Equipo portátil conectado para medida de fuerzas

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Autor:

D. José Andrés Lorenzo Robles

Tutor:

Dr. D. Jesús Manuel Gómez de Gabriel

Cotutor:

D. Juan Manuel Gandarias Palacios

UNIVERSIDAD DE MÁLAGA

Departamento de Ingeniería de Sistemas y Automática
ESCUELA DE INGENIERÍAS INDUSTRIALES

SEPTIEMBRE 2018

Resumen

En esta memoria se presenta el desarrollo de un equipo de instrumentación portátil para la medida de fuerzas en una dirección. Además de la fuerza, se incluye un acelerómetro que permita conocer la orientación del equipo en todo momento. Finalmente se transmiten los datos leídos a un equipo externo para su posterior almacenamiento y representación. La transmisión de los datos se realiza por puerto serie a un ordenador, o con técnicas basadas en *Internet of Things*, se mandan a un servidor web vía Wi-Fi. Para su desarrollo, se describe el proceso de conexión de componentes, el diseño de una carcasa o base estructural del equipo, la programación *software*, filtros de la señal que se aplican, la calibración de los sensores y comunicación tanto por puerto serie como con distintos servidores web. Por último se realizan experimentos para comprobar qué configuración presenta mejor comportamiento para la realización de su propósito. Tanto para comprobar qué modelo del sensor presenta menor error, como para comprobar qué servidores web presenta mejores características de comunicación.

Abstract

This project shows the development of a portable instrumentation device which is able to measure forces in one direction. Apart from forces, it includes an accelerometer which allows to know the orientation of the device in every moment. Finally, the information collected from the readings of the device is transmitted to another external device, such as a computer, for further storage and graphic representation. The transmission of data is made in two ways: By means of the serial port to a computer and by means of *Internet of Things* techniques for sending the data to a web server via Wi-Fi. For its development, it is described the wiring of the components, the case or structural basis 3D designing of the device, the software programming, filters applied to the electrical signal from sensors, the calibration process of sensors and the communication between the device and other external devices, such as the computer or web servers. To sum up, some experiments are carried out to verify which configuration presents a better behavior for the execution of its final purpose. Experiments are made to verify which model of the sensor presents less error and to check which web server presents better communication characteristics.

Agradecimientos

En primer lugar quiero agradecer a toda aquella persona que con el más mínimo acto, haya contribuido a la realización de este proyecto. Por otro lado, este proyecto no habría sido posible realizarlo sin las personas que me han ayudado y apoyado a seguir en los peores momentos. Por las personas que me han aguantado día tras día con mis "problemas" y me siguen aguantando. Esas personas son mis familiares, amigos y tutores. Todos, de alguna forma u otra han contribuido enormemente en mi propósito y por ellos estoy escribiendo esta nota.

A mi familia por lo que han sufrido conmigo el proyecto y por lo que celebrarán conmigo también cuando de una vez lo termine. A mis amigos, por la paciencia y el intento de entender lo que les explico acerca del proyecto. Y a mis tutores Jesús Gómez y Juan Manuel Gandarias por su inagotable paciencia y ayuda, así como por todo lo que he aprendido con ellos.

Declaración de Autoría

Yo, José Andrés Lorenzo Robles, estudiante del Grado en Ingeniería en Tecnologías Industriales en la Escuela de Ingenierías Industriales de la Universidad de Málaga, en relación con este Trabajo de Fin de Grado, titulado: "Desarrollo de equipo de instrumentación portátil para la medida de fuerzas", declaro que asumo la originalidad de dicho trabajo, entendida en el sentido de que no se han utilizado fuentes sin citarlas debidamente.

FIRMADO: ..... FECHA: .....

ÍNDICE GENERAL

Índice de Figuras

Figura **Página**

Índice de Tablas

Tabla **Página**

1

INTRODUCCIÓN

1.1 Descripción del proyecto

Este proyecto corresponde al Trabajo de Fin de Grado (TFG) del Grado en Ingeniería en Tecnologías Industriales (GITI) de la Universidad de Málaga. Lo que se pretende con él, es facilitar la realización de experimentos en los que la magnitud principal a conocer sea la fuerza. Tanto la fuerza que puede recibir un cuerpo, antes de deformarse o partirse, como la que un cuerpo ejerce sobre otro. Del mismo modo, conocer la fuerza presente en un experimento en cada momento, permite también conocer la dinámica del mismo. De esta forma, se podría conocer si la respuesta del experimento se asemeja al comportamiento amortiguado de un muelle o bien si es rígido, etc.

Como funcionalidad extra se considera incluir un acelerómetro de forma que también se pueda conocer la orientación del equipo en cada momento. Aunque no sea tan importante como la lectura de la fuerza, se considera interesante conocer el ángulo de incidencia con el que se aplica la fuerza sobre el cuerpo.

Para darle mayor libertad al usuario del equipo se pretende que sea portátil, de modo que se puedan realizar experimentos en cualquier situación y sin necesidad de cables. En cada momento se muestran las lecturas por una pantalla integrada en el mismo. Asimismo, se desea que de forma automática y en tiempo real, se recopilen los datos medidos para posteriormente procesarlos y representarlos durante la etapa de obtención de conclusiones del experimento.

Para efectuar la recopilación de datos, se consideran varias opciones. Una de ellas es mediante la transmisión de datos por puerto serie (USB) a un ordenador. La otra opción pasa por aplicar

técnicas de *Internet of Things* dando al equipo la posibilidad de comunicarse por Wi-Fi con un servidor externo que recopila los datos que recibe. Esta última característica daría al equipo toda la versatilidad y autonomía que se pretende obtener al finalizar el proyecto.

1.2 Motivación y Justificación

En el campo de la robótica basada en la manipulación de objetos por medio de robots, o el desarrollo de equipos hápticos[1], es necesario conocer las propiedades de los cuerpos a examinar, también conocido como la caracterización de un objeto. Por ejemplo se necesita conocer cuanta fuerza admite antes de llegar a deformar o a partir. O por ejemplo conocer cuanta fuerza recibe un cuerpo en un impacto.

En la actualidad, el procedimiento general para caracterizar es mediante la colocación de cuerpos con masas conocidas sobre el objeto en cuestión. Se comprueba el efecto que produce dicha masa sobre el objeto. Finalmente se incrementa el peso hasta que se percibe la deformación límite o deseada. Esto hace que resulte muy laborioso el procedimiento dando mayor posibilidad a errores o problemas de origen humano.

Por otro lado, existen de una gran variedad de equipos de medida de diferentes magnitudes físicas, como por ejemplo equipos de medida de tensión e intensidad, como el multímetro, de presión, como el manómetro, de temperatura, velocidad, etc. La gran mayoría de estos equipos se consideran portátiles, pues suelen ser pequeños, manejables e incluso simples de usar. Sin embargo, no existe un equipo que integre en uno la posibilidad de medir fuerzas y orientación de manera activa, como pasiva, y que además recopile en tiempo real los datos del experimento de manera libre y sin cables.

Existen equipos portátiles de medida de fuerzas, pero la gran mayoría solo son capaces de leer fuerzas de tracción y compresión y si se desean almacenar los datos, es necesario que estén conectados por cable a un ordenador con un *software* de adquisición de datos. Por otro lado, existen otros dispositivos de medición de fuerzas pero tienen un diseño y utilidad muy específica, como las básculas portátiles, o los dinamómetros convencionales.

Con el equipo que se desarrolla, se permite trabajar de forma activa (fuerza ejercida sobre un cuerpo externo) o de forma pasiva (fuerza recibida por un cuerpo externo), de forma que se pueda ejercer una fuerza gradualmente mayor hasta llegar a un estado límite, o se pueda comprobar la fuerza que un proceso realiza sobre un cuerpo, que podría ser la fuerza que reciba un ser humano.

[1] Que tienen relación con el sentido del tacto

Por otro lado, con nuestro equipo de medida se puede comprobar el comportamiento dinámico de un experimento tanto en modo activo como pasivo. Mediante la curva de fuerza obtenida en el experimento se puede asemejar a un comportamiento amortiguado, elástico o rígido, etc. De este modo el equipo permite conocer más características del cuerpo y no solo la carga máxima que aguanta.

1.3 Objetivos

Este proyecto pretende ser un equipo portátil, autónomo, versátil, fiable y fácil de usar. Para ello se deben cumplir una serie de objetivos que de forma natural impone el proyecto.

Para que el equipo sea portátil, debe disponer de una batería y que se muestre el nivel de batería por pantalla. Además de esto, debe poder realizar medidas y recopilarlas en un fichero sin necesidad de ningún cable. Para que cumplir este requerimiento, es necesario que el equipo sea capaz de transmitir los datos a un servidor. Para ello debe incluirse un módulo Wi-Fi o Bluetooth que ofrezca la oportunidad de comunicarse y además conseguir recibir los datos en el servidor deseado.

Para experimentos cuya fiabilidad sea muy necesaria, se desea poder realizar una comunicación por puerto serie con un ordenador. Esto se debe a que las ventajas que ofrece una comunicación por cable, muy difícilmente las presente una comunicación sin cable y gratuita.

Por otro lado, para que se considere versátil, debe poder adaptarse a las diferentes exigencias de los experimentos. Para ello debe realizarse un diseño en 3D de una pieza envolvente que integre todos los componentes en un equipo. Además debe ser cómodo de utilizar, fácil de fijar a una superficie y resistente.

Para que se considere fiable, el equipo debe ofrecer lecturas de fuerza e inclinación con un error menor al 3% ($E_{adm} < 3\%$) en todo el rango de medidas. Para ello se deben conectar y programar adecuadamente los sensores que ofrecen estas lecturas.

Por último, para que se considere fácil de usar, debe ofrecer una navegación por el programa y una utilidad simple e intuitiva. Se deben mostrar por pantalla el estado en el que se encuentre el aparato así como las lecturas de cada sensor sin posibilidad de interpretación errónea.

Cumpliendo estos objetivos, se espera poder obtener un equipo que cumpla las necesidades que surjan durante un experimento cualquiera que requiera medir fuerzas en una dirección. Para comprobar si se han cumplido o no, se realizarán una serie de experimentos que determinen la fiabilidad del equipo ante estas exigencias.

1.4 Estructura de la Memoria

La memoria se encuentra estructurada de la siguiente manera. En el capítulo "2. Hardware" se describen los componentes electrónicos utilizados en el equipo y la conexión realizada. En el capítulo "3. Modelado e Impresión 3D" se presenta el diseño y obtención del prototipo 3D de carcasa para el equipo. En el capítulo "4. Software" se presenta todo lo relacionado con la programación de la placa. Desde la lectura de la señal de los sensores, calibración de los mismos, filtrado de la señal así como la transmisión de datos. En el capítulo "5. Experimentos y Resultados" se describen los experimentos y se presentan los respectivos resultados obtenidos. En los capítulos "6. Conclusiones" y "7. Líneas de Trabajo Futuras" se discuten los resultados, se presentan las conclusiones obtenidas y se detallan las posibles líneas de desarrollo futuras, respectivamente. Por último el capítulo "8. Anexos" se adjuntan los planos y el programa implementado en el procesador, entre otros documentos necesarios para el desarrollo del equipo.

1.5 ¿Qué es el IoT?

El término IoT corresponde a las siglas de "Internet of Things" el cual se refiere a la interconexión de distintos dispositivos electrónicos con Internet [1]. Estos dispositivos están integrados en objetos cotidianos, cosas, de forma que recopile información sobre factores cuantificables a través de sensores incluidos. Posteriormente esos datos se transmiten a una nube o servidor donde son recibidos y procesados.

Una vez en Internet, mediante una computación en la nube, se monitorizan y se podrían realizar acciones en función de los valores recibidos. De esta manera se puede conocer el estado o realizar un control automático de cualquier proceso, entre otros objetivos finales.

1.5.1 Descripción

Una utilidad muy importante de nuestro equipo es que sea capaz de poder subir las lecturas de fuerza tomadas a una nube o servidor web. Una vez recibidos en la nube se podría almacenar cada experimento en un archivo para posteriormente ser representados o procesados. De esta manera facilitaría la obtención e interpretación de las medidas o puntos de interés del experimento.

Se disponen de dos modos "Wireless" para transferir datos IoT. Dicho con otras palabras, tenemos dos formas de enviar los datos a una nube. Estas dos formas son vía Wi-Fi y vía Bluetooth que vienen integradas en el ESP-32 de serie. Dada la facilidad y versatilidad de conectarse a una red, además de ofrecer velocidades de transmisión mucho mayores (hasta 10Gbps del Wi-Fi

frente a 10Kb/s del Bluetooth BLE (*Bluetooth Low Energy.*), se decide transmitir los datos vía Wi-Fi.

El protocolo idóneo para la comunicación del equipo es mediante MQTT (*Message Queue Telemetry Transport*) enfocado sobretodo para la comunicación de pequeños dispositivos con bajos recursos, ya que resulta muy ligera.

- **¿En qué consiste el protocolo MQTT?**

  Es una arquitectura basada en una topología en estrella. El servidor o "broker" se encarga de mantener activo el canal y de gestionar los mensajes. Por otro lado, los clientes (dispositivos) mandan y reciben los mensajes mediante "topics" (temas). Los topics son los canales de comunicación y se representan a través de cadenas mediante una estructura jerárquica. Si un cliente quiere recibir unos datos en concreto, simplemente ha de suscribirse al topic correspondiente. Finalmente, si la comunicación es más amplia, pueden existir nodos donde confluyan diversos clientes situados en la misma jerarquía [2].

2

Hardware

Para cumplir con el objetivo final del equipo, es necesario el uso de componentes físicos adecuados que transformen la fuerza aplicada a unos valores numéricos e interpretables por el ser humano. Dichos componentes físicos son en esencia un sensor para la medición de fuerzas en una única dirección, otro para medir la orientación del equipo, una tarjeta de amplificación de la señal de medición y un microprocesador que procese los datos de entrada y muestre los resultados. Por último para que el equipo sesa portátil, necesita la presencia de una batería que alimente cada dispositivo.

El proceso de transformación que va desde la aplicación de una fuerza hasta obtener un valor numérico legible empieza con la deformación producida sobre nuestro sensor (célula de carga) que transforma dicha deformación en una diferencia de potencial eléctrico. Esta diferencia de potencial es muy pequeña (del orden del mV). Para adaptar la señal se necesita una etapa de adaptación donde se amplifique con una ganancia constante y previamente regulada. Para ello se utiliza la tarjeta de adquisición de datos CJMCU-333. Posteriormente la señal es recibida, filtrada, procesada y representada por un microcontrolador, que en nuestro caso es el módulo "ESP-32 TTGO Wifi & Bluetooth, 18650 Battery, 0.96 inch OLED" de la compañía Wemos. Por último será enviada, junto con los datos de orientación, a un servidor mediante técnicas de IoT.

Los datos de la orientación los proporciona el acelerómetro integrado en nuestro equipo cuyo modelo es MPU-92/95 y que son recibidos y procesados por el ESP-32.

2.1 Descripción de los componentes

2.1.1 Célula de carga

La célula de carga es un sensor basado en galgas extensiométricas pegadas sobre una estructura metálica y son éstas las verdaderas encargadas de medir la deformación que sufre la célula. Las galgas extensiométricas están conectadas formando un puente de Wheatstone en puente completo. Con esta configuración se ofrece la mayor sensibilidad posible de las galgas y resulta libre de los efectos de temperatura que puedan distorsionar la señal de salida. Nuestra célula de carga presenta una carga máxima de hasta 1 Kg, por lo que ofrece lecturas de hasta 1 Kg en ambos sentidos de la dirección de la fuerza. Es decir, es capaz de leer hasta +1 Kg, donde ofrecería la tensión de salida máxima, y -1 Kg, donde ofrecería la tensión de salida más pequeña. De este modo, el rango de trabajo presentado es de 2 Kg.

- **Fundamento teórico de las galgas extensiométricas:**

  En definitiva, la función que realizan las galgas, como se describe en el documento de *National Instruments* [3], es la de una resistencia eléctrica que varía con la deformación que es transmitida desde la estructura metálica a ellas. A partir de la fórmula genérica de resistencia eléctrica vemos que:

  $$R = \rho \frac{l}{S} \tag{2.1}$$

  Donde ρ es la resistividad eléctrica característica de cada material, l es la longitud del material por la que circulará la corriente eléctrica y S es la sección transversal del material por la que circulará dicha corriente.

  Cuando la deformación es transmitida a las galgas se produce una variación de longitud y sección, variando así su resistencia y por consiguiente la caída de tensión que se produce en ella.

De esta forma es como produce las pequeñas variaciones de tensión en mV anteriormente nombradas, que es la señal que posteriormente se procesará.

Figura 2.1: Sensor célula de carga.

2.1.2 CJMCU-333

Esta tarjeta de adquisición de datos es un amplificador de instrumentación digital. Integra el chip INA333 de Texas Instruments. Esta tarjeta trabaja a 3.3V, por lo que resulta ideal para aplicaciones de baja potencia que dependen de una batería de litio recargable, ya que su consumo siempre debe ser optimizado. Su función se basa en amplificar la señal de entrada con una ganancia constante y regulable mediante un potenciómetro. De esta manera se pasa de una señal, del orden de milivoltios (mV), a una señal de trabajo adaptada al rango de lectura del procesador, del orden de voltios (V).

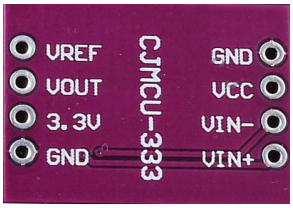

Figura 2.2: Tarjeta de adaptación de la señal CJMCU-333.

2.1.3 ESP-32 TTGO

Tarjeta que integra en sí mismo el microcontrolador ESP-32, desarrollado por la compañía Espressif Systems, con módulos Bluetooth y Wi-FI. Incluye también una pantalla OLED de 0.96" y soporte adaptado para una batería extraíble, la cual se recarga automáticamente cuando la tarjeta es conectada por micro-USB. Ofrece 32 pines de propósito general, GPIO (entrada o salida). El procesador ESP-32 integra dos convertidores analógico-digital (ADC) de 12-bit *Successive Approximation Register* (SAR) y ofrece lecturas desde 18 canales (pines analógicos). Dispone también de 2 botones (Reset y Boot) los cuales son utilizados para “resetear” nuestro equipo y navegar por el programa. Por último, trabaja a 3.3V, siendo un dispositivo de bajo consumo, ideal para una aplicación portátil alimentada por batería.

Todas estas características son ideales para el equipo y el propósito que tendrá el mismo. Además es compatible con el IDE de Arduino, facilitando aún más el trabajo en la etapa de programación.

Figura 2.3: Microcontrolador ESP-32 TTGO.

2.1.4 Sensor MPU-92/65

Se utiliza la tarjeta MPU-92/65, que corresponde a una Unidad de Medición Inercial (IMU). Ofrece 9 grados de libertad mediante 3 lecturas del acelerómetro, 3 lecturas del giroscopio y otras 3 del magnetómetro para cada eje. Internamente incorpora un MPU-9250 fabricado por Invensense de la compañía TDK, donde integra el acelerómetro y giroscopio, y aparte integra el magnetómetro AK8963 fabricado por Asahi Kasei Microdevices Corporation.

Dispone de conversores analógicos digitales (ADC) de 16 bits y los rangos de trabajo pueden ser ajustados desde ±2g, ±4g, ±8g y ±16g para el acelerómetro, ±250, ±500, ±1000 y ±2000°/s para el giroscopio y hasta ±4800μT para el magnetómetro. Además trabaja tanto a 5V como a 3V y es capaz de comunicarse tanto por I2C como por SPI.

Aclarar que aunque la IMU que se usa ofrece en un mismo dispositivo tres sensores, solo será utilizado el acelerómetro, pues los otros dos sensores no son de interés para nuestro objetivo.

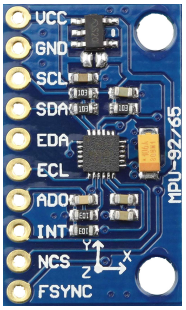

Figura 2.4: Unidad de Medición Inercial MPU9265.

2.1.5 Batería

Para darle una aplicación portátil y no necesite estar conectado por cable cuando se desee utilizar, es necesaria una fuente de alimentación interna en el equipo. Para ello se incluye una batería recargable de ion-litio 18650, fabricada por Samsung, con una tensión nominal de 3.7V y capacidad de 2600mAh. Para un uso adecuado, su rango de temperaturas de trabajo van desde -20 a 60°C. Esta batería presenta unas características muy adecuadas tanto en tensión de trabajo como por autonomía pues la tensión de trabajo es de 3.3V y presenta una autonomía para varios días en uso continuo del equipo.

Figura 2.5: Batería.

2.2 Conexionado del sistema

En la siguiente figura se representan mediante un esquema eléctrico, las conexiones realizadas para cada componente.

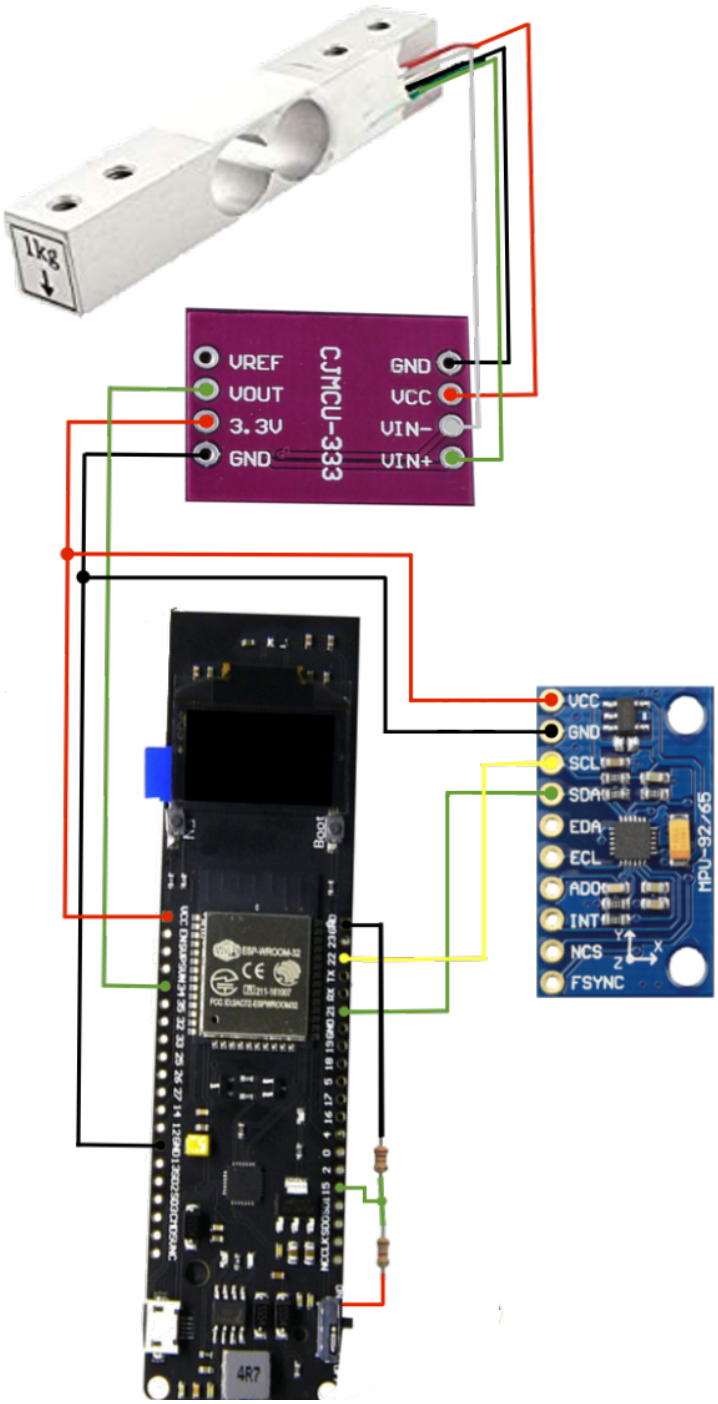

Figura 2.6: Esquema eléctrico del conexionado de los dispositivos.

A continuación se comenta más detalladamente la elección de cada conexión realizada:

Aunque existen células de carga de 6 hilos, la utilizada en nuestro caso es de 4 hilos. De estos 4 hilos, 2 son de alimentación (VCC y GND) y los otros 2 son de salida para la comunicación (VOUT+ y VOUT-). Estos 4 pines van conectados a la tarjeta de amplificación CJMCU-333. Los hilos de VCC y GND se conectan a los pines de VCC, cuya entrada es de 3.3V, y GND respectivamente. Los hilos de salida de la señal se conectan a los pines VIN+ y VIN- respectivamente.

La tarjeta de adquisición de datos presenta 8 pines. De estos pines, 2 corresponden a la alimentación de la tarjeta (3.3V y GND), otros 2 pertenecen a la alimentación del sensor (VCC y GND), otros 2 pines corresponden con la señal de entrada del sensor (VIN+ y VIN-), 1 pin para la salida de la señal amplificada (VOUT) y 1 pin de entrada para la regulación del *offset*[1] (VREF).

Los pines de 3.3V y GND se conectan a los pines de alimentación VCC y GND del ESP-32, respectivamente. VOUT es conectado a un pin GPIO cualquiera del ESP-32 de donde se leerá la señal del sensor. Se toma el pin 34 por su cercanía. El pin VREF no se utilizará pues no es necesaria para nuestra aplicación.

Por otro lado, el IMU utilizado dispone de 10 pines de conexión. De ellos, 2 pines son de alimentación (VCC y GND), 4 pines para la comunicación tanto por SPI como por I^2C (SDA, SCL, EDA, ECL, ADO e INT), 1 pin para elegir dispositivos en caso de haber varios conectados a un microcontrolador por SPI y 1 pin para distintos usos en función de la configuración dada (FSYNC). En nuestro caso, el IMU se comunica por I^2C y solo se necesitan conectar los pines de alimentación junto con los pines de alimentación, SDA y SCL. Los pines VCC y GND se conectan a sus correspondientes pines en el ESP-32 (VCC y GND) y los pines SDA (línea de transmisión de datos) y SCL (linea de transmisión para la señal de reloj) se conectan a los pines 21 y 22 respectivamente, que son los habilitados para la comunicación I^2C.

Por último, para poder conocer el nivel de batería en cada momento, se realiza un divisor de tensión desde el pin ON del interruptor hasta un pin GPIO cualquiera, en este caso el pin 15 por su cercanía. El interruptor presenta 3 pines, uno en ON, otro en OFF y un último que cortocircuita con el pin del estado presente. Se pueden apreciar estos pines en la imagen 2.7:

Figura 2.7: Detalle del interruptor del ESP-32.

[1]Valor de la componente de continua o desplazamiento vertical de la señal desde el origen a cero voltios.

Se dan dos situaciones:

1. Interruptor en OFF: La tensión del pin OFF es de 5.14V mientras que la tensión del pin ON es de 0.1V.

2. Interruptor en ON: La tensión del pin OFF es de 0.2V y la tensión del pin ON es de 5.05V.

Tomando la tensión de entrada al divisor desde el pin ON y no desde otro, o desde la batería directamente, aseguramos que con el equipo apagado el divisor de tensión no seguirá consumiendo corriente de la batería.

Asimismo, se tiene el divisor de tensión:

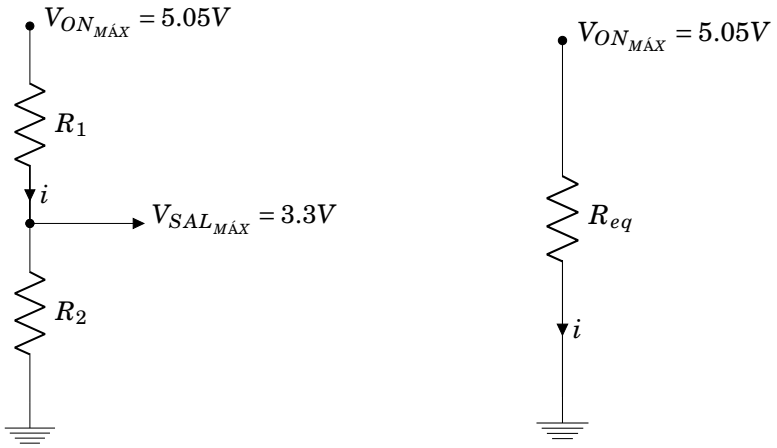

Figura a) Figura b)

Figura 2.8: Esquema del divisor de tensión a) y divisor de tensión equivalente b)

De la figura 2.8 b) se deduce que:

$$i = \frac{V_{ON_{MÁX}}}{Req} = \frac{V_{ON_{MÁX}}}{R_1 + R_2} \tag{2.2}$$

De esta forma, se obtiene la expresión del divisor de tensión para la figura 2.8 a):

$$V_{SAL_{MÁX}} = i \cdot R_2 \rightarrow V_{SAL_{MÁX}} = \frac{V_{ON_{MÁX}}}{R_1 + R_2} \cdot R_2 \tag{2.3}$$

Lo que se sabe es que V_{ON} siempre será menor que 5.05V y debemos calcular R_1 y R_2 para que V_{SAL} sea menor que 3.3V. Se decide probar con $R_1 = 1.8\text{K}\Omega$ y $R_2 = 2.7\text{K}\Omega$. Para estos valores, la tensión de salida del divisor de tensión es:

$$V_{SAL_{MÁX}} = \frac{5.05}{1800 + 2700} \cdot 2700 = 3.03V < 3.3V \tag{2.4}$$

Se observa que la tensión de entrada al microcontrolador es suficientemente grande sin superar el valor límite, por lo que definitivamente se toman estos valores para el divisor de tensión $R_1 = 1.8\text{K}\Omega$ y $R_2 = 2.7\text{K}\Omega$.

3

MODELADO E IMPRESIÓN 3D

En este capítulo se describe el proceso de obtención de un prototipo de carcasa para nuestro equipo de medida. En él se explica como se ha realizado el modelo en 3D de la pieza con técnicas CAD y las decisiones tomadas para su elaboración. Se describe cómo finalmente se pasa de un modelo 3D en un software a un objeto físico, mediante técnicas CAM, y qué equipos y materiales se utilizan. Por último se especifica la disposición de todos los componentes hardware ensamblados.

