Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

DISEÑO INTEGRADO DE UNA PLATAFORMA LOW-COST PARA MONITORIZADO DE ESTRUCTURAS

Autor: D. Pedro Blanco Fragua
Tutor: D. Álvaro Magdaleno González
Tutor 2: D. Antolín Lorenzana Ibán

Valladolid, mes, 2023

RESUMEN

Los sistemas de adquisición de datos son un elemento esencial en el entorno de la ingeniería, proporcionando las medidas del sistema sobre las que trabajar y experimentar. Sin embargo, los sistemas comerciales, si bien precisos, son costosos económicamente.

Por ello, el presente trabajo tiene como objetivo desarrollar un prototipo de sistema de medida que sea capaz de tratar la señal de células de carga y potenciómetros lineales y mostrar las medidas a un usuario cualquiera de forma accesible y en tiempo real.

El propósito es crear un sistema menos costoso que los sistemas comerciales de este tipo y que pueda ser utilizado en medida y control de estructuras. Para ello, utilizaremos plataformas, programas y componentes de relativa fácil accesibilidad.

Palabras clave

Sensor, tiempo real, Raspberry, PCB, Eagle, Fusion360

ABSTRACT

Data acquisition systems are an essential element in engineering environment, providing the measurements from the system on which to work and experiment. However, the commercial system, although precise, they are economically expensive.

Thus, the present work has a goal to develop a prototype of measurement system capable of processing the signal from load cells and lineal potentiometers and showing the measurements to any user in real time and an accessible way.

The purpose is to create a lower cost system than the commercial systems that can be used in the measure and control of structures. For it, we will use platforms, programs and components of relative accessibility.

Keywords

Sensor, real time, Raspberry, PCB, Eagle, Fusion360

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto

El presente trabajo de fin de máster o TFM se desarrolla en el Departamento de Construcciones Arquitectónicas, Ingeniería del Terreno y Mecánica de los Medios Continuos y Teoría de Estructuras como conclusión a los estudios del Máster en Ingeniería Industrial.

Como tal, se trata de un proyecto transversal en áreas de la ingeniería industrial que cubre aspectos principalmente de electrónica, programación y diseño, aplicados a usos del entorno de la ingeniería mecánica en el control de estructuras.

El trabajo se ha realizado, especialmente toda la sección práctica y experimental, en laboratorio de dinámica de estructuras de la EII de la Uva, sede de Paseo del Cauce y gracias a los medios ofrecidos por el Departamento para el desarrollo.

1.2. Objetivos del proyecto

El objetivo del presente TFM es el desarrollo de un prototipo de sistema de adquisición de datos orientado a células de carga y potenciómetros lineales con capacidades de visualización en tiempo real, de adaptación a distintos sensores y escalabilidad, a fin de ser utilizado en control y medida en el tiempo de estructuras

Es decir, desarrollar un sistema que pueda tomar las señales eléctricas de los sensores, convertirlos en una señal digital con la que poder trabajar y mostrarle a un usuario dicha información de forma interpretable al tiempo que se está generando. Al tiempo que el sistema tenga flexibilidad para aumentar el número y tipo de sensores de forma cómoda.

Se tendrá como referente de partida el sistema comercial Sirius, con el que se asegurará la compatibilidad de sensores (entre Sirius y nuestro prototipo) y del que trataremos de imitar capacidades, pero por un coste de equipo mucho menor.

Se prestará una especial atención al fácil acceso de los datos para cualquier usuario, por lo que se buscará una forma de basar la aplicación en web para que cualquiera con navegador pueda hacerlo. Además, dichos datos deben ser actualizados en tiempo real.

Para ello, se escogerá un sistema de tratamiento de datos, un procesador que ejerza de puente entre los sensores utilizados y el usuario humano para entregarle a este último la información importante de la forma más clara posible.

Además, se diseñará un sistema electrónico para el tratamiento de la señal de sensores previa al procesamiento. El objetivo de diseño de dicho circuito electrónico será emplear componentes comerciales y el diseño de una PCB (*Printed Circuit Board*) para conseguir una señal estable y así como que el sistema diseñado sea fácilmente reproducible.

Finalmente, se diseñará una envolvente para todo el prototipo en su conjunto, es decir, procesador y PCB, para protegerlo de daños y agentes externos. Esta envolvente se ideará para su impresión en 3D, como apoyo a la reproducibilidad del sistema.

1.3. Metodología del trabajo

Para conseguir los objetivos descritos en el anterior subapartado, se hará uso de programas de diseño electrónico y de modelado 3D, de plataformas y ayudas de código libre que faciliten alcanzar dichos objetivos, de la información proporcionada por fabricantes de diferentes componentes, además de herramientas físicas, como son impresora 3D, multímetros y fuente de alimentación de tensión variable.

Como se presentaba en el contexto, estas herramientas se pusieron a disposición en el laboratorio de dinámica de estructuras, en dónde se llevó a cabo enteramente la parte práctica y pruebas realizadas sobre el prototipo que se describirán a lo largo del trabajo. A lo largo del mismo se fueron tomando notas y registros de las ideas, pruebas y acciones acometidas, que culminan en redacción de esta memoria de TFM.

El proceso de trabajo, como es la naturaleza de un proceso de desarrollo de prototipo como el que se ha planteado, es puramente iterativo. Es decir, se plantea una posibilidad, se investigan las bases sobre ella, se hace una aproximación o simulación de ser posible y se prueba. La forma de progresar es, por tanto, prueba y error, identificando los fallos y corrigiéndolos según los recursos disponibles, en medida de lo posible.

Es por ello que muchas de las partes se han desarrollado en paralelo, o se ha identificado un problema en una etapa posterior del desarrollo que ha obligado a volver atrás y modificar una parte que ya se había considerado trabajada previamente.

Por último, comentar que no se ha querido dejar en un diseño simplemente teórico, si bien, junto con algunas simulaciones es la base del trabajo, y que se ha trabajado sobre componentes y piezas reales hasta que se ha logrado un prototipo funcional.

1.4. Estructura del trabajo

Como consecuencia directa del proceso iterativo que se explicaba en el apartado anterior, se habla en esta memoria de puntos que pueden tener relación bien con apartados anteriores, bien con posteriores, por esta naturaleza del desarrollo. Es posible que durante una lectura inicial pudiera haber puntos que no resulten claros, pero se espera que, con la lectura del documento completo, las dudas queden resueltas.

Se comenzará dando en una breve introducción a la referencia principal de este TFM, para después entrar en la parte de programación, explicando la plataforma utilizada y el desarrollo del programa de cabecera. En esta parte se incluye un segmento que no llegó al prototipo final, pero se considera importante referenciar como contrapunto que comparar a la solución finalmente escogida, de más sencilla implementación, que también aparece completamente explicada.

Seguidamente, se recorre el proceso de diseño electrónico de una PCB que cubra las necesidades del prototipo y sustituya las conexiones de prueba que son utilizadas hasta ese momento. Concluye la sección de diseño con el proceso para crear una caja que albergue PCB y procesador conjuntamente.

Como última parte, se presentan los resultados alcanzados durante el desarrollo, así como conclusiones alcanzadas y propuestas a futuro, finalizando así el presente TFM.

2. Estado de la técnica: Sirius

Se presenta en el presente apartado de estado de la técnica solamente este aparato, ya que es el que vamos a utilizar como referencia para el desarrollo de nuestro equipo.

Este, en su variante comercial modular, que se muestra en la figura 1, es el sistema que se emplea en el laboratorio de dinámica de estructuras de la EII de la UVa para la adquisición de datos precisa en los experimentos de estructuras que se llevan a cabo. Y esto se debe a la gran capacidad de recogida de datos del Sirius, con una frecuencia de muestreo muy elevada y así como a su gran precisión, dada por su conversor ADC (*Analog-Digital Converter* o conversor analógico-digital) de 24 bits.

Figura 1: Sirius modular [1]

Esto, sumado a su capacidad para aceptar prácticamente cualquier sensor lo vuelve ideal como sistema de cabecera en un taller de investigación. Sin embargo, su elevado coste (entorno a los 11260 €), hace que no sea tan idóneo para otros usos más sencillos, como pudieran ser unas prácticas de laboratorio.

Es por ello que usaremos el sistema Sirius como referencia para nuestro proyecto, pero buscando como objetivo un prototipo más económico, que, además, respete la compatibilidad de sensores con Sirius. Es decir, que los sensores con conectores adaptados para ser leídos por Sirius (principalmente potenciómetros lineales y células de carga de puente completo) puedan ser leídos tanto por Sirius como pro nuestro prototipo.

