void loop() {
33    // put your main code here, to run repeatedly:
34  sensor.getAcceleration(&ax, &ay, &az);
35  sensor.getRotation (&gx, &gy, &gz);
36
37
38   /*Serial.print("a[x y z] g[x y z]:\t");
39    Serial.print(ax); Serial.print("\t");
40    Serial.print(ay); Serial.print("\t");
41    Serial.print(az); Serial.print("\t");
42    Serial.print(gx); Serial.print("\t");
43    Serial.print(gy); Serial.print("\t");
44    Serial.println(gz);*/
45
```

Figura 20.-Función Void Loop (I).

En esta primera parte del códio dentro de la función loop (Figura 20), que se repetirá indefinidamente mientras la placa tenga alimentación, tomaremos valores de los tres ejes tanto del acelerometro, función *sensor.getAcceleration (&ax,&ay,&az)* como del girocospio, función *sensor.getRotation(&gx,&gy,&gz)*.
El código entre la línea 38 y la línea 44 está comentado, esto quiere decir que para la placa ese trozo de código no existe y por tanto no gastará recursos para su gestión, por está razón está sombreado. Este trozo de código está desarrollado para ver en el monitor serie que valores está tomando el Arduino del sensor. Podremos verlo mientras el ordenador y la placa de Arduino esten conectados, nos sería de gran ayuda para comprobar el funcionando de nuestro programa en un momento dado. Para que el monitor serie nos muentre dichos valores tendremos que tener el código sin comentar, quitandole la barra y asterisco del principio /* y final de bloque */. Otra opción para comentar parte del código es añadir dos barras inclinadas // al principio de una línea y así la dejaremos comentadas. Resulta muy útil comentar y descomentar las partes del programa que estamos creando para poder hacer pruebas.

```
52
53   if(ax>=0){
54    eep.write (address,'+');
55   }else{
56    eep.write (address,'-');
57   }
58   signo= eep.read(address);
59  Serial.println (signo);
60
61  int axt=abs(ax);
62  Serial.println(axt);
63
64    if (axt>10000){
65      ax1=axt/10000;
66      ax2=(axt-(ax1*10000))/100;
67      ax3=axt-(ax1*10000)-(ax2*100);
68  }
69    if (axt<10000 && axt>100){
70      ax1=0;
71      ax2=axt/100;
72      ax3=axt-(ax1*10000)-(ax2*100);
73  }
74    if (axt<100){
75      ax1=0;
76      ax2=0;
77      ax3=axt;
78     }
79
80   eep.write(++address, ax1);
81
82   eep.write(++address, ax2);
83
84   eep.write(++address, ax3);
85
```

Figura 21.-Fución Void Loop (II).

En la Figura 21 podemos ver la parte del código que nos permite guardar en la memoria el valor de la aceleración en el eje x. Como ya dijimos, las bibliotecas se desarrollan específicamente para el uso de dispositivos con unos parámetros determinados. Al crear el programa e ir haciendo pruebas nos encontramos que los valores numéricos mayores de 255 fallaban cuando se leían desde la memoria, esto es debido a que la biblioteca está desarrollada para guardar los datos en una posición o hueco de la memoria y esto quiere decir 1 byte de cada vez. Por ello, se ha realizado un algoritmo que parte el dato en números menores de 3 dígitos, asegurando valores menores a 255, antes de grabarlos en la memoria. Vamos a ver, paso a paso, en que consiste realmente este algoritmo y como se implementa:

- Debemos de guardar en la memoria los datos, hueco a hueco, es decir datos no mayores a un byte, un carácter o un número entre 0 y 255.
- En primer lugar, utilizaremos una función *If* que guarda en la posición de la memoria *address* el signo positivo + si *ax* es mayor o igual que cero y si no, comando *else*, el sigo negativo -.
  Como podremos observar en el resto del programa se utiliza el comando *++address*, que suma 1 a la variable para realizar un contador incremental que

nos ayude, de forma sencilla, a realizar grabaciones en las sucesivas celdas de la memoria.

- Después, designaremos una nueva variable *int axt* el valor absoluto de la variable *ax* para poder partirlo en datos que se puedan guardar en un byte de memoria. En este punto tendremos tres opciones:
    - Que el número sea mayor de 10000, cabe recordar que el rango de medición del sensor varía entre 32768 y -32768, en este caso dividiremos dicho número entre 1000 y guardaremos el resultado en la variable *ax1*. Después, restaremos del valor completo de la variable *axt* el valor *ax1* multiplicado por 10000 y dividiremos el resultado entre 100, este valor se guardará en la variable *ax2*. De esta forma tendremos los dos primeros dígitos del valor total de *ax*, empezando por la izquierda, en la variable *ax1* y los dos siguientes en la variable *ax2*. Ahora ya solo nos quedará obtener los dos últimos números de nuestra variable para guardar en la variable *ax3*, seguiremos para ello la misma mecánica, restando los valores de *ax1* multiplicado por 10000 y *ax2* multiplicado por 100 al valor total, *axt*.
    - Los otros dos casos se abordan de igual manera, pero teniendo en cuenta que, si el número fuera menor de 10000 e incluso de 100, los primeros números serían 0 y obtendríamos un valor erróneo en las divisiones por lo que usaremos la función *if* anidada para determinar si nuestro valor *axt* es de 2, 4 o 5 dígitos.
- Ahora ya solo queda grabar los valores en la memoria con la función *epp.write(n° de celda, nombre de la variable).*

```
295
296   if (address < eep.length()){
297     (address++);
298   }else{
299     exit(0);
300   }
301
302     delay(500);
303   }
304
```

Figura 22.-Fin de void loop.

En la Figura 22, se ve que al final del código se compara el valor de la variable *address* con la capacidad de la memoria, en el caso de que todavía quede espacio le sumamos 1 y seguimos con el programa que se repetirá en bucle con una pausa al final del código igual al valor numérico, en milisegundos, introducido en la función *delay()*, en este caso 500 ms. Si el valor de la variable *address* fuese superior a la capacidad de la memoria la fusión *exit(0)* detendrá el programa.

RECUPERACIÓN DE DATOS

Una vez que pongamos nuestro prototipo a funcionar, la electrónica estará dentro de la envolvente adquiriendo datos y guardándolos en la memoria. Necesitaremos recuperar los datos cuando hayamos acabado la medición por lo que necesitaremos un sketch que, simplemente, nos entregue todos los valores guardados para después poder enviarlos a Excel y realizar el tratamiento de dichos datos.
Para realizar la lectura de datos buscaremos en los ejemplos que la propia biblioteca de la memoria posee. En este caso encontramos un sketch, Figura 23, al que tan solo tendremos que cambiarle la dirección I2C de la memoria.

i2c_eeprom_read.ino