Puesto que el equipo de instrumentación está diseñado para ser portátil, es necesario que todos los componentes utilizados estén fijos sobre una base estructural. Esto permite que pueda ser transportado y no existan movimientos relativos entre componentes. Además debe ser cómodo y fácil de utilizar. De esta manera, el soporte físico debe ser diseñado y fabricado para soportar una serie de condiciones de uso y permitir una serie de condiciones de funcionalidad. Por ello se desea que el equipo esté abierto a diversos tipos de aplicaciones, intentando hacerlo lo más versátil posible.

En un principio, se diseña el equipo teniendo en cuenta que se debe de poder utilizar tanto de forma portátil, o libre, como fijada sobre una superficie. De esta manera podrá medir tanto si realiza fuerzas a un objeto, como si es el equipo el que recibe las fuerzas externas.

Antes de profundizar más, aclarar que el diseño 3D que se realiza de la carcasa no es más que un prototipo y no un diseño cerrado. Por ello dejar claro que está abierto a los rediseños y ampliaciones que se estimen oportunas. Este diseño no es más que una solución de las muchas posibles y se debe tomar como punto de partida para un futuro rediseño.

3.1 Consideraciones para el diseño

Para el diseño de la pieza se han tomado una serie de consideraciones en base al uso que se le espera dar en un futuro, adaptando el diseño a estas. Dichas consideraciones son:

- Todos los componentes deben ir completamente fijados para que no existan movimientos relativos entre ellos, sobretodo el acelerómetro pues será calibrado para una posición exacta.
- El equipo debe ser en la medida de lo posible ergonómico y manejable, pero a su vez podrá ser fijarlo sobre una superficie plana y estable.
- Para evitar posibles influencias de la carcasa hacia el sensor y que lo toque lo mínimo posible, este debe estar colocado en voladizo y únicamente fijado al equipo por los tornillos.
- La pieza no deberá de dejar inutilizables tanto los botones, como el interruptor ON-OFF, la entrada micro-USB o la pantalla OLED integrada en la tarjeta.

3.1.1 Modelado en 3D

En base a las consideraciones anteriores, se comienza el modelado mediante un software CAD (*Computer-Aided Design*) para posteriormente imprimirlo en 3D. El software CAD utilizado es *SolidWorks*, desarrollado por *SolidWorks Corp.*

El procedimiento realizado durante el diseño en SolidWorks, se basa en el croquizado y extrusión de tal croquis, tal y como se explica en *El Gran Libro de SolidWorks* [4]. Aplicando este procedimiento para distintas partes del diseño se obtiene la pieza final. Finalmente se obtiene el diseño que se muestra en las figuras 3.1 y 3.2:

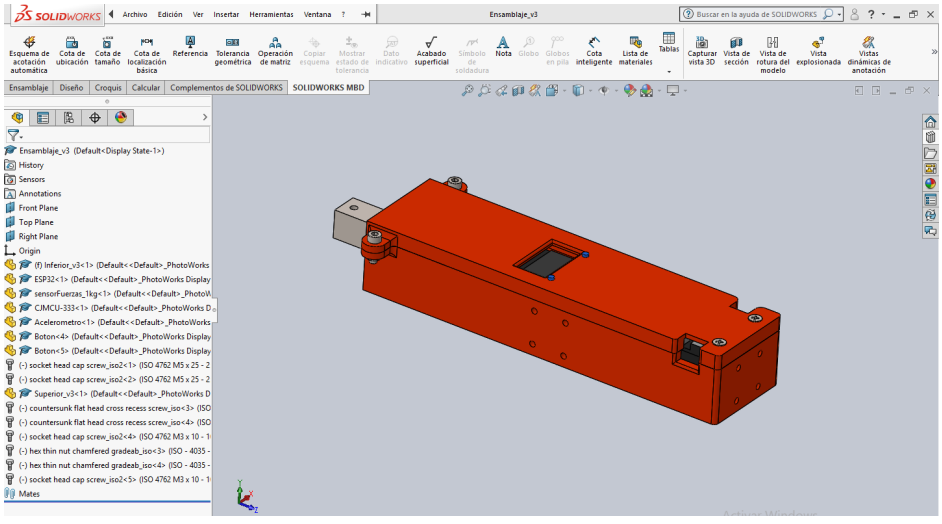

Figura 3.1: Imagen renderizada del prototipo de carcasa para el equipo.

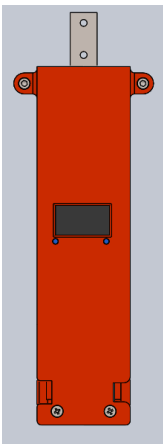

Figura 3.2: Vista en planta.

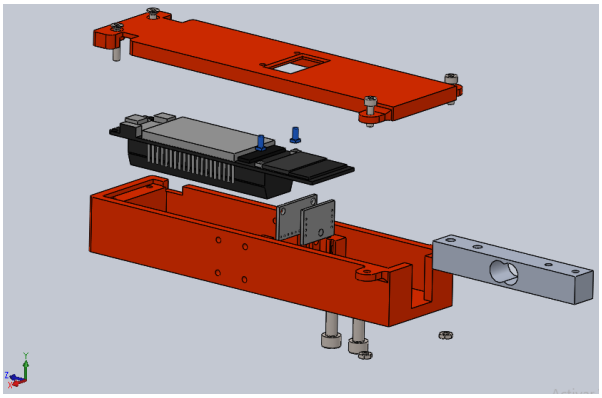

Figura 3.3: Vista explosionada del conjunto.

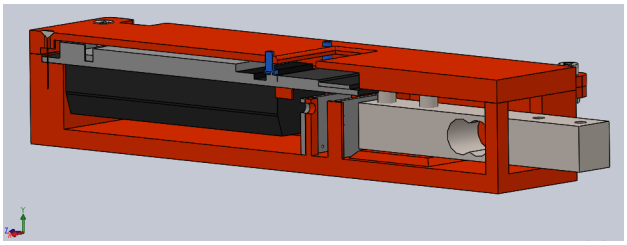

Figura 3.4: Vista seccionada del conjunto.

Para realizar de forma más fácil el diseño y asegurar que posteriormente todas las piezas encajan, inicialmente se han diseñado todos los componentes descritos en el capítulo "2. Hardware" (sensores y procesador) que son los que van integrados en el equipo. Finalmente se han diseñado las piezas de la carcasa en base a los componentes ya presentes en SolidWorks.

Referente a la carcasa, se observa que ha sido diseñada tomando piezas por separado. Se ha diseñado una pieza inferior donde irán los componentes ensamblados, otra pieza superior que cierre el conjunto y por último unos botones que hagan accesibles desde fuera los pulsadores integrados en el controlador.

A continuación se muestra por separado cada elemento diseñado:

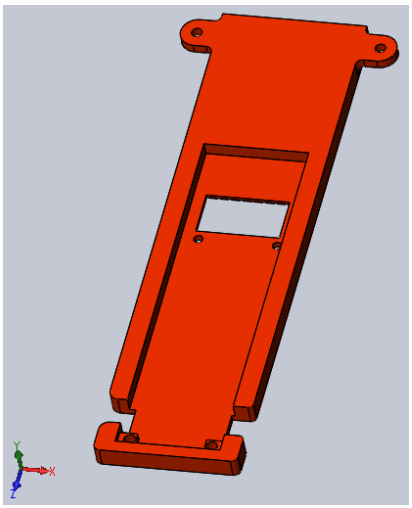

Figura 3.5: Diseño de pieza superior vista trasera.

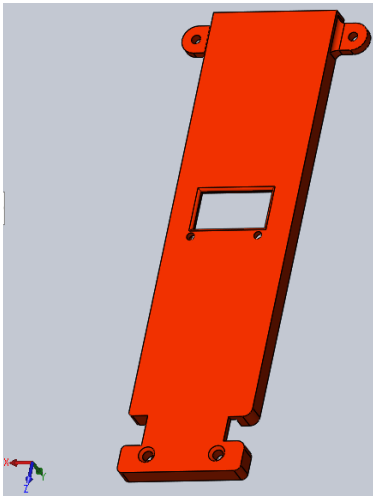

Figura 3.6: Diseño de pieza superior vista delantera.

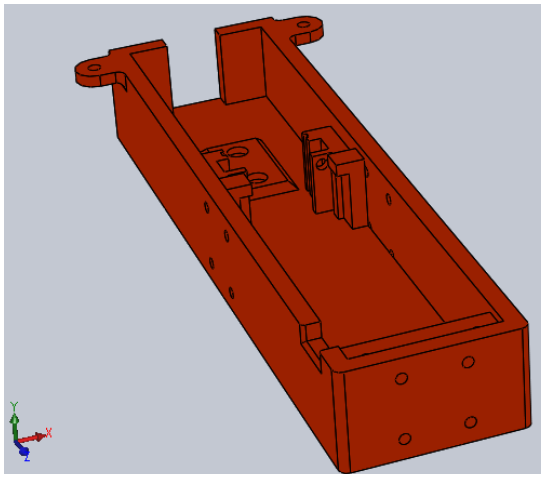

Figura 3.7: Diseño de pieza inferior vista superior.

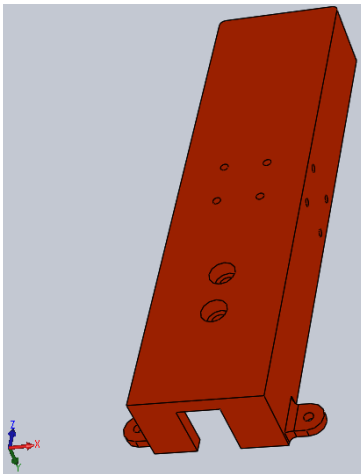

Figura 3.8: Diseño de pieza inferior vista inferior.

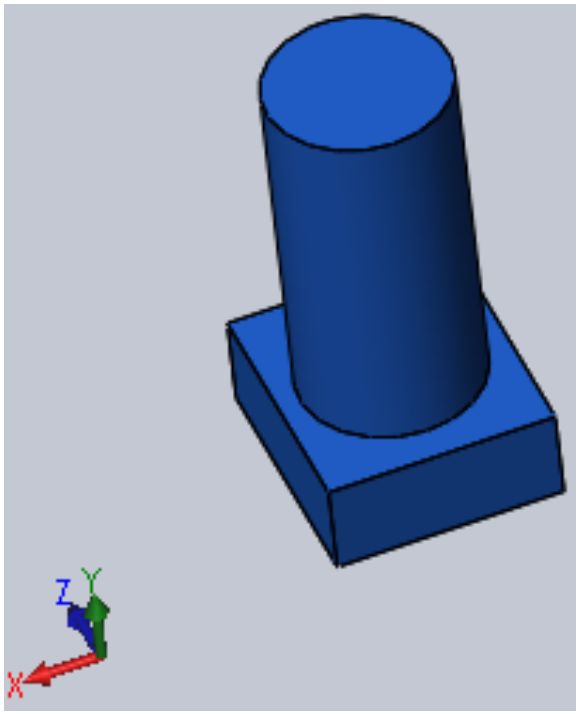

Figura 3.9: Diseño 3D del botón vista superior.

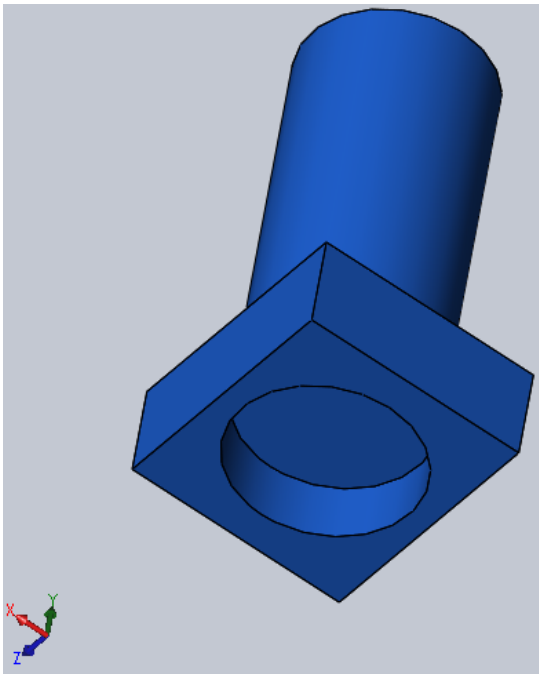

Figura 3.10: Diseño 3D del botón vista inferior.

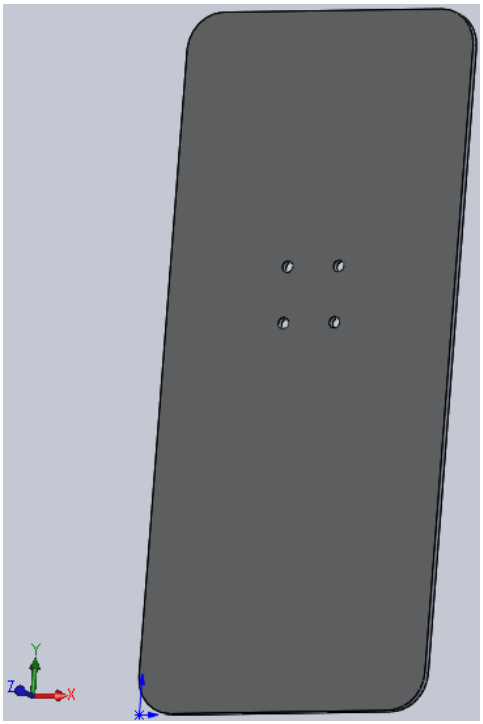

Figura 3.11: Diseño en 3D de soporte.

3.1.2 Impresión 3D

Una vez terminado el modelo de la pieza, se imprime en 3D en una impresora modelo *Sigma* de la compañía *BCN3D* presente en el Taller 0.27II de la Escuela de Ingenierías Industriales de la Universidad de Málaga. La impresora utilizada emplea tecnología *Fused Deposition Modeling* (FDM) y fue impresa en PLA (Poliácido Láctico).

Se elige el PLA para la impresión ya que es uno de los materiales más adecuados para la impresión por FDM. Es un material termoplástico que presenta unas buenas propiedades mecánicas ya que es ligero (ρ=1.25 g/cm^3), rígido (E=3.5 GPa) y resistente (Relación fuerza-peso=40 KN·m/Kg). Con estas características se espera obtener unas piezas lo suficientemente rígidas y resistentes para aguantar el uso que se espera que tenga el equipo.

Para efectuar la impresión se almacenan los modelos realizados en el formato ".STL" y se carga en el software CAM (*Computer-Aided Manufacturing*) Cura. Este programa se encarga de establecer las propiedades de impresión (relleno, disposición de las piezas, velocidad de impresión, soportes para voladizos,etc.) y generar el código G para que el extrusor realice el movimiento de forma correcta, de manera que capa por capa se genere el objeto deseado.

Una vez generado el código de movimiento, en un documento de texto plano ".gcode", se carga en la impresora. Automáticamente comienza a generar las piezas tal y como se ha especificado en el software Cura. A continuación se muestran unas imágenes de la elaboración de uno de los varios modelos que se han impreso.

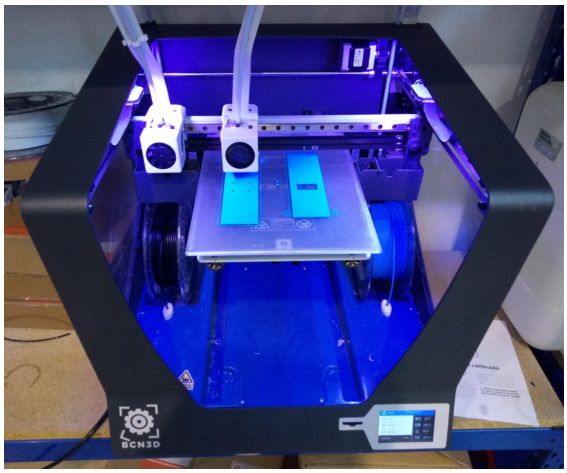

Figura 3.12: Impresión de un modelo de carcasa.

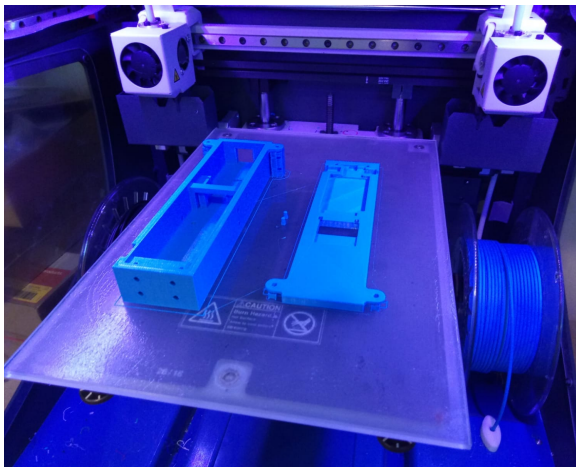

Figura 3.13: Impresión del modelo una vez finalizada.

Aunque las imágenes no corresponden con el último modelo impreso, se puede observar en la figura 3.12 como trabaja la impresora. Para elaborar el diseño deseado rellena una capa sobre otra hasta llegar a la figura 3.13, donde ha terminado de imprimir. También se pueden apreciar los soportes que se imprimen para las orejeras en voladizo.

El motivo de los soportes es debido a que la tecnología FDM deposita el filamento caliente sobre una capa anterior y consistente de material. Sin embargo, en los elementos en voladizo no existe tal capa, por lo que de no haber apoyos el material caería al suelo o el voladizo no estaría recto. Para ello, en cada capa se rellena, con mucho menos material, ese espacio sirviendo de soporte al voladizo que queremos.

4

Software

En esta sección se describe el proceso de programación del microcontrolador ESP-32. Una vez realizado el ensamblaje y con todos los componentes correctamente conectados se puede proceder a la programación software. En este proceso se transforman las medidas recibidas desde los diferentes sensores, en voltios, a valores numéricos de fuerza en newtons (N) y orientación en grados sexagesimales (^{o}). Posteriormente se muestran dichas medidas por pantalla y son enviadas a una nube o servidor web. Una vez transferidas, se almacenan para su posterior representación gráfica o procesado.

El proceso de programación software definitivo se desarrolla tras la calibración de los sensores de fuerza y acelerómetro. Durante la calibración también es necesaria la realización de un pequeño programa para leer y representar los valores de entrada al microcontrolador. Una vez calibrados los sensores, se aplican filtros a la señal para atenuar el ruido. Por último se realiza el programa final donde se integran todas las partes que realizan lo descrito en el párrafo anterior.

Toda la programación del microcontrolador se realiza a través del entorno de desarrollo de software libre de Arduino, Arduino IDE (*Integrated Development Environment.*). Este software es compatible con nuestro procesador mediante una serie de configuraciones previas para adaptarlo.

En definitiva, en esta sección se describe como se toman, transforman y representan las medidas. Además se explica como se comunica, mediante técnicas IoT, nuestro equipo con los servidores con los que se baraja la transmisión de datos.

4.1 Lectura de datos

Los datos procedentes de la célula de carga no son recibidos de la misma manera que los procedentes del acelerómetro. El motivo es que la célula de carga y el acelerómetro utilizan distintos protocolos de comunicación con el microcontrolador. Por ello, la programación de las lecturas de los datos se realiza de distinta manera.

4.1.1 Lectura analógica

La lectura de la célula de carga se realiza mediante una lectura analógica del pin al que se encuentra conectado el circuito de adaptación de la señal dada por la célula de carga. Dicho pin es concretamente el GPIO34, que corresponde con la entrada analógica ADC6. Para ello, se declara dicho pin como un pin de entrada en la función "setup()"[1], del programa de Arduino. Seguidamente, en la función "loop()"[2], se realiza la lectura del pin de entrada y se guarda su valor en una variable.

Lo mismo ocurre para la lectura del nivel de la batería a traves del divisor de tensión. La salida de éste se conecta al GPIO15, que corresponde con la entrada ADC13. Como su lectura es a través de un bus analógico, se procede de la misma manera que para la célula de carga.

4.1.2 Lectura por I^2C

Puesto que el acelerómetro es un sensor digital y se comunica por I^2C, cuyo protocolo permite comunicar el microcontrolador con diversos periféricos desde un mismo puerto, su comunicación se debe realizar de forma distinta a la lectura analógica. Para ello, es necesario seguir una serie de pasos para su correcta comunicación

1. Puesto que puede haber diversos periféricos conectados a un mismo puerto se debe declarar la dirección de cada uno de ellos. Además se debe declarar la escala a la que trabaja el sensor.

2. Llegado el momento de comunicarnos con un periférico en concreto, el dispositivo maestro debe informar al esclavo que desea recibir sus datos. Para ello, el dispositivo maestro manda por el bus serie un byte. Este byte contiene 7 bits con la dirección del dispositivo deseado, junto con un último bit cuyo valor puede ser 0 para escritura o 1 para lectura al esclavo. En ese momento, cada dispositivo compara la dirección recibida con su propia dirección. Si coincide, el dispositivo esclavo manda un bit[3] de retorno donde indica que reconoce la solicitud y está en condiciones de comunicarse. Entonces comienza la comunicación entre

[1] Función de inicialización de variables, pines o librerías. Es llamada al iniciar el programa por primera vez.
[2] Función que se ejecuta en bucle durante el tiempo de funcionamiento del programa.
[3] El bit de reconocimiento es mandado cada vez que se reciba un byte desde el otro dispositivo, independientemente si es esclavo o maestro.

los dispositivos. Este proceso se realiza mediante una función de escritura en el dispositivo maestro, en este caso el ESP-32.

3. Una vez enlazados los dispositivos, el dispositivo maestro comienza a leer los datos transmitidos por el esclavo. La lectura recibida para cada instante se almacena en una variable tipo array. Cada posición del array contiene 8 bits con las lecturas de los ejes del espacio. Puesto que el IMU tiene un registro de 16 bits, a cada eje corresponde 2 posiciones del array. Finalmente para obtener la lectura completa de la aceleración en cada eje, se concatenan 2 posiciones del array por cada eje. Hay que tener en cuenta que estas lecturas son datos sin tratar, por lo que necesitan ser posteriormente procesados.

Previamente habría que haber incluido la librería "Wire.h" en Arduino la cual ofrece instrucciones para la inicialización, lectura y escritura con dispositivos por I^2C.

4.2 Calibración

En este capítulo se describe cómo se ha desarrollado la calibración de los distintos sensores integrados en nuestro equipo de medida. En general, la calibración es un procedimiento que relaciona un valor eléctrico con un valor mecánico. El objetivo de este proceso es que las lecturas realizadas sean coherentes y se aproximen lo máximo posible a los valores reales tanto de fuerza como de aceleración.

Hay que aclarar que para la lecturas de fuerza, como el sensor es analógico y la magnitud de entrada ses de voltios, para poder medirla y trabajar con ella, el procesador lo convierte a un valor discreto. Esto lo hace a través del conversor analógico-digital(ADC) de 12 bits que integra este, por lo que las lecturas se convierten en niveles de 1 a 2^{12} bits, o bien, de 0 a 4095 bits. De este modo, la calibración se hace para pasar de valores en bits a fuerzas en newtons. Igualmente, existe una relación lineal entre la tensión en voltios y su valor en bits por lo que lo que se realice en bits es extrapolable a voltios.

El proceso de calibración realizado para ambos sensores consiste en la toma de datos para distintas condiciones de estos. Posteriormente se compara el valor de salida respecto al valor real de entrada o "ideal". Cuando se comparan los datos leídos respecto a los reales, para una cantidad considerable de condiciones, obtenemos el comportamiento del sensor en su rango de trabajo. De esta manera existe la posibilidad de aproximar dicho comportamiento a un modelo matemático que, aunque pueda presentar un cierto error, deberá ser más pequeño que un cierto error que será considerado admisible.

4.2.1 Calibración de la célula de carga

4.2.1.1 Descripción del procedimiento

Respecto a la calibración de la célula de carga, se realiza el procedimiento anteriormente descrito. Dado que su carga máxima es de 1Kg y presenta lecturas para ambos sentidos de aplicación de la fuerza, su rango de trabajo es de ±1Kg.

Lo que queremos medir es fuerza en newtons y disponemos de medidas en voltios. Asimismo sabemos que la fuerza que genera cierta masa, según la Segunda Ley de Newton dada por $\sum F = m \cdot a$, es proporcional a la aceleración. Para las condiciones de uso del equipo, dicha aceleración es igual a la gravedad en la superficie de la Tierra, 9.80665 m/s^2. De esta forma haciendo una calibración del sensor para masas, podemos pasar fácilmente a la magnitud de fuerzas.

Idealmente, se podría aproximar el comportamiento de la célula de carga a una regresión lineal para la que solo serían necesarias 2 medidas con las que conformar una recta. Puesto que en la realidad no se comporta como tal, para reducir el error y aproximar en todo lo posible el modelo a su comportamiento real se tomarán más medidas de las 2 estrictamente necesarias. De esta manera se obtiene un sistema "sobredimensionado" pero lógicamente mucho más fiable.

Además, para reducir en todo lo posible el error en el modelo matemático, se toman distintas consideraciones iniciales:

- La fuerza siempre se aplica sobre el mismo punto y se incrementará de manera proporcional.
- Se toman 20 medidas para cada zona de trabajo (cada sentido de la fuerza) que es equivalente a una medida cada 50g.
- Para detectar un posible efecto de histéresis sobre nuestro sensor, se distingue entre las medidas con incremento de peso de las medidas con decremento de peso. Por lo que habrá 2 medidas para cada peso.
- Se calibran por separado ambos sentidos de aplicación de la fuerza (zona de trabajo positiva y negativa) ya que pueden tener distinta respuesta. Para ello, se girará el equipo 180° respecto a su eje longitudinal cuando se calibre el sensor para masas negativas.
- La calibración se realizará tomando medidas con una precisión al menos 20 veces superior a la de nuestro equipo.

De este modo, tomando 2 veces 20 puntos para las 2 zonas de trabajo, se obtiene una calibración de 80 puntos en los que en cada uno se tomarán varias medidas. Con ese número de datos se estima que es suficiente para determinar el comportamiento de la célula de carga.

Para realizar la calibración se necesitan una serie de masas que permitan realizar medidas cada 50g de una manera constante. Para ello se han utilizado un juego de masas o pesas de calibración que ofrecen medidas de hasta 1Kg con un paso mínimo de 10g. Además presenta un error de ± 0.03g, menor de 20 veces el error del equipo, que será de 1g. Asimismo, para que el peso se aplique siempre respecto al mismo punto, es necesaria una plataforma donde situar las pesas de forma estática. Para ello se ha fabricado un plato de madera sobre el que se coloquen las masas y sea colgado en el sensor.

Figura 4.1: Pesas utilizadas para la calibración.

Figura 4.2: Plato donde colocar las masas durante la calibración.

Para la obtención de las lecturas se realiza un pequeño programa en el entorno de programacion de Arduino IDE. En el programa se toma una lectura analógica del pin de entrada del sensor de fuerza (o célula de carga). Seguidamente se observa la salida por el "Monitor Serie"[4] o por el "Serial Plotter"[5], dependiendo en cada momento de lo que más convenga.

Una vez representada la señal obtenida, se observa que la lectura oscila entorno a un valor sin un patrón definido, dificultando la lectura del valor real de la medida. Dicha oscilación es producida por el ruido inherente en la señal, que puede proceder de diversas fuentes (como por ejemplo las imperfecciones del propio sensor, las soldaduras frías, los conectores, posibles pequeñas oscilaciones del plato de medida, etc). Se observa que la medida varía en un rango de 10 gramos con algunos picos que alcanzan 50 gramos sobre la medida. Ante esto, se decide no tomar un solo valor por cada medida, pues la incertidumbre sería muy grande, y se toman 50

[4]Herramienta integrada en el entorno de desarrollo de Arduino que presenta el valor numérico de la variable medida en cada momento de lectura del procesador.

[5]Herramienta integrada en el entorno de desarrollo de Arduino que presenta la representación gráfica de la variable medida respecto del tiempo.

medidas para cada peso, de esta manera el error se debería compensar y el valor medio de las medidas aproximarse al valor real.

4.2.1.2 Recopilación de datos

Las 100 medidas (incrementando y decrementando) para cada uno de los 40 pesos (en ambas zonas) han sido recopiladas en un archivo Excel y cargadas en un programa de MATLAB donde se representan los pesos aplicados en gramos respecto a las lecturas en voltios. De esta manera obtenemos la siguiente gráfica:

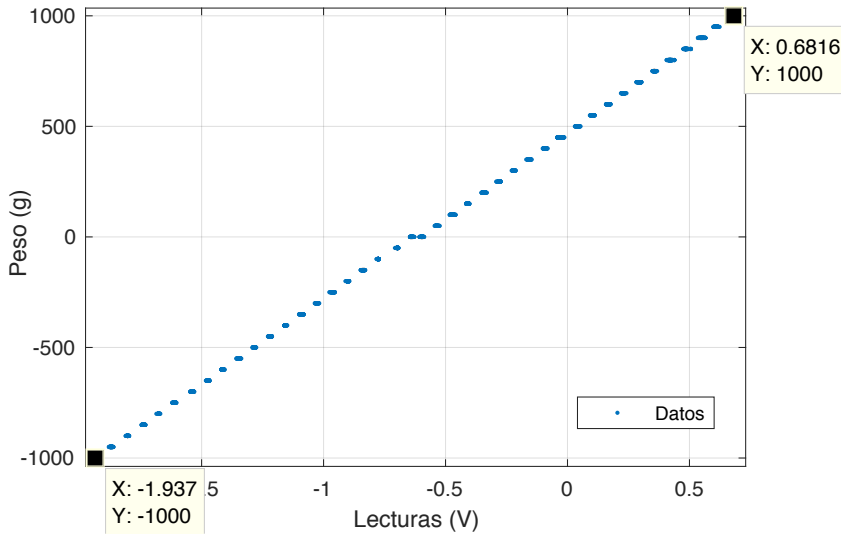

Figura 4.3: Representación de los datos obtenidos tras la calibración para los distintos pesos aplicados.