2.1. Características

Otras características destacables de Sirius son [1]:

- Conversores ADC de 24 bits.
- Frecuencia de muestreo de 200 kS/s (*kiloSamples per second*).
- Dos conversores ADC por cada canal de entrada, uno de ganancia media y otro de ganancia alta, para asegurar la conversión del rango completo de la señal. La estructura de estos puntos se muestra en la figura 2.
- Muy alto aislamiento galvánico entre canales y de canal a tierra, útil para la medición de altas tensiones y de tensiones en modo común.

- Posibilidad de configurar salidas analógicas para generación de funciones, aplicaciones de control, acondicionamiento de señales o reproducción de datos previamente grabados.
- Sincronización de datos con diferentes frecuencias de muestreo.
- Protección contra polvo IP50 e impermeabilidad IP67, que, sumado a la carencia de ventilador, lo hace perfectamente resistente a entornos industriales.

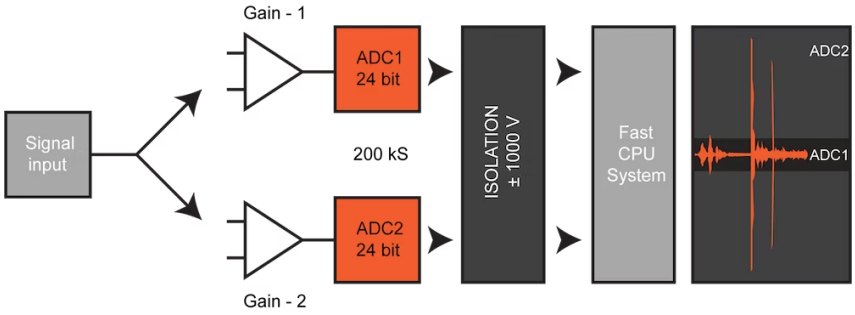


Figura 2: Esquema de entradas Sirius [1]

2.2. Entrada DB9

Para la entrada de sensores al bloque Sirius, la conexión se debe realizar a través de un conector tipo DB9, de los que el terminal hembra, representado en la figura 3, se encuentra en el Sirius. Esto es porque Sirius tiene asignados a cada uno de los pines una función concreta para la alimentación y muestreo de datos de los sensores. Estas funciones de los pies se presentan en la tabla 1 inferior.

1	*Exc+*	Excitación positiva del sensor
2	*In+*	Salida positiva del sensor
3	*Sns-*	Alimentación negativa del sensor
4	*GND*	Referencia a tierra
5	*R+*	Resistencia de Shunt
6	*Sns+*	Alimentación negativa del sensor
7	*In-*	Salida negativa del sensor
8	*Exc-*	Excitación negativa del sensor
9	*TEDS*	Integrado TEDS de Sirius

Tabla 1: Pines DB9 Sirius [2]

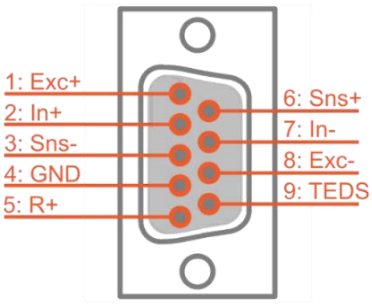


Figura 3: Distribución de pines entrada Sirius [2]

Esto es importante de tener en cuenta para poder asegurar la compatibilidad entre Sirius y nuestro prototipo. No deben de existir interferencias eléctricas entre uno y otro, ya que el prototipo desarrollado debe de poder aceptar los sensores preparados con conector DB9 para conectarse al Sirius sin necesitar de ningún cambio. Para ello, los pines de dicho prototipo deben de cumplir la misma función que en el Sirius en cuanto a la alimentación y recepción de señales del sensor.

Los dos tipos de sensores que vamos a conectar son potenciómetros lineales, que, a efectos prácticos, se conectan igual que cualquier otro tipo de potenciómetro; y células de carga. Las células de carga funcionan utilizando un puente de Wheatstone completo, con cuatro resistencias variables (galgas extensiométricas) según la fuerza ejercida sobre la célula. Dentro del programa asociado Sirius, se puede ver el conexionado que exigido para estos dos tipos (potenciómetros y puente completo) de sensores, con los puentes del cableado en el interior del propio conector DB9. Resumido en un esquema eléctrico en la figura 4 encontramos que es:

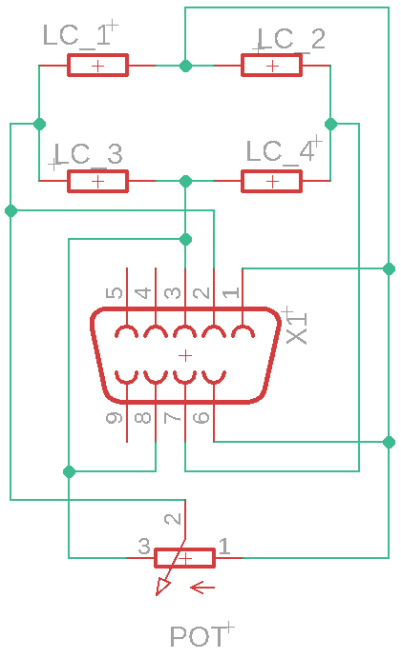


Figura 4: Esquema de conexionado DB9

Este será por tanto el esquema de conexionado que se deba de tener en cuenta en el prototipo para asegurar la compatibilidad de los conectores con Sirius. Como se puede ver, quedan tres pines sueltos (4, 5 y 9), que quedan libres de utilizar a para otras funciones que se consideren pertinentes, ya que no interfieren con las funciones de Sirius. Este esquema se recuperará más adelante en el trabajo cuando se presente el proceso de diseño electrónico del prototipo.

3. Desarrollo de la aplicación

3.1. Introducción

En una definición general, el prototipo que se tiene por objetivo desarrollar es, al igual que el Sirius, un sistema de adquisición de datos (o DAS por sus siglas en inglés *Data Aquisition System*). Es decir, en términos generales, un dispositivo capaz de leer información de un sensor, usualmente n forma de voltaje, y sea capaz de mostrárselo a un usuario humano de forma que este la pueda interpretar.

Estos sistemas están conformados por los sensores, que nos vienen fijados por el uso previsto (potenciómetros y células de carga); un circuito de tratamiento de la señal, conversores analógico-digitales (ADC) y un controlador que gestione los datos por software. Un esquema de estos elementos, representando el flujo recorrido por la información, se presenta en la figura 5 inferior. La parte de filtrado físico es prescindible para el uso previsto del prototipo.

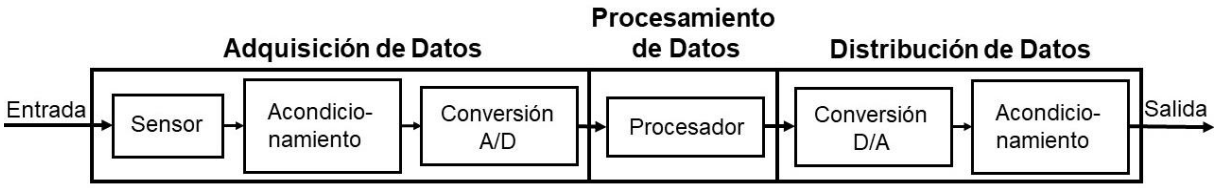


Figura 5: Sistema de adquisición de datos

3.2. Los sensores

Las magnitudes que se han de recoger en cualquier sistema de control de estructuras o resistencia de materiales son la fuerza y el desplazamiento. Por tanto, sensores que se utilizan para este proyecto y que aparecerán referenciados a lo largo de la memoria son dos, que tienen esta capacidad de medida: células de carga y potenciómetros lineales.

Primeramente, las células de carga, son sensores que convierten la fuerza que se aplica sobre ellas en una señal eléctrica medible y proporcional a dicha fuerza. Se utilizan generalmente para medidas de peso y su forma puede variar. La que utilizaremos en concreto, mostrada en la figura 6, en el presente TFM es de tipo viga de flexión con capacidad máxima de 10 kg, si bien más adelante veremos que se prevé el uso de otras.