```
1
2   #include <AT24Cxx.h>
3   #define i2c_address 0x57
4
5   int address = 0;
6   byte value;
7
8   AT24Cxx eep(i2c_address, 32);
9
10  void setup() {
11    Serial.begin(9600);
12    while (!Serial) {
13      ;
14    }
15  }
16
17  void loop() {
18    value = eep.read(address);
19
20    Serial.print(address);
21    Serial.print("\t");
22    Serial.print(value, DEC);
23    Serial.println();
24
25    address = address + 1;
26    if (address == eep.length())
27      address = 0;
28    }
29
30    delay(500);
31  }
32
```

Figura 23.- Sketch para lectura de datos.

Observamos que la declaración de la variable de lectura por parte de este skecth es del tipo *byte value*, por lo que leerá cada dato como un valor numérico. En el caso de los signos + y – nos mostrará su valor de código ASCII. Es decir, mostrará el número 43

cuando sea negativo y el 45 cuando sea positivo. Tendremos en cuenta este hecho cuando hagamos el tratamiento de datos.

TRATAMIENTO BÁSICO DE DATOS

En primer lugar, para realizar un tratamiento de los datos debemos de pasar estos del IDE de Arduino a un programa que nos ayude a realizar su análisis. Como ya viene sucediendo, las opciones para realizar esta toma de datos y su análisis posterior son variadas, nosotros nos decantaremos por realizar el análisis con Excel, ya que es un programa que nos es familiar. Además, existe un software específico para la adquisición directa de los datos desde el Arduino a través del puerto serie, el PLX-DAQ. Veremos, a continuación, como funciona.

PLX-DAQ

El PLX-DAQ es una macro especialmente desarrollada para tomar los datos de un puerto serie y plasmarlos en una hoja de Excel. Se puede descargar de internet ya que es un software libre, tendremos especial cuidado de descargar una versión que sea compatible con nuestro PC, poniendo especial atención en si el sistema operativo es de 64 o 32 bits, porque no todas las versiones soportan ambos, pero no se especifica claramente en el software.
Cuando abrimos la macro nos encontramos con una ventana emergente como la de la Figura 24. Prestaremos especial atención al número de puerto que podremos ver en esquina inferior izquierda del propio IDE Arduino y la velocidad de transmisión de datos, los Baudios, que habremos configurado en nuestro programa de lectura de datos de la EEPROM del Arduino.

Figura 24.- Ventana emergente PLX-DAQ.

Para enviar los datos desde nuestro Sketch de lectura de la EEPROM, los haremos como si lo mandasemos al propio Monitor Serie. Solamente tendremos que añadir la instrucción *Serial.print ("DATA,TIME,")* que realiza el envio de los datos a nuestra página de Excel. Dentro de las pecualiridades de este software esta que no reconoce la instrucción típica *Serial.print ("\t")* para la tabulación y en lugar de ella debemos de utilizar *Serial.print (",")*.

Aunque no es estrictamente necesario utilizaremos un par de instrucciones más que insertaremos dentro de la función *Void Setup ()*, para que se ejecuten solo una vez. La función *Serial.println ("CLEARDATA")*, que limpia los datos de la hoja Excel antes de empezar, y la función *Serial.println("LABEL, Time, Address, Valor")* que nos permite nombrar las columnas en las que vamos a recibir los datos.

Por tanto, el skecth para realizar esta operación nos quedaría como se muestra en la Figura 25.

Eeprom_send.ino

```
#include <AT24Cxx.h>
#define i2c_address 0x57

byte address = 0;
byte value;

AT24Cxx eep(i2c_address, 32);

void setup() {

  Serial.begin(9600);
  while (!Serial) {
    ;
  }
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,Address,Valor");
}

void loop() {

  value = eep.read(address);
  Serial.print("DATA,TIME,");
  Serial.print(address);
  Serial.print(",");
  Serial.print(value, DEC);
  Serial.println();