De la gráfica se deduce el comportamiento del sensor del cual se pueden comentar varios aspectos clave:

- Como se puede apreciar, el valor del peso máximo no corresponde con el valor de la lectura máxima, 3.3V, al igual que el valor de peso mínimo no corresponde con la lectura de -3.3V. El valor máximo corresponde con 0.7V aproximadamente, el valor mínimo con -1.9V, por lo que su rango de lectura es de 2.6 voltios aproximadamente.
  Esta discordancia es debida a que la ganancia proporcional de la tarjeta de adquisición y adaptación de la señal (CJMCU-333) no es lo suficientemente grande como para abarcar todo el rango de lectura del ESP-32 (-3.3 a 3.3V) con el rango de trabajo (-1 a 1Kg). De esta manera, podría seguir leyendo para entradas superiores (e inferiores) a 1Kg hasta que sature el sensor o la electrónica de amplificación de señal. Sin embargo, no se asegura ni la linealidad del sensor ni la precisión de la medida, por lo que nos centramos en el rango de trabajo que nos asegura el fabricante.

- Se puede observar que el sensor no presenta histéresis, pues las medidas se realizaron inicialmente para incrementos de peso y una vez llegado al máximo, se fue decrementando hasta llegar a peso nulo. De esta forma, si el sensor presentase histéresis habría una diferencia considerable entre incrementos y decrementos. Pero no existe tal diferencia.
- El sensor se comporta de forma muy lineal en todo su rango de trabajo. Para todo su rango no hay cambio de pendiente o un comportamiento no lineal, aproximándose más a una curva que a una línea recta.
- Aunque es muy pequeña, se puede apreciar la existencia de una zona muerta en torno al 0. Dicha zona muerta afecta a 0.056V, que corresponde a un 2.17% de toda la zona de trabajo. Aunque sea muy pequeña dicha zona muerta puede distorsionar bastante la medida para pequeñas fuerzas. Para evitar dicha distorsión, se estudia calcular un modelo matemático donde se diferencie la zona positiva y de la zona negativa.

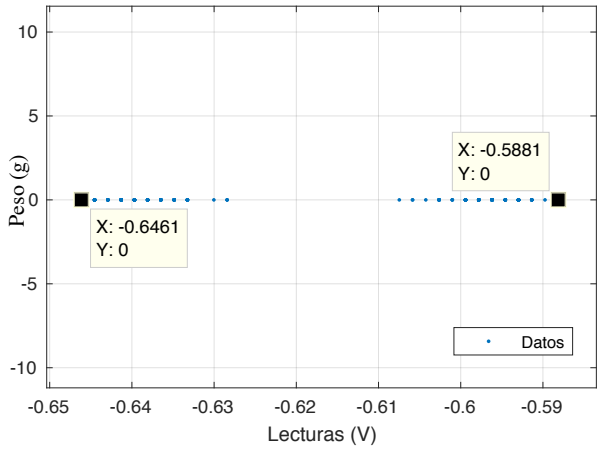

Figura 4.4: Ampliación de la zona muerta de la calibración del sensor.

4.2.1.3 Obtención del modelo matemático

Una vez recopilados los datos y estimado el comportamiento del sensor, estamos en condiciones de poder obtener un modelo matemático que se asemeje lo máximo posible a dicho comportamiento, para obtener así la relación existente entre voltios y gramos.

Para su obtención se elabora un pequeño programa en MATLAB donde se ofrecen una serie de herramientas matemáticas que facilitan enormemente el trabajo. Por otro lado, MATLAB dispone de diversas maneras de obtener un modelo matemático basado en una nube de puntos. En este caso se ha optado por utilizar la función perteneciente a la *Statistics and Machine Learning Toolbox*, "fitlm(X,Y,MODELSPEC)", la cual tiene como entradas los valores X e Y, que son las variables independiente y dependiente respectivamente de la nube de puntos. MODELSPEC representa la forma del modelo deseado, que puede ir desde lineal hasta polinómica o una forma

concreta que deseemos. Su salida será una serie de coeficientes que conforman la ecuación del modelo resultante. Además, dicho comando también muestra su coeficiente de correlación de Pearson, R^2, el cual muestra la relación lineal entre 2 variables cualitativas aleatorias.

4.2.1.4 Regresión lineal para la nube de puntos

Comenzamos realizando un modelo basado en una regresión lineal para todas las medidas realizadas, es decir los 80 puntos con los 50 datos tomados para cada uno. Aunque anteriormente se dice que se considerarían por separado las zonas positiva y negativa, no se quiere descartar esta opción. Si ofrece buenos resultados, es mejor desde el punto de vista computacional, pues no habría que diferenciar entre zonas de trabajo. Más adelante se examinará y se comprobará si es apta.

Tras cargar en MATLAB las variables independiente (X, lecturas en voltios) y dependiente (Y, peso en gramos), se ejecuta el comando "fitlm(X,Y,'linear')" para obtener la regresión lineal de las variables introducidas. Se obtiene:

$$y = a \cdot x + b \rightarrow \left.\begin{cases} a = 1.2269 \\ b = -2037.9 \end{cases}\right\} \rightarrow y = 1.2269 \cdot x - 2037.9 \;\; \text{con} \;\; R^2 = 1 \tag{4.1}$$

donde la variable independiente x corresponde a la lectura en voltios e y es la variable dependiente y representa la equivalencia en gramos de dicha lectura. A continuación se representa la nube de puntos con el modelo basado en la regresión lineal:

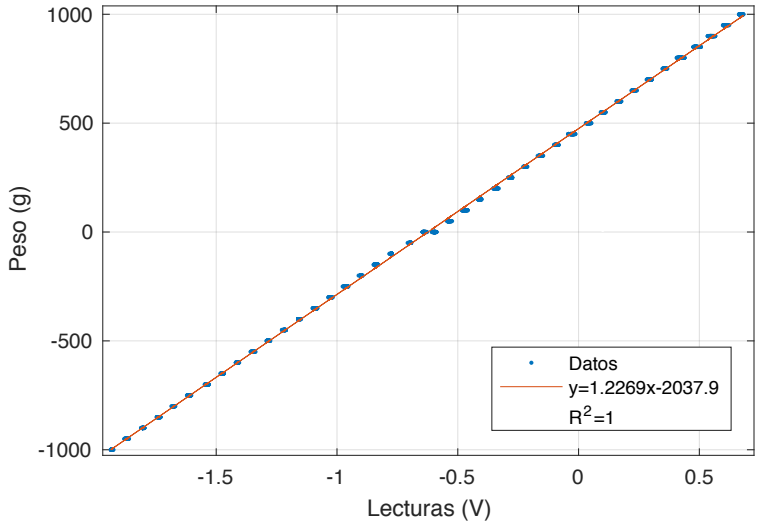

Figura 4.5: Representación de la nube de puntos junto con su modelo matemático.

Como se puede apreciar, el coeficiente de correlación obtenido es 1, por lo que la aproximación del modelo basado en la regresión lineal es perfecta. Sin embargo, esto no significa que el modelo

se aproxime con un error $E_{adm} < 3\%$ en todo el rango de trabajo, ya que la pequeña zona muerta puede provocar unos errores muy superiores al admisible para pesos reducidos. En el capítulo "5. Experimentos y Resultados", se analiza y comenta el comportamiento y errores de éste y los demás modelos deducidos.

4.2.1.5 Regresión lineal separando zona positiva y negativa

Con el fin de ajustar el modelo al comportamiento real y minimizar el error al mínimo, se toma un modelo del sensor para lecturas de fuerzas positivas y otro para las negativas. De esta forma, la pequeña zona muerta no influirá sobre las zonas más lineales de los resultados.

Por consiguiente, se realiza el mismo procedimiento que en el caso anterior aplicándolo a cada zona de trabajo. Es decir, se toman los puntos de la zona positiva y posteriormente de la negativa. De esta manera se obtienen dos modelos distintos. Para ello, volvemos a cargar los datos en las variables X e Y y ejecutamos el comando "fitlm(X,Y,'linear')", obteniendo los resultados:

- Zona positiva:

$$y = a \cdot x + b \rightarrow \left\{ \begin{matrix} a = 1.2633 \\ b = -2117.2 \end{matrix} \right\} \rightarrow y = 1.2633 \cdot x - 2117.2 \ \text{ con } \ R^2 = 1 \tag{4.2}$$

A continuación se muestra la representación gráfica de los datos para pesos positivos junto con el modelo obtenido.

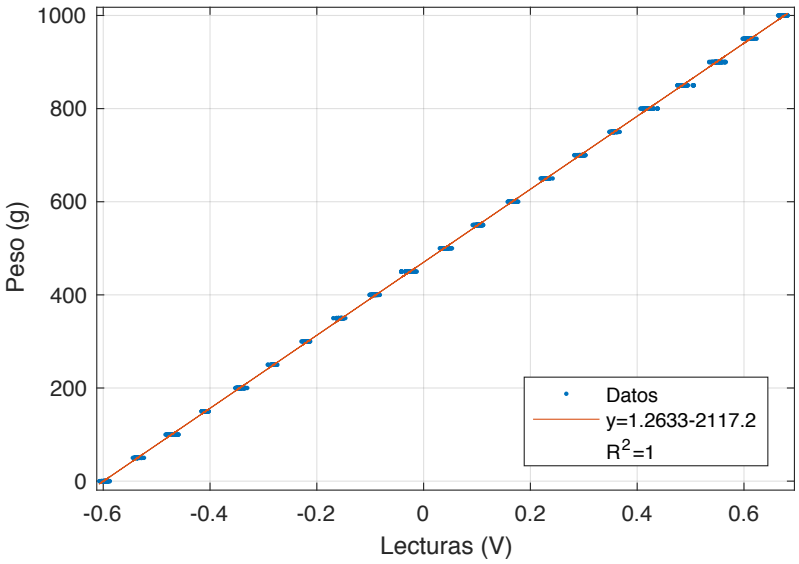

Figura 4.6: Representación de los datos de la zona positiva obtenidos junto con su modelo matemático basado en una regresión lineal.

- Zona negativa:

$$y = a \cdot x + b \rightarrow \left\{ \begin{array}{l} a = 1.2463 \\ b = -2058.2 \end{array} \right\} \rightarrow y = 1.2463 \cdot x - 2058.2 \ \text{ con } \ R^2 = 1 \tag{4.3}$$

A continuación se muestra la representación gráfica de los datos para pesos negativos junto con el modelo obtenido.

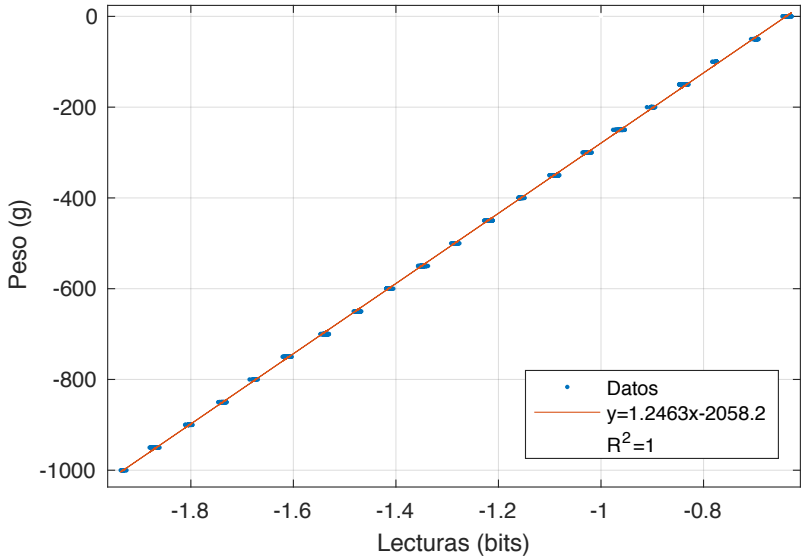

Figura 4.7: Representación de los datos de la zona negativa junto con su modelo matemático.

Como se puede apreciar, ambos coeficientes de correlación son exactamente 1 y además hemos eliminado la zona muerta de nuestro modelo. De este modo, para pesos pequeños será capaz de ofrecer una lectura más próxima a la real que el caso anterior. También se puede observar que el coeficiente a para cada caso no es exactamente igual, por lo que no tienen el mismo comportamiento (pendiente) ambas zonas. Con este modelo se espera afinar más las lecturas.

Se podría seguir obteniendo modelos mediante una regresión polinómica en lugar de lineal, pero apenas produciría diferencia. Ésto es debido a que los coeficientes de mayor orden (orden 2), no lineales, serían excesivamente pequeños, produciendo un efecto poco apreciable. Y a su vez ocurre esto debido al comportamiento lineal del sensor. Además, computacionalmente es más costoso calcular funciones cuadráticas en lugar de funciones lineales simples.

Una vez obtenidos los modelos que presentan un comportamiento aproximado al real y pueden tener un error admisible, se puede dar por concluida la etapa de calibración. Por último queda por ver qué modelo presenta errores menores para cada peso. Esto será incluido en el capítulo "5. Experimentos y Resultados".

4.2.2 Calibración del acelerómetro

Para la calibración del acelerómetro se realiza el procedimiento descrito en la introducción de este apartado, "4.2. Calibración", que es el mismo realizado que para la célula de carga. Sin embargo, la calibración de éste resulta mucho más simple. El motivo de ello es porque aplicar valores constantes y exactos de aceleración, para obtener una curva como en el caso anterior es casi imposible. Por lo que solo se puede calibrar en relación a un valor conocido que es el de la gravedad en la superficie terrestre, que tomamos como 9.80665 m/s^2. De este modo, el sensor es calibrado a partir de éste valor de gravedad, asumiendo que presenta un comportamiento lineal.

4.2.2.1 Descripción del procedimiento

El procedimiento a seguir es el de convertir las lecturas tomadas en cada eje en el valor de la gravedad mediante un simple factor de conversión. Para calibrar por completo el acelerómetro lo correcto es calibrar por separado cada eje del espacio (X, Y y Z). Una vez obtenido un valor para cada eje, se relaciona directamente con el valor de la gravedad anteriormente escrito.

Para tomar las lecturas de la medida en cada eje se realiza un pequeño programa en el entorno de Arduino IDE. En el programa simplemente guardamos en tres variables (una para cada eje) las lecturas de aceleración que sufre el sensor en cada momento. Posteriormente se representan las 3 variables en el "Monitor Serie" o "Serial Plotter" y se toman los valores que se muestran por pantalla. Siempre habrá una lectura que sea mucho mayor que las otras 2, por lo que esa es la variable relacionada con la dirección de mayor aceleración, que será la que se esté analizando.

Una vez representadas las lecturas, se observa que también existe un ruido que hace oscilar las señales en torno a un valor sin ningún patrón definido. Para que el ruido no afecte demasiado a la elección del valor correspondiente a la gravedad, se decide tomar no solo una medida sino 150, de manera que el error se compense y el valor medio se parezca al real.

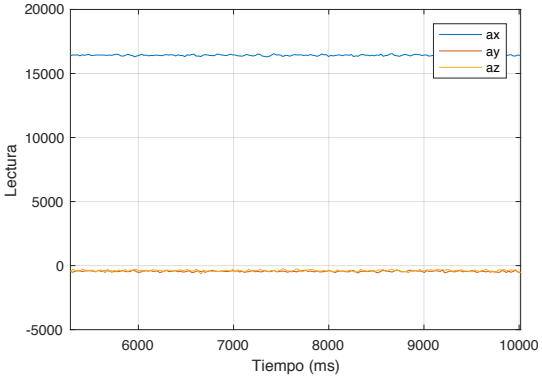

Figura 4.8: Señal de cada componente.

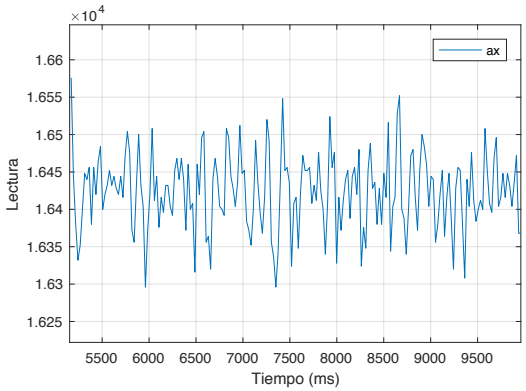

Figura 4.9: Señal del eje X.

Para calibrar correctamente cada eje hay que realizar varios experimentos. En cada uno de ellos se debe colocar el dispositivo de tal manera que la variable de dirección de máxima lectura sea distinta. Además se deben realizar dos experimentos por cada eje, uno para una lectura positiva de la gravedad y otro para una lectura negativa.

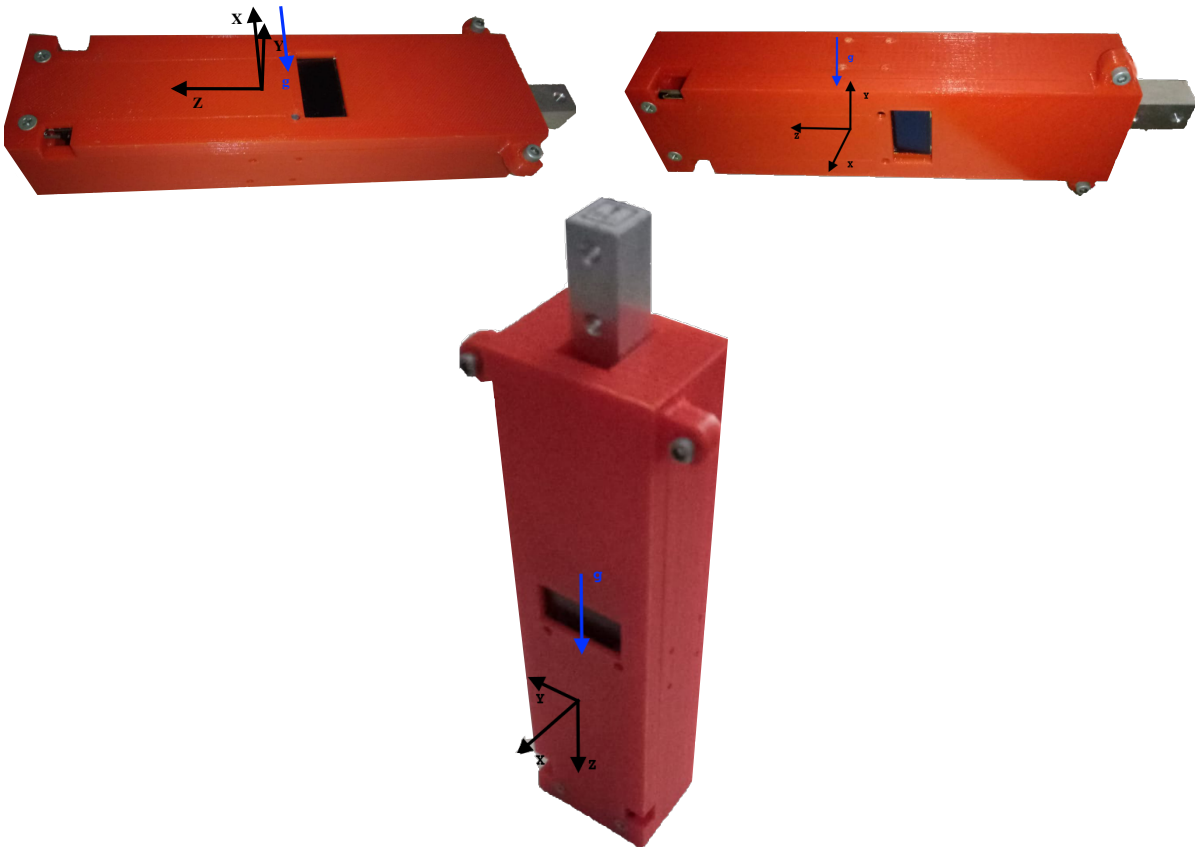

Figura 4.10: Diferentes disposiciones del equipo para la calibración del acelerómetro para cada eje del espacio.

En la figura 4.10 se observan las disposiciones del equipo para la calibración de cada eje. Para calibrarlo completamente se debe girar cada experimento 180^o y obtener otras 150 lecturas en cada eje. Finalmente se hace el promedio de los 6 experimentos, por separado. Puesto que el sensor se considera lineal, las lecturas deben ser iguales (o muy parecidas) pero de signo opuesto.

Una vez calibrado y con las lecturas de aceleración de cada componente, se puede obtener la orientación del equipo mediante simples operaciones trigonométricas. Se definen dos ángulos sobre los que se calculará la orientación. En la figura 5.5 se muestran dichos ángulos sobre el equipo.

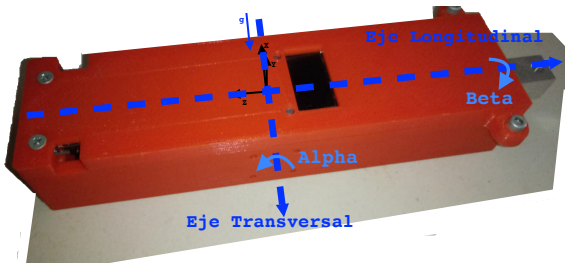

Figura 4.11: Representación sobre el equipo de los ejes y ángulos de orientación.

A partir de esto, se deducen las expresiones para el cálculo de ambos ángulos. Se calculan mediante las operaciones 4.4:

$$\alpha = atan2(a_z, a_x) \quad || \quad \beta = atan2(a_y, a_x) \tag{4.4}$$

En el capítulo 5. Experimentos y Resultados", se expone el resultado de la calibración y el error que presenta para cada componente.

4.3 Filtros

Una vez calibrados los sensores, se observa que las señales recibidas presentan un ruido inherente que dificulta la lectura real de sus datos. En ellas, se puede apreciar que existe distintos tipos de ruido, afectando a todo o gran parte de su espectro de frecuencias. En las figuras 4.12 y 4.13 se muestran las señales recibidas desde los respectivos sensores sin filtrar y con tiempo de muestreo, T_s, igual a 0:

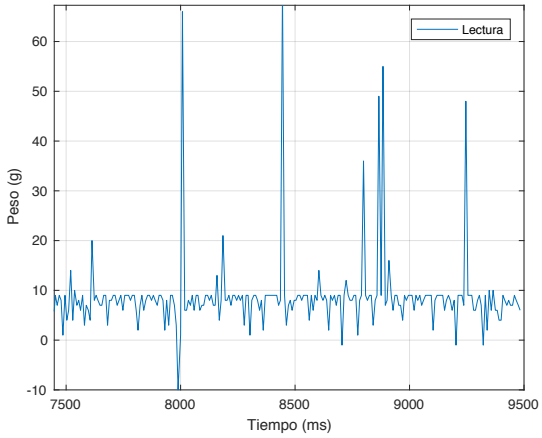

Figura 4.12: Señal de lectura de la célula de carga.

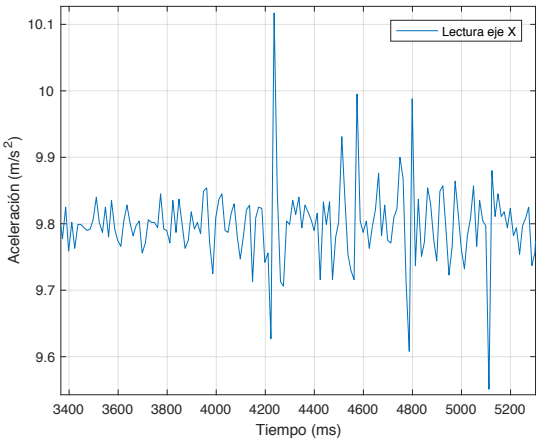

Figura 4.13: Señal de lectura del acelerómetro.

Volviendo a la señal sin filtrar, se observa que para una entrada de peso constante, la medida presenta unos picos que se producen de forma aleatoria y que no corresponden con la realidad. Estos picos son producidos por un ruido tipo "sal y pimienta"[6], o impulso, y se caracteriza por originar unos valores de salida extremos tanto positivos como negativos. Otro efecto que se puede observar es que la señal oscila sin ningún patrón definido en torno a un valor. Esto es producto por la existencia de un ruido uniforme, originando una inexactitud en la medida de forma permanente.

Ambos ruidos nombrados corresponden a un mismo grupo, que es el "ruido blanco". Se llama así debido a que, como la luz blanca, en un análisis frecuencial está presente en todo el espectro. Esto es debido a que para cualquier frecuencia posee la misma densidad espectral de potencia (PDS). Es decir, presenta el mismo tipo y cantidad de ruido para todo el espectro de frecuencias.

Para atenuar los efectos del ruido en la medida, se decide aplicar una serie de filtros basados en modelos matemáticos implementados en el código del programa. También se podrían utilizar filtros paso bajo y paso alto analógicos, pero necesitan de unos cálculos previos y montaje del circuito del filtro sobre nuestro equipo. De esta modo permite probar "in situ" qué filtro se comporta mejor mediante prueba y error además de poder ahorrar espacio. Los filtros utilizados son el filtro de la media móvil, filtro EMA y el filtro mediana. A continuación se describe como trabaja cada uno y las ventajas e inconvenientes que presentan:

4.3.1 Filtro de media móvil

Este filtro está basado en la media aritmética ($\overline{X}$) de una serie de datos ($X_1, X_2, ..., X_n$). La cantidad de datos, n, es constante para el filtro y su número influye en lo bueno que sea el filtro [5].

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{4.5}$$

Se llama media móvil debido a que los datos sobre los que se hace la media se actualizan con cada medida, donde se incluye la última lectura realizada y se suprime la más antigua.

- Ventajas: Filtro muy intuitivo y fácil de implementar. Permite suavizar la lectura y suaviza en cierta medida el ruido de alta frecuencia. Todo dependiendo del valor de datos tomados n.

- Inconvenientes: Muy vulnerable ante ruido impulso, donde un solo dato anómalo, desvirtúa una serie de medidas. En cada momento, el valor resultante no es el valor real sino la media de los n valores. Una gran cantidad de datos (n grande), produce que en régimen transitorio se comporte como un sistema de primer orden. De esta forma ante entradas impulso puede no acercarse al valor de fuerza ejercido.

[6]También conocido como *salt and pepper noise*

En las figuras 4.14, 4.15, 4.16 y 4.17 se representa la señal sin procesar y el comportamiento del filtro para distintos valores de n. Se estudia su comportamiento tanto en régimen estacionario (con peso nulo) como en transitorio:

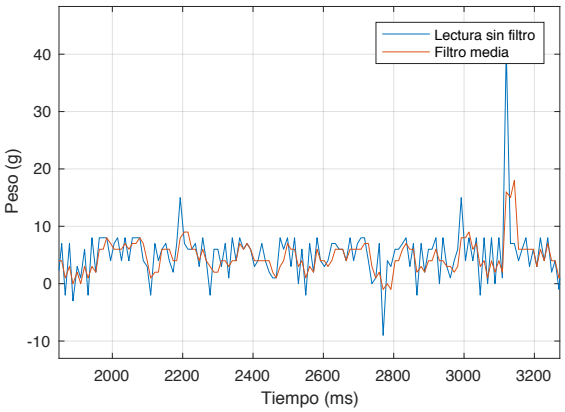

Figura 4.14: Filtro de media móvil con n=3 en régimen estacionario.

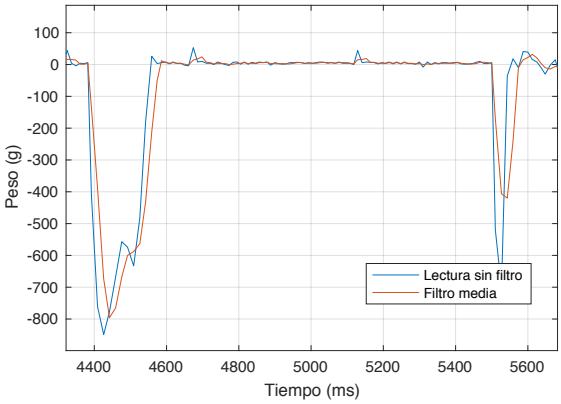

Figura 4.15: Filtro de media móvil con n=3 en régimen transitorio.

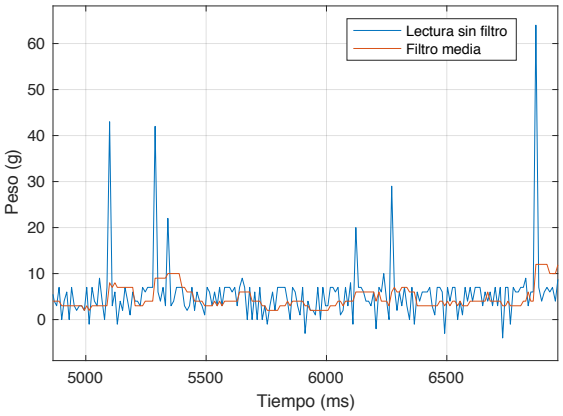

Figura 4.16: Filtro de media móvil n=10 en régimen estacionario.

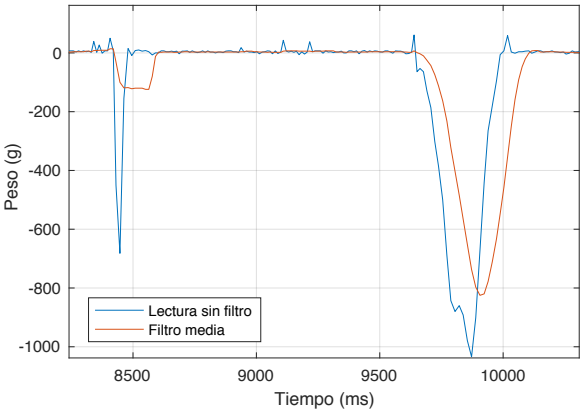

Figura 4.17: Filtro de media móvil con n=10 en régimen transitorio.

Aclaración: Se puede apreciar que la lectura de peso no varía entorno a cero. Esto es debido a que no se ha implementado aún la función de tarar el equipo, por lo que en ocasiones puede existir este error estacionario en las gráficas mostradas.

En las gráficas en régimen estacionario (Figuras 4.14 y 4.16) se observa que cuanto mayor es n más atenuada es la señal filtrada. Aún así, se aprecia que resulta vulnerable ante ruido impulso, ya que un solo punto muy alejado, es capaz de variar considerablemente el valor filtrado. En régimen transitorio (Figuras 4.15 y 4.17) se observa que cuanto mayor es n mayor es el retraso

de la señal filtrada y más se asemeja a un sistema de primer orden, haciendo que para entradas impulso, la medida resulte imprecisa.