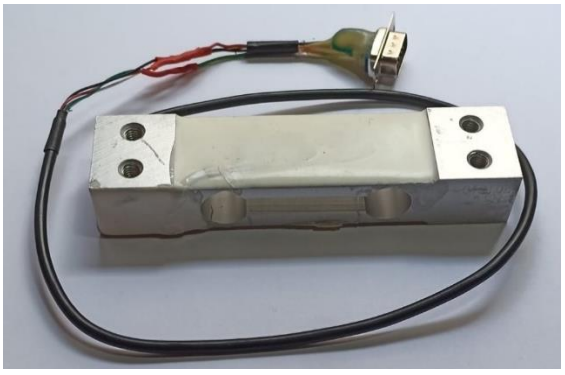


Figura 6: Célula de carga

El funcionamiento interno está basado en un puente de Wheatstone de 4 galgas extensiométricas de resistencia variable, dispuestas de modo que dos se extiendan en sentido positivo y otras dos en negativo, variando por tanto su resistencia de forma opuesta dos a dos. Su conexionado consta de cuatro hilos diferenciados por colores. Dos de alimentación (rojo positivo y negro negativo) y dos para la salida diferencial (verde positivo y blanco negativo). Este esquema se representa en la siguiente figura 7.

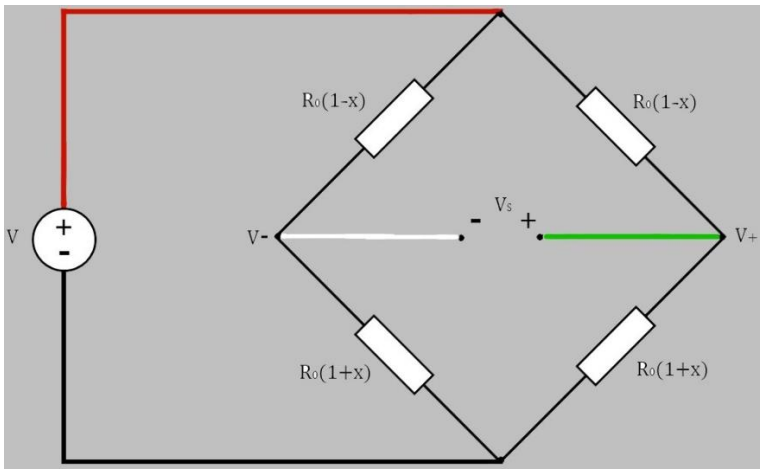


Figura 7: Cableado célula de carga

El segundo sensor, son los potenciómetros lineales. Un potenciómetro estándar se basa en una resistencia variable según las vueltas dadas al eje, dando lugar a un divisor de tensión, cuya tensión de salida es proporcional a la variación de resistencia inducida con cada vuelta del eje. Los potenciómetros lineales constan un potenciómetro de eje descrito y de un cable recogido en un carrete de tambor de retorno por resorte. Dicho cable se fija en su extremo, y la extensión del mismo se transforma en giro de eje por el carrete, convirtiendo la variación de resistencia en dependiente del desplazamiento. El conexionado de un potenciómetro, por tanto, consta de tres hilos, siendo los extremos la alimentación a tensión y a tierra; y siendo el central la salida del divisor de tensión.

El que será utilizado para las pruebas a lo largo de este trabajo, presentado en la figura 8, es un potenciómetro lineal que carece del carrete de retorno. Dicho carrete es fácilmente dañado por extensiones y recogidas rápidas y repetidas, por lo que se opta por utilizar solo el cuerpo del potenciómetro para tener la capacidad de variar la resistencia de forma más libre para las pruebas. Pero el funcionamiento, como se describía en el párrafo anterior, es equivalente.

Figura 8: Potenciómetro

3.3. El procesador: Raspberry Pi

Como componente central del sistema, será necesario un controlador capaz de manejar todos los datos adquiridos y mostrárselos al usuario. Además, este controlador es preferible que sea un dispositivo portátil para facilitar el transporte, por lo que un ordenador convencional queda descartado. También sería deseable un controlador que nos diese flexibilidad y opciones a la hora de gestionar dichos datos, con suficiente potencia como para hacer representaciones gráficas.

La última y quizá más importante de las exigencias que ha de cumplir el sistema de computación es que permita al usuario ver los datos online conectándose a la red del sistema. El número de usuarios conectados no debe de afectar a la posibilidad de visualización. El objetivo de este punto es mejorar la comodidad del usuario.

El mejor dispositivo comercial que cubre las necesidades mencionadas y nos asegura unas opciones de tratamiento, incluyendo cálculos, almacenamiento y formas de mostrárselo al usuario; casi infinitas y apoyadas por una comunidad activa dada su naturaleza de código libre, es la Raspberry Pi, mostrada en la figura 9.

Figura 9: Raspberry Pi 4

Otras dos opciones han sido consideradas antes de esta decisión:

- **Arduino:** Es también una plataforma de software de programación y hardware de microcontroladores abiertos, bastante popular en ámbitos educativos y con una amplia comunidad online. El problema que tiene es que, para usar las funcionalidades WiFi, que son necesarias para cubrir el objetivo de mostrar los datos al usuario online, es preciso un *shield* o módulo externo. Algunas placas nuevas de Arduino incorporan ya este módulo, pero la interfaz entre este módulo y Arduino complica la programación en todos los casos. [3]
- **ESP32:** Es un microcontrolador fabricado por Espressif y cuya programación es compatible con el IDE (*Integrated Development Environment*) de Arduino. Se desarrolló para aplicaciones de IoT y ya viene con el módulo WiFi integrado a un coste comparativamente reducido respecto a lo ofrecido por Arduino. Sin embargo, su potencia para el tratamiento y, sobre todo, almacenamiento de datos, se quedaría corta rápidamente. [4]

La Raspberry resulta más ventajosa frente a las otras opciones consideradas por su muy superior potencia (RAM de mínimo 1 Gb), capacidad de almacenamiento interno (tarjeta SD) de la información y las casi infinitas posibilidades de tratamiento de los datos.

El sistema operativo de la Raspberry, Raspbian, está basado en Linux y es por tanto *Open Source*. Esto implica que existe una gran cantidad de software colaborativo, tutoriales, y foros de comunidad dedicados a desarrollar aplicaciones gratuitas para este sistema. Aplicaciones que se pueden adaptar con relativa facilidad a las necesidades concretas del prototipo. La programación software dentro de la Raspberry en sí misma se realiza principalmente en Python, lenguaje que ha ganado bastante popularidad en los últimos años, gracias a su relativa sencillez en comparación con otros competidores como C o JavaScript. [5]

Las dos últimas versiones de Raspberry disponibles para compra son la 3 y la 4, siendo las principales diferencias entre estos dos modelos la capacidad de la memoria RAM (1 Gb en el modelo 3, 2/4/8 Gb en el 4), la velocidad del procesador (mayor en el 4) y el posicionamiento de los puertos USB, Ethernet y HDMI. La localización precisa de los mismos se puede encontrar en sus correspondientes dibujos mecánicos. [6]

Como interfaz para ser utilizada en aplicaciones como microcontrolador, la Raspberry cuenta con 40 pines, organizados en dos filas, que se distribuyen entre funciones específicas de comunicación o alimentación y una serie de pines GPIO. GPIO significa *General Purpose Input-Output*, lo que significa que son pines genéricos que tienen capacidad de lectura/escritura analógica/digital. Este bloque de pines 2x20 se encuentran en el mismo lugar en las Raspberry de versión 3 y 4. Las funciones de cada pin, resumidas en la figura 10, deberán de tenerse en cuenta para las conexiones futuras de cualquier dispositivo externo con la Raspberry. [7]

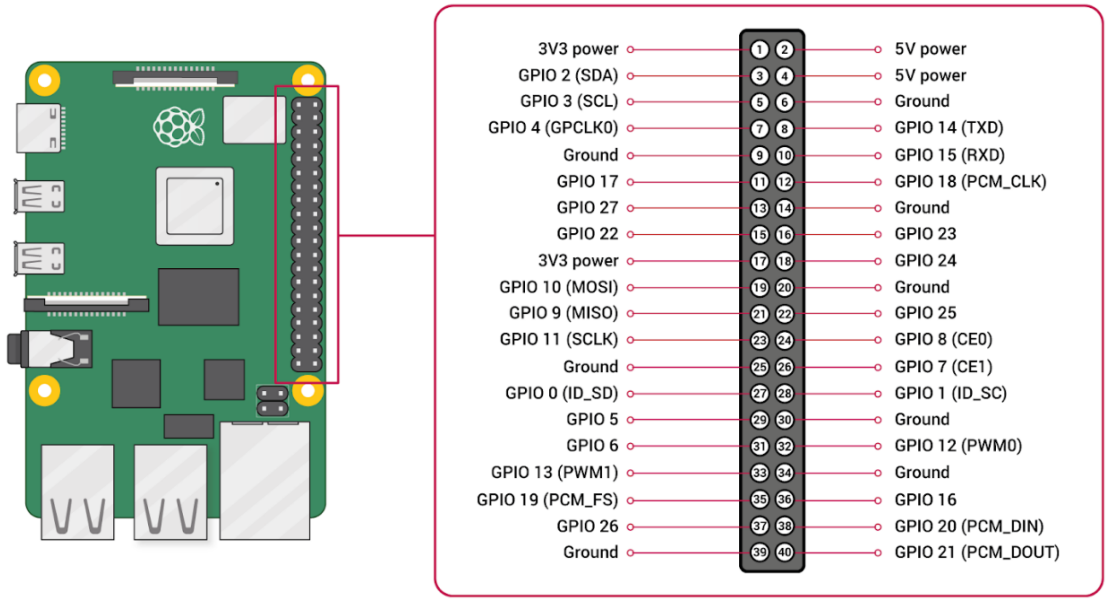


Figura 10: Raspberry Pi pinout [7]