  address = address + 1;
  if (address == eep.length()) {
    address = 0;
  }

```

Figura 25.- Sketch para enviar datos a Excell.

Y obtendríamos una hoja Excel como la mostrada en la Figura 26.

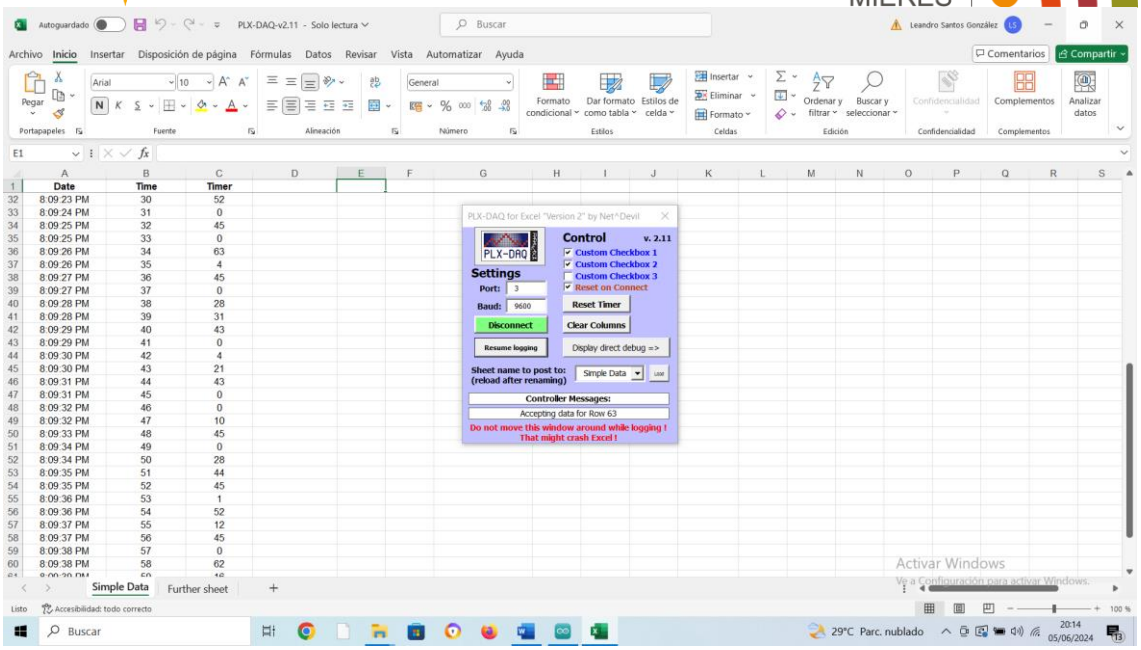


Figura 26.- Excel con datos leídos de la EEPROM.

Se desarrolla una macro de Excel para ordenar convenientemente, como se puede observar en la Figura 27, los datos para su posterior tratamiento mediante un algoritmo de procesamiento de datos de sistemas de orientación y rumbo con el fin de obtener, finalmente, los valores medidos de forma ordenada y que podamos interpretar.

Time	Dirección	Valor					ax	ay	az	gx	gy	gz
4:21:29 PM	0	43	872	-872	-0,053223							
4:21:29 PM	1	0					-0,053223	-0,086914	-0,383545	-20,9313	-4,045802	-0,122137
4:21:30 PM	2	8					-0,052002	-0,087891	-0,382324	-21,83206	-3,206107	-0,015267
4:21:30 PM	3	72					-0,055908	-0,084961	-0,391113	-21,44275	-3,396947	-0,221374
4:21:31 PM	4	43	1424	-1424	-0,086914		-0,046875	-0,083008	-0,386963	-21,50382	-3,152672	-0,007634
4:21:31 PM	5	0					-0,051758	-0,085693	-0,378662	-21,28244	-3,274809	-0,061069
4:21:32 PM	6	14					-0,055176	-0,084229	-0,382813	-21,41985	-3,412214	-0,267176
4:21:32 PM	7	24					-0,050781	-0,084717	-0,38501	-22,0916	-2,877863	-0,053435
4:21:33 PM	8	43	6284	-6284	-0,383545		-0,053711	-0,07959	-0,399414	-23,75573	-3,358779	-0,099237
4:21:33 PM	9	1					-0,04834	-0,089111	-0,381836	-22,22901	-2,328244	-0,091603
4:21:34 PM	10	52					-0,054932	-0,084229	-0,393799	-21,83206	-3,022901	-0,137405
4:21:35 PM	11	84					-0,056396	-0,079346	-0,388672	-21,16794	-3,389313	-0,038168
4:21:35 PM	12	45	2742	-2742	-20,9313		-0,050049	-0,088867	-0,388916	-21,72519	-3,229008	-0,061069
4:21:36 PM	13	0					-0,044678	-0,085205	-0,379639	-22,03053	-3,648855	-0,099237
4:21:36 PM	14	27					-0,046387	-0,084473	-0,38916	-21,41221	-3,030534	-0,267176
4:21:37 PM	15	42					-0,050049	-0,086426	-0,391602	-21,68702	-3,099237	-0,061069
4:21:37 PM	16	43	530	-530	-4,045802		-0,051025	-0,094971	-0,379639	-21,71756	-2,839695	-0,152672
4:21:38 PM	17	0					-0,050781	-0,088623	-0,379639	-21,96947		
4:21:38 PM	18	5										
4:21:39 PM	19	30										
4:21:39 PM	20	43	16	-16	-0,122137							
4:21:40 PM	21	0										
4:21:40 PM	22	0										
4:21:41 PM	23	16										
4:21:41 PM	24	43	852	-852	-0,052002							
4:21:42 PM	25	0										
4:21:42 PM	26	8										
4:21:43 PM	27	52										
4:21:44 PM	28	43	1440	-1440	-0,087891							
4:21:44 PM	29	0										
4:21:45 PM	30	14										
4:21:45 PM	31	40										
4:21:46 PM	32	43	6264	-6264	-0,382324							
4:21:46 PM	33	1										
4:21:47 PM	34	52										
4:21:47 PM	35	64										
4:21:48 PM	36	45	2860	-2860	-21,83206							
4:21:48 PM	37	0										
4:21:49 PM	38	28										
4:21:49 PM	39	60										
4:21:50 PM	40	43	420	-420	-3,206107							
4:21:50 PM	41	0										
4:21:51 PM	42	4										
4:21:51 PM	43	20										
4:21:52 PM	44	45	2	-2	-0,015267							
4:21:53 PM	45	0										
4:21:53 PM	46	0										

Figura 27.- Tratamiento de datos en Excel.

Los tres algoritmos de procesamiento de datos más utilizados en sistemas de referencia de orientación y rumbo son el filtro de Kalman, el de Mahoy y el de Madgwick [12]. Estos algoritmos se suelen usar para el cálculo, en tiempo real, del posicionamiento de las naves, utilizando los datos adquiridos para rectificar los comandos enviados a los propulsores consiguiendo con ello el control necesario para los movimientos precisos de dichas naves. En nuestro caso vamos a utilizar, a modo de ejemplo una simplificación del filtro de Madgwick directamente implementado en el propio Excel. Teniendo en cuenta que en el proyecto que nos ocupa no necesitamos las respuestas del cálculo de manera automática, sería interesante en una fase más avanzada del proyecto aplicar los tres filtros. Tendremos especial cuidado con ver que no filtramos las vibraciones que estamos buscando.