4.3.2 Filtro EMA

Este filtro está basado un modelo exponencial de la media de una serie de valores [6]. De ahí su nombre EMA, *Exponential Moving Average*. El modelo matemático del filtro es el siguiente:

$$A_n = \alpha M_n + (1-\alpha)\cdot A_{n-1} \tag{4.6}$$

donde M es la nueva lectura realizada (sin filtrar), A_n es el valor de salida filtrado, A_{n-1} es el valor filtrado anterior y α es el coeficiente de influencia de cada componente comprendido entre 0 y 1. El valor de α está relacionado con la frecuencia de corte del filtro y es el que determina lo bueno que es el filtro. Su valor está determinado por las características de la señal. Si α es igual a 1, la salida será la señal sin filtrar. Sin embargo, si α es nulo o casi nulo, la salida será constante pues prescinde de nueva información aportada por el sensor. De esta manera, cuanto menor sea α más se filtrará la señal.

Este modelo es un caso concreto de un modelo de media de pesos ponderados. En el modelo de media de pesos ponderados, el resultado es la media de unos valores, x_i. Sin embargo, en este caso cada valor tiene un efecto (o peso), w_i, concreto sobre el cálculo final del resultado.

$$\overline{x} = \frac{\sum_{i=1}^{n}(w_i \cdot x_i)}{\sum_{i=1}^{n} w_i} \tag{4.7}$$

Aunque a simple vista pueda parecer que no es un modelo exponencial, se puede demostrar fácilmente que sí lo es aplicándolo a una serie de mediciones finitas. Para ello partiendo de la ecuación genérica del modelo:

$$A_n = \alpha M_n + \beta \cdot A_{n-1}$$

donde β correspondería con el término $(1-\alpha)$, entonces:

$$A_0 = \alpha M_0$$

$$A_1 = \alpha M_1 + \beta \cdot A_0 = \alpha M_1 + \beta\alpha M_0$$

$$A_2 = \alpha M_2 + \beta \cdot A_1 = \alpha M_2 + \beta\alpha M_1 + \beta^2 \alpha M_0$$

Generalizando para n medidas, se obtiene:

$$A_n = \alpha M_n + \beta \cdot A_{n-1} = \alpha \cdot \sum_{t=0}^{n}(\beta^t \cdot M_{n-t}) = \alpha \cdot \sum_{t=0}^{n}((1-\alpha)^t \cdot M_{n-t})$$

De esta manera, el valor de salida filtrado A_n es proporcional a un factor exponencial $(1-\alpha)^t$, por lo que es filtro exponencial. Puesto que α es menor que 1, pero mayor que 0, β también lo es, por lo que cuanto mayor sea el exponente t menor será el efecto producido sobre la salida. Sin embargo, a consecuencia del redondeo que se produce en el programa, su efecto es nulo pasadas unas pocas iteraciones.

- Ventajas: Computacionalmente es muy eficiente. Su implementación es muy simple y consume muy poca memoria, ya que solo guarda el último valor filtrado. Permite eliminar ruido de alta frecuencia, suavizando la señal. Con un buen valor para α, no resulta tan vulnerable ante ruido impulso.

- Inconvenientes: Elegir un valor de α demasiado bajo puede eliminar componentes frecuenciales de interés. Cuanto menor sea α, mayor es el tiempo de respuesta, asemejando su comportamiento a un sistema de primer orden. Además, en los casos más extremos, introduce un retraso importante en la medida.

En las figuras 4.18, 4.19, 4.20 y 4.21 se representa la señal sin procesar junto con el filtro para distintos valores de α:

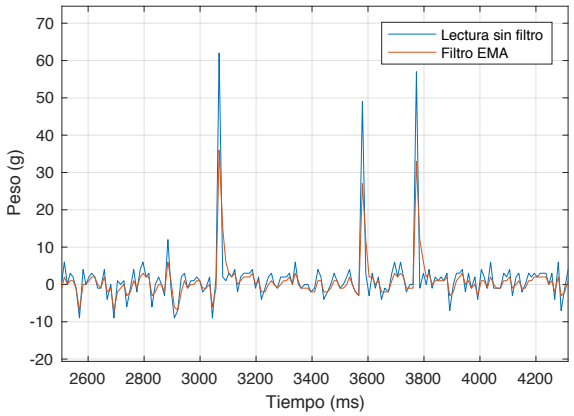

Figura 4.18: Señal filtrada con α=0.6 frente a la señal original en régimen estacionario.

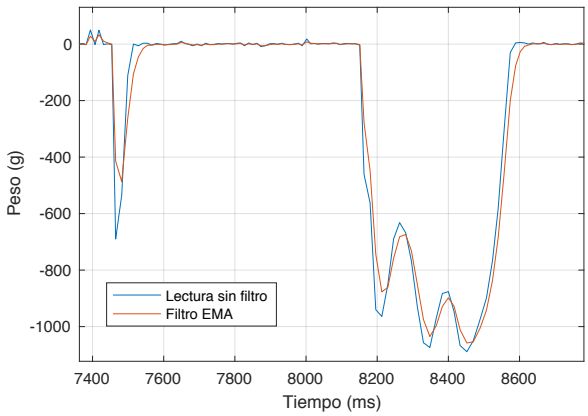

Figura 4.19: Señal filtrada con α=0.6 frente a la señal original en régimen transitorio.

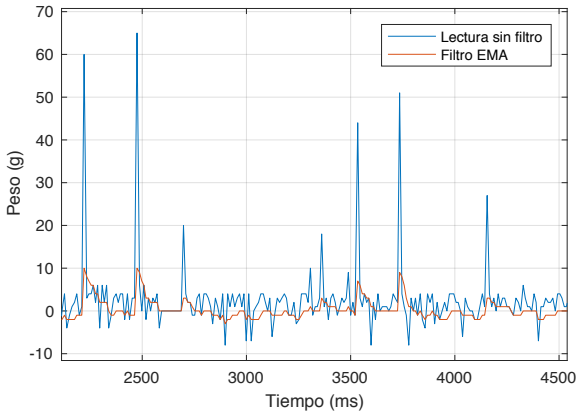

Figura 4.20: Señal filtrada con α=0.2 frente a la señal original en régimen estacionario.

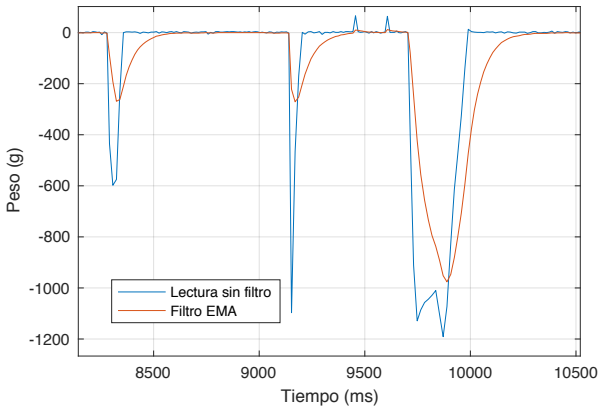

Figura 4.21: Señal filtrada con α=0.2 frente a la señal original en régimen transitorio.

En las gráficas en régimen estacionario (Figuras 4.18 y 4.20) se puede observar que cuanto menor es α mejor es filtrada la señal, atenuando el ruido en altas frecuencias. Sin embargo, se observa que este filtro también resulta vulnerable ante ruido tipo impulso. Además en régimen transitorio (Figuras 4.19 y 4.21) el tiempo de respuesta aumenta notablemente cuanto menor es α ya que se asemeja aún más a un sistema de primer orden que el filtro media. De este modo, ante entradas impulso muy grandes, puede haber una gran diferencia entre el valor real y el leído.

4.3.3 Filtro mediana móvil

Este filtro está basado en el modelo matemático de la mediana estadística para un conjunto de datos. De este modo, el filtro lo que realiza es una lectura de una cantidad de datos constantes n, los ordena de menor a mayor según su valor, y devuelve el valor central del conjunto de datos. Por ello, es importante que el conjunto de datos tomados siempre sea impar, pues de no ser así, sería necesario hacer la media de los valores centrales, perdiendo parte de la capacidad del filtro. El procedimiento descrito matemáticamente sería:

$$x_1, x_2, x_3, ..., x_n \;\; con \;\; n = impar \;\; y \;\; x_1 < x_2 < x_3 < ... < x_n \rightarrow x_{filtrada} = x_{\frac{n+1}{2}} \tag{4.8}$$

Como para los filtros anteriormente descritos, esta cantidad de datos determinará el funcionamiento del filtro, así como la calidad de los datos de salida. Al igual que el filtro media, los datos a filtrar se actualizan con cada iteración. Se incluye la última lectura realizada y se suprime la más antigua.

- Ventajas: Existe una librería para Arduino que implementa el filtro [7], por lo que la implementación resulta muy simple. Presenta mejor comportamiento que los filtros anteriormente nombrados ante lecturas puntuales anómalas. De esta forma, eliminaría en gran medida y de forma sencilla el ruido tipo impulso. De forma general, el resultado del filtro es un valor de lectura real, no un valor aproximado del real o la media de ciertos medidas. Aumentar el valor de datos no aumenta el tiempo de establecimiento de la medida, por lo que la señal filtrada tiene la misma forma que la original.

- Inconvenientes: Computacionalmente es más costoso. El proceso de ordenación de las lecturas consume más tiempo que si se aplicase una operación matemática simple como en los filtros anteriores. Además, cuantos más datos se tomen, mayor es el retraso que sufre la lectura filtrada durante la espera para recibir los datos. De esta forma, más desfase se produce entre la señal original y la filtrada.

En las figuras 4.22, 4.23, 4.24 y 4.25 donde se representan la señal sin procesar y el comportamiento del filtro para distintos valores de n. Se estudia su comportamiento tanto para régimen estacionario, como transitorio:

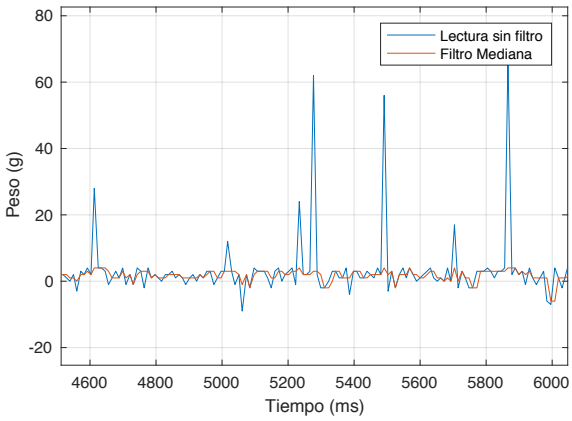

Figura 4.22: Filtro de mediana móvil con n=3 en régimen estacionario.

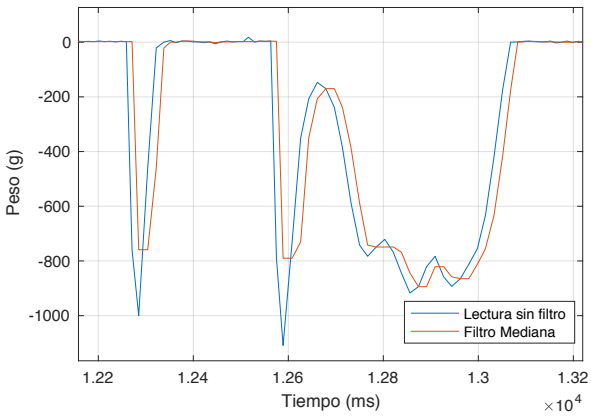

Figura 4.23: Filtro de mediana móvil con n=3 en régimen transitorio.

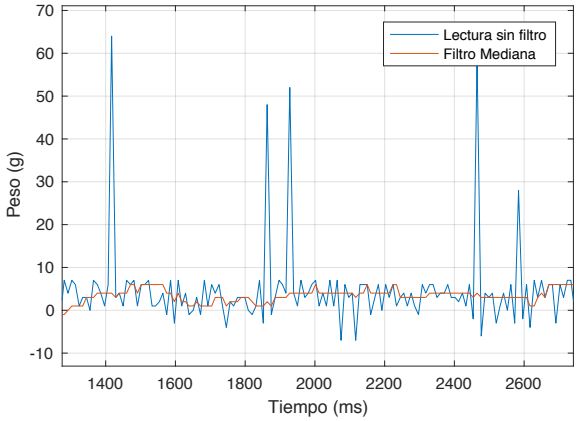

Figura 4.24: Filtro de mediana móvil con n=11 en régimen estacionario.

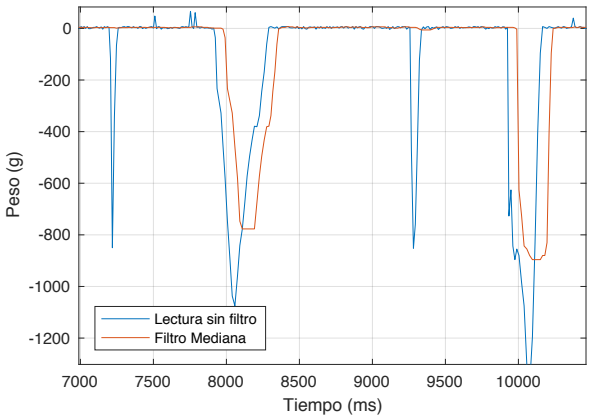

Figura 4.25: Filtro de mediana móvil con n=11 en régimen transitorio.

En las gráficas en régimen estacionario (Figuras 4.22 y 4.24) se puede observar que cuanto mayor es n mejor es filtrada la señal. Como se dijo anteriormente, este filtro presenta muy buenos resultados ante ruido tipo impulso, ya que hasta para un bajo valor de n, como en la Figura 4.22, no este ruido no perturba la señal filtrada. Sin embargo en régimen transitorio (Figuras 4.23 y 4.25) se aprecia que cuanto mayor es n mayor es el retraso o desfase de la señal filtrada. Sin embargo, no se asemeja tanto a un sistema de primer orden como los filtros anteriores.

Como se puede deducir, no existe un único filtro que proporcione una señal ideal, sin ruido y con tiempo de respuesta nulo. Además cada filtro tiene unas características que los hacen necesarios. De este modo la solución viene por combinarlos de forma adecuada, de manera puedan aportar sus ventajas a las lecturas sin que sus inconvenientes influyan demasiado. Por ello se necesita encontrar un equilibrio entre una buena lectura filtrada y un buen tiempo de respuesta.

4.3.4 Filtro para la célula de carga

Como se ha comentado anteriormente, la manera correcta de obtener una señal suficientemente filtrada sin que afecte demasiado a su tiempo de respuesta es mediante la combinación de filtros. De esta manera, la combinación del filtro mediana permite filtrar los puntos anómalos de forma eficiente, junto con el suavizado y submuestreo aportado por los filtros EMA y media móvil. Por otro lado, el tiempo de respuesta de la señal depende sobretodo de la robustez de los filtros y no tanto de cuántos filtros se utilizan.

Se decide aplicar primero el filtro de mediana a la señal original con el objetivo de eliminar el ruido tipo impulso, de forma que no afecten al resto de filtros. De lo contrario, estos puntos podrían ser suavizados haciendo que el filtro mediana resulte menos efectivo. Aparte de esta particularidad, el orden de los filtros no debería influir en la salida pues producen un efecto parecido sobre la señal. De esta manera, se comprueba el resultado de aplicar el filtro mediana móvil, posteriormente el filtro EMA y por último el filtro de media móvil. Se utiliza unos coeficientes de $n = 3$ para el filtro mediana, $\alpha = 0.6$ para el filtro EMA y $n = 3$ para la media. Es decir, los coeficientes menos robustos, para que afecten lo mínimo posible al tiempo de respuesta de la señal.

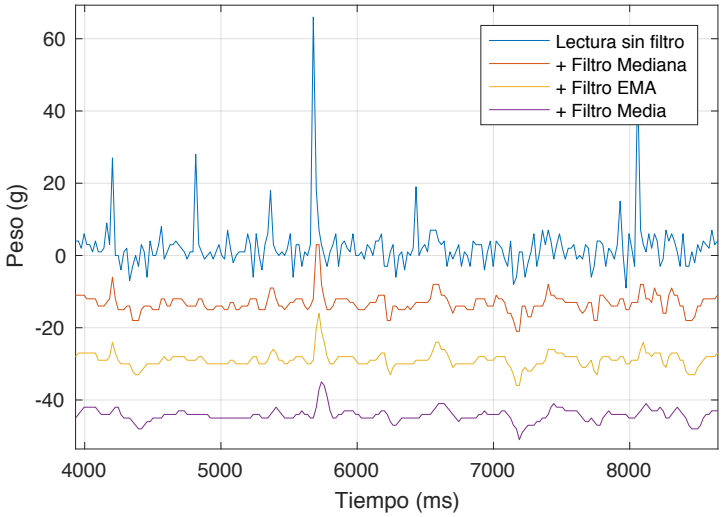

Figura 4.26: Comparación de los filtros utilizados en la célula de carga.

Para una fácil interpretación de las señales, se desplaza cada una 10 gramos respecto a las adyacentes para evitar que se superpongan. Como se puede observar, con cada filtro que se aplica, menos ruido en altas frecuencias existe. Se comparan ahora la señal más filtrada (+ Filtro Media) respecto a la original tanto para régimen estacionario como transitorio.

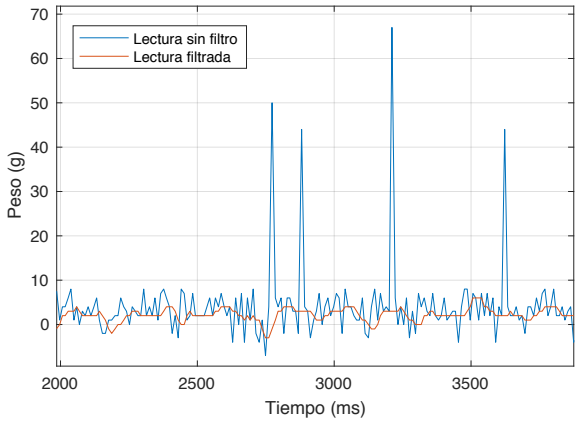

Figura 4.27: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.6$ y $n_{media} = 3$ en régimen estacionario.

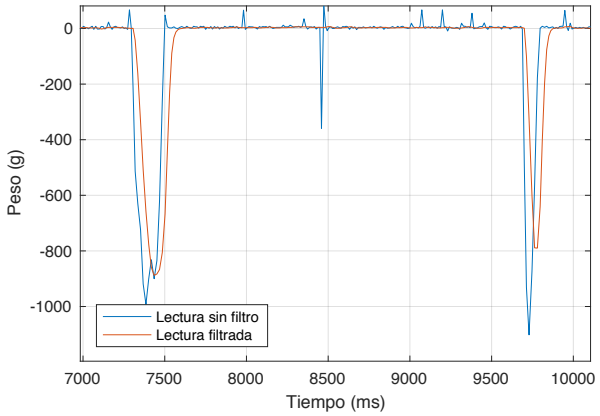

Figura 4.28: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.6$ y $n_{media} = 3$ en régimen transitorio.

Se puede apreciar que la señal resultante responde muy bien ante ruidos tipo impulso. Sin embargo, los ruidos de alta frecuencia siguen produciendo efectos no deseados sobre la señal. Se puede hacer más robusto el filtro aumentando el coeficiente de cada filtro. No obstante es importante no modificar aquel que no sea necesario, ya que el filtro de la mediana ofrece una buena señal ante ruido impulso y tomar más datos produciría un mayor retraso. Como no se conoce qué coeficiente es más adecuado modificar, se comprueban distintos valores tanto de n_{media} como de α. Finalmente se espera obtener un filtro que presente un equilibrio entre un buen comportamiento en régimen permanente y una buena respuesta dinámica.

En las figuras 4.29 y 4.30, se estudia el comportamiento de la señal tanto en régimen estacionario como transitorio para $n_{media} = 5$ y $\alpha = 0.6$. En las figuras 4.31 y 4.32 se estudia el comportamiento para $n_{media} = 3$ y $\alpha = 0.5$.

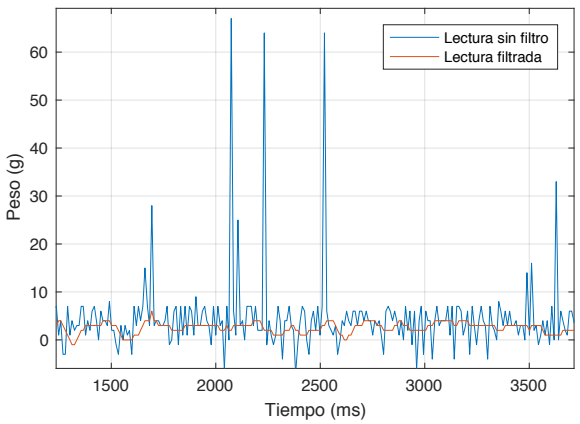

Figura 4.29: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.6$ y $n_{media} = 5$ en régimen estacionario.

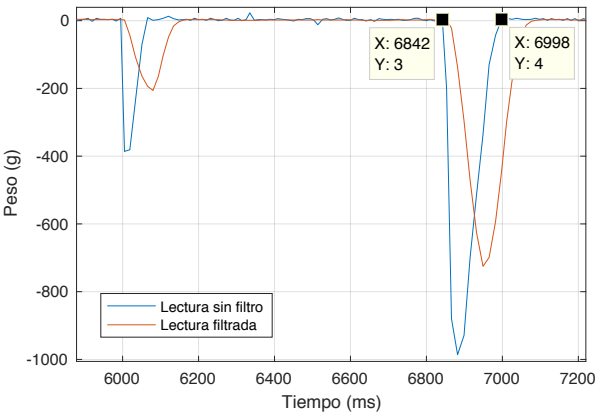

Figura 4.30: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.6$ y $n_{media} = 5$ en régimen transitorio.

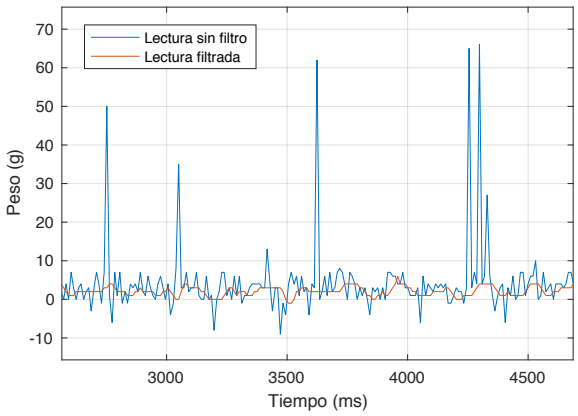

Figura 4.31: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.5$ y $n_{media} = 3$ en régimen estacionario.

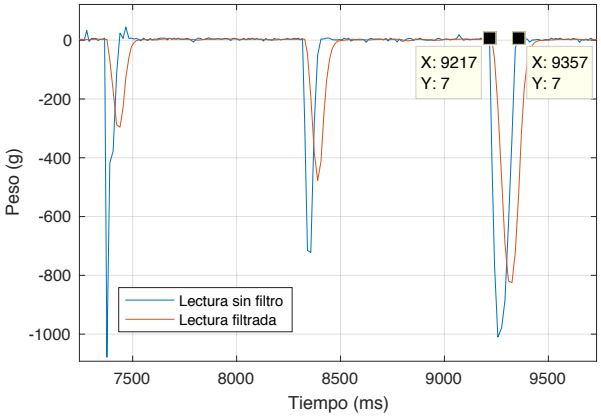

Figura 4.32: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.5$ y $n_{media} = 3$ en régimen transitorio.

Comparando el comportamiento transitorio de las figuras 4.30 y 4.32, se puede observar que aumentar la robustez del filtro media es más costoso que aumentarlo en el filtro EMA. El retraso es menor en el segundo, cuando la duración de las fuerzas es aproximadamente la misma en ambos casos (156ms en el filtro media y 140ms en el filtro EMA). Para la misma entrada de -1000g, en el primero la respuesta es de -700g aproximadamente, mientras que en el segundo llega a los -800g. Por otro lado, en régimen permanente la capacidad de filtrado es prácticamente igual para ambos casos.

Por ello, se llega a la conclusión de que es más efectivo y adecuado hacer más robusto el filtro EMA y aplicar un valor más conservador al filtro media. De esta manera, se obtiene el siguiente comportamiento:

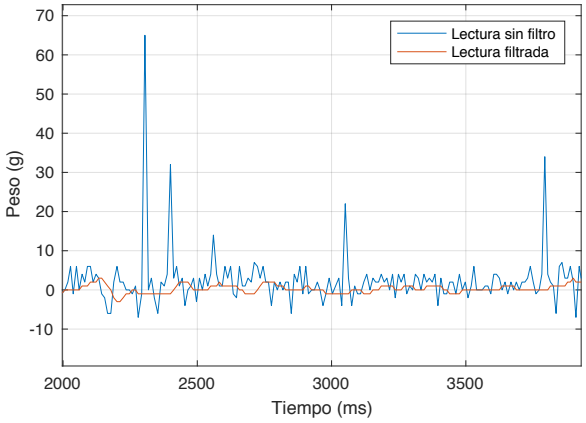

Figura 4.33: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.45$ y $n_{media} = 4$ en régimen estacionario.

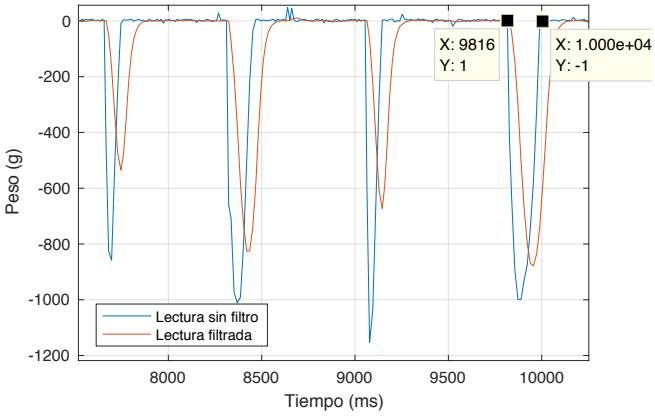

Figura 4.34: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.45$ y $n_{media} = 4$ en régimen transitorio.

Como se puede observar, ofrece una lectura mucho más estable en régimen estacionario que los dos filtros anteriores, mientras que en régimen transitorio ofrece una señal aproximada a la real. Para una entradas de fuerza con una duración superior a 184ms se puede considerar que la lectura es fiable y aproximada. Por esta razón, se decide implementar este filtro, con coeficientes $n_{mediana} = 3$, $\alpha = 0.45$ y $n_{media} = 4$.

Por último, y no menos importante, aumentando el tiempo de muestreo, se consigue estabilizar aún más la señal. Sin embargo, si el dato leído tiene un error, éste se mantendrá valor durante más tiempo mientras no se vuelva a muestrear, por lo que no es una herramienta que realmente filtre la señal. Tomando un tiempo de muestreo $T_s = 25ms$ al filtro anterior, se obtiene la señal:

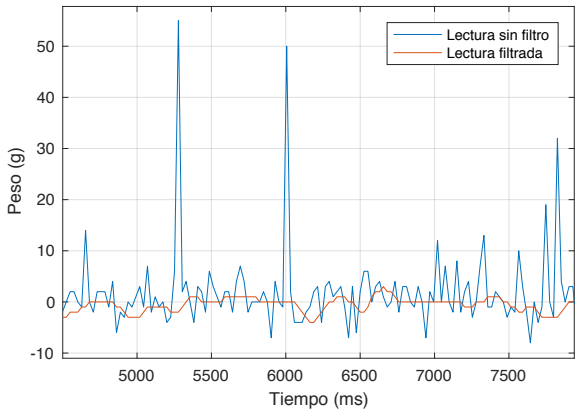

Figura 4.35: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.45$, $n_{media} = 4$ y $T_s = 25ms$ en régimen estacionario.

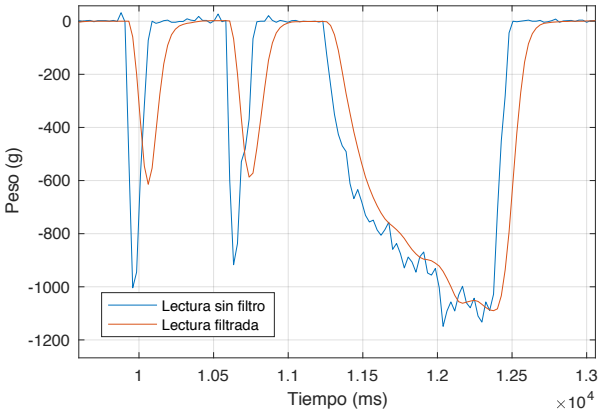

Figura 4.36: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.45$, $n_{media} = 4$ y $T_s = 25ms$ en régimen transitorio.

Se puede observar que la señal sin filtrar es más limpia. Esto es debido a que toma 1 dato cada 25ms, a diferencia que con $T_s = 0$ donde habría tomado al menos 25 datos en ese tiempo. Por otro lado, la señal filtrada no muestra una gran diferencia respecto a la señal con $T_s = 0$. Es por ello por lo que no aumentar T_s se considera como un filtro de la señal propiamente dicho.

4.3.5 Filtro para el acelerómetro

Como se puede ver en la figura 4.13, la señal procedente del acelerómetro también presenta un ruido que dificulta la lectura de sus datos, por lo que es necesario filtrarla. Al igual que para la célula de carga, se aplica primero el filtro de la mediana para eliminar los puntos de lecturas anómalas y posteriormente se suaviza con los filtros EMA y media.

En la figura 4.42 se muestra la señal filtrada mediante el mismo procedimiento que para la célula de carga:

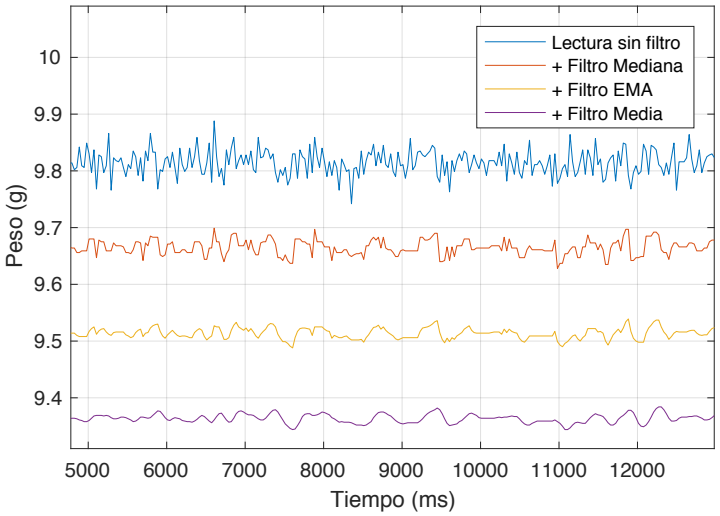

Figura 4.37: Comparación de los filtros utilizados en el acelerómetro.