3.3.1. Primeros pasos con Raspberry

Sirva este apartado como una breve nota sobre la puesta a punto inicial de la Raspberry para comenzar a trabajar sobre ella, asumiendo que se parte desde cero. Se necesitará una pantalla que conectar por HDMI a la Raspberry, un teclado y ratón USB, una tarjeta de memoria MicroSD de al menos 8Gb y un ordenador con acceso a Internet.

En el ordenador es necesario instalar el software "Raspberry Pi Imager", programa mostrado en la figura 11, que será el encargado de instalar el sistema operativo Raspbian en la tarjeta MicroSD. Se descarga desde la página propia de Raspberry, en dónde se escoge el sistema operativo que se desee instalar. Para el uso que se le dará a lo largo de este trabajo, se recomienda la última versión de "*Raspberry Pi OS with Desktop and recommended software*". La versión del OS con escritorio hace que sea más cómodo navegar entre ficheros y demás funcionalidades, y será conveniente su uso especialmente a la hora de utilizar el navegador web. Además, se puede ahorrar tiempo en instalaciones futuras si ya se instala desde el primer momento parte del Software recomendado, que incluye por ejemplo Chromium e IDEs de programación. [8]

Todos los enlaces de descarga pueden encontrase en:

https://www.raspberrypi.com/software/

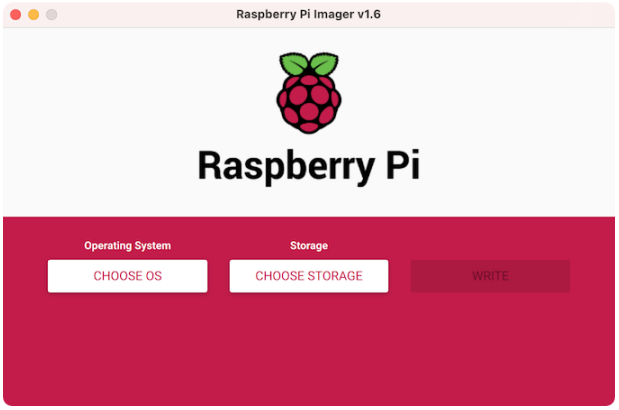


Figura 11: Raspberry Pi Imager [8]

Una vez descargado el SO y conectada la MicroSD a nuestro PC, se selecciona el archivo del SO descargado y se graba en la tarjeta utilizando el *Imager*. Hecho esto, se puede introducir la MicroSD en la Raspberry y encenderla por primera vez. En este primer encendido, el sistema guiará automáticamente por una serie de configuraciones iniciales, en dónde se podrá seleccionar la localización, la contraseña del usuario predeterminado y la red WiFi principal. Completada la configuración inicial, se reiniciará y en esta ocasión se abrirá ya la pantalla inicial del escritorio.

Desde el escritorio de la Raspberry es posible acceder ya a todas las funciones, como documentos, opciones de configuración avanzadas y la barra de comandos habitual.

3.4. Conversor ADC

La conversión analógico digital es el proceso interfaz entre el mundo real y el controlador que gestiona la aplicación. Es decir, los sensores transducen una magnitud física en un valor eléctrico. Este valor eléctrico en el caso de los sensores utilizados es, en ambos casos, un valor de tensión variable. Pero este valor variable es una magnitud analógica continua, que un controlador no puede interpretar. Para convertir esta señal continua en digital, la señal analógica se cuantifica en niveles según los bits de resolución del ADC.

Dando un ejemplo, en una señal de 5 V que entra en un ADC de 2 bits de resolución, a los 5 V le corresponderá en binario "11" y a 1,25 V le corresponderá "01". Así, la señal analógica se convierte en un discontinuo de números binarios que el controlador puede interpretar como naturales para ajustarse a la realidad medida por el sensor. La medida continua tendrá más cuantos en que dividirse y por tanto asemejará ser más continua y cercana a la analógica del sensor cuanta más resolución (bits) tenga el ADC.

Por lo general, los microcontroladores cuentan con un ADC ya integrado de base. La resolución de este es, según las especificaciones, de 10 bits en el caso de Arduino (2^{10} = 1023 niveles) y de 12 bits en el caso del ESP32 y la Raspberry (2^{12} = 4096 niveles). Sin embargo, para el prototipo desarrollado, el cual tiene como objetivo poder utilizarse para mediciones, entre otros, de deformaciones en estructuras, resulta conveniente conseguir la mayor resolución posible.

3.4.1. El ADS1115

Por los motivos presentados anteriormente, se seleccionará el ADS1115, mostrado en la figura 12, para ejercer como ADC en sustitución de los integrados en la placa de la Raspberry. Este ADC tiene una resolución de 16 bits, que corresponde con 2^{16} = 65536 niveles, muy superior al integrado en placa de 12 bits, y proporcionará una mayor precisión en la medida.

Este ADC puede encontrarse como circuito integrado independiente o adquirir el modelo embebido en placa con componentes auxiliares, destinados principalmente para reducción de ruido en la comunicación, de la empresa Adafruit. Se opta por utilizar esta opción por resultar más fácil de adquirir en este formato y por la inclusión de esas protecciones contra ruido. Además, es sencillo de conectar en una *protoboard* para realizar pruebas de adquisición de datos, así como de adaptarse para su integración en una PCB. Se trata también de un producto sujeto a licencias de código abierto, por lo que es sencillo encontrar medidas, especificaciones y librerías de control y ejemplos de su uso en proyectos.

Figura 12: ADS1115 [9]

El integrado se comunica como esclavo por I2C con un dispositivo maestro, que sería la Raspberry Pi. I2C es un protocolo de comunicación serie sencillo, muy habitual en la comunicación con microcontroladores, pensado para el intercambio de información entre un maestro (que coordina la comunicación) y varios esclavos (a la espera de peticiones del maestro). Se basa en dos cables, uno SCl (señal de reloj) y otro SDA (intercambio de datos). Cada dispositivo está identificado con una dirección única propia. [9]

La dirección del ADS1115 esclavo se puede seleccionar, según la tabla 2, dependiendo dónde se conecte el pin ADDR.

Pin ADDR conectado a	Dirección I2C del Esclavo
GND	1001000
VDD	1001001
SDA	1001010
SCL	1001011

Tabla 2: Selección ADDR esclavo [10]

Esto restringe a un máximo de 4 ADS1115 conectados al mismo bus I2C, y, por tanto, a una misma Raspberry, que tiene solamente dos pines de bus I2C. Pero cada uno de los ADS1115 cuenta con 4 canales de entrada, a los que se conecta la señal de salida de los sensores presentados. Esto hace un total de 16 sensores posibles conectados al mismo tiempo, que se considera suficiente para este diseño.

En cuanto a la capacidad de interconexión con Raspberry, es perfectamente compatible con ella. Puede ser alimentado con los 5V que tenemos en pines 2 y 4 y con cualquiera de las referencias a GND y se comunica con ella por los pines I2C 3 (SDA) y 5 (SCL).

3.5. Programa de lectura de datos

El objetivo a conseguir en esta primera versión del programa es lograr leer correctamente los datos de los sensores. Sin cumplimentar este primer paso clave, no habrá información que mostrar al usuario.

Los datos de los sensores se reciben del ADS1115, que actúa como interfaz entre sensores y la Raspberry, para conseguir la mayor precisión según se mencionaba anteriormente. Para establecer comunicación con el ADS1115 no solo es necesario el canal I2C, sino también la biblioteca de control del ADS1115, adaptada para su uso en programación de la Raspberry.