Los datos que obtendremos de aplicar el filtro serán cuaterniones del tipo (q0, q1, q2, q3) en los que el componente 'q0' corresponderá al componente escalar y los componentes 'q1', 'q2', 'q3', a los componentes vectoriales del cuaternión. Se muestran estos valores en la Figura 28.

	A	B	C	D	E	F	G	H	I	J	K	L
1	ax	ay	az	gx	gy	gz						
2	-0,05322266	-0,08691406	-0,38354492	-20,9312977	-4,04580153	-0,1221374		0,99428883	-0,10490671	-0,01959423	-0,0006072	
3	-0,05200195	-0,08789063	-0,38232422	-21,8320611	-3,20610687	-0,01526718		0,97640291	-0,21309254	-0,03503916	-0,00109142	
4	-0,0559082	-0,08496094	-0,39111328	-21,4427481	-3,39694656	-0,22137405		0,94709714	-0,3168108	-0,05131197	-0,00223484	
5	-0,046875	-0,08300781	-0,38696289	-21,5038168	-3,15267176	-0,00763359		0,90648799	-0,41709865	-0,06558106	-0,00271374	
6	-0,05175781	-0,08569336	-0,37866211	-21,2824427	-3,27480916	-0,0610687		0,85559405	-0,51145635	-0,07976199	-0,00304223	
7	-0,05517578	-0,08422852	-0,3828125	-21,4198473	-3,41221374	-0,26717557		0,79421869	-0,6002839	-0,09413185	-0,00388528	
8	-0,05078125	-0,0847168	-0,38500977	-22,0916031	-2,8778626	-0,05343511		0,72143733	-0,68450614	-0,10462684	-0,0057219	
9	-0,05371094	-0,07958984	-0,39941406	-23,7557252	-3,35877863	-0,09923664		0,63309765	-0,76536806	-0,11555184	-0,00684587	
10	-0,04833984	-0,08911133	-0,38183594	-22,2290076	-2,32824427	-0,09160305		0,54251262	-0,83111057	-0,12172637	-0,01086148	
11	-0,05493164	-0,08422852	-0,39379883	-21,8320611	-3,02290076	-0,13740458		0,44639547	-0,88548578	-0,12848758	-0,01170306	
12	-0,05639648	-0,0793457	-0,38867188	-21,1679389	-3,38931298	-0,03816794		0,34762845	-0,92793098	-0,13415272	-0,01008003	
13	-0,05004883	-0,08886719	-0,38891602	-21,7251908	-3,22900763	-0,0610687		0,24232294	-0,96027806	-0,13804807	-0,0094008	
14	-0,04467773	-0,08520508	-0,37963867	-22,0305344	-3,64885496	-0,09923664		0,13236422	-0,98110721	-0,14093874	-0,00668089	
15	-0,04638672	-0,08447266	-0,38916016	-21,4122137	-3,03053435	-0,26717557		0,02463377	-0,98945569	-0,1425936	-0,00613767	
16	-0,05004883	-0,08642578	-0,39160156	-21,6870229	-3,09923664	-0,0610687		-0,08375678	-0,98633729	-0,14175028	-0,00551505	
17	-0,05102539	-0,0949707	-0,37963867	-21,7175573	-2,83969466	-0,15267176		-0,19090903	-0,97156176	-0,1399304	-0,00640175	
18												

Figura 28.- Filtro de Madgwick.

GRÁFICAS

Para hacernos una idea de los valores con que estamos trabajando vamos a pasar los quaterniones a grados Euler. Es decir, analizaremos las rotaciones registradas en el prototipo en cada uno de los tres ejes con respecto a un sistema de ejes fijos. Los valores que obtendremos serán, Roll o rotación con respecto del eje x, Pitch o rotación con respecto del eje y, Yaw o rotación con respecto del eje z. En la Figura 29 se muestra la gráfica en el caso del prototipo quieto durante un minuto tomando valores cada 0,5 segundos. Se realiza, también, la gráfica para el prototipo girando durante 5 segundos tomando valores cada 0,01 segundos, como se observa en la Figura 30.

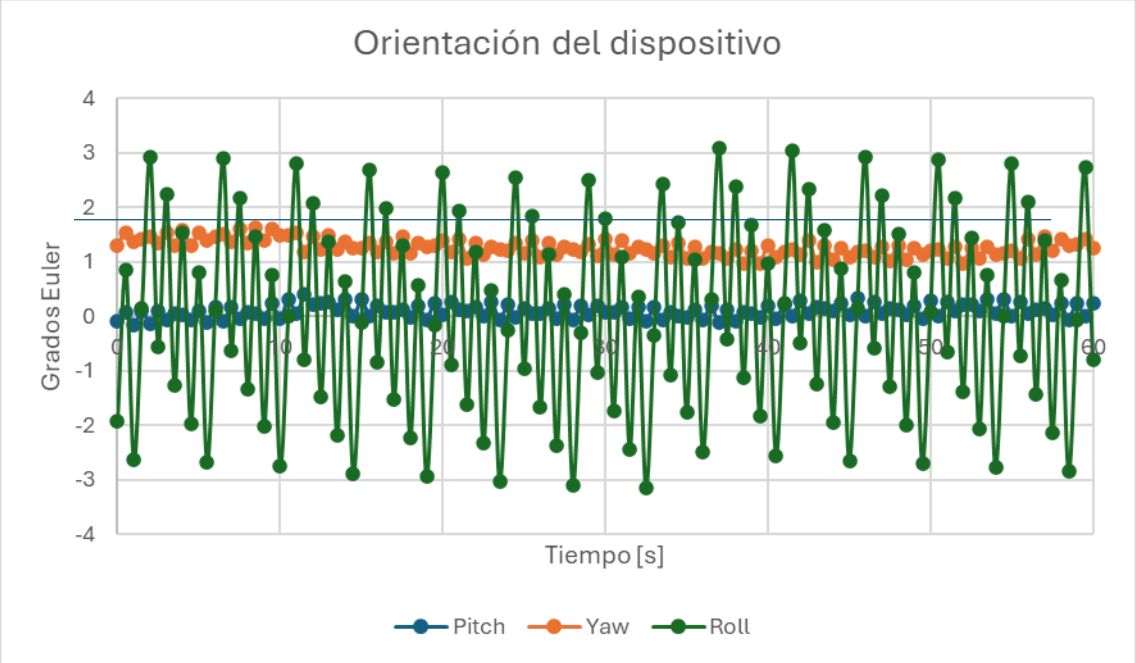