Se puede apreciar el efecto que ejerce cada filtro sobre la señal anterior. Los coeficientes aplicados a cada filtro son $n_{mediana} = 3$, $\alpha = 0.45$ y $n_{media} = 4$, iguales que para la célula de carga. Se observa que la señal filtrada por la mediana, EMA y media ofrece una lectura suavizada de la señal. Sin embargo, se pueden hacer más robustos los filtros, pues las lecturas en régimen transitorio no necesitan ser tan aproximadas o fieles a las reales como para la célula de carga. Para ello se decide aumentar el valor de n_{EMA} a 0.4 y n_{media} a 5.

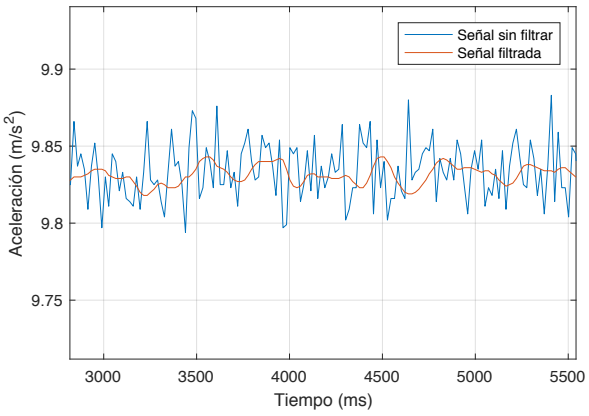

Figura 4.38: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.4$ y $n_{media} = 5$ en régimen estacionario.

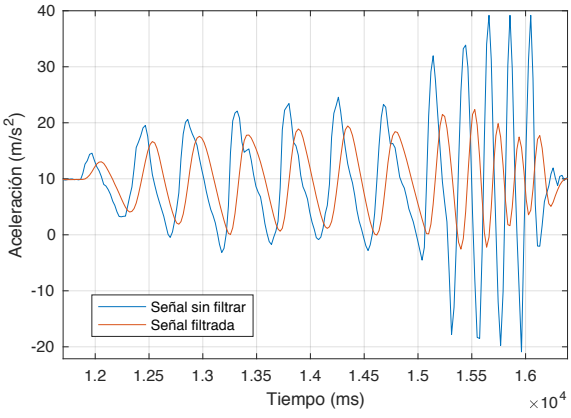

Figura 4.39: Filtro con coeficientes $n_{mediana} = 3$, $\alpha = 0.4$, $n_{media} = 5$ en régimen transitorio.

En este caso, la señal se ve algo más suavizada, pero no está a un valor constante que sería lo deseado. Sin embargo, la oscilación tiene una amplitud de $0.02 m/s^2$ aproximadamente.

Por otro lado, la respuesta en régimen transitorio no resulta muy retrasada de la original. De cualquier modo, durante un experimento estándar del equipo, no sería normal que tenga unos movimientos excesivamente bruscos o sea necesario leer giros rápidos con alta precisión. De esta manera, con giros normales, la señal filtrada puede seguir la real con suficiente precisión.

Con estos resultados, se decide tomar finalmente el filtro compuesto por el filtro mediana con $n_{mediana} = 3$, el filtro EMA con , $\alpha = 0.4$, y el filtro media con $n_{media} = 5$.

4.4 Salida de datos por pantalla

Una vez obtenidas y filtradas las lecturas de fuerza y orientación, se muestran por pantalla con el fin de poder conocer los valores en tiempo real sin necesidad de equipos externos a nuestro equipo de medida. Para ello se utiliza la pantalla integrada en la tarjeta del ESP-32 usada en este proyecto. La pantalla se comunica con el procesador a través de los buses SDA y SCL de I_2C, como el acelerómetro.

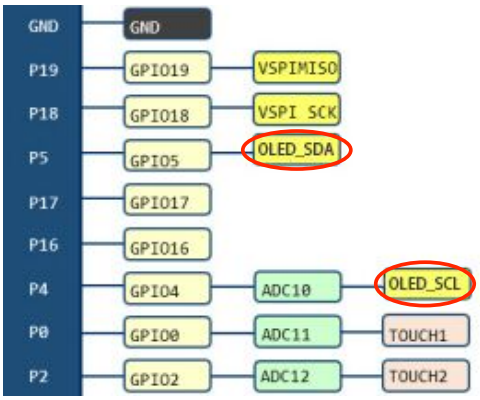

Figura 4.40: Detalle de los pines de conexión I_2C de la pantalla.

En la imagen se muestran los pines de conexión de la tarjeta. Se observa que los pines 4 y 5 corresponde con los pines OLED_SCL y OLED_SDA respectivamente. Es decir, que la pantalla OLED internamente está conectada a estos pines de comunicación I_2C, por lo que simplemente queda programarla. Para ello, se debe incluir en el programa la librería "Wire.h", para poder realizar la comunicación I_2C, y la librería "SSD1306.h" la cual ofrece funciones exclusivas para la pantalla como el tamaño de la letra y tipografía, disposición del texto, etc.

Una vez incluidas las librerías, se declara un objeto del tipo SSD1306 con los pines de I_2C y su dirección para la comunicación, bajo el nombre "display". De esta manera, se podrán utilizar todas las funciones de la librería SSD1306 sobre el objeto "display".

Antes de mostrar datos por pantalla, se debe inicializar el objeto "display" en la función "setup()". Es en este momento donde se establece la tipografía y tamaño de letra, contraste de la pantalla, orientación y origen del texto (a izquierda o a derecha), etc. Finalmente en la función "loop()" se incluyen todas las funciones de representación. Básicamente con la función "display.drawString(X, Y, String);", se muestra en las coordenadas X e Y de la pantalla, la variable tipo "string" que se desee. En el anexo "1.2. Anexo de programación del procesador" se muestran la programación tanto de la pantalla, como de todos los sensores.

4.5 Comunicación

Una vez recibidas y procesadas las lecturas de fuerza y orientación, es necesario recopilar todos estos datos para que el usuario pueda analizarlos posteriormente. Puesto que no se puede utilizar la memoria interna del ESP-32 para el almacenamiento de datos, es necesario transmitir los datos a un equipo secundario que sea capaz de almacenar y representarlos. Para ello existen 2 posibilidades:

1. Transmitir los datos por el puerto serie (USB) a un ordenador.
2. Transmitir los datos por Wi-Fi, mediante aplicación de IoT (*Internet of Things*), a un servidor donde puedan ser almacenados para la disposición del usuario.

4.5.1 Comunicación por puerto serie

La comunicación por puerto serie consiste en la transmisión de los datos obtenidos por el procesador ESP-32 a un equipo mediante el puerto USB. Para hacerlo posible, el ordenador debe estar preparado para ello y ordenar dicha adquisición de datos. Conforme se reciben, el equipo tiene la posibilidad de representarlos, guardarlos o ambas.

Una gran ventaja que ofrece este método, es que no ofrece ninguna limitación en la cantidad de datos a transmitir, como ninguna limitación en velocidad de transmisión[7]. No obstante, tiene la desventaja de necesitar la presencia del ordenador, perdiendo gran parte de la autonomía que se quiere dar al equipo. En función del tipo de experimento y de la necesidad de precisión, se utilizará este método u otro.

Para la comunicación y recepción de los datos se ha utilizado el software MATLAB. Se utiliza este debido a su gran versatilidad, dando soporte para lecturas por puerto serie además de ofrecer herramientas de procesado, representación y almacenamiento de datos.

[7]Dependiendo del software utilizado, algunos ofrecen mejores tiempos de lectura (tiempo real), mientras que otros presentan cierto retraso de la señal

Una vez en MATLAB, se ha creado una función "Datos_PuertoSerie.m" donde, para una cantidad de lecturas establecidas "numero_muestras", automáticamente se conecta al ESP-32, realiza la lectura y representación de los datos. Por último los guarda en un archivo .csv para su posterior análisis.

4.5.2 Comunicación por IoT

Para la comunicación por IoT, primero se debe conocer qué servidor puede satisfacer nuestras necesidades. Cada uno presenta unas características que otros no tienen, pero no todos son válidos para nuestro objetivo. Los aspectos más importantes que se han tenido en cuenta al elegir el servidor son que ofrezca soporte para comunicación por MQTT, una cantidad de datos a subir suficientes como para nuestro uso, que tenga una velocidad de subida de datos alta, para poder tener los datos en tiempo real, y que sea lo más fácil posible de implementar en nuestro equipo.

En el mercado ya existen una gran cantidad de plataformas IoT a los que enviar los datos directamente desde el equipo vía Wi-Fi. Cada uno de ellos presenta una serie de características que resultan más atractivas para nuestro equipo y su finalidad. Para ello, se consideran 3 plataformas con soporte para MQTT. Estas son ThingSpeak, Amazon Web Service (AWS) y Cayenne. A continuación se enumeran las ventajas e inconvenientes de cada uno de ellos:

1. ThingSpeak

    - Ventajas: Es el servicio IoT desarrollado por MathWorks, por lo que está muy integrado en el entorno de desarrollo de MATLAB. Permite implementar código MATLAB en la nube para analizar y procesar los datos. Y por último, la programación para su conexión resulta muy simple.
    - Inconvenientes: Aunque para la versión de pago no existe este problema, con la versión gratuita solo puede subir un dato cada 15 segundos. Por otro lado, limita los datos a subir hasta 8.200 al día o 3.000.000 de datos al año.

2. Amazon Web Service

    - Ventajas: Ofrece en un mismo lugar una amplísima colección de servicios de computación en la nube.
    - Inconvenientes: Está más orientado a la gran empresa donde se procesan gran cantidad de datos desde gran cantidad de dispositivos o fuentes. De esta manera, ofrece tal multitud de servicios que su uso no resulta tan intuitivo como los otros servicios considerados. De este modo, la conexión con el servidor no resulta igual de simple que con otros servidores. Por último, ofrece acceso gratuito durante un año, pero no es un servicio totalmente gratuito.

3. Cayenne

   - Ventajas: Interfaz de usuario muy intuitiva. Al igual que en ThingSpeak, con unas pocas líneas de código se asocia el equipo al servidor. Ofrece 60 lecturas por minuto, sin importar el tiempo entre subida de datos. Fácil interpretación de las lecturas mediante distintas representaciones (Valor numérico, gráfico, indicador,...). Permite descargar las lecturas en un fichero .csv fácilmente. Por último, aunque hay packs de pago, su uso es totalmente gratuito.
   - Inconvenientes: Solo permite 60 lecturas por minuto, que aunque es de los servidores que más lecturas ofrece, presenta esa limitación.

4.5.2.1 Implementación en el programa

A continuación se describe paso a paso la conexión de nuestro *thing* a cada servidor web.

1. **ThingSpeak**

   La limitación de subir un dato cada 15 segundos, hace que sea inviable para el proyecto que atañe, dado que las medidas obtenidas apenas aportarían información de los experimentos realizados. Aún así, se considera este servidor como una mera comprobación de la posibilidad de mandar datos a un servidor por Wi-Fi. Se hace así ya que es el servidor que más documentación y ayudas en Internet ofrece, por lo que sirve como punto de partida.

   Se procede a la implementación en el programa. Para ello, una vez registrados en su página web, se crea un nuevo canal por el que realizar la comunicación y se le asigna un nombre, al canal y a los datos a recibir, "Medidas sensor" y "Fuerza" para nuestro caso.

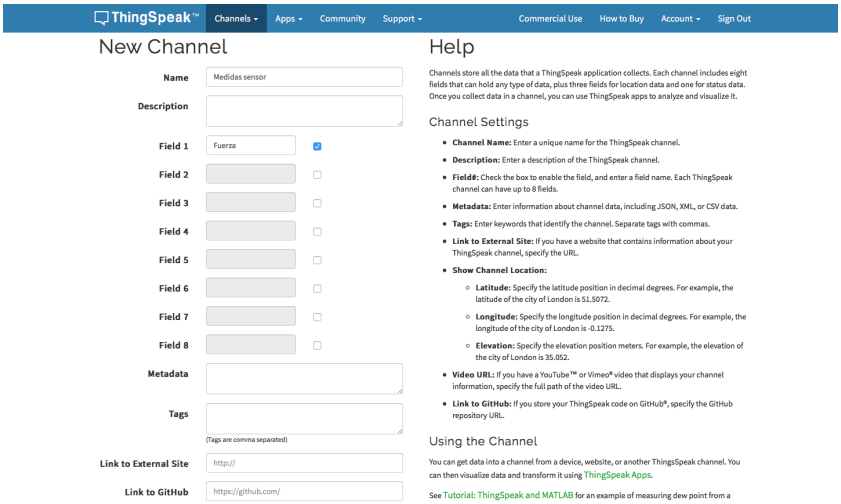

Figura 4.41: Creación de un nuevo canal en ThingSpeak.

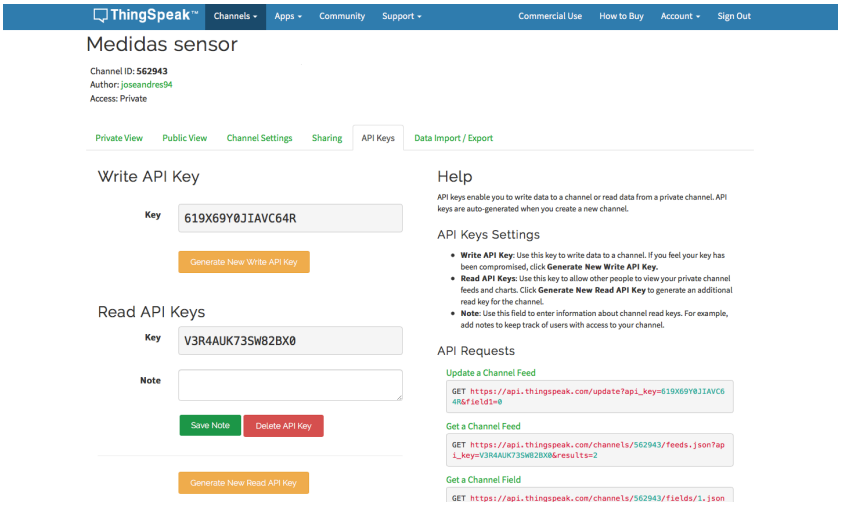

Figura 4.42: Obtención de las credenciales necesarias para la comunicación.

Una vez dentro del canal, en la pestaña "API Keys" se dispone de las credenciales necesarias para una correcta asociación. Para poder escribir en el servidor, se toma la ubicada en "Write API Key".

Por otro lado, en el IDE de Arduino se crean las variables de "ssid" (Service Set IDentifier) y "password" donde guardar el nombre y contraseña de la red Wi-Fi a la cual conectarse. Además, se crean las variables "writeAPIkey" y "server" donde almacenar las credenciales obtenidas y el nombre del servidor. Por último se crea un cliente que pueda comunicarse con una dirección IP o servidor ("server") a través de un puerto específico. Se utiliza la librería "WiFiClient" junto con el nombre deseado, "client" en nuestro caso. En este momento, el programa ya se puede comunicar con el servidor y mandar los datos con la función "client.print()".

En el anexo "**??**. Función "Publicación_ThingSpeak()"" se puede observar el programa completo utilizado para la transmisión de datos al servidor, donde se aplica lo anteriormente descrito.

2. Amazon Web Service (AWS)

Antes de comenzar la explicación de la conexión con AWS, aclarar que solo se ha conseguido conectar con AWS a través de la plataforma Node-RED, mediante la comunicación por puerto serie. Para realizar la comunicación por Wi-Fi se necesitaría un dispositivo (como una Raspberry Pi), junto con la plataforma MOSQUITTO que hiciera de nexo entre AWS y el ESP-32 y donde ejecutar Node-RED.

Una vez registrados, para realizar la comunicación con AWS, se selecciona la herramienta "IoT Device Management" dentro de la categoría "Internet of Things". Se crea el objeto o dispositivo que manda los datos a AWS en la pestaña "Administración". Para ello se le asigna un nombre, en nuestro caso "ESP-32" y pulsamos "siguiente".

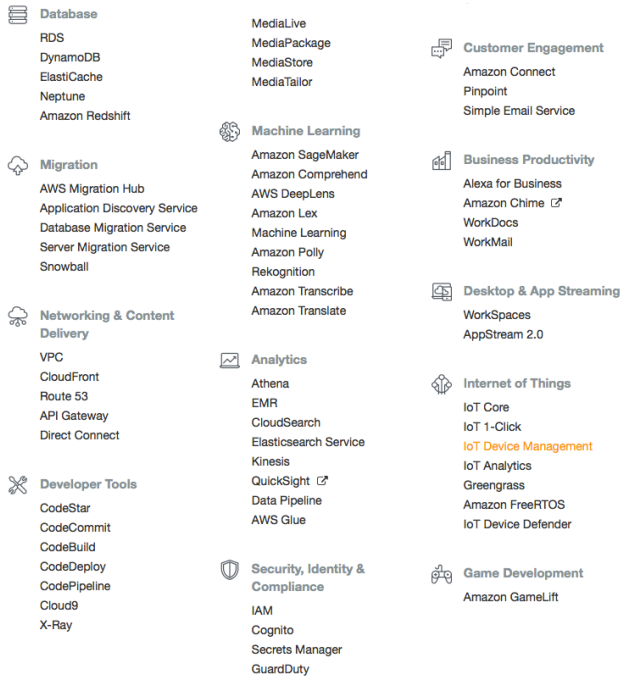

Figura 4.43: Menú de herramientas de AWS.

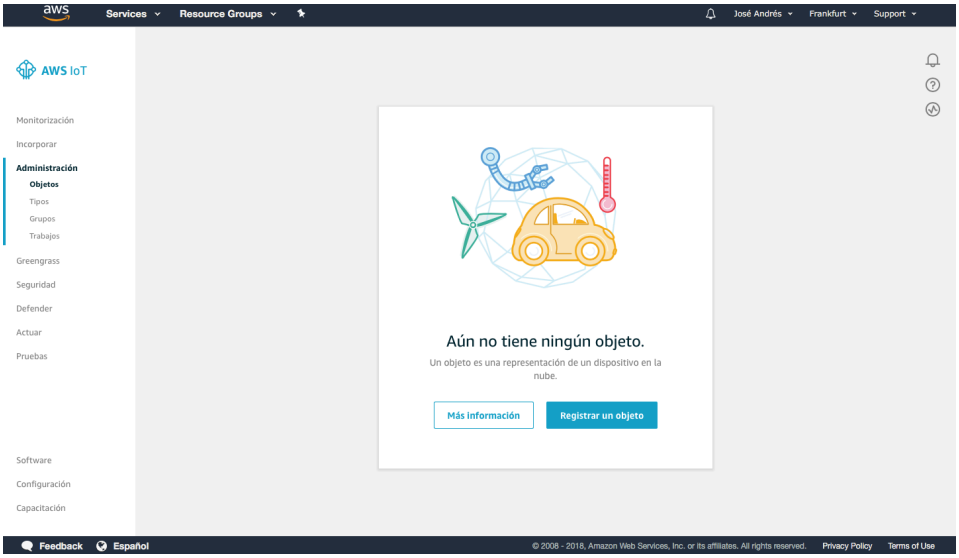

Figura 4.44: Pantalla de IoT Device Management.

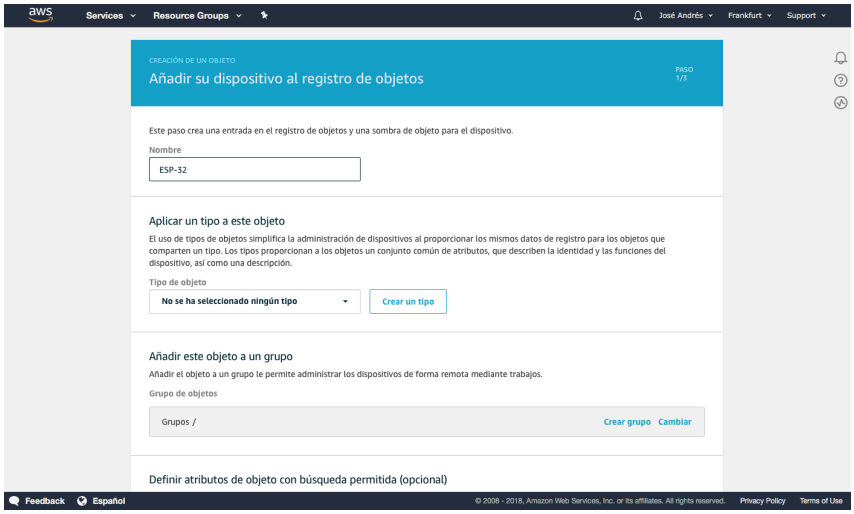

Figura 4.45: Creación del dispositivo u objeto.

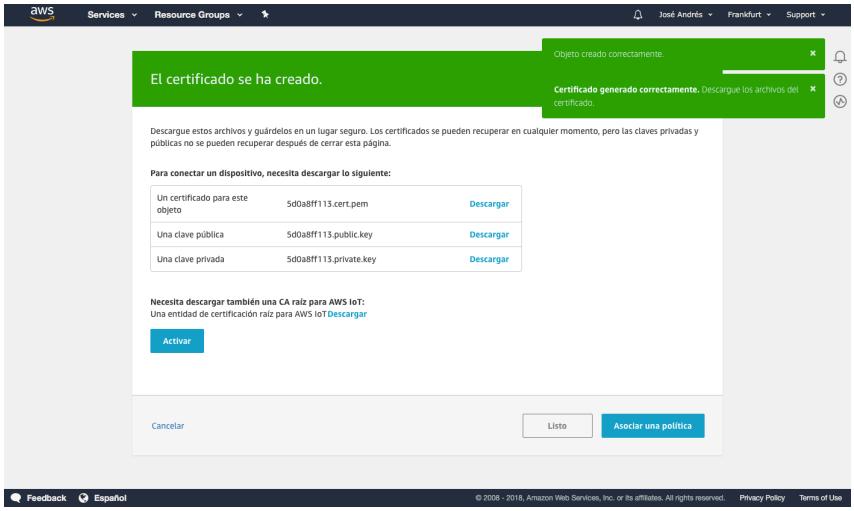

Figura 4.46: Creación de los certificados del objeto.

Seguidamente creamos el certificado del objeto pulsando en la opción "Creación de un certificado con un clic". En ese momento se crea el objeto con sus certificados. Antes de seguir, se debe activar los certificados pulsando "Activar" y descargar tanto el certificado como las claves, pública y privada, y la "Root CA", el cual es una cadena de texto del tipo:

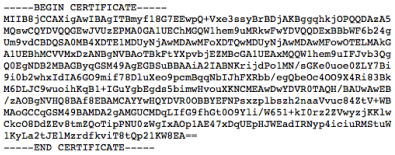

Figura 4.47: Certificado raíz "Root CA".

Para descargar "Root CA", se copia el texto en un archivo de texto sin formato con la extensión ".pem". Una vez descargados los certificados anteriormente descritos, se termina de definir el dispositivo. Lo siguiente es asociar una política de comunicación, donde se establecen los permisos de escritura del dispositivo.

Antes de asociar la política, hay que crearla. Para ello, desde la pestaña "Seguridad", en "Políticas" se pulsa sobre "Crear política". Se le asigna un nombre y unas acciones.

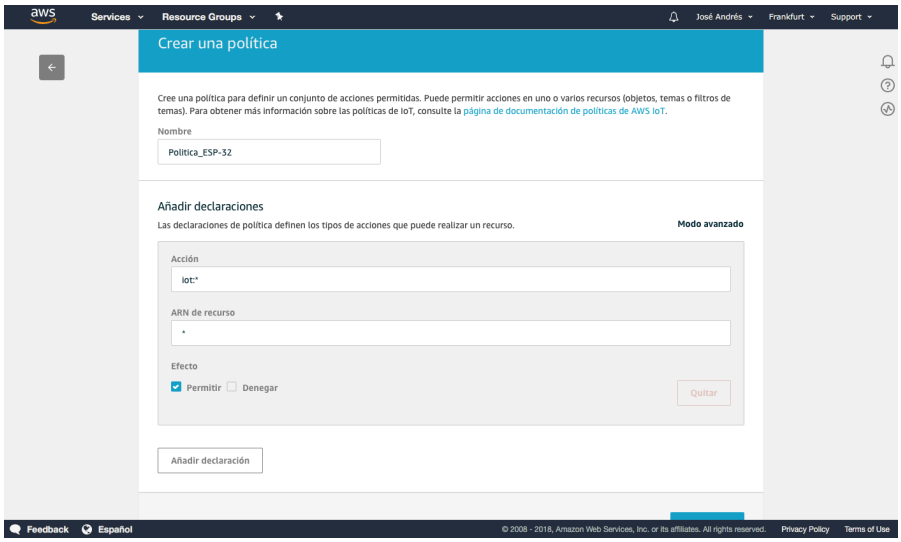

Figura 4.48: Creación de la política de recursos.

Es muy importante realizar la sección "Añadir declaraciones" correctamente, pues de no darle la libertad suficiente al dispositivo, no podrá escribir, o conectarse al servidor. Para ello, asignamos "iot:*" en "Acciones" de manera que se permite realizar cualquier acción sobre AWS y en "ARN de recurso" asignamos "*" de forma que las acciones se puedan realizar para cualquier ARN[8]. De esta forma estamos dando total libertad de recursos al dispositivo. Por último, pulsando sobre los 3 puntos del certificado anteriormente creado, asociamos la política al certificado y el certificado al objeto.

[8]Nombre de recurso de Amazon.

Figura 4.49: Asociación de los certificados a la política y al objeto.

De este modo, el servidor AWS está completamente definido y listo para recibir datos. Sin embargo, se puede observar que se han obtenido certificados que no se implementan de la misma forma en el programa del ESP-32 que como se realizó en ThingSpeak. Para ello, se utiliza la plataforma Node-RED, basado en Node.js y con una programación mediante flujo de datos de forma más visual que las anteriormente vistas.

Para utilizar Node-RED se debe descargar y tenerlo ejecutado durante el tiempo deseado de comunicación. Además, para poder utilizarlo es necesario tener instalado Node.js. Desde la página de oficial de Node-RED ofrecen la opción de descargar ambas herramientas como un paquete, escribiendo en "Terminal" el comando: *sudo npm install -g –unsafe-perm node-red*. Una vez instalados, se puede ejecutar Node-RED escribiendo en Terminal: *node-red*.

```
Last login: Mon Aug  6 12:31:49 on console
192:~ joseandreslorenzorobles$ node-red
23 Aug 23:52:50 - [info]

Welcome to Node-RED
===================

23 Aug 23:52:50 - [info] Node-RED version: v0.18.6
23 Aug 23:52:50 - [info] Node.js  version: v8.11.2
23 Aug 23:52:50 - [info] Darwin 16.7.0 x64 LE
23 Aug 23:52:51 - [info] Loading palette nodes
23 Aug 23:52:54 - [info] Dashboard version 2.9.1 started at /ui
23 Aug 23:52:55 - [info] Settings file  : /Volumes/Macintosh HDD/Users/joseandreslorenzorobles/.
node-red/settings.js
23 Aug 23:52:55 - [info] User directory : /Volumes/Macintosh HDD/Users/joseandreslorenzorobles/.
node-red
23 Aug 23:52:55 - [warn] Projects disabled : set editorTheme.projects.enabled=true to enable
23 Aug 23:52:55 - [info] Flows file     : /Volumes/Macintosh HDD/Users/joseandreslorenzorobles/.
node-red/flows_192.168.1.130.json
23 Aug 23:52:55 - [info] Creating new flow file
23 Aug 23:52:55 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

23 Aug 23:52:55 - [info] Starting flows
23 Aug 23:52:55 - [info] Started flows
23 Aug 23:52:55 - [info] Server now running at http://127.0.0.1:1880/
```

Figura 4.50: Activación de Node-RED por Terminal.

Para acceder al entorno de trabajo, se navega hasta la dirección escrita en la última línea de la Terminal, que es la dirección del servidor en el se que está ejecutando, en el puerto 1880. Una vez en Node-RED se tiene un entorno en blanco donde colocar los bloques.

En la imagen anterior se observa el entorno de programación junto con los bloques necesarios para la obtención y envío de datos a AWS. Para ello, se coloca el bloque "serial", el

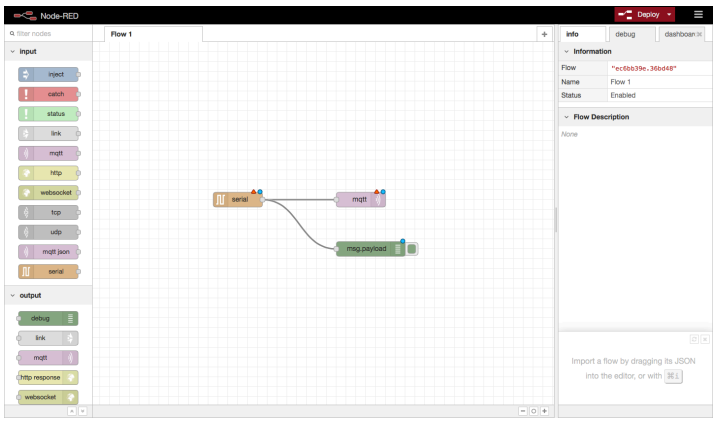

Figura 4.51: Entorno de programación de Node-RED.

cual es el encargado de recibir los datos por puerto serie desde nuestro aparato de medida. Por otro lado, el bloque "mqtt" es el encargado de transmitir los datos por protocolo MQTT hasta AWS. Finalmente el bloque "msg.payload" simplemente representa los datos que llegan desde el bloque "serial".

Cada bloque debe ser configurado de forma específica para el uso que tiene. Para la configuración del bloque "serial", se debe configurar el puerto serie a utilizar, la velocidad de transmisión (baudios), número de bits y el tiempo en el que se divida la entrada de datos.