Para comenzar, se instalarán dos paquetes de herramientas que permiten interactuar con dispositivos I2C. El primero permite hacer uso del protocolo en sí y el segundo permite usar programación en Python para comunicarse con dispositivos I2C. Se instalan desde ventana de comandos de Raspberry con el comando [11]:

```
sudo apt install -y i2c-tools python3-smbus
```

Anotar que la comunicación I2C viene deshabilitada por defecto en Raspberry, de modo que para establecer comunicación se debe habilitar el uso del bus I2C de la Raspberry desde las opciones de configuración, a las que se accede mediante el comando:

```
sudo raspi-config
```

Ejecutado el comando, se utilizan las flechas de dirección para desplazarse hasta la opción 5 "*Interfacing Options*", seleccionado con intro la opción, aparece una nueva pantalla en la que se debe seleccionar nuevamente "P5 I2C". Se confirma la activación de la interfaz I2C. Después de realizar este tipo de operaciones de instalación y cambio de configuración, es conveniente reiniciar la Raspberry. [11]

Ahora, se instalará la biblioteca de Adafruit, que gestiona las comunicaciones con el ADS1115. Es muy importante remarcar que, después de un conjunto de pruebas fallidas, se debe de instalar la versión de la biblioteca escrita en Python3, así como compilar el programa con esta versión de Python para que la biblioteca funcione correctamente. En caso de instalar una versión desactualizada de la biblioteca o de no compilarse en Python3, el programa dará errores de lectura. Se utiliza el comando [12]:

```
pip3 install adafruit-circuitpython-ads1x15
```

3.5.1. Circuito de pruebas

Una vez instalados los prerrequisitos necesarios, se utilizará una *protoboard* y un conjunto de cables jumper (cables con pines de conexionado fácil para prototipos) para preparar el circuito sobre el que se trabajará y realizarán las pruebas necesarias. Se han empleado dos versiones de dicho circuito de pruebas, con la única diferencia entre ellos siendo el conexionado de la célula de carga. El esquema del mismo se presenta en la figura 13 a continuación.

Se utilizarán, para las pruebas con los programas que se muestran a lo largo del presente trabajo, los siguientes sensores: una célula de carga de 10 kg y 290 Ω de resistencia de galgas en reposo y dos potenciómetros. Siendo estos últimos, como se introducía anteriormente en su apartado, potenciómetros lineales sin cuerda y ni resorte.

En esta versión del circuito, se alimenta tensión de 5 V desde la Raspberry a los sensores y a el ADS1115. Se conectan los potenciómetros a al ADS1115 en los canales 0 y 1, quedando los canales 2 y 3 para los cables de la salida de tensión diferencial de la célula de carga.

La célula de carga se fija a la mesa con un sargento, como se muestra en la figura 14, y ejerciendo fuerza con la mano sobre ella para comprobar si se recibe una variación en los datos o utilizando una pesa de 2 kg depositada en el extremo para tener un valor fijo del peso con el que trabajar.

El conexionado físico de estos elementos, tal y como se ha descrito en este apartado, se muestra en la posterior figura 15.

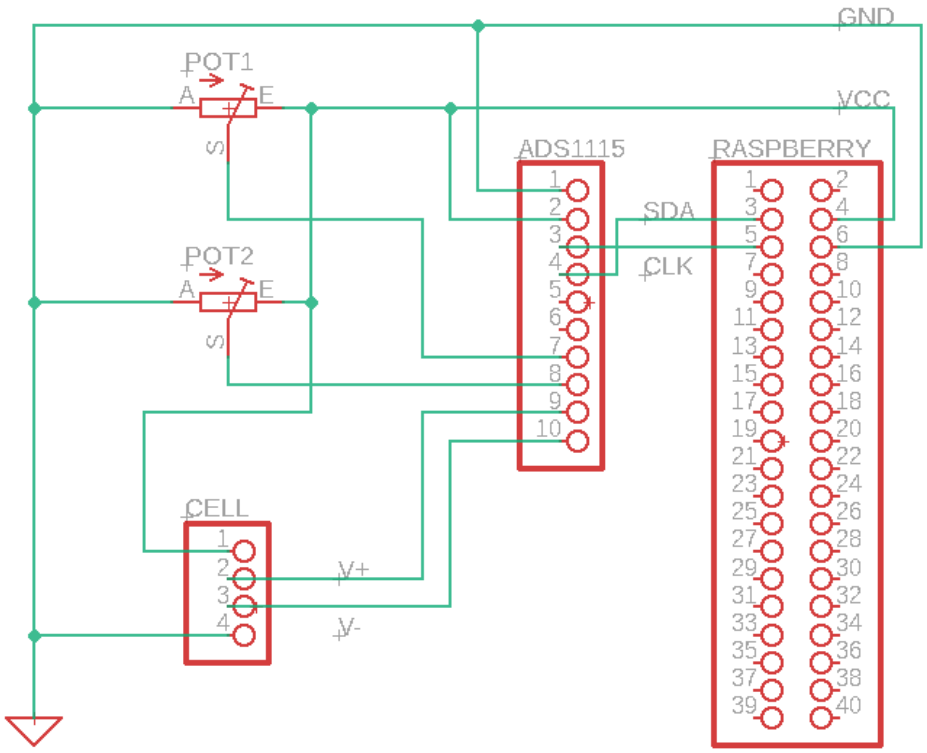


Figura 13: Esquema de conexión circuito de pruebas

Figura 14: Célula de carga fijada

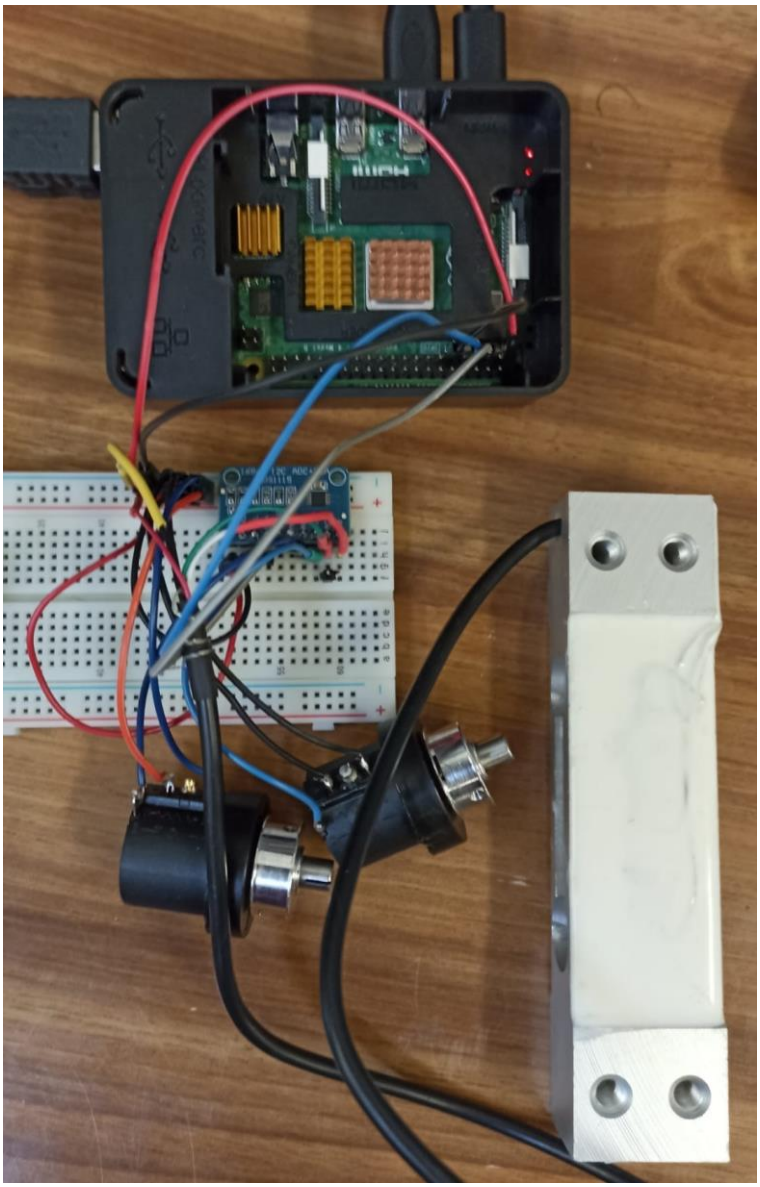


Figura 15: Montaje circuito pruebas protoboard

3.5.2. Primer programa

Para comenzar el programa, se utiliza como referencia el programa de ejemplo de uso de la librería controladora del ADS1115 en GitHub, para comenzar a leer los datos de los canales. En cabecera del programa, se importan todas las bibliotecas necesarias y se declaran los elementos auxiliares exigidos por la librería del ADS1115. Se lee de los canales de la célula de carga (2 y 3) una señal diferencial, declarando el uso de dichos canales. Mediante un bucle *while* infinito, se leen continuamente las variaciones en las lecturas. A causa de este bucle, para detener la ejecución del programa se debe introducir la combinación “ctrl+c”. Se recoge tanto la tensión original leída como su conversión analógica, que sería el dato realmente relevante, sin embargo, conocer la tensión en este punto ayudar a comparar medidas, como veremos a continuación. En la figura 16 se muestra este programa, comentado para su comprensión.