Figura 29.- Prototipo estático.

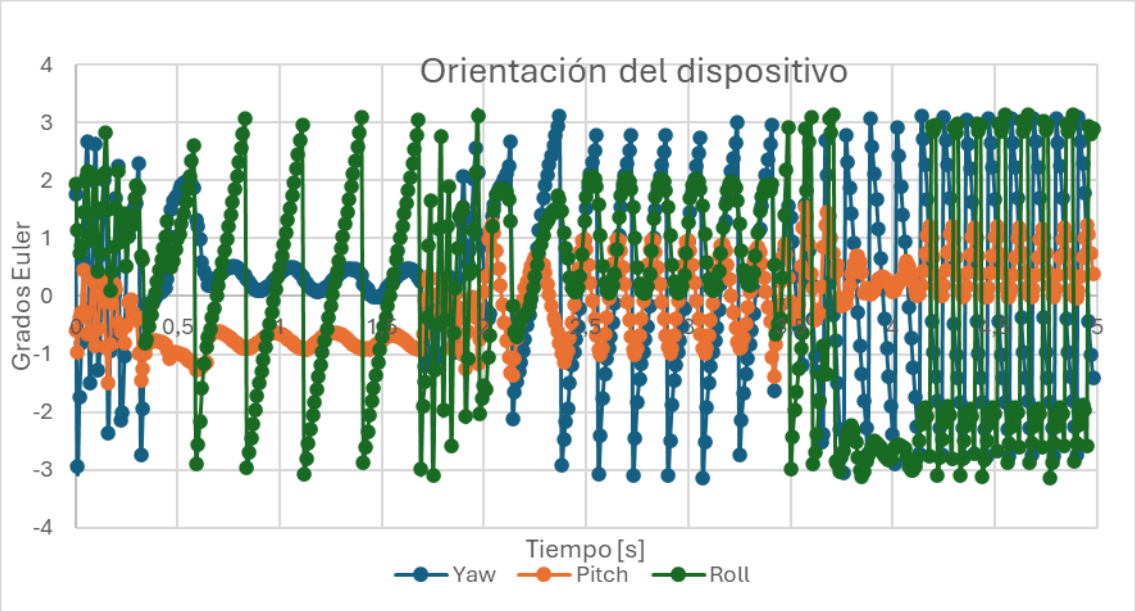


Figura 30.-Prototipo en movimiento caótico.

CONCLUSIONES

Las conclusiones que obtenemos del presente proyecto son:

- Para un desarrollo ágil de la envolvente un programa de CAD, como puede ser el Fusion 360 permite generar un prototipo a base de bocetos con facilidades para implementar modificaciones.
- Los programas de rebanado y de reparación de mallas dotan al proyecto de calidad en la impresión y disponen de herramientas de preselección de

parámetros como el modelo de impresora o el material de impresión que la facilitan.

- La impresión 3D con PETG agiliza el desarrollo del prototipo y le confiere una calidad suficiente para la realización de pruebas. Este tipo de material no requiere de una temperatura de impresión elevada y tiene una baja contracción térmica lo que facilita la impresión y da como resultado una envolvente de buenas características mecánicas: elevado módulo elástico, estabilidad química y resistencia térmica.
- La tecnología Arduino es de aplicación para este tipo de proyectos gracias a la versatilidad de sus aplicaciones, la cantidad de periféricos con los que puede interactuar y la disponibilidad de bibliotecas disponibles.

  Su entorno de desarrollo facilita la implementación del código necesario para controlar los sensores y la memoria. Es de código abierto y posee una gran comunidad de desarrolladores que nos facilita la resolución de problemas que surgen durante la programación.
- El módulo MPU 6050 GY-521 con acelerómetro y giroscopio basado en tecnología MEMS, así como el módulo DS323 con memoria EEPROM de 32K, nos permiten la adquisición de datos de aceleración y giro del dispositivo, así como su guardado para el posterior análisis.
- El programa Excel puede utilizarse para obtener los cuaterniones y ángulos Euler que permitirían el posicionamiento del dispositivo, así como el graficado de los mismos, sin embargo, es recomendable el uso de otras herramientas específicas para el análisis de grandes volúmenes de datos.

RETOS Y FUTURAS LÍNEAS DE INVESTIGACIÓN

En este proyecto hemos diseñado un prototipo que pretende detectar las anomalías causadas en un flujo de agua a través de una tubería debido a una fuga. Los retos que debe de superar este proyecto serán de diferente naturaleza:

- En primer lugar, deberíamos de comprobar si alguno de los algoritmos de procesamiento de datos nos muestra la anomalía que buscamos de forma clara y en que rangos es capaz de hacerlo. Podría ser interesante implementar un programa que realizase el análisis con varios algoritmos a la vez para analizar las diferencias entre ellos.
- Las dimensiones del prototipo se han determinado en función de los dispositivos electrónicos que alberga, pero sería práctico desarrollar una envolvente de menor tamaño para utilizar en conducciones de menor sección.
  Se podría diseñar un modelo que utilice placas de menores dimensiones o incluso desarrollar una única placa con todos los dispositivos necesarios embebidos.
  Podríamos utilizar baterías que se adapten a las necesidades exactas, tanto desde un punto de vista de capacidad como de dimensiones.

Para continuar con este proyecto se recomienda la realización de pruebas en un entorno controlado para la adquisición de datos que permitan realizar los ajustes necesarios tanto en la envolvente como en los algoritmos de filtrado de datos para con las tecnologías