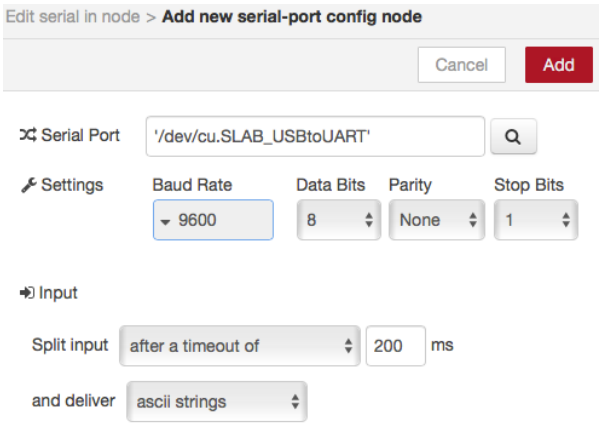

Figura 4.52: Configuración del bloque "serial".

Finalmente se asigna un nombre y se guarda. Seguidamente se configura el bloque "mqtt". Se debe configurar el servidor web al que publicar los datos, el tipo de dato que se publica (topic) y la calidad del servicio (QoS). El valor de QoS varía entre 0 y 2. Con valor 0, el dato se puede enviar una o más veces. Con valor 1, el dato se envía una vez y espera a que llegue un mensaje de respuesta del servidor. Para QoS igual a 2, AWS no ofrece soporte.

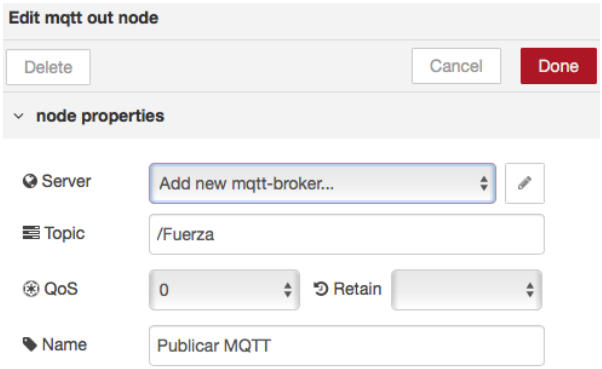

Figura 4.53: Configuración del bloque "mqtt".

Para la configuración del servidor (*Server*) se necesita el puerto de conexión, el "punto de enlace personalizado" proporcionado por AWS y por último configurar los certificados SSL/TLS, obtenidos anteriormente con la creación del objeto. El puerto de conexión con AWS siempre es 8883.

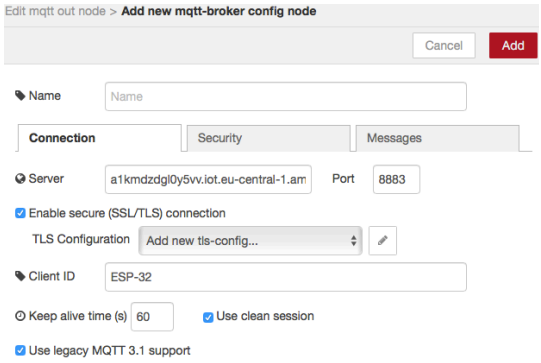

Figura 4.54: Configuración del servidor o broker.

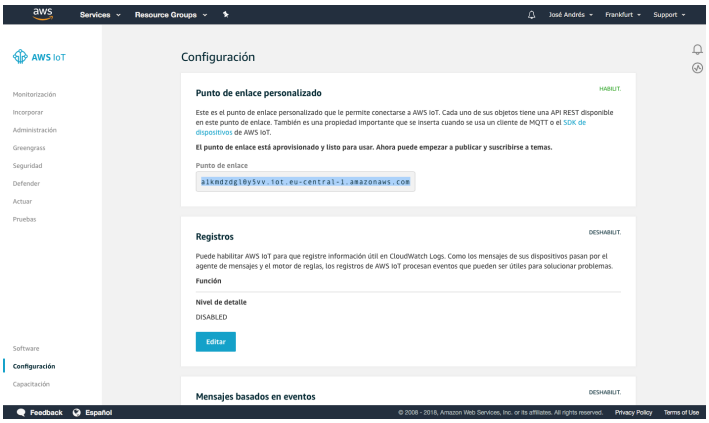

Figura 4.55: Punto de enlace personalizado.

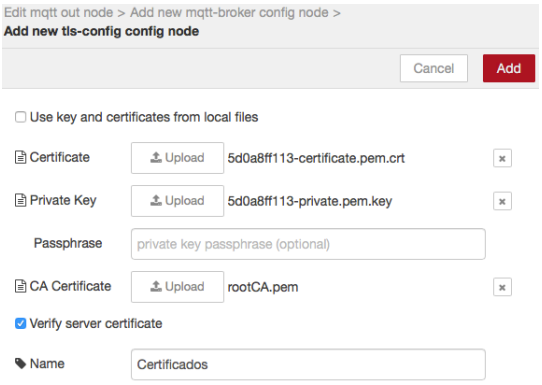

Figura 4.56: Configuración de certificados en Node-RED para la comunicación con AWS.

De esta forma, ya tenemos totalmente definida la comunicación entre Node-RED y AWS, por lo que los datos recibidos por puerto serie a Node-RED son publicados a AWS mediante el protocolo MQTT. Además, gracias al bloque "msg.payload" se podrá comprobar cómo los datos recibidos en AWS son los mismos que los recibidos en Node-RED.

Para mandar los datos desde Arduino por puerto serie, simplemente se utiliza la función *Serial.println()*. Finalmente ejecutando el programa se comprueba que todos los bloques se conectan correctamente.

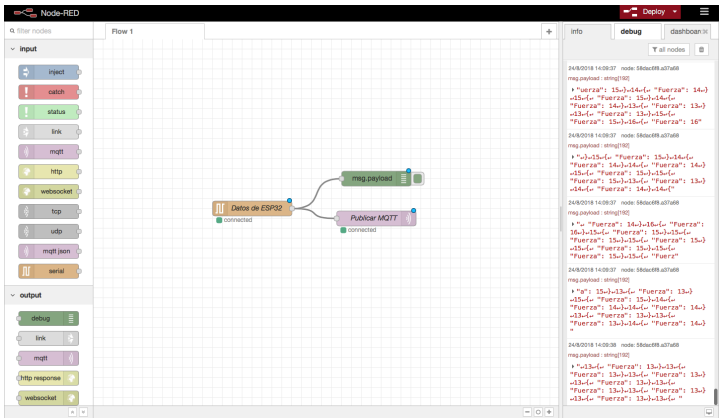

Figura 4.57: Programación de Node-RED acabada.

Se puede observar que en la barra lateral derecha, se muestran los datos recibidos junto con la fecha y la hora. Esos valores son los obtenidos por el bloque "serial" representados sobre el bloque "msg.payload". En el siguente apartado se comprueba que los datos mostrados en Node-RED corresponden con los recibidos en AWS.

Para obtener los datos en AWS, se debe suscribir al *topic* deseado, en nuestro caso "/Fuerza", que es como se nombró en Node-RED. Una vez suscrito, automáticamente comienzan a mostrarse las medidas del equipo en AWS.

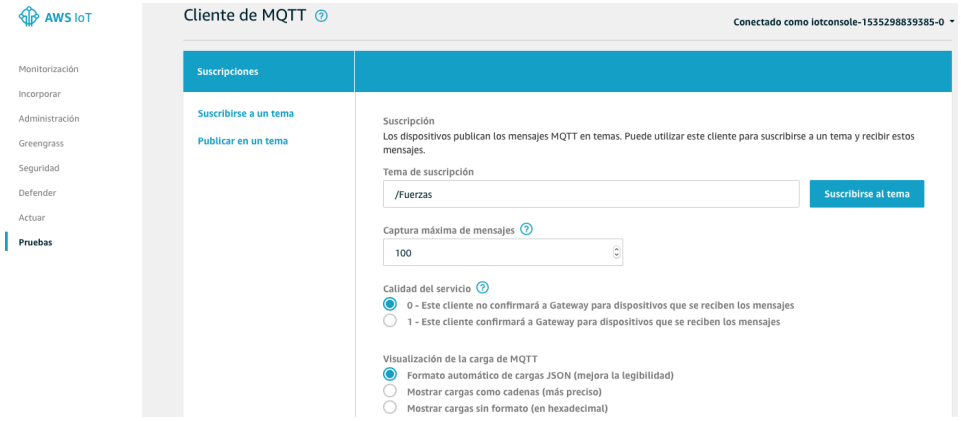

Figura 4.58: Suscripción al *topic* en AWS.

En el capítulo "5. Experimentos y Resultados" se muestra como se representan los datos desde AWS, junto con el resto de servidores.

3. Cayenne

La implementación del equipo en este servidor resulta muy intuitiva y simple, muy parecida a la realizada en ThingSpeak. Una vez registrados en su página web de "myDevices", se elige qué tipo de dispositivo se comunicará con el servidor.

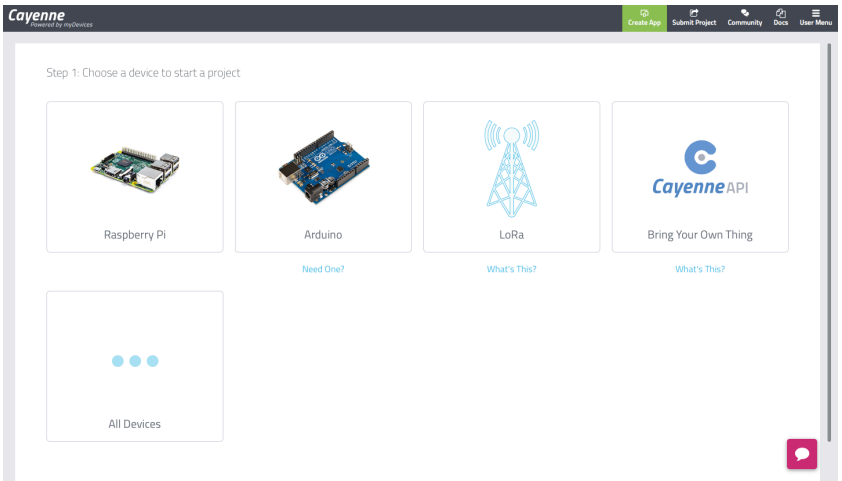

Figura 4.59: Ventana de registro de dispositivo en Cayenne.

Para nuestro caso, el ESP-32, se selecciona "All Devices", donde se encuentra el dispositivo "Generic ESP8266". Se selecciona este, ya que es compatible con el "ESP-32".

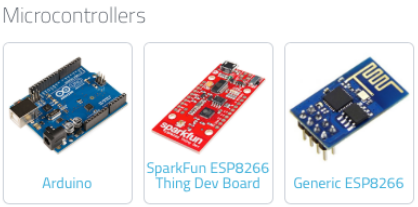

Figura 4.60: Microcontroladores disponibles en Cayenne.

A continuación se muestran los pasos y credenciales necesarias para la correcta asociación y conexión.

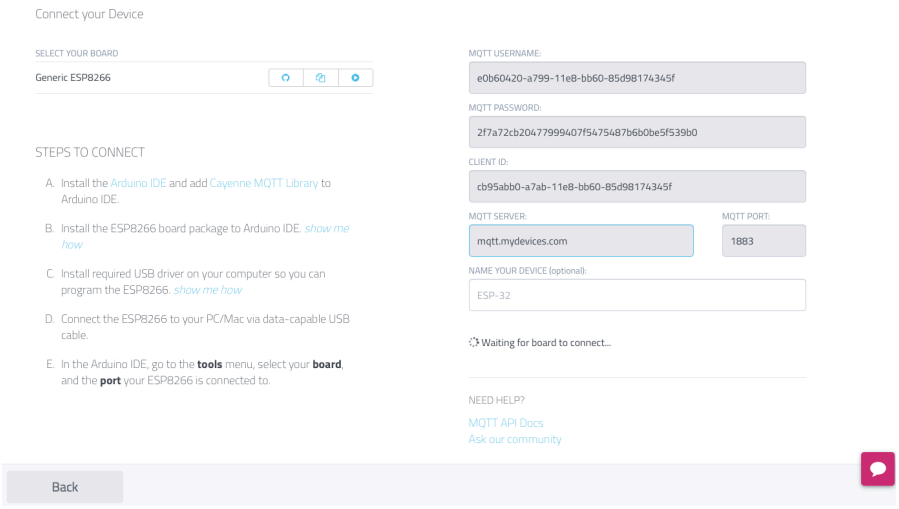

Figura 4.61: Pasos y credenciales para la conexión con Cayenne.

Para efectuar la conexión se descarga y se incluye la librería "CayenneMQTTESP32.h". Se incluye la función "Cayenne.begin(username, password, clientID, ssid, wifiPassword)" en la función "setup()" y "Cayenne.loop()" en la función "loop()". Se guarda en variables las credenciales anteriores. Una vez se cargue el programa con lo anteriormente descrito, automáticamente se asocia el dispositivo al servidor y está listo para usarse.

Para mostrar los datos en el servidor, se utiliza la función *Cayenne.VirtualWrite(Canal, Variable)* donde se introduce la variable que se desea transmitir. Una vez ejecutado, automáticamente se genera un nuevo *widget* donde se muestra el valor. Este *widget* inicialmente es de tipo valor numérico, pero puede modificarse a una gráfica, o indicador. Para disponer de más de un *widget* a la vez, se puede añadir pulsando sobre "Add new..." y en la

pestaña "Device/Widget". Una vez dentro, en el apartado "Custom Widgets" se encuentran los *widget* disponibles.

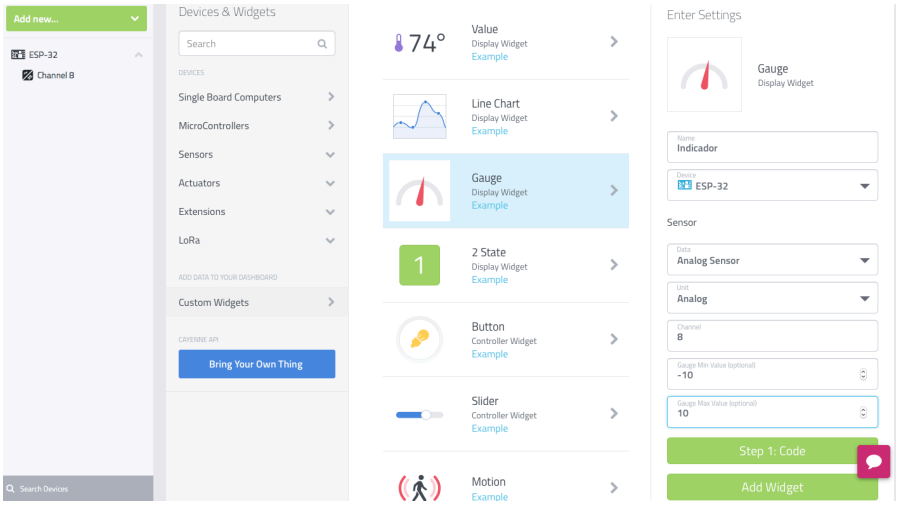

Figura 4.62: Representaciones de medidas disponibles en Cayenne.

Finalmente se añade un indicador de fuerza con el rango de trabajo del equipo (desde -10 a 10N), una gráfica y el indicador numérico generado automáticamente.

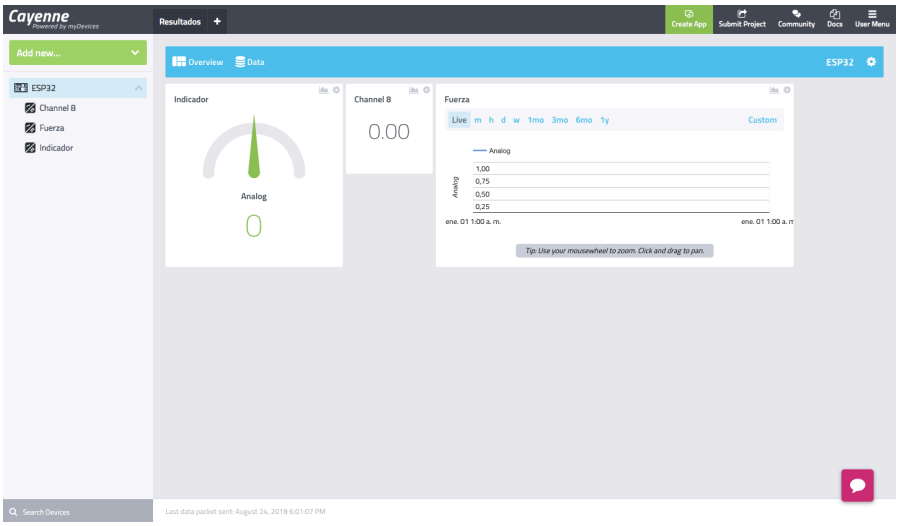

Figura 4.63: Panel de lecturas de Cayenne.

En el anexo "8.2.7.2. Funciión "Publicacion_Cayenne()"" se encuentra el código realizado para la conexión y publicación de datos en Cayenne.

5

Experimentos y resultados

En este capítulo se exponen los distintos experimentos realizados sobre el equipo para determinar la mejor configuración del mismo. Para ello, se realizan experimentos para comprobar cual es el mejor modelo de los obtenidos para la célula de carga. Por otro lado, se comprueba el comportamiento del acelerómetro una vez calibrado, para el cálculo de la orientación. Para cada caso, se determina el error que presenta cada medida y se comprueba que sea menor que un cierto valor de error que se considera admisible. Si el error, en valor absoluto, fuese superior a este valor admisible, no podría considerarse bueno el modelo analizado.

Por otro lado, también se expone el procedimiento y resultados de los experimentos realizados sobre la comunicación de nuestro equipo. Tanto para puerto serie como para los servidores IoT, se comprueba cual de las técnicas de comunicación utilizadas ofrece una mayor capacidad de adquisición y almacenamiento de datos.

Una vez obtenidos los resultados de cada experimento, se decide cual es la configuración más apropiada para lograr lecturas más fieles y una mayor utilidad para nuestro equipo.

Por requerimiento expreso, se desea que las medidas no presenten un error mayor del 3%, por lo que un error superior de este invalidaría el modelo o calibración en cuestión.

5.1 Célula de carga

En este apartado se comprueban los modelos elaborados tras la calibración del sensor de fuerza. Los modelos son obtenidos en el apartado "4.2.1.3. Obtención del modelo matemático", del capítulo "4.2. Calibración". Se comprueba el error que presenta cada uno, y se decide qué modelo es el que se implementa en el programa del procesador.

El experimento que se realiza para cada modelo, consiste en la comprobación de la lectura del equipo para los 40 pesos aplicados en la calibración. Puesto que la medida que ofrece el equipo no es estática ni constante, sino que varía entorno a un pequeño umbral, se toman para cada peso, el valor de lectura medio y el valor que resulte más alejado del real.

El error de cada medida se representa de tres formas, que son error absoluto, error relativo y error estático. El error absoluto representa la diferencia entre la medida real o "ideal" y la lectura obtenida. Se calcula mediante la resta de ambos. Por otro lado, el error relativo representa el porcentaje del error absoluto, respecto a la medida ideal. Se calcula como la razón entre el error absoluto y la medida real, en valor absoluto. Por último el error estático representa la diferencia que tiene el valor medio o estático de la lectura en régimen permanente respecto al valor ideal. Es el error absoluto aplicado al valor medio de la señal y no al más alejado.

$$Error_{absoluto} = X_{lectura} - X_{ideal}$$

$$Error_{relativo}(\%) = \left| \frac{Error_{absoluto}}{X_{ideal}} \right| \cdot 100$$

$$Error_{estático} = X_{medio} - X_{ideal}$$

Asimismo, el error absoluto y relativo que se representan, puesto que se basan en la lectura más alejada registrada, son los máximos errores que se producen. De esta manera se asegura que para el modelo que se elija, el error máximo producido sea menor que el error admisible exigido. Por otro lado, se representa el valor medio y el error estático, para comprobar cuánto se acerca la lectura media o predominante, al valor real.

5.1.1 Modelo basado en la regresión lineal para la nube de puntos

Se examina el modelo basado en la regresión lineal para la nube de puntos (modelo 4.1), cuya ecuación es $y = 1.2269 \cdot x - 2037.9$. Los resultados que se obtienen a través este modelo son:

Regresión lineal nube de puntos											
Peso	Lect. E_{max}	Lect. Med.	E. Absoluto	E. Relativo (%)	E. Estático	Peso	Lect. E_{max}	Lect. Med.	E. Absoluto	E. Relativo(%)	E. Estático
0	-1	0	1		0	0	-2	0	2		0
50	44	46	-6	12	-4	-50	-48	-50	-2	4	0
100	94	96	-6	6	-4	-100	-96	-99	-4	4	-1
150	139	141	-11	7.3333	-9	-150	-154	-152	4	2.6667	2
200	188	190	-12	6	-10	-200	-204	-202	4	2	2
250	236	239	-14	5.6	-11	-250	-247	-249	-3	1.2	-1
300	287	290	-13	4.3333	-10	-300	-297	-299	-3	1	-1
350	334	336	-16	4.5714	-14	-350	-346	-347	-4	1.1429	-3
400	381	384	-19	4.75	-16	-400	-394	-396	-6	1.5	-4
450	437	439	-13	2.8889	-11	-450	-443	-445	-7	1.5556	-5
500	486	489	-14	2.8	-11	-500	-494	-497	-6	1.2	-3
550	532	533	-18	3.2727	-17	-550	-542	-543	-8	1.4545	-7
600	580	583	-20	3.3333	-17	-600	-590	-592	-10	1.6667	-8
650	629	631	-21	3.2308	-19	-650	-639	-640	-11	1.6923	-10
700	678	681	-22	3.1429	-19	-700	-687	-689	-13	1.8571	-11
750	727	728	-23	3.0667	-22	-750	-738	-739	-12	1.6	-11
800	776	778	-24	3	-22	-800	-792	-794	-8	1	-6
850	823	825	-27	3.1765	-25	-850	-839	-841	-11	1.2941	-9
900	874	877	-26	2.8889	-23	-900	-890	-891	-10	1.1111	-9
950	923	925	-27	2.8421	-25	-950	-938	-938	-12	1.2631	-12
1000	971	975	-29	2.9	-25	-1000	-985	-987	-15	1.5	-13

Tabla 5.1: Tabla de resultados del modelo aplicado a la nube de puntos.

Se puede apreciar que se presentan unos resultados muy dispares. En la dirección positiva, el modelo tiene un error muy alto mientras que para la dirección negativa el error se encuentra dentro de los límites admisibles. Que error relativo sea prácticamente constante en gran parte de su dominio, es consecuencia de que el error absoluto aumenta gradualmente con el peso.

A continuación se muestra gráficamente la desviación que sufren las lecturas estacionarias representándolas respecto a los valores de peso reales.

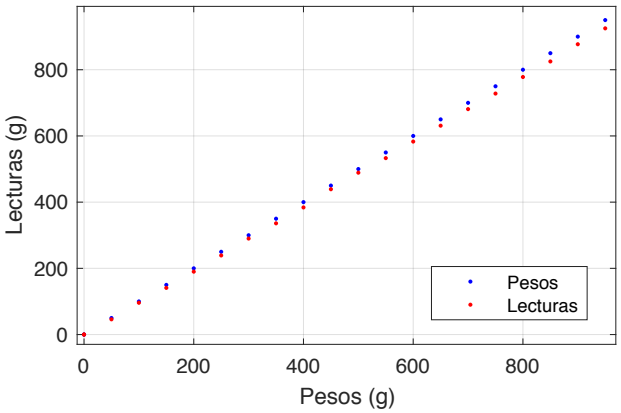

Figura 5.1: Desviación de las lecturas positivas respecto a los valores reales.

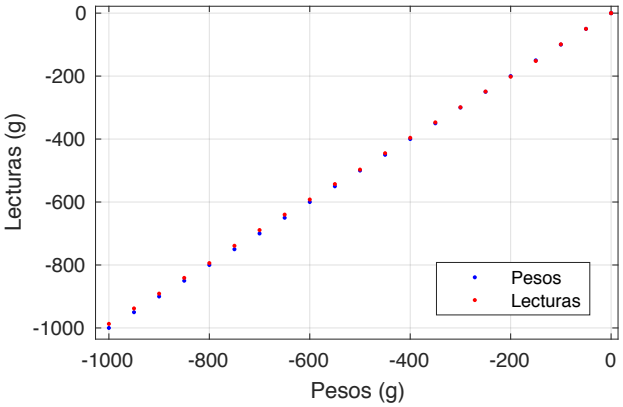

Figura 5.2: Desviación de las lecturas negativas respecto a los valores reales.

5.1.2 Modelo basado en la regresión lineal diferenciando zona positiva y negativa

En esta ocasión se estudia el comportamiento del modelo obtenido mediante la regresión lineal de la zona positiva y negativa por separado. Se han obtenido 2 modelos (modelos 4.2 y 4.3) cuyas ecuaciones son $y = 1.2633 \cdot x - 2117.2$ para la zona positiva y $y = 1.2463 \cdot x - 2058.2$ para la zona negativa. Los resultados aplicando estos modelos son:

Regresión lineal nube de puntos											
Peso	Lect. E_{max}	Lect. Med.	E. Absoluto	E. Relativo (%)	E. Estático	Peso	Lect. E_{max}	Lect. Med.	E. Absoluto	E. Relativo(%)	E. Estático
0	1	0	1		0	0	-1	0	-1		0
50	47	49	-3	6	-1	-50	-48	-50	-2	4	0
100	98	100	-2	2	0	-100	-96	-98	-4	4	-2
150	145	147	-5	3.3333	-3	-150	-153	-152	3	2	2
200	197	198	-3	1.5	-2	-200	-204	-202	4	2	2
250	246	247	-4	1.6	-3	-250	-253	-251	3	1.2	1
300	298	300	-2	0.6667	0	-300	-303	-301	3	1	1
350	344	346	-6	1.7143	-4	-350	-351	-350	1	0.2857	0
400	394	396	-6	1.5	-4	-400	-396	-399	-4	1	-1
450	453	451	3	0.6667	1	-450	-452	-451	2	0.4444	1
500	503	501	3	0.6	1	-500	-504	-502	4	0.8	2
550	553	550	3	0.5454	0	-550	-553	-551	3	0.5454	1
600	603	600	3	0.5	0	-600	-598	-600	-2	0.3333	0
650	648	651	-2	0.3077	1	-650	-649	-650	-1	0.1538	0
700	704	702	4	0.5714	2	-700	-697	-699	-3	0.4286	-1
750	751	751	1	0.1333	1	-750	-749	-750	-1	0.1333	0
800	802	800	2	0.25	0	-800	-806	-803	6	0.75	3
850	852	851	2	0.2353	1	-850	-855	-852	5	0.5882	2
900	905	903	5	0.5556	3	-900	-907	-905	7	0.7777	5
950	954	952	4	0.4211	2	-950	-953	-952	3	0.3158	2
1000	1005	1002	5	0.5	2	-1000	-1003	-1002	3	0.3	2

Tabla 5.2: Tabla de resultados del modelo aplicado diferenciando zona positiva y negativa.

Se puede observar en este caso que los resultados de error máximo se aproximan mucho más a los valores reales que el caso anterior. Además, no presenta una diferencia tan considerable tanto para una dirección de fuerza como para la otra, como en el caso anterior. A continuación se muestra la representación gráfica de las lecturas junto con los pesos reales aplicados:

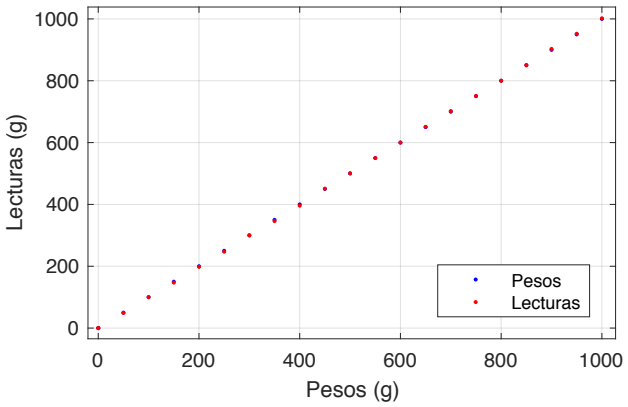

Figura 5.3: Desviación de las lecturas positivas respecto a los valores reales.

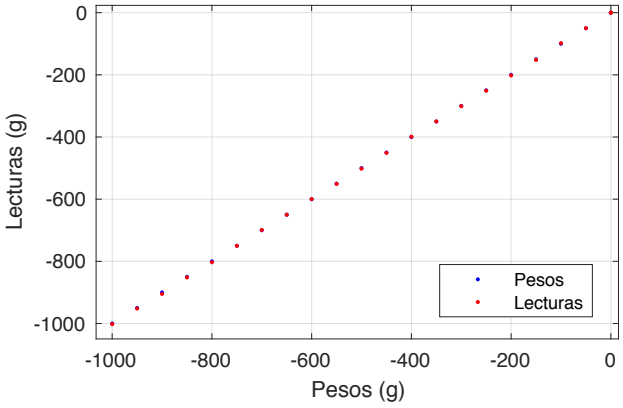

Figura 5.4: Desviación de las lecturas negativas respecto a los valores reales.

En este caso, se aprecia que a diferencia del caso anterior, las lecturas de error máximo son muy próximas a los valores reales. Asimismo, el error absoluto no se incrementa por lo que el error relativo cada vez se va reduciendo con el aumento de peso.

5.2 Acelerómetro

En este apartado se comprueban las medidas de la orientación del equipo realizadas por el acelerómetro. Para ello, se comprueba la lectura para diferentes posiciones concretas del equipo y por otro lado se comprueba la lectura de un giro de 360º en ambos ejes. En la figura 5.5 se muestran los ejes de giro del equipo y los ángulos de lectura que se toman. En esta configuración se considera en condiciones iniciales, cuyos ángulos son de 0º.