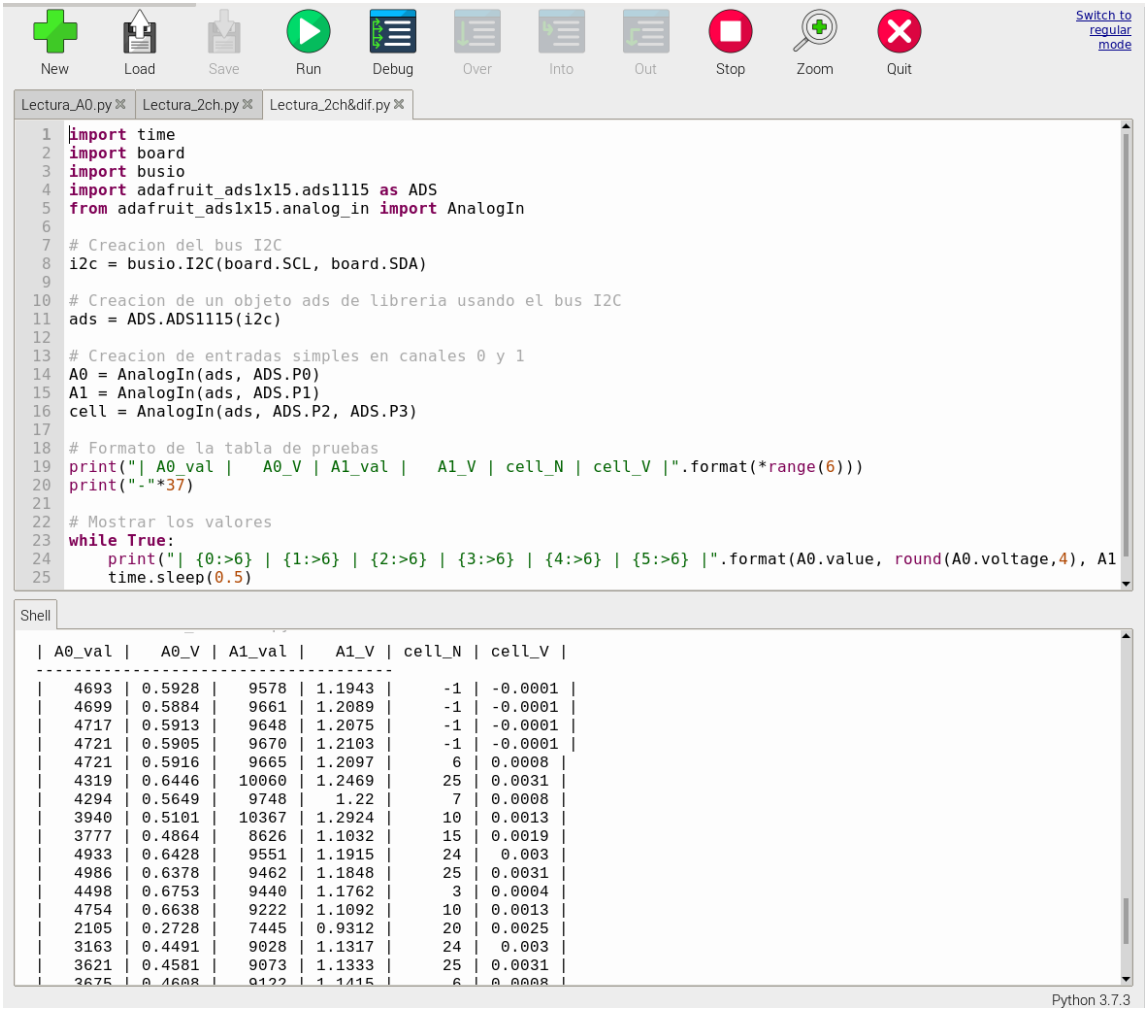


Figura 16: Prueba primer programa

En la figura 16 se observa que el programa siendo ejecutado en Thonny, un intérprete de código de la versión Desktop instalada en la Raspberry. Este intérprete compila el programa ya automáticamente en Python3, como predeterminado, pero recordar que, si la compilación-ejecución se hiciera desde la ventana de comandos, se debe de indicar "python3" para que la biblioteca funcione correctamente [12]. Se remarca esto porque fue uno de los problemas encontrados durante el desarrollo.

Con este programa sencillo ya se ve que hay unas buenas lecturas de los potenciómetros, que reaccionan bien a los cambios. Sin embargo, el valor de la célula de carga es muy pequeño en comparación con ellos. Esto se debe a que, aunque esté siendo alimentada a 5 V, el comportamiento del circuito de medida (puente de Wheatstone) hace que su salida diferencia sea del orden de los milivoltios. Este dato suele venir indicado en las células comerciales como "X mV/V", es decir, que se obtienen X milivoltios en la salida por cada voltio de alimentación a la entrada. Esto causa que las medidas sean muy dispares con las de los potenciómetros, y que se pierda mucha precisión en la medida, ya que un milivoltio puede equivaler a un cambio de varios gramos.

Para tratar de solucionar este problema, se agrega una línea de código al programa anterior, sobre el de la figura 16. Se añade una función de ganancia incluida en la biblioteca del ADS1115 para multiplicar el número convertido a digital, como se muestra en la figura 17 [12].