actuales, como herramientas de inteligencia artificial y machine learning definir exactamente los puntos de fuga.

RELACIÓN DEL TFG CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE

ODS CON LOS QUE SE RELACIONA EL TFG

El presente TFG se desarrolla con la idea de mejorar el mantenimiento de las redes y canalizaciones de distribución y recogida de aguas por lo que se alinea con algunos de los objetivos marcados por los Estados Miembro de las Naciones Unidas en la Agenda 2030 para el desarrollo sostenible. A través de un proyecto de innovación se mejorarían las redes de agua limpia y saneamiento ahorrando energía en tratamientos de agua y bombeo y haciendo con ello las ciudades más sostenibles.

Tabla 6.- Desarrollo sostenible.

Referencias

[1] Raskin, P.; Gleick, P. H.; Kirshen, P.; Pontius, R. G. Jr; and Strzepek, K., 1997. Comprehensive assessment of the freshwater resources of the world. Stockholm Environmental Institute, Sweden. 5th session of the United Nations Commission on Sustainable Development, 1997.

[2] Velázquez, M. P., Escriva-Bou, A., & Sorribes, H. M. (2020). Balance hídrico actual y futuro en las cuencas en España, déficits estructurales e implicaciones socioeconómicas (No. eee2020-38). FEDEA.

[3] Aksela, K., Aksela, M., & Vahala, R. (2009). Leakage detection in a real distribution network using a SOM. Urban Water Journal, 6(4), 279-289.

[4] Ragan, R. R. (1984). Inertial technology for the future. *IEEE transactions on aerospace and electronic systems*, *20*(4), 414-444.

[5] N. Yazdi, F. Ayazi, y K. Najafi, «Micromachined inertial sensors», Proceedings of the IEEE, vol. 86, no . 8, pp. 1640–1659, 1998.

[6] Giménez Martínez, J., Loeff, E., y García Moreno, E. (2014). Estudio de la navegación de un planeador autónomo submarino. En: XXXV Jornadas de Automática, Valencia, España, 3-5 de septiembre. Comité Español de Automática (CEA), p. 45. ISBN-13: 978-84-697-0589-6.

[7] https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/RM-MPU-6000A.pdf.

[8] https://tienda.bricogeek.com/download/RTC-0004/24C32.pdf.

[9] Limón.R. Elpais.Tecnología.2022-10-27. Recuperado de https://elpais.com/tecnologia/2022-10-27.

[10] https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf.

[11] https://sp.arduino-france.site/i2c-scanner/

[12] Chérigo, C., & Rodríguez, H. (2017). Evaluación de algoritmos de fusión de datos para estimación de la orientación de vehículos aéreos no tripulados. *I+ D Tecnológico*, *13*(2), 90-99.

Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

Julio 2024

ESCUELA POLITÉCNICA DE MIERES
UNIVERSIDAD DE OVIEDO

PLANOS

LEANDRO SANTOS GONZÁLEZ

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Mención en Recursos Energéticos

Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

PLANOS

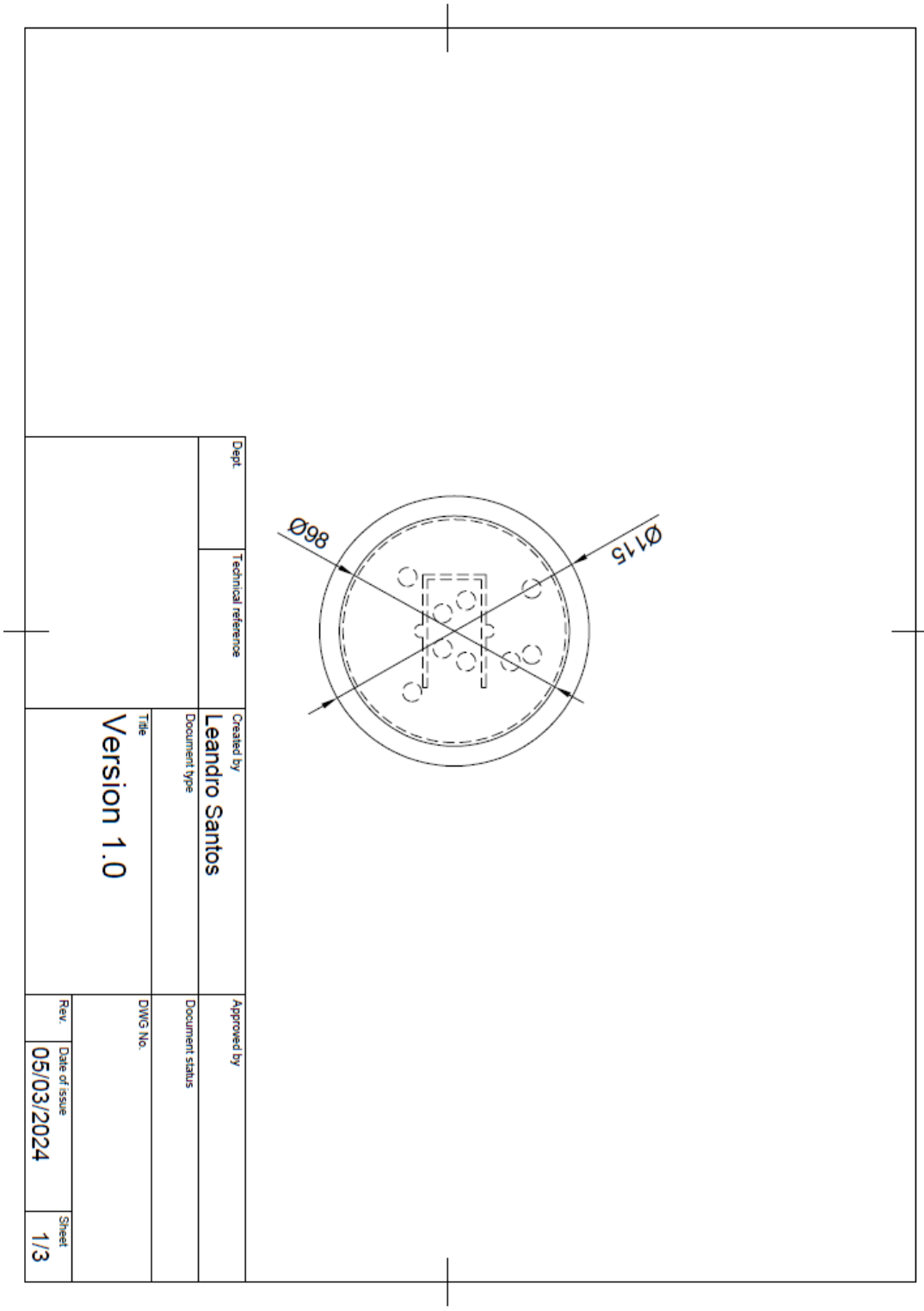
Julio, 2024

Universidad de Oviedo

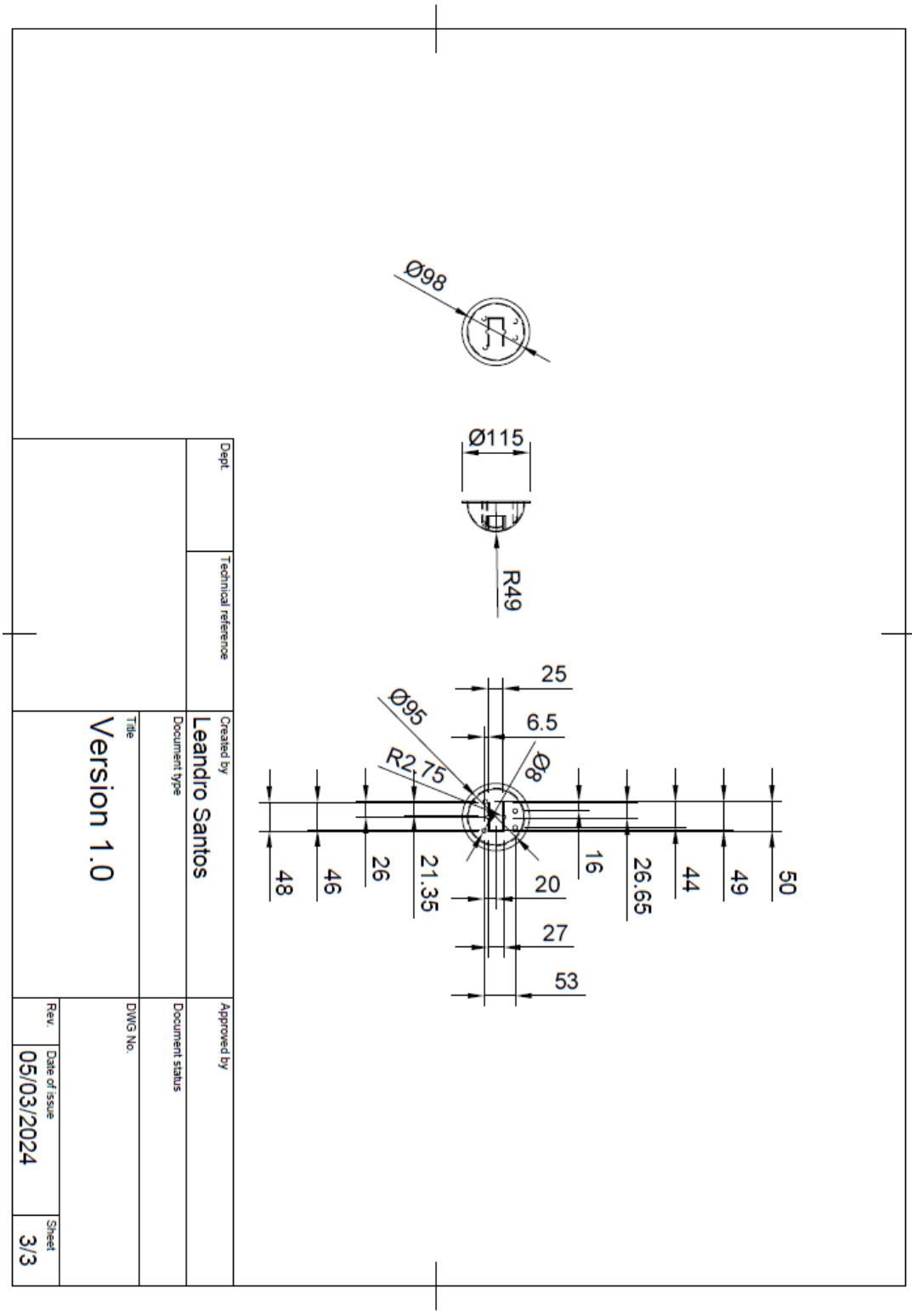
EPM ESCUELA POLITÉCNICA DE MIERES

Contenido

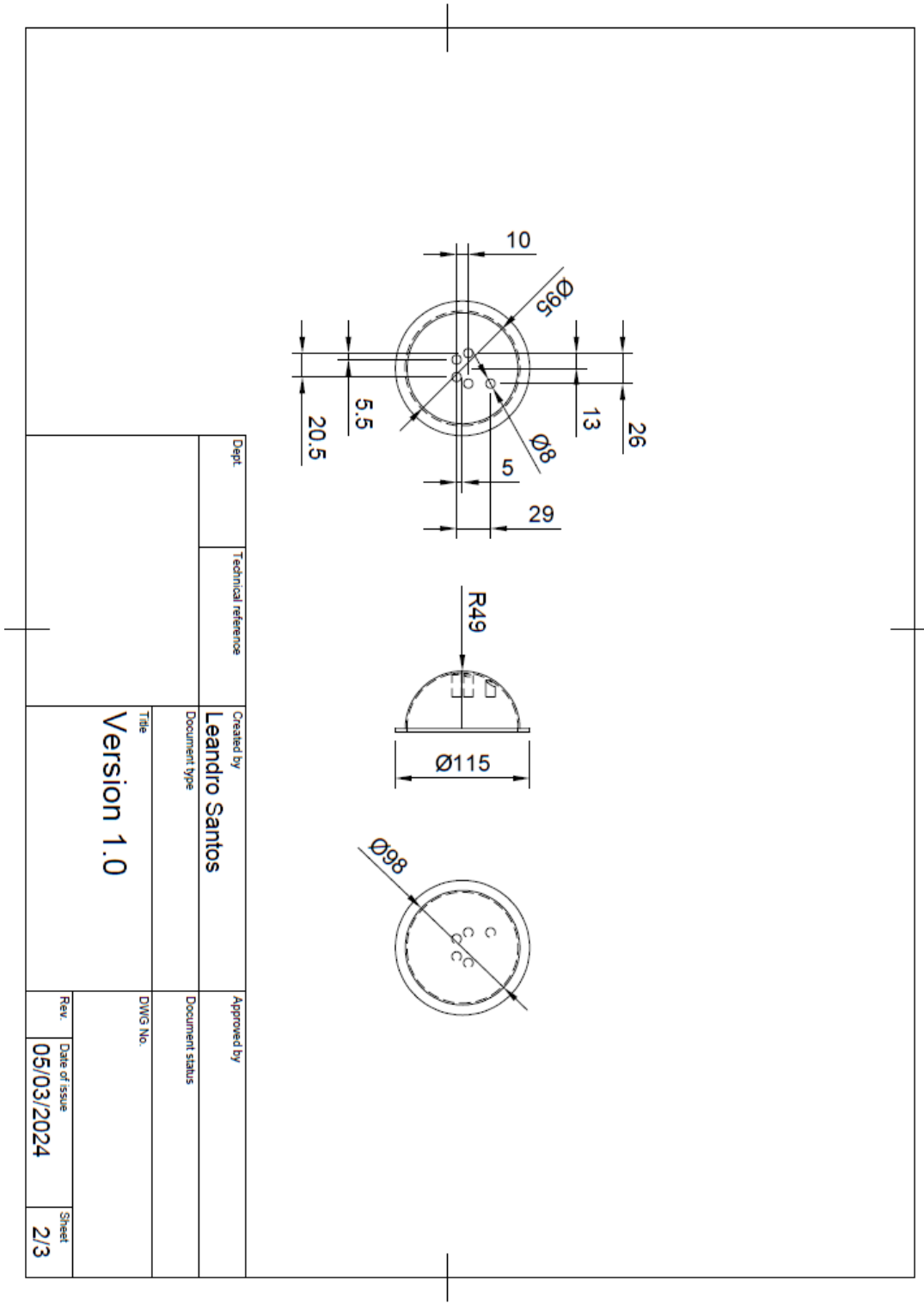
VISTA GENERAL



PLANO SEMIESFERA INFERIOR



PLANO SEMIESFERA SUPERIOR



Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

Julio 2024

ESCUELA POLITÉCNICA DE MIERES
UNIVERSIDAD DE OVIEDO

PROGRAMACIÓN

LEANDRO SANTOS GONZÁLEZ

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Universidad de Oviedo

EPM ESCUELA POLITÉCNICA DE MIERES

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Mención en Recursos Energéticos

Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

PROGRAMACIÓN

Julio, 2024

Contenido

ARDUINO

PROGRAMACIÓN PROTOTIPO