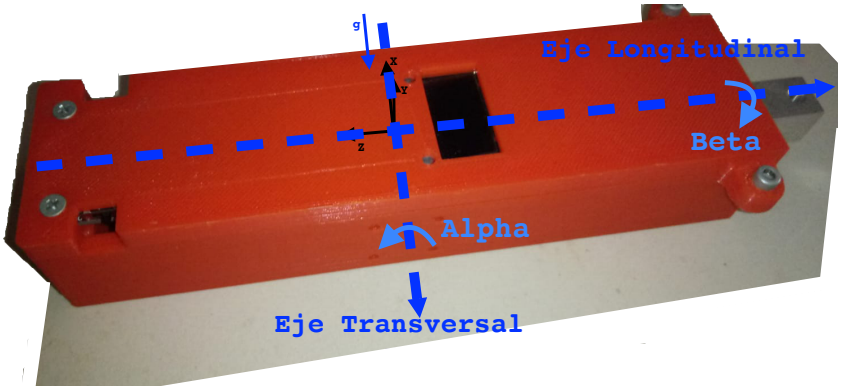

Figura 5.5: Representación sobre el equipo de los ejes y ángulos de orientación.

5.2.1 Experimento estático con posiciones concretas

La única forma de comprobar con exactitud que el ángulo que se introduce es correcto, es mediante ángulos rectos para los distintos ejes. Si el equipo mide correctamente y con poco error para estas posiciones, también lo debe hacer para el resto, pues las componentes de aceleración se compensan de una posición a otra y la operación que realiza el cálculo es la misma.

Ángulos	α	β	Ángulos	α	β	Ángulos	α	β	Ángulos	α	β
α=0^o, β=0^o	0	0	α=0^o, β=90^o	0	90	α=0^o, β=180^o	180	180	α=0^o, β=270^o	0	-90
α=90^o, β=0^o	90	0	α=90^o, β=90^o	90	0	α=90^o, β=180^o	90	0	α=90^o, β=270^o	90	0
α=180^o, β=0^o	180	180	α=180^o, β=90^o	0	-90	α=180^o, β=180^o	0	0	α=180^o, β=270^o	0	90
α=270^o, β=0^o	-90	0	α=270^o, β=90^o	-90	0	α=270^o, β=180^o	-90	0	α=270^o, β=270^o	-90	0

Tabla 5.3: Experimento estático del acelerómetro.

Se observa que muchas de las lecturas que aparecen no corresponde con el valor de entrada realizado. Esto ocurre debido a que existen diversos caminos por los que llegar a una posición exacta. Por ejemplo cuando el equipo se encuentra vertical, α=90^o o α=270^o, el ángulo β no va a

influir en la lectura, independientemente de su valor antes de estar el equipo vertical. Otro caso es cuando el equipo se encuentra horizontal boca arriba, donde α=0^o y β=0^o, mientras que si está boca abajo α=180^o y β=180^o. En definitiva, no hay ninguna lectura realmente incorrecta, sino que se muestra la salida más simple de las posibles y que se cumpla para α y β por separado.

5.2.2 Experimento dinámico para giros completos

En este caso se comprueba que el equipo sea capaz de realizar la lectura para giros completos en ambos ejes. De esta manera también se comprueba que no haya ningún ángulo en todo el rango de trabajo que presente ningún problema o un comportamiento no deseado.

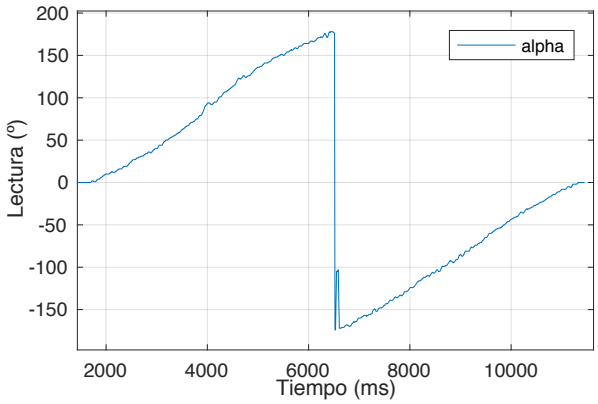

Figura 5.6: Representación del angulo α en el experimento dinámico.

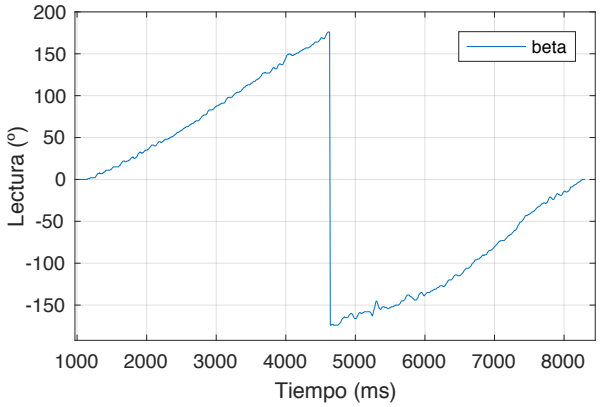

Figura 5.7: Representación del angulo β en el experimento dinámico.

De esta forma, se puede observar que el equipo es capaz de leer en todo el rango de lectura (360^o) para ambos ejes de rotación con las particularidades descritas el experimento 5.2.1. De este modo, se puede considerar que el acelerómetro realiza correctamente las lecturas de orientación, por lo que se ha sido calibrado correctamente.

5.3 Comunicación

En este apartado, se comparan los resultados obtenidos en cada servidor para un mismo experimento. Para que el experimento sea el mismo para cada caso, se realiza el mismo procedimiento para cada uno. Empezando con 0g, coloca inicialmente 200g sobre el equipo, seguidamente otros 100g y por último se quita todo el peso.

5.3.1 Puerto Serie

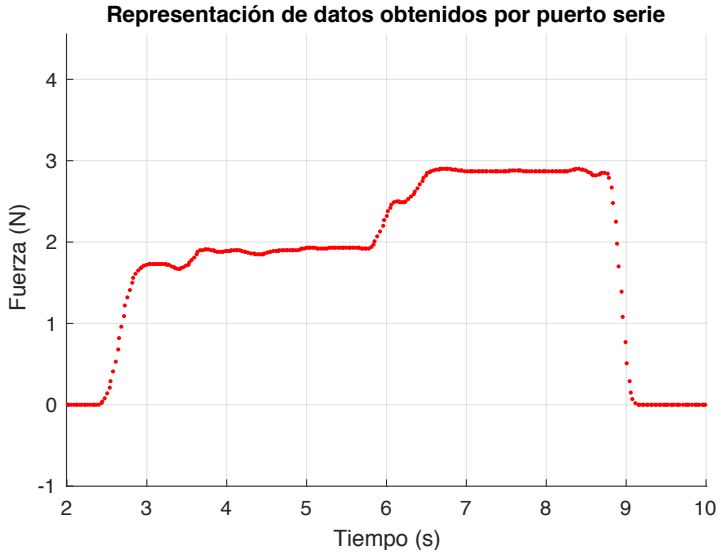

Figura 5.8: Representación de la fuerza obtenida por puerto serie.

En la imagen anterior se muestra cada lectura realizada con un punto rojo. Para el experimento se han tomado 500 lecturas, pero se ha ampliado a la zona de interés.

Puesto que no hay retardos, el tiempo de muestreo en este caso es el que se ha dado por programa de $25ms$.

Se aprecia que la gráfica resultante tienen una forma muy continua sin cambios brusco. Esto es debido a que como el tiempo entre lecturas es tan bajo que es capaz de detectar cualquier pequeño cambio, por rápido que se realice. Esta es una de las grandes ventajas que ofrece la lectura de datos por puerto serie. Por otro lado ni existen límites de datos transferidos ni límites de velocidad. De esta manera los resultados obtenidos son muy fieles a los reales.

5.3.2 ThingSpeak

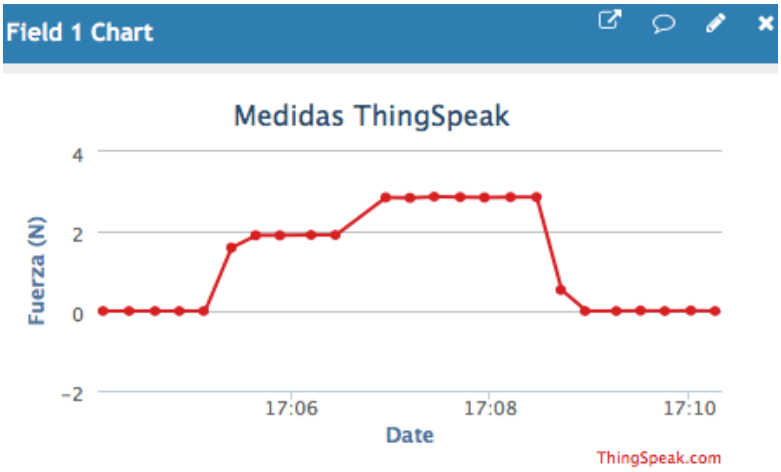

Figura 5.9: Representación de la fuerza obtenida por ThingSpeak.

En la gráfica anterior se representa cada dato por un punto rojo. Se aprecia que los cambios son más bruscos y no tan continuos. Esto es debido a que al publicar un dato cada 15 segundos (Tiempo de muestreo de 15s), no es capaz de detectar los cambios de fuerza de manera gradual.

Por otro lado, el experimento completo dura alrededor de 2 minutos para que se asemeje al realizado por puerto serie que se considera "ideal". Si se hubiese realizado el experimento en el mismo tiempo, apenas habría sido capaz de leer alguna variación de fuerza. Además, presenta un pequeño problema y es que no siempre publica un dato cada 15 segundos. En ocasiones se salta la publicación que debería hacer. Se puede apreciar esto cuando pasa por 3N.

5.3.3 Node-RED+AWS

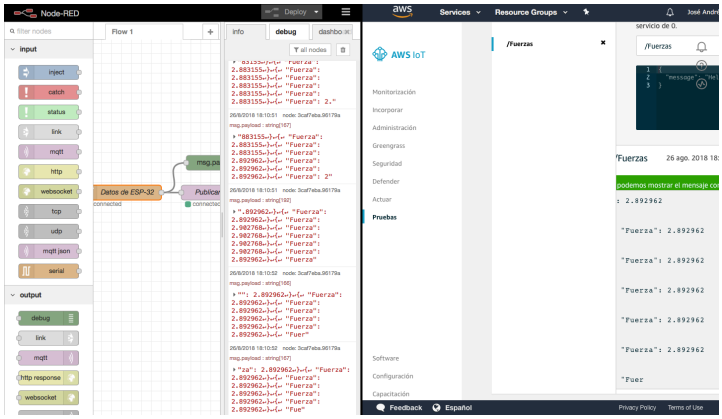

Figura 5.10: Representación de la fuerza obtenida por AWS + Node-RED.

En este caso, no se consigue representar gráficamente los datos leídos. En Node-RED se separan los datos cada 200 milisegundos y son enviados a AWS en paquetes de datos. Por otro lado, no se consigue almacenar los datos que llegan a AWS con el procedimiento explicado en su implementación. Es muy probable que en AWS se puedan representar y guardar los datos de manera muy simple pero como se dijo anteriormente, en las desventajas que presentaba utilizar este servidor, tiene tantas herramientas, que no resulta tan intuitivo para un uso más simple que consista en obtener datos y representarlos en una gráfica.

Por otro lado, la velocidad de lectura en Node-RED y publicación en AWS resulta muy buena, pues no presenta un retraso considerable. Sin embargo, dicha velocidad es proporcionada por la comunicación por puerto serie, por lo que no presenta ninguna novedad frente al primer caso.

5.3.4 Cayenne

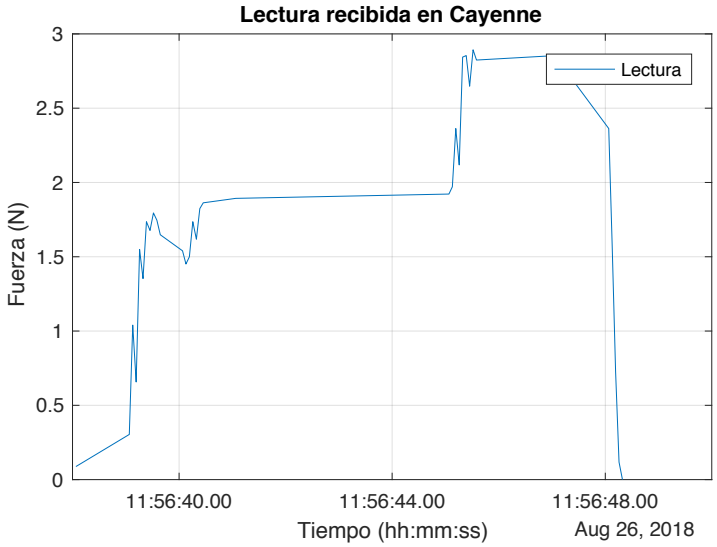

Figura 5.11: Representación de la fuerza obtenida por Cayenne.

En esta gráfica, como se observa si existen muchos cambios bruscos. Sin embargo se puede apreciar que el experimento se ha realizado en 10 segundos aproximadamente. Se observa que ha sido capaz de recibir todos los datos, o al menos los más significativos, en muy poco tiempo.

Un pequeño problema que presenta este servidor es que cuando se desea subir muchos datos de forma rápida, algunos de ellos no se publican en orden real, sino que se cambia su valor con el siguiente o el anterior, provocando alguno de los picos existentes. Aún así, no es algo que ocurra siempre. Por otro lado, como se dijo anteriormente, solo permite publicar 60 datos por minuto, si el experimento es demasiado largo, no es capaz de recopilar todos sus datos.

5.4 Problemas encontrados

Hasta este punto, el equipo de medida es capaz de leer datos del sensor de fuerzas. También realiza lecturas del nivel de batería disponible. Muestra los datos por pantalla. Por otro lado es capaz de leer la orientación del equipo. Y por último es capaz de transmitir los datos, bien a un servidor o por puerto serie a un ordenador. Sin embargo, aún no se han integrado todos los subsistemas en uno.

Para hacerlo posible, se pretende hacer un único programa donde se realicen llamadas a cada función que controla un sensor u operación. A la hora de implementarlos, se observa que el acelerómetro y pantalla no pueden trabajar juntos en un mismo programa.

Se encuentra que mientras no se produzcan llamadas a la función "Lectura_Acelerometro()" la pantalla funciona sin problemas. Por el contrario cuando se llama a esta función, el acelerómetro realiza las lecturas de orientación pero la pantalla queda apagada. Como posible solución, se decide ejecutar la función "Lectura_Acelerometro()" solo durante unos milisegundos, en los que la pantalla podría estar apagada, realizar la lectura de orientación y volver al programa principal para mostrarlos por pantalla. Sin embargo no se percibe ninguna diferencia, pues la pantalla sigue sin funcionar.

Investigando sobre cómo solucionar este problema, se observa que dentro del subprograma "Lectura_Acelerometro()", cuando se habilita la instrucción "Wire_begin()", la pantalla deja de funcionar. La función "Wire_begin()" lo que hace es inicializar la librería "Wire.h" para comenzar la comunicación por I^2C con el dispositivo que se desee. En la función "Lectura_Acelerometro()" sin esta instrucción no se podría realizar la comunicación I^2C entre el sensor y el procesador, por lo que es necesaria.

Como se describe en los apartados 4.1.2 y 4.4, tanto el acelerómetro como la pantalla se comunican por I^2C. Aunque este tipo de comunicación permite al procesador comunicarse con un único dispositivo aunque haya más conectados en los mismos pines I^2C, el acelerómetro no se ha conectado en los pines OLED_SDA y OLED_SCL. El motivo de ello es porque se observa que cuando es conectado a dichos pines, el acelerómetro no funciona y no es reconocido por el microprocesador. Sin embargo, cuando se conecta a los pines 21 y 22, sí es reconocido por el microprocesador y sí se pueden obtener medidas de él. Por esta razón se decide dejar en los pines 21 y 22 el acelerómetro, que son los pines SDA y SCL respectivamente.

De esta manera, una vez que se ejecuta "Wire_begin()", se inicializa y establece la comunicación con los pines 21 y 22, dejando los pines 4 y 5 sin comunicación. Por otro lado no se conoce de la existencia de ninguna función "Wire_end()" o similar, que interrumpa a "Wire_begin()"

o anule sus efectos. Por lo que una vez ejecutado "Wire_begin()" se mantiene durante toda la ejecución del programa.

Puesto que no se consigue que funcionen ambos periféricos a la vez, se debe decidir cual es más importante para el equipo y prescindir de uno de los dos. De este modo, se considera más importante poder leer la fuerza desde el propio equipo de medida, que necesitar siempre estar conectado a un equipo o red Wi-Fi para ver las lecturas de fuerza y orientación.

Si se prescinde de la pantalla, el equipo pierde versatilidad. Sin embargo, si se prescinde del acelerómetro, solamente se pierde la lectura de orientación, que al fin y al cabo no es vital en los experimentos. Por este motivo, se decide prescindir del acelerómetro.

6

CONCLUSIONES

En este capítulo se exponen las conclusiones realizadas una vez terminada la etapa de experimentación. Se desarrollan las conclusiones alcanzadas sobre el modelo y el modo de comunicación a utilizar. Se describe además la solución tomada por utilizar el que mejor características presente.

6.1 Conclusiones de los experimentos sobre los modelos de la célula de carga

Una vez recopilados los resultados de ambos modelos, se puede comprobar que el modelo basado en la regresión lineal diferenciando las zonas de trabajo, 5.1.2, presenta unos errores mucho menores que el basado en la nube de puntos, 5.1.1. Se puede observar que el error relativo máximo para ambas direcciones es mucho menor que el admisible, y por otro lado el error estático es prácticamente nulo.

Esto es debido a que si se toman modelos por zonas, aparte de eliminar la histéresis, se consigue que cada modelo se aproxime más al comportamiento real que si se pretende aproximar para toda la nube de puntos. Por otro lado, se ha comprobado que ambos sentidos de lectura no tienen la misma pendiente. Por lo que no se puede caracterizar utilizando el mismo modelo.

De esta manera, se ha conseguido obtener un modelo del comportamiento de la célula de carga, que cumple la condición del $E_{admisible} < 3\%$ para errores máximos, de hecho, el error absoluto máximo que se encuentra es de ± 7 gramos, pero realmente el error estacionario es prácticamente nulo para casi todo su rango de trabajo.

6.2 Conclusiones sobre la comunicación

Vistos los resultados para cada tipo de comunicación, se observa que por puerto serie no presenta limitación de datos y su velocidad de transmisión es mucho más rápida que por IoT. Sin embargo tiene la limitación de estar conectado por cable mientras se realiza el experimento, por lo que pierde la característica de ser portátil. De este modo, la comunicación por puerto serie no puede ser la única y definitiva.

Por otro lado, los servidores IoT presentan características muy distintas. Por un lado AWS se consideró por la gran variedad de herramientas que ofrece, pero su uso ha sido bastante más complicado de lo esperado. Además no se ha conseguido representar los datos, ni guardarlos en la nube, ni comunicarse sin utilizar el puerto serie. Por lo que no resulta útil esta opción.

El servidor de ThingSpeak tiene el gran problema de solo poder publicar un dato cada 15 segundos. De esta forma las medidas y experimentos no resultan concluyentes. Si todos los experimentos hubiesen tenido la misma duración, ThingSpeak apenas hubiese reconocido variaciones en la fuerza realizada.

Por último el servidor de Cayenne es el que mejor resultados ofrece. Se pueden realizar experimentos sin tiempo de espera entre medidas. Además los resultados obtenidos se asemejan muy bien a los obtenidos por puerto serie, que se pueden considerar como los más fieles.

De esta manera, se incluyen ambos métodos: Comunicación por puerto serie con MATLAB y comunicación con técnicas IoT con Cayenne. De esta forma el usuario podrá elegir cuando utilizar una u otra en función de si se necesita más precisión o autonomía para el experimento.

6.3 Conclusiones generales

Una vez concluidas todas las etapas de montaje, calibración, programación y experimentación, se obtiene un equipo de instrumentación capaz de medir las fuerzas aplacadas sobre el equipo con un error máximo de ± 0.07N. Además se consigue mostrarlas por pantalla en el mismo instante de aplicación y transmitirlas a un equipo externo para su almacenamiento y posterior representación.

El modo de trabajo del equipo consiste en una lectura de fuerza proporcionada por la célula de carga. Seguidamente se procesa la lectura adquirida. El procesado de la señal consiste inicialmente en el filtrado para eliminar ruidos inherentes en ella. Finalmente, se pasa de una magnitud eléctrica en voltios (V), a un valor de fuerza en newtons (N). A continuación se muestra por pantalla y se transmite vía Wi-Fi al servidor de Cayenne donde se representan y almacenan

los datos para finalmente descargarlos.

Por otro lado, se ha desarrollado un programa en MATLAB que realiza dos funciones. Una de ellas es que, una vez conectado por USB al ordenador, automáticamente se enlaza y comienza a representar y almacenar las lecturas en un archivo ".csv". La otra función es la de representar los datos de fuerza respecto al tiempo, almacenados en un fichero ".csv".

De este modo el equipo presenta 2 posibilidades de trabajo. Una de ellas ofrece unos resultados más precisos transmitiendo por el puerto serie, sin interrupciones ni límites en la medida. La otra opción es más versátil y libre, permitiendo guardar los datos en un servidor web vía Wi-Fi. Esta opción sería la ideal, pero presenta limitaciones de velocidad y cantidad de medidas.

Además de la fuerza, se pretendía dar otra funcionalidad al equipo, presentando por pantalla la orientación que tiene en cada momento. Sin embargo, por un problema en la conexión o programación, no se ha conseguido hacer funcionar el acelerómetro y la pantalla del ESP-32 en el mismo programa. Se decide así prescindir de esta lectura, para dar prioridad a la pantalla. Sin embargo, era uno de los objetivos iniciales y el diseño del equipo está totalmente adaptado para disponer de un acelerómetro totalmente funcional e integrado. Por lo que se propone como una posible y futura mejora y ampliación del equipo.

Por último, para utilizar todos los componentes de manera sólida y portátil, pero sin que sufran ningún riesgo, se ha diseñado e impreso en 3D un prototipo de carcasa que sirva como base estructural a los componentes. Con esta carcasa, junto con la batería, el equipo resulta más autónomo y fácil de usar.

Aclarar también que este equipo es un prototipo de un medidor de fuerzas al que le quedan muchas mejoras por añadir. Es decir, es un proyecto que aún no está terminado ni está cerrado a nuevas funcionalidades. Es por ello por lo que los planos que se incluyen en los anexos deben servir de base o guía para aquella persona que decida continuar con el desarrollo del equipo. Del mismo modo, en el CD junto a la memoria, se adjuntan los modelos 3D diseñados en SolidWorks. Se presentan como base para continuar con el proyecto, pudiendo así modificarse o mejorarse los aspectos del diseño que se consideren necesarios.

7

Líneas de Trabajo Futuras

Como se comenta anteriormente, el equipo aún no está terminado y además está abierto a múltiples ampliaciones y rediseños para mejorar tanto la experiencia del usuario, como precisión de las medidas o funcionalidades extra. Para ello se proponen algunas como líneas de trabajo futuras que potenciarían las características y funcionalidad respecto a lo realizado hasta ahora. A continuación se describen algunas de las mejoras o trabajos pendientes que habría que realizar.

La primera de las mejoras podría ser solucionar el problema entre el acelerómetro y la pantalla permitiendo así conocer la orientación del equipo cuando recibe la fuerza y poder mostrarla en cada momento. Esta funcionalidad sería útil para tener mayor exactitud y control en el experimento. Permitiría conocer el ángulo de incidencia de una fuerza aplicada sobre un cuerpo y ver cómo se comporta ante distintas formas de aplicación. Por otro lado, el proyecto está diseñado para disponer de un acelerómetro, tanto por el diseño de la carcasa, como por la conexión, programación y calibracion de este. Por lo que se considera una mejora natural al equipo.

Otra posible mejora podría ser incluir más direcciones de lectura de la fuerza, permitiendo así descomponerla en dos o las tres componentes del espacio. Para ello habría que añadir 2 sensores más, con sus respectivas tarjetas de amplificación, además de calibrar y procesar sus datos. Asimismo, los sensores deberían de tener una disposición concreta para que cada uno lea en una dirección perpendicular a las otras dos. Con la fuerza descompuesta en tres direcciones, se podría conocer exactamente su dirección y magnitud en el espacio. Por ejemplo, a la hora de examinar un cuerpo se podría observar la dirección de mayor fuerza que se opone a la realizada. De este modo se podrían obtener mejores conclusiones de los experimentos.

Otra posible ampliación al equipo sería el de utilizar un sensor con mayor rango, como 5 Kg por ejemplo. De este modo se alcanzarían medidas de hasta $\pm 49N$, dando al equipo un alcance y una utilidad mayor.

Otra posible mejora podría ser la de obtener unas mejores prestaciones en la comunicación por IoT. Una solución podría pasar por encontrar otro servidor gratuito que ofrezca mejores características que Cayenne, especialmente que ofrezca mayor cantidad de lecturas. Otra solución sería utilizar un servidor de pago que ofrezca unas prestaciones que no ofrezca uno gratuito y se adapten a las necesidades del usuario. Otra solución sería crear un servidor IoT basado en Raspberry Pi, Node-RED y Mosquitto al que se conectaría automáticamente nuestro equipo y publica sus datos. En este caso tampoco existirían problemas de cantidad de datos pero sí seguirían existiendo problemas de latencia en el momento de publicar los datos. Además esta solución resulta bastante más laboriosa que la de tomar un servidor ya existente.

Por último, también se propone realizar un modelo 3D más pequeño o ergonómico del elaborado. Así como, utilizar para la impresión 3D otros materiales más resistentes en caso de necesidad y diferentes condiciones de uso. Que soporten mayores temperaturas, que resulte más resistente a impactos o que se estropee menos con el uso. etc.

8

ANEXOS

8.1 Anexo de planos de modelo 3D

En esta sección se adjuntan los planos obtenidos tras el diseño en 3D de las piezas del prototipo de la carcasa.

8.1.1 Plano de pieza superior

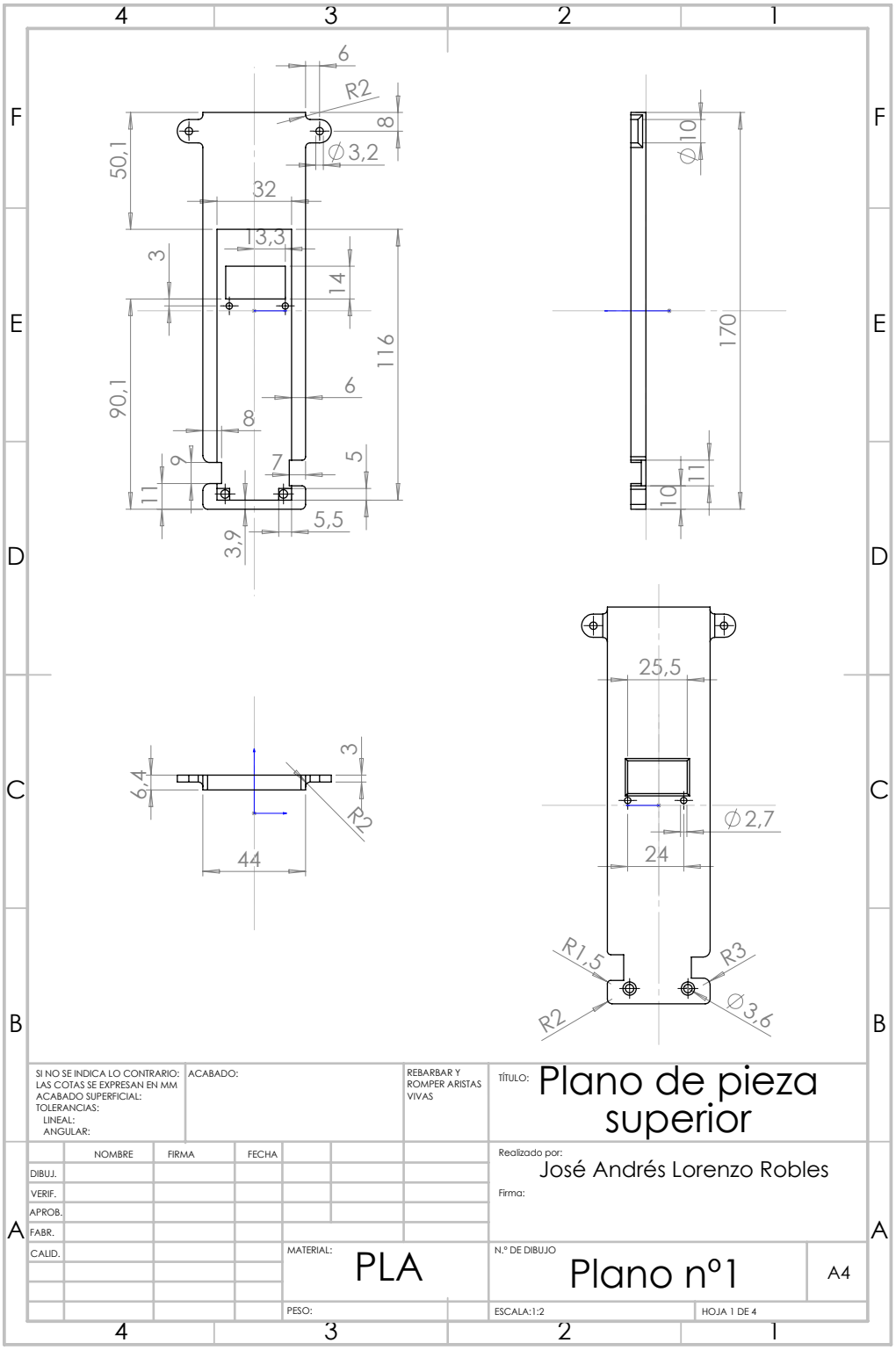

Figura 8.1: Plano de pieza superior de la carcasa.