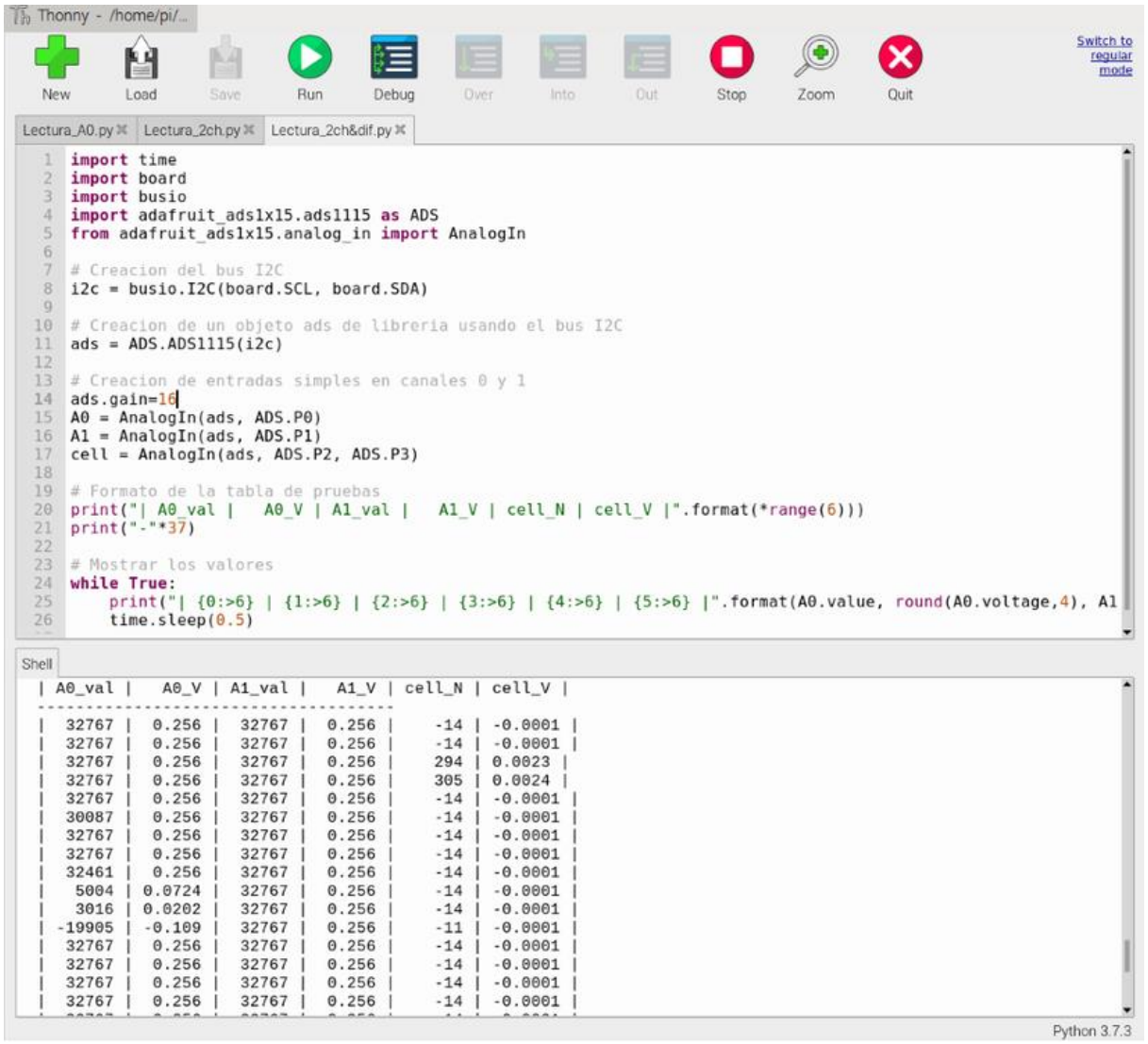


Figura 17: Prueba programa con ganancia

El problema que aparece es que, si bien el valor numérico de la célula se ha incrementado, la ganancia establecida afecta globalmente a todo el ADS1115, no solamente a la respuesta diferencial de la célula de carga. Esto hace que los potenciómetros, cuyo valor de tensión va a ser mayor por norma general, lleguen al número máximo de su escala con mucha facilidad. Escala que en el ADC ADS1115 utilizado de 16 bits cubre el rango (-32767, 32768). Estos valores, que se acercan la saturación del conversor, no aportan realmente información, no se ve una variación real. Además, si se sigue incrementando la tensión recibida, el ADC alcanza saturación, desborda y se corta la comunicación, apareciendo un fallo de ejecución.

Para poder conseguir mayor precisión de medida de la célula de cargar, se tendrá por tanto que conseguir amplificar solamente su señal diferencial de salida. Amplificación que no se puede alcanzar de forma precisa desde software, en consecuencia, se recurrirá a un circuito de electrónica analógica.

3.5.3. Amplificador de instrumentación

Para amplificar este tipo de señales diferenciales se utiliza, por lo general, en las aplicaciones de sensorización un amplificador de instrumentación (AI). Esto es, un tipo de circuito electrónico conformado por amplificadores operacionales (AO) y resistencias, que toma dos entradas de tensión y amplifica su diferencia a la salida, según una ganancia dependiente de los valores de las resistencias.

Es un circuito muy utilizado para el condicionamiento de señales de sensores con salida diferencial debido a ese compartimiento de su salida. Y es que proporciona unas ganancias de tensión estables y muy altas, en conjunto con un rechazo en modo común muy elevado. Esto es, se amplifica solo la parte diferencial existente entre las señales de entrada, no la común. Así, un sensor con dos salidas diferenciales del orden de milivoltios, conectado a un AI, da una respuesta ya diferencial que se puede mover a rangos más elevados, mejorando la precisión.

Existen dos versiones principales de AI, la formada por dos y la formada por tres amplificadores operacionales. Utilizaremos este segundo, representado en la figura 18, porque tiene mejores características teóricas en todos los factores mencionados en el párrafo anterior que hacen deseable el uso de un amplificador de instrumentación. La ecuación que rige el amplificador, siendo A_v la ganancia es:

$$A_v = \frac{V_{out}}{V_2 - V_1} = \left(1 + \frac{2R_1}{R_G}\right)\frac{R_3}{R_2}$$

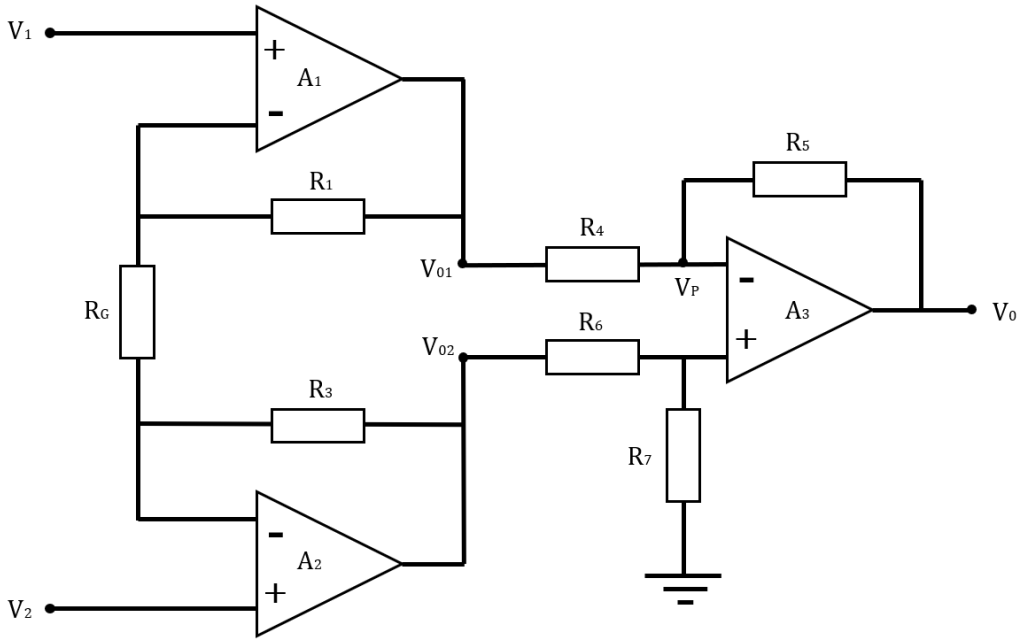


Figura 18: Esquema AI de 3 AO

Este circuito puede construirse con los componentes individuales, adaptándolo a una situación específica, o puede adquirirse una versión comercial monolítica. Se opta por esta segunda opción porque los AI comerciales son más estables ante ruidos y temperatura de que lo que se pudiera construir con las partes individuales, y ofrecen un rango amplio de ganancias, por lo general, ajustables mediante resistencia externa que ofrecen un buen rango de adaptación.

Se buscará en el catálogo del proveedor comercial Farnell, filtrando las opciones de búsqueda según las necesidades del prototipo. Siendo las principales, que tenga estructura de 3 AO internos para potenciar el rechazo en modo común, que se pueda alimentar entre 5 V y GND, para aprovechar la fuente de la Raspberry; que su ganancia se pueda ajustar externamente con una resistencia, y que tenga una envolvente tipo DIP, preferiblemente DIP8, de pines de soldadura pasante, para facilitar su conexión en protoboard y posteriormente su soldadura a mano en PCB.

Se encuentran, entre los más vendidos, el modelo INA118 de Texas Instruments, representado en la figura 19, que cumple con todas las exigencias descritas anteriormente con un precio razonable del entorno a los 15 €. En su hoja de características (o *datasheet*) se lee que tiene unas resistencias internas muy elevadas, con objetivo de lograr que la ganancia se ajuste finamente con una resistencia externa situada entre los pines 1 y 8, siguiendo la ecuación proporcionada por el fabricante para ganancia "G" [13]:

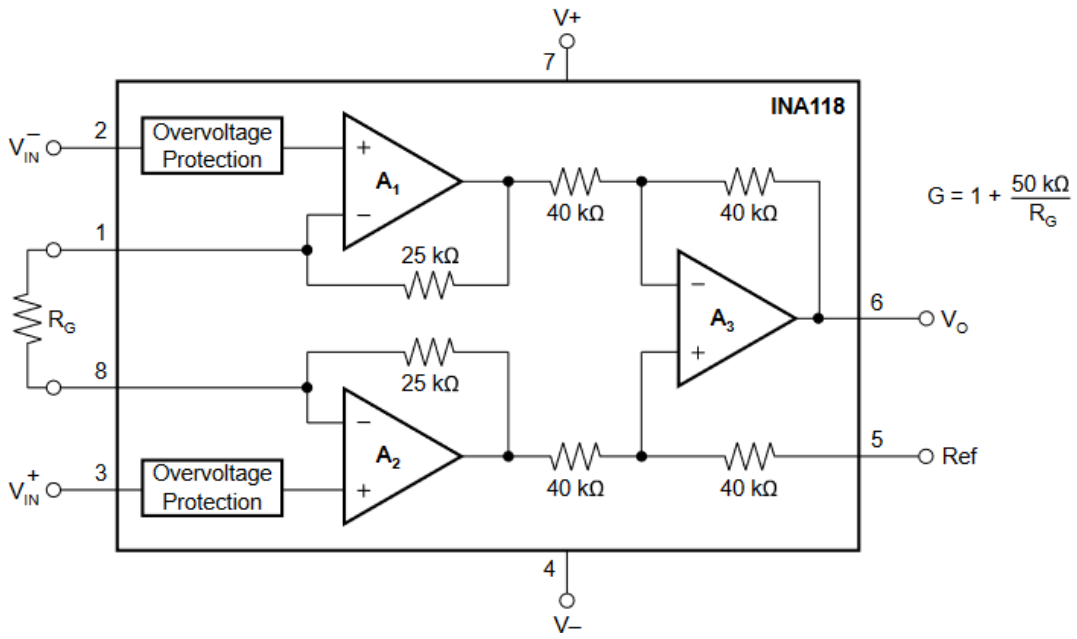


Figura 19: Esquema INA118 [13]