```
#include "I2Cdev.h"
#include "Wire.h"
#include "MPU6050.h"
#include "AT24Cxx.h"
#include "DS3231.h"

MPU6050 sensor (0x69);
AT24Cxx eep (0x57 , 32);

//Para la variable ax

int address=0;
//int axn

int ax, ay, az;
int gx, gy, gz;

//int axt;

char menos = "-";
char mas  = "+";

byte ax1, ax2, ax3;
byte ay1, ay2, ay3;
byte az1, az2, az3;
byte gx1, gx2, gx3;
byte gy1, gy2, gy3;
byte gz1, gz2, gz3;

char signo;

void setup() {
  // put your setup code here, to run once:
Serial.begin (9600);
Wire.begin();
sensor.initialize();
  while (!Serial){
 ;
  }
  delay(10000);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:

sensor.getAcceleration(&ax, &ay, &az);
sensor.getRotation (&gx, &gy, &gz);


Serial.print("a[x y z] g[x y z]:\t");
  Serial.print(ax); Serial.print("\t");
  Serial.print(ay); Serial.print("\t");
  Serial.print(az); Serial.print("\t");
  Serial.print(gx); Serial.print("\t");
  Serial.print(gy); Serial.print("\t");
  Serial.println(gz);

//Para la variable ax

 if(ax>=0){
  eep.write (address,'+');
 }else{
  eep.write (address,'-');
 }
 signo= eep.read(address);
Serial.println (signo);

long axt=abs(ax);
Serial.println(axt);

  if (axt>10000){
    ax1=axt/10000;
    ax2=(axt-(ax1*10000))/100;
    ax3=axt-(ax1*10000)-(ax2*100);
}
  if (axt<10000 && axt>100){
    ax1=0;
    ax2=axt/100;
    ax3=axt-(ax1*10000)-(ax2*100);
}
  if (axt<100){
    ax1=0;
    ax2=0;
    ax3=axt;
  }

 eep.write(++address, ax1);
  byte axl= eep.read (address);
```

```
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(axl);


 eep.write(++address, ax2);
  byte bxl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(bxl);

eep.write(++address, ax3);
  byte cxl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(cxl);



  //Para la variable ay

 if(ay>0){
   eep.write (++address,'+');

 }else{
   eep.write (++address,'-');
 }
 signo= eep.read(address);
Serial.println (signo);

long ayt=abs(ay);
Serial.println(ayt);

  if (ayt>10000){
    ay1=ayt/10000;
    ay2=(ayt-(ay1*10000))/100;
    ay3=ayt-(ay1*10000)-(ay2*100);
}
  if (ayt<10000 && ayt>100){
    ay1=0;
    ay2=ayt/100;
    ay3=ayt-(ay1*10000)-(ay2*100);
}
  if (ayt<100){
    ay1=0;
    ay2=0;
    ay3=ayt;
  }

 eep.write(++address, ay1);
```

```
  byte ayl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(ayl);

 eep.write(++address, ay2);
  byte byl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(byl);

 eep.write(++address, ay3);
  byte cyl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(cyl);


  //Para la variable az

 if(az>0){
   eep.write (++address,'+');

 }else{
   eep.write (++address,'-');
 }
 signo= eep.read(address);
Serial.println (signo);

long azt=abs(az);
Serial.println(azt);

  if (azt>10000){
    az1=azt/10000;
    az2=(azt-(az1*10000))/100;
    az3=azt-(az1*10000)-(az2*100);
}
  if (azt<10000 && azt>100){
    az1=0;
    az2=azt/100;
    az3=azt-(az1*10000)-(az2*100);
}
  if (azt<100){
    az1=0;
    az2=0;
    az3=azt;
  }

 eep.write(++address, az1);
  byte azl= eep.read (address);
```

```
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(azl);


 eep.write(++address, az2);
  byte bzl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(bzl);

 eep.write(++address, az3);
 byte czl= eep.read (address);
 Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(czl);


  //Para la variable gx

 if(gx>0){
   eep.write (++address,'+');

 }else{
   eep.write (++address,'-');
 }
 signo= eep.read(address);
Serial.println (signo);

long gxt=abs(gx);
Serial.println(gxt);

  if (gxt>10000){
    gx1=gxt/10000;
    gx2=(gxt-(gx1*10000))/100;
    gx3=gxt-(gx1*10000)-(gx2*100);
}
  if (gxt<10000 && gxt>100){
    gx1=0;
    gx2=gxt/100;
    gx3=gxt-(gx1*10000)-(gx2*100);
}
  if (gxt<100){
    gx1=0;
    gx2=0;
    gx3=gxt;
  }

 eep.write(++address, gx1);
  byte dxl= eep.read (address);
```

```
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(dxl);
 eep.write(++address, gx2);

  byte exl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(exl);
 eep.write(++address, gx3);
  byte fxl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(fxl);

  //Para la variable gy

 if(gy>0){
   eep.write (++address,'+');