8.1.2 Plano de pieza inferior

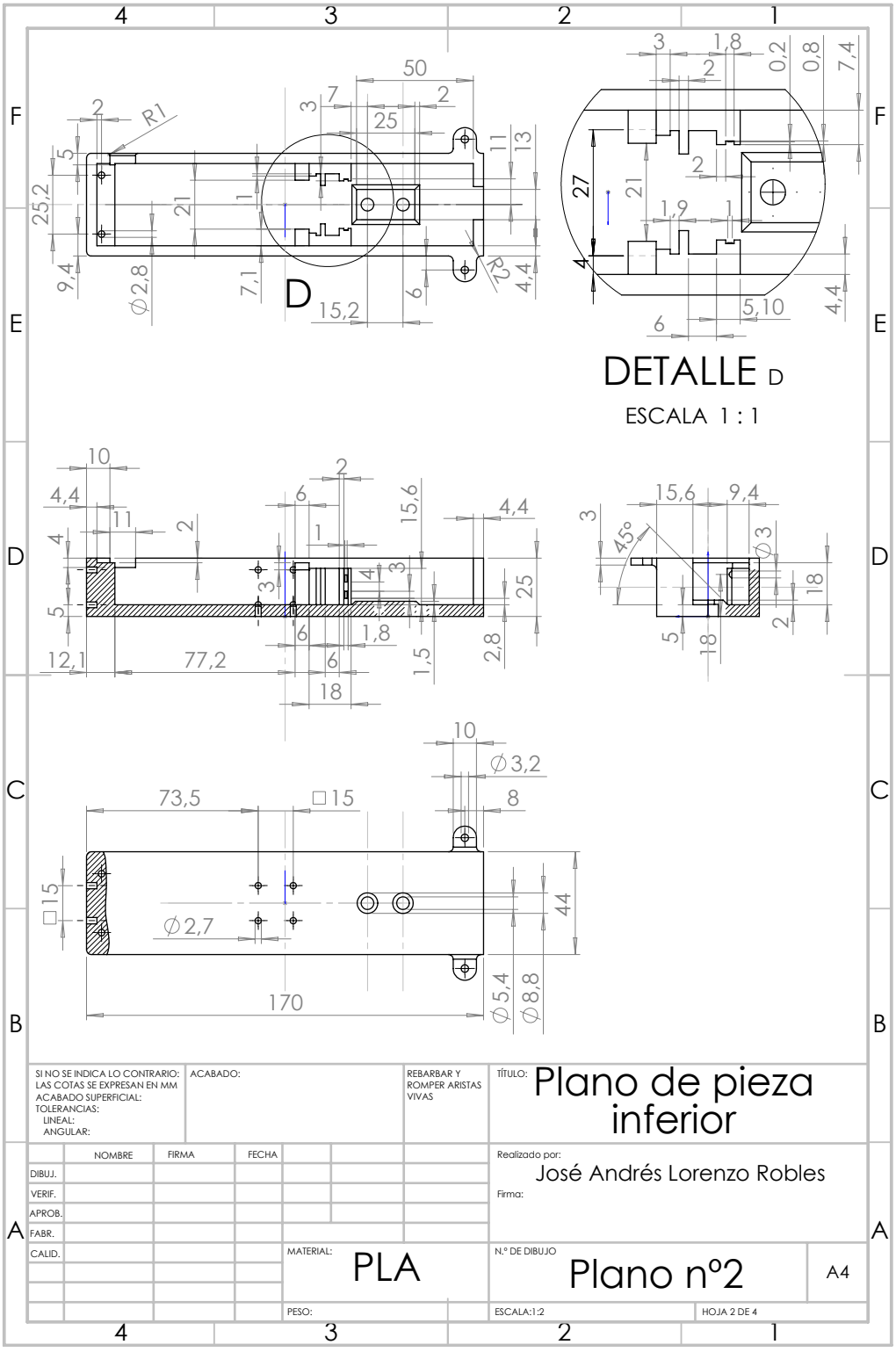

Figura 8.2: Plano de pieza inferior de la carcasa.

8.1.3 Plano de botón

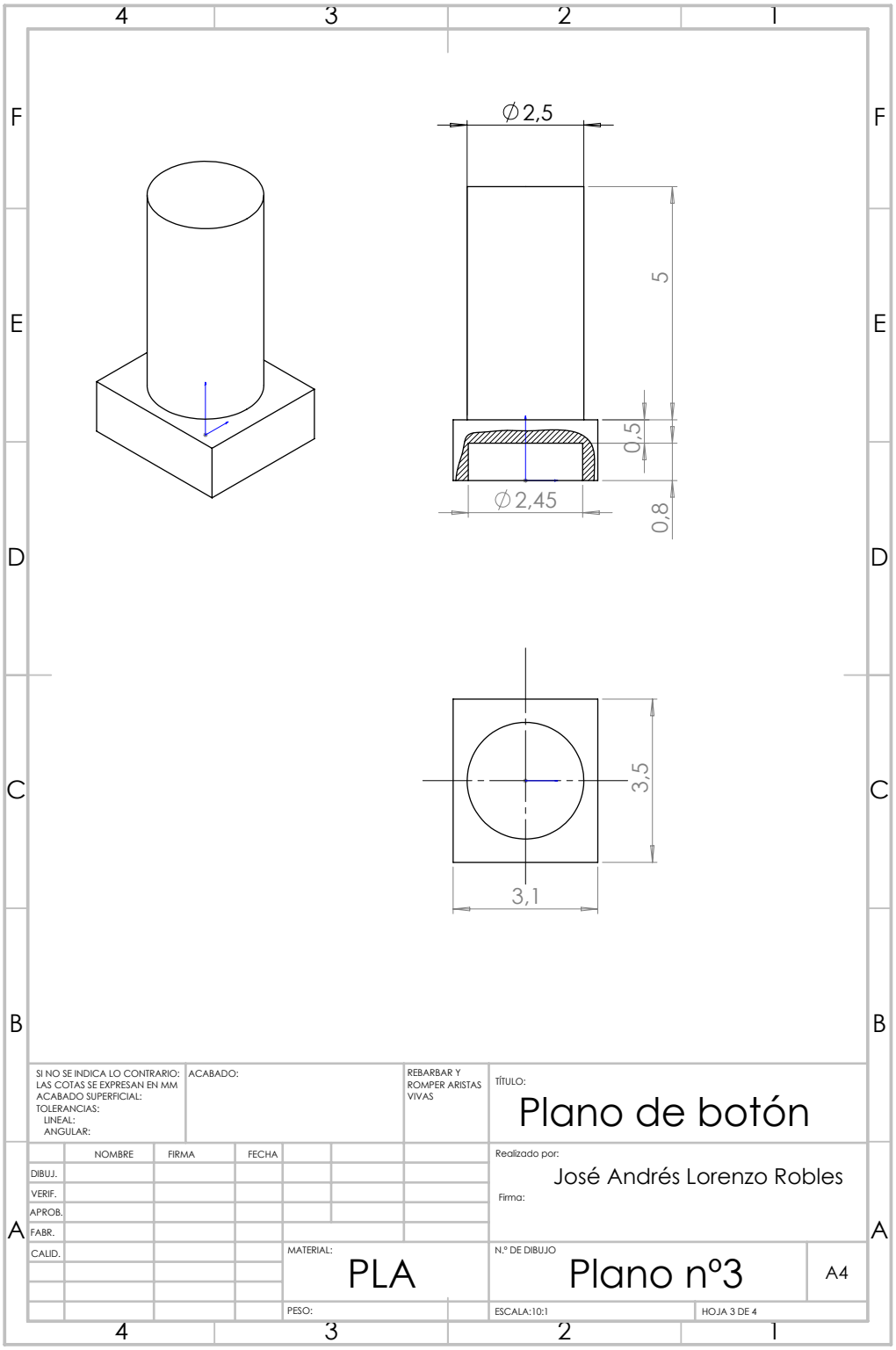

Figura 8.3: Plano del botón.

8.1.4 Plano de base

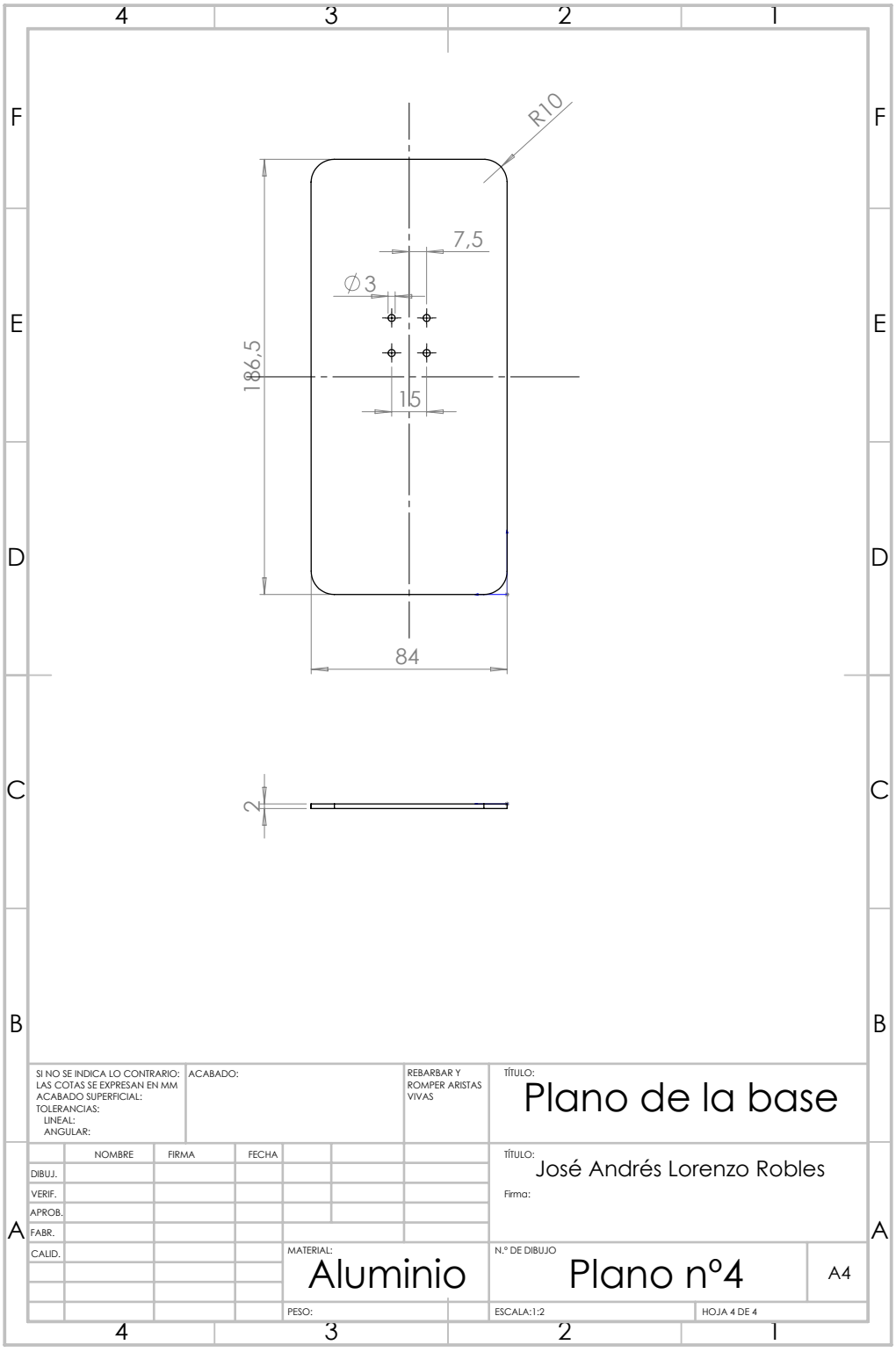

Figura 8.4: Plano de soporte fijo.

8.2 Anexo de programación del procesador

En esta sección se incluye el programa principal y las funciones que controlan lo sensores o realizan operaciones muy especializadas.

8.2.1 Programa implementado

8.2.1.1 Programa principal "Programa_equipo"

```
// LIBRERÍAS
#include <Wire.h> // Librería para comunicarse con el acelerómetro por I2C.
#include <WiFi.h> //Librería para utilizar el módulo WiFi.
#include "MedianFilterLib.h"  // Librería para el filtro de la mediana.
#include "SSD1306.h"      // Librería para utilizar la pantalla OLED.
#include <CayenneMQTTESP32.h> //Librería para comunicarse con el servidor Cayenne.

// DEFINICIÓN DE RED WI-FI
char ssid[] = "XXXXXXXXXXXXXX"; // Nombre de la red Wi-Fi
char wifiPassword[] = "XXXXXXXXXXXX"; // Contraseña


// AUTENTIFICACIÓN PARA SERVIDOR CAYENNE PARA COMUNICACIÓN MQTT
char username[] = "5f591db0-90e0-11e8-b90f-f7e25f9cf9c8";
char password[] = "06800159fcd67886a54cf5c76ee6ee3b533efd59";
char clientID[] = "ebc04fd0-929d-11e8-8fba-979723b0b5e0";

// DEFINICIÓN DE BOTON "BOOT"
#define boton 0

// DEFINICIÓN DE LA PANTALLA
SSD1306 display(0x3c, 5, 4);          // Definimos la pantalla.

// DEFINICIÓN DE ACELERÓMETRO
#define    MPU9265_ADDRESS           0x68   //Dirección del MPU9265 para I2C
#define    ACC_FULL_SCALE_4_G        0x08   //Escala de trabajo del acelerómetro

// DEFINICIÓN DE VARIABLES
float lectura_fuerza;
int lectura_bateria;
int inicio=70;
```

```
int posicion=0;
unsigned long tiempo = 0;
unsigned long t_actualizado = 0;
unsigned long t_delay =25;

int alpha_acel;
int beta_acel;

int sensor;
int salida;

float Fuerza;
float Fuerza_anterior;
bool Fuerza_nula;
int Ultima_Subida;

bool Boot;
bool Comenzado=false;

MedianFilter<int> FiltroMediana(3);   // Definimos el filtro de tamaño 3.

void setup() {

  Serial.begin(9600);

  // CONEXIÓN A INTERNET
    Cayenne.begin(username, password, clientID, ssid, wifiPassword);

  // INICIALIZAMOS ENTRADAS
    pinMode(boton,INPUT); // Pin de entrada del boton Boot
    pinMode(A6,INPUT);    // Pin de entrada de CJMCU-333
    pinMode(A13,INPUT);   // Pin de entrada de Div.Tensión


  // INICIALIZAMOS PANTALLA
    display.init(); // Inicializamos pantalla.
    display.setContrast(255);
    display.setFont(ArialMT_Plain_16); // Estilo.
```

```
    display.setTextAlignment(TEXT_ALIGN_LEFT); // Origen en la izquierda.

  // CONFIGURACIÓN DEL ACELERÓMETRO
    I2CwriteByte(MPU9265_ADDRESS, 28, ACC_FULL_SCALE_4_G);
}

void loop() {

  // PANTALLA DE INICIO
  while (Comenzado==false){
    display.clear(); // Limpiar pantalla
    display.displayOff();
    display.displayOn();
    display.setFont(ArialMT_Plain_10); // Estilo.
    display.drawString(posicion, 0, "Pulse botón derecho para");
    display.drawString(posicion, 15, "comenzar.");
    display.drawString(posicion, 35, "Pulse botón izquierdo para");
    display.drawString(posicion, 50, "resetear.");
    display.display(); // Muestra lo que haya en buffer
    Boot=digitalRead(boton);
    if (Boot==LOW){
      Comenzado=true;
    }
  }

  // COMIENZA A MEDIR
  display.clear(); // Limpiar pantalla
  display.setFont(ArialMT_Plain_16); // Estilo.
  Comenzado=true;
  Cayenne.loop();
  tiempo = millis();

  lectura_bateria=Lectura_Bateria();
  //Lectura_Acelerometro();
  if (tiempo>t_actualizado+t_delay){
    t_actualizado = tiempo;
    lectura_fuerza=Lectura_Fuerzas();
    if(Fuerza_max<lectura_fuerza and lectura_fuerza>0){
```

```
      Fuerza_max=lectura_fuerza;
    }
    if(Fuerza_min>lectura_fuerza and lectura_fuerza<0){
      Fuerza_min=lectura_fuerza;
    }

// REPRESENTACIÓN
  Serial.println(lectura_fuerza);

  // Mostrar datos por pantalla
  if(inicio==0){
    display.clear(); // Limpiar pantalla
    display.displayOff();
    display.displayOn();
    display.setFont(ArialMT_Plain_16); // Estilo.
    display.drawString(posicion, 0, String(lectura_fuerza));
    display.drawString(posicion+7*((String(lectura_fuerza)).length()), 0, " N");
    //display.drawString(posicion, 25, "Alpha= " + String(alpha_acel)); //Puesto que no
    //funciona, se sustituye por el valor máximo de fuerza
    display.drawString(posicion, 25, "Valor Máx.= " + String(Fuerza_max));
    //display.drawString(posicion, 45, "Beta= " + String(beta_acel)); //Puesto que no func
    //se sustituye por el valor mínimo de fuerza
    display.drawString(posicion, 45, "Valor Mín.= " + String(Fuerza_min));
    display.setFont(ArialMT_Plain_10); // Estilo.
    display.drawString(100, 0, String(lectura_bateria) + "%");
    display.display(); // Muestra lo que haya en buffer
  }

  // MIENTRAS SE INICIALIZAN LAS VARIABLES
  else{
    display.clear(); // Limpiar pantalla
    display.displayOff();
    display.displayOn();
    display.drawString(posicion, 0, "Inicializando...");
    display.display(); // Muestra lo que haya en buffer
  }

  // PUBLICACIÓN DE CADA DATO EN EL SERVIDOR DE CAYENNE
```

```
  if ((millis()-Ultima_Subida>65)and(abs(Fuerza)>0.05 or (Fuerza==0 and Fuerza_nula==false))
   and abs(Fuerza-Fuerza_anterior)>=0.02 and (inicio==0)){
      Cayenne.virtualWrite(8, Fuerza);
      Ultima_Subida=millis();
      Fuerza_anterior=Fuerza;
      if(Fuerza_anterior==0){
        Fuerza_nula=true;
      }
      else Fuerza_nula=false;
      }

  }
}
```

8.2.2 Función "Filtro_EMA()"

```
int sensor_EMAa;
int sensor_EMA;
float alpha=0.45;

int Filtro_EMA(int sensor){
  sensor_EMAa=sensor_EMA;
  sensor_EMA=alpha*sensor+(1-alpha)*sensor_EMAa;

  return(sensor_EMA);
}
```

8.2.3 Función "Filtro_EMA_Acel()"

```
int sensor_EMAa_Acel;
int sensor_EMA_Acel;
float alpha_Acel=0.4;

int Filtro_EMA_Acel(int sensor){
  sensor_EMAa_Acel=sensor_EMA_Acel;
  sensor_EMA_Acel=alpha_acel*sensor+(1-alpha_Acel)*sensor_EMAa_Acel;

  return(sensor_EMA_Acel);
}
```

8.2.4 Función "Lectura_Acelerómetro()"

```
// LECTURA ACELERÓMETRO //

// DEFINICIÓN DE PARÁMETROS
#define    MPU9265_ADDRESS          0x68

#define    ACC_FULL_SCALE_2_G       0x00
#define    ACC_FULL_SCALE_4_G       0x08
#define    ACC_FULL_SCALE_8_G       0x10
#define    ACC_FULL_SCALE_16_G      0x18

// DEFINICIÓN DE VARIABLES
float ax_conv;
float ay_conv;
float az_conv;

float X;
float Y;
float Z;

int a;
int b;

int a1;
int a2;
int a3;
int a4;
int a5;
int a_fMED;
int a_fMED_EMA;
int a_fMED_EMA_MEDIA;

int b1;
int b2;
int b3;
int b4;
int b5;
int b_fMED;
```

```
int b_fMED_EMA;
int b_fMED_EMA_MEDIA;

// FUNCIÓN DE LECTURA
void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data){
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.endTransmission();
    Wire.requestFrom(Address, Nbytes);
    uint8_t index = 0;
    while (Wire.available())
        Data[index++] = Wire.read();
}

// FUNCIÓN DE ESCRITURA
void I2CwriteByte(uint8_t Address, uint8_t Register, uint8_t Data){
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.write(Data);
    Wire.endTransmission();
}

void Lectura_Acelerometro(){
 Wire.begin();
 // LECTURAS DEL ACELERÓMETRO EN LOS EJES X,Y,Z.
  uint8_t Buf[14];
  I2Cread(MPU9265_ADDRESS, 0x3B, 14, Buf);

  int16_t ax = -(Buf[0] << 8 | Buf[1]);    // Invierto los valores porque el acelerómetro
  int16_t ay = -(Buf[2] << 8 | Buf[3]);
  int16_t az = -(Buf[4] << 8 | Buf[5]);

  //Wire.endTransmission();


  // CONVERSIÓN DE LAS LECTURAS.
    // Conversión a gravedad (m/s^2)
      ax_conv=-ax*9.80665/8207.066225+0.027654;
```

```
  ay_conv=-ay*9.80665/8226.456689-0.170296;
  az_conv=az*9.80665/8185.788079-0.144666;
  if(ax_conv<0.3){
    ax_conv=round(ax_conv);
  }
  if(ay_conv<0.3){
    ay_conv=round(ay_conv);
  }
  if(az_conv<0.3){
    az_conv=round(az_conv);
  }

  // Redondeo al tercer decimal de los valores de aceleración
    ax_conv=round(ax_conv*1000);
    ay_conv=round(ay_conv*1000);
    az_conv=round(az_conv*1000);

    ax_conv=ax_conv/1000;
    ay_conv=ay_conv/1000;
    az_conv=az_conv/1000;

// Conversión de aceleración a orientación
  Y = RAD_TO_DEG * (atan2(-ay_conv, -az_conv));
  X = RAD_TO_DEG * (atan2(-ax_conv, -az_conv));
  Z = RAD_TO_DEG * (atan2(-ay_conv, -ax_conv));

  a = RAD_TO_DEG * (atan2(az_conv, ax_conv));
  b = RAD_TO_DEG * (atan2(ay_conv, ax_conv));

  if(a==180 and b!=180){
    a=a-180;
  }

  if(b==180 and a!=180){
    b=b-180;
  }

// Filtrado de las señales de orientación
```

```
        a_fMED=FiltroMediana.AddValue(a);
        a_fMED_EMA=Filtro_EMA_Acel(a_fMED);
        a5=a4;
        a4=a3;
        a3=a2;
        a2=a1;
        a1=a_fMED_EMA;
        a_fMED_EMA_MEDIA=(a1+a2+a3+a4+a5)/5;

        b_fMED=FiltroMediana.AddValue(b);
        b_fMED_EMA=Filtro_EMA_Acel(b_fMED);
        b5=b4;
        b4=b3;
        b3=b2;
        b2=b1;
        b1=b_fMED_EMA;
        b_fMED_EMA_MEDIA=(b1+b2+b3+b4+b5)/5;

        alpha_acel=a_fMED_EMA_MEDIA;
        beta_acel=b_fMED_EMA_MEDIA;

}
```

8.2.5 Función "Lectura_Bateria()"

```
// LECTURA DE BATERIA

int salida_bateria;

int Lectura_Bateria(){
  int bateria=analogRead(A13);    // Leemos el pin del D.Tensión

  if (bateria>3200){
    int bateria_llena=3700;   // Valor máximo de la lectura de la bateria desconectado.
    salida_bateria=100*bateria/bateria_llena;
    if (salida_bateria>100) salida_bateria=100;
  }

  else if (bateria>3100 and bateria<3200){
    salida_bateria=100;
  }

  else {
    int bateria_llena=3030;   // Valor máximo de la lectura de la bateria conectado.
    salida_bateria=100*bateria/bateria_llena;
    if (salida_bateria>100) salida_bateria=100;
  }
  return(salida_bateria);
}
```

8.2.6 Función "Lectura_Fuerzas()"

```
// LECTURA SENSOR FUERZA

#define boton 0
//int inicio;
int cero;
int entrada;
//int salida;

int sensor_media;
int sensor10=0;
int sensor9=0;
int sensor8=0;
int sensor7=0;
int sensor6=0;
int sensor5=0;
int sensor4=0;
int sensor3=0;
int sensor2=0;
int sensor1=0;

int salida4=0;
int salida3=0;
int salida2=0;
int salida1=0;

int salida_fMED;
int salida_fMED_EMA;
int salida_fMED_EMA_MEDIA;

int contador_Boot=0;
unsigned int tiempo_Boot=0;

float Lectura_Fuerzas(){
  sensor=analogRead(A6);  // Señal de salida del sensor.

  // CÁLCULO DE FUERZA
```

```
entrada=sensor;

// Condicionantes para tarar solo cuando se pulse el botón "Boot" durante un instante.
Boot=digitalRead(boton);
if(Boot==LOW and contador_Boot==0){
  contador_Boot++;
  tiempo_Boot=millis();
}
else if(Boot==HIGH){
  contador_Boot=0;
  tiempo_Boot=0;
}

// Función de tara de la medida
if((tiempo_Boot!=0 and tiempo-tiempo_Boot<100) or inicio!=0){
  cero=entrada;
  inicio--;
  if (inicio<0) inicio=0;
}


if(entrada<cero-1){     // Fuerzas en sentido contrario al positivo del sensor.
  salida=(entrada-cero)*1.2463;
}

else if (entrada>cero+1){    // Fuerzas en mismo sentido que el positivo del sensor.
  salida=(entrada-cero)*1.253;
}

else{
  salida=0;
}

// FILTROS

  // Filtro mediana
    salida_fMED=FiltroMediana.AddValue(salida);
```

```
        // Filtro EMA a la señal filtrada con mediana
          salida_fMED_EMA=Filtro_EMA(salida_fMED);

            // Filtro media móvil a la señal filtrada con EMA y mediana
            salida4=salida3;
            salida3=salida2;
            salida2=salida1;
            salida1=salida_fMED_EMA;
            salida_fMED_EMA_MEDIA=(salida1+salida2+salida3+salida4)/4;

  // CONVERSIÓN DE PESO (g) A FUERZA (N)
    Fuerza=(9.80665*salida_fMED_EMA_MEDIA)/1000; //Cálculo de la fuerza.
    if(inicio!=0) Fuerza=0; //Mientras inicializa, la salida se hace cero.

  return(Fuerza);
}
```

8.2.7 Programas no implementados

A continuación se adjuntan los programas que se han utilizado para demostrar la conexión a los distintos servidores, pero que finalmente no han sido implementados.

8.2.7.1 Función "Publicacion_ThingSpeak()"

En la declaración de variables del programa principal, se incluye:

```
// Configuración WiFi
const char* ssid= "XXXXXXXXXXXXXX";
const char* password="XXXXXXXXXXXX";
WiFiClient client;

// Seteo de datos para ThingSpeak.com
const int channelID = 486457;//Identificación del canal (se obtiene de ThingSpeak.com)
String writeAPIKey = "RWP7JL0T59M2Y164"; //Clave de escritura API para su canal
// (se obtiene de ThingSpeak.com)
const char* server = "api.thingspeak.com";
const int intervalo = 15000; // Publicar datos cada 15 segundos( el mínimo
// es 15 segundos en la versión de prueba)
```

En la función "setup()" debe estar incluido:

```
// CONEXIÓN A INTERNET
  WiFi.begin(ssid,password); //Conexión a red WiFi
  pinMode(GPIO2,OUTPUT); //Pin GPIO2 salida
  while (WiFi.status()!=WL_CONNECTED){
    delay(500);
  }
```

Y en la función "Publicacion_ThingSpeak()":

```
void Publicacion_ThingSpeak(float Fuerza){
// Si el cliente está conectado al servidor "server", en el puerto 80 y han pasado
    //15 segundos, puede publicar un dato.
    if (client.connect(server, 80) and tiempo-tiempo_subida>intervalo) {
      tiempo_subida=tiempo;
    // Construir el cuerpo de solicitud de API
    String body = "field1=";
           body += String(Fuerza);
    client.print("POST /update HTTP/1.1\n");
```

```
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + writeAPIKey + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(body.length());
    client.print("\n\n");
    client.print(body);
    client.print("\n\n");
  }
  client.stop();
  //Genera un pequeño parpadeo indicando se ha enviado un dato
  digitalWrite(GPIO2, HIGH);
  delay(200);
  digitalWrite(GPIO2, LOW);
}
```

8.2.7.2 Función "Publicacion_Cayenne()"

En la declaración de variables del programa principal, se incluye:

```
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

// Configuración WiFi
char ssid[] = "MIWIFI_2G_q9rh";
char wifiPassword[] = "4r29s5tihvha";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "e0b60420-a799-11e8-bb60-85d98174345f";
char password[] = "2f7a72cb20477999407f5475487b6b0be5f539b0";
char clientID[] = "cb95abb0-a7ab-11e8-bb60-85d98174345f";
```

En la función "setup()" debe estar incluido:

```
// CONEXIÓN A INTERNET
    Cayenne.begin(username, password, clientID, ssid, wifiPassword);
    //Esta función establece la conexión WiFi con el servidor.
```

En la función "loop()" debe estar incluido:

```
Cayenne.loop(); //Realiza la comunicación (manda cada dato)
```

Y en la función "Publicacion_Cayenne()":

```
void Publicacion_Cayenne(float Fuerza){

    if ((millis()-ultima_subida>65) and
    (abs(Fuerza)>0.05 or (Fuerza==0 and Fuerza_nula==false)) and
    abs(Fuerza-Fuerza_anterior)>=0.02 and (inicio==0)){
      Cayenne.virtualWrite(8, Fuerza);
      ultima_subida=millis();
      Fuerza_anterior=Fuerza;
      if(Fuerza_anterior==0){
        Fuerza_nula=true;
      }
      else Fuerza_nula=false;
      }
  }
```

REFERENCIAS

[1] Félix Armando Fermín Pérez y Jorge Guerra Guerra. Internet de las cosas. *Perspectiv@s*, 10(11):45–49, 2017.

[2] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtt-s — a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pages 791–798, Jan 2008.

[3] National Instruments. Guía de acondicionamiento de señales para ingenieros. *The International Journal of Robotics Research*, 28(23):6–7, 8 2008.

[4] Sergio Gómez. El gran libro de solidworks. *México: Editorial Marcombo, Alfaomega*, 2008.

[5] Luís Llamas. Implementar un filtro de media móvil rápido en arduino.

[6] Luís Llamas. Filtro paso bajo y paso alto exponencial (ema) en arduino.

[7] Luis Llamas. Implementar un filtro mediana móvil rápido en arduino.