 }else{
  eep.write (++address,'-');
 }
 signo= eep.read(address);
Serial.println (signo);

long gyt=abs(gy);
Serial.println(gyt);

  if (gyt>10000){
    gy1=gyt/10000;
    gy2=(gyt-(gy1*10000))/100;
    gy3=gyt-(gy1*10000)-(gy2*100);
}
  if (gyt<10000 && gyt>100){
    gy1=0;
    gy2=gyt/100;
    gy3=gyt-(gy1*10000)-(gy2*100);
}
  if (gyt<100){
    gy1=0;
    gy2=0;
    gy3=gyt;
  }

 eep.write(++address, gy1);
  byte dyl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(dyl);
 eep.write(++address, gy2);
```

```
  byte eyl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(eyl);

 eep.write(++address, gy3);
  byte fyl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(fyl);


  //Para la variable gz

 if(gz>0){
    eep.write (++address,'+');
 }else{
   eep.write (++address,'-');
 }
 signo= eep.read(address);
Serial.println (signo);

long gzt=abs(gz);
Serial.println(gzt);

  if (gzt>10000){
    gz1=gzt/10000;
    gz2=(gzt-(ax1*10000))/100;
    gz3=gzt-(gz1*10000)-(gz2*100);
}
  if (gzt<10000 && gzt>100){
    gz1=0;
    gz2=gzt/100;
    gz3=gzt-(gz1*10000)-(gz2*100);
}
  if (gzt<100){
    gz1=0;
    gz2=0;
    gz3=gzt;
  }

 eep.write(++address, gz1);
  byte dzl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(dzl);
 eep.write(++address, gz2);
  byte ezl= eep.read (address);
  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(ezl);
```

```
 eep.write(++address, gz3);
  byte fzl= eep.read (address);


  Serial.print ("address");Serial.print
(address);Serial.print("\t");Serial.println(fzl);



if (address < eep.length()){
  (address++);
}else{
  exit(0);
}

  delay(10);
}
```

LECTURA DE DATOS

```
#include <AT24Cxx.h>
#define i2c_address 0x57

// start reading from the first byte (address 0) of the EEPROM
int address = 0;
byte value;

// Initilaize using AT24CXX(i2c_address, size of eeprom in KB).
AT24Cxx eep(i2c_address, 32);

void setup() {
  // initialize serial and wait for port to open:

  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

}

void loop() {
  // read a byte from the current address of the EEPROM
  value = eep.read(address);

  Serial.print(address);
  Serial.print("\t");
  Serial.print(value, DEC);
  Serial.println();

  address = address + 1;
  if (address == eep.length()) {
    address = 0;
  }


  delay(500);
}
```

ENVÍO DE DATOS A EXCELL

```
#include <AT24Cxx.h>
#define i2c_address 0x57

int address = 0;
byte value;

AT24Cxx eep(i2c_address, 32);

void setup() {

  Serial.begin(9600);
  while (!Serial) {
    ;
  }
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,Address,Valor");
}

void loop() {

  value = eep.read(address);
  Serial.print("DATA,TIME,");
  Serial.print(address);
  Serial.print(",");
  Serial.print(value, DEC);
  Serial.println();

  address = address + 1;
  if (address == eep.length()) {
    address = 0;
  }



  delay(10);
}
```

Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

Julio 2024

ESCUELA POLITÉCNICA DE MIERES
UNIVERSIDAD DE OVIEDO

PRESUPUESTO

LEANDRO SANTOS GONZÁLEZ

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Universidad de Oviedo

EPM ESCUELA POLITÉCNICA DE MIERES

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE LOS RECURSOS MINEROS Y ENERGÉTICOS

Mención en Recursos Energéticos

Sistema autónomo de captación y medida de aceleraciones para la monitorización de flujo en tuberías

PRESUPUESTO

Julio, 2024

Contenido

CUADRO DE PRECIOS UNITARIOS

Cuadro de Precios nº 1			
N°	Designación	Importe	
		En cifra (Euros)	En letra (Euros)
1	Desarrollo de envolvente con software 3D e impresa con material plástico PETG, mediante tecnología de impresión 3D aditiva.	1.324,30	Mil trescientos veinticuatro con treinta céntimos
2	Desarrollo hardware. Incluyendo selección de equipos, estañado de cables para conexión entre módulos, así como la fijación de estos a la envolvente.	685,20	Seiscientos ochenta y cinco con 20 céntimos
3	Desarrollo del software de Arduino para la toma de datos del dispositivo, así como su lectura y volcado en Excel.	10.494,00	Diez mil cuatrocientos noventa y cuatro

CUADRO DE PRECIOS DESCOMPUESTOS

Cuadro de Precios nº 2			
N˚	Designación	Importe	
		Parcial (Euros)	Total (Euros)
1	Desarrollo de envolvente con software 3D e impresa con material plástico PETG, mediante tecnología de impresión 3D aditiva.		
	Mano de obra	1200	
	Materiales	1,42	
	Equipos	48	
	6% Costes Indirectos	74,88	
			1.324,30
2	Desarrollo hardware. Incluyendo selección de equipos, estañado de cables para conexión entre módulos, así como la fijación de estos a la envolvente.		
	Mano de obra	600	
	Materiales	49,20	
	6% Costes Indirectos	36	
			685,20
3	Desarrollo del software de Arduino para la toma de datos del dispositivo, así como su lectura y volcado en Excel.		
	Mano de Obra	9900	
	6% Costes Indirectos	594	
			10.494,00

CUADRO DE MANO DE OBRA

Cuadro de Mano de Obra				
		Precio (Euros)	Cantidad (Horas)	Total (Euros)
1	Ingeniero Junior	30	380	11400
			Importe total	11400

CUADRO DE MATERIALES

Cuadro de Materiales				
		Precio (Euros)	Cantidad Empleada	Total (Euros)
1	Placa Arduino uno Rev3	22,94	1	22,94
2	Módulo MPU-6050 GY-521	3,15	1	3,15
3	Módulo Memoria Ds3231/ AT24C32	10,26	1	10,26
4	Cable Impresora USB 2,0 Tipo A- Tipo B	3,94	1	4,94
5	Anycubic PETG Filamento 1,75mm 1Kg	14,21	0,1	1,42
6	Baterías de Litio Recargables de 9V, 1000mAh	7,9	1	7,90
			Importe total	50,62

ANÁLISIS PORCENTUAL DE UNIDADES DE OBRA

Análisis porcentual unidades de obra			
N°	Designación	Importe Total	%PEM
1	Desarrollo de envolvente con software 3D e impresa con material plástico PETG, mediante tecnología de impresión 3D aditiva.	1324,30	10,59
2	Desarrollo hardware. Incluyendo selección de equipos, estañado de cables para conexión entre módulos, así como la fijación de estos a la envolvente.	685,20	5,48
3	Desarrollo del software de Arduino para la toma de datos del dispositivo, así como su lectura y volcado en Excel.	10.494,00	83,93

PRESUPUESTO DE EJECUCIÓN MATERIAL

Presupuesto de ejecución material					Importe(€)
1 Desarrollo de envolvente					1.324,30
2 Desarrollo Hardware					685,20
3 Desarrollo Software					10.494,00
			Total		12.503,50
Asciende el presupuesto de ejecución material a la expresada cantidad de DOCE MIL QUINIENTOS TRES CON CINCUENTA CÉNTIMOS					