Se indica también la posición en el DIP8 y uso de los pines, que se debe tener presente para las conexiones en la *protoboard* y en el rutado de la futura PCB. La referencia de posición/orientación viene da una pequeña muesca en la envolvente física del integrado.

De la *datsheet* se extrae también un dato importante, que resultará clave en el diseño del circuito. Y es que el uso para el que se le pretende (amplificación de una salida diferencial) aparece listado como aplicaciones habituales del integrado en la *datasheet*. Indica que, en caso de alimentar el INA118 entre una tensión y tierra, en lugar de a tensión diferencial de uso habitual, si se utiliza el INA118 como amplificador para la salida diferencial de un circuito, se tendrá que asegurar que las tensiones de entrada al AI sean de un mínimo de 1,2 V. De no ser así, la salida del AI no será la diferencia amplificada, sino una tensión fija de fuga. Para logar estas tensiones mínimas, se deberá de colocar una resistencia R1 antes del puente de Wheatstone correspondiente [13]. En la figura 20 se resume esquemáticamente este punto.

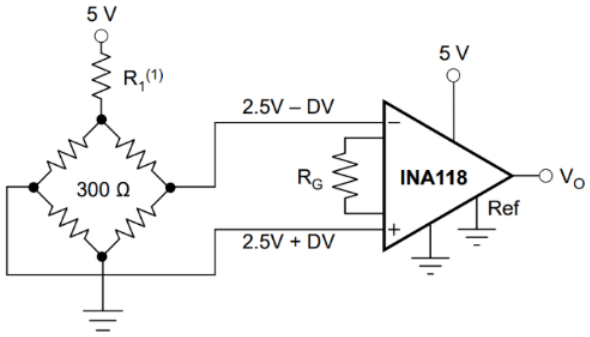


Figura 20: Esquema de conexión AI - célula de carga [13]

3.5.3.1 Cálculos para el AI

Se debe calcular por tanto el valor de esta resistencia pre-puente de Wheatstone que se denominará Rx a partir de ahora. Su valor depende tanto de la tensión de alimentación como de los valores de las resistencias del puente. Para calcular dicho valor, se aplicará la transformación de circuito equivalente de Thevenin al puente, según la figura 21.

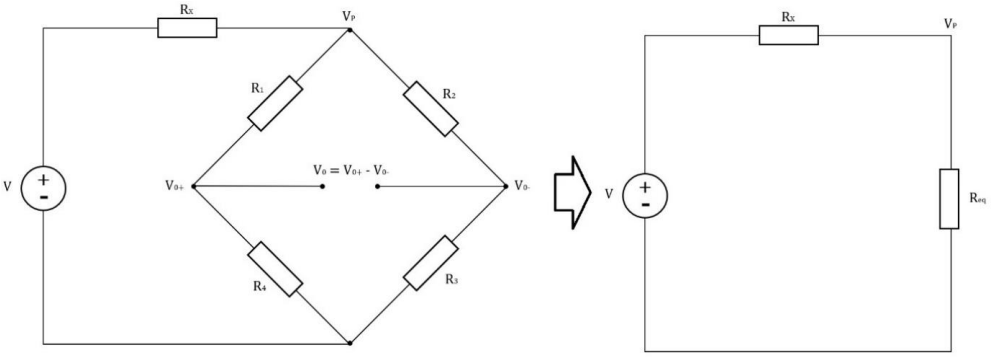


Figura 21: Circuito equivalente

Se calcula la resistencia equivalente de Thevenin a partir de las relaciones en serie y paralelo entre las resistencias del puente, para poder relacionar el valor de la tensión previa al puente con los valores de las resistencias del mismo y de Rx. Para relacionar las resistencias del puente entre sí, se asume que su valor es el mismo entre todas ellas con el puente en reposo. Se toma de la misma forma una tensión de alimentación fija de 5V. Igualando después las dos corrientes que circulan por cada brazo del puente, de nuevo, idénticas en reposo, se llega a un valor de tensión mínima (V_{min}) a la entrada del AI dependiente solamente de los valores de resistencia del puente y de Rx pre-puente.

$$\frac{5 - V_P}{R_x} = \frac{V_P - 0}{R_{eq}} \rightarrow 5R - V_P R = V_P R_x \rightarrow V_P = \frac{5R}{R + R_x}$$

$$(V_0 = V_{min}) \rightarrow R(5 - V_P) = 2R_x(V_P - V_{min}) \rightarrow V_{min} = \frac{5R - RV_P - 2R_x V_P}{-2R_x}$$

$$V_{min} = \frac{5R}{R + R_x} + \frac{R\dfrac{5R}{R + R_x}}{2R_x} - \frac{5R}{2R_x} = \frac{5R^2}{2R_x(R + R_x)} + \frac{5R}{R + R_x} - \frac{5R}{2R_x}$$

La idea inicial es utilizar una resistencia normalizada, de las disponibles comercialmente, así que se transada la ecuación final obtenida en el desarrollo anterior a Excel para calcular los distintos valores de V_P que se pueden conseguir con el conjunto de resistencias normalizadas para un puente con galgas extensiométricas en reposo de 290 Ω de resistencia, medidos de la célula presentada en el apartado "3.2. Los sensores" con un multímetro. Se recogen los resultados en la tabla 3.

Resistencia Normalizada	Rcell			=	290	ohm
	Resistencia Pre-Puente (R_X)					
	x1	x10	x100	x1000	x10000	x100000
1	2,491	2,417	1,859	0,562	0,071	0,007
1,2	2,490	2,401	1,768	0,487	0,059	0,006
1,5	2,487	2,377	1,648	0,405	0,047	0,005
1,8	2,485	2,354	1,543	0,347	0,040	0,004
2,2	2,481	2,324	1,422	0,291	0,033	0,003
2,7	2,477	2,287	1,295	0,243	0,027	0,003
3,3	2,472	2,245	1,169	0,202	0,022	0,002
3,9	2,467	2,204	1,066	0,173	0,019	0,002
4,7	2,460	2,151	0,954	0,145	0,015	0,002
5,1	2,457	2,126	0,906	0,135	0,014	0,001
5,6	2,453	2,095	0,853	0,123	0,013	0,001
6,8	2,443	2,025	0,747	0,102	0,011	0,001
8,2	2,431	1,949	0,653	0,085	0,009	0,001

Tabla 3: Cálculos en Excel

3.5.3.2 Pruebas con el AI

Habiendo hecho los cálculos, se seleccionan una resistencia Rx previa al puente de 245 Ω, que debiera proporcionar una tensión de entrada mínima del entorno a los 1,3 V, según la tabla 3 superior, calculados para una célula de carga de 290 Ω de resistencia de las galgas en reposo. Las pruebas se realizarán con potenciómetros ajustables, para lograr ajustar el valor de resistencia exacto predicho por los cálculos, aspirando en un futuro a que sean sustituidos por resistencias fijas normalizadas, con una configuración serie-paralelo que permita conseguir el valor más cercano posible al calculado.

Realizando unas primeras pruebas, hechas sobre circuito en protoboard conformado solamente por el AI, la célula de carga y la resistencia de ganancia, todo ello alimentado a 5 V mediante la Raspberry, se midió, mediante multímetro que la tensión de entrada era de 0,9 V Esto supone 0,3 V por debajo del valor predicho por los cálculos. Para confirmar la corrección de los cálculos teóricos realizados, se optó por hacer una simulación rápida en Eagle (programa que será presentado más adelante). El esquema simulado se muestra en la siguiente figura 22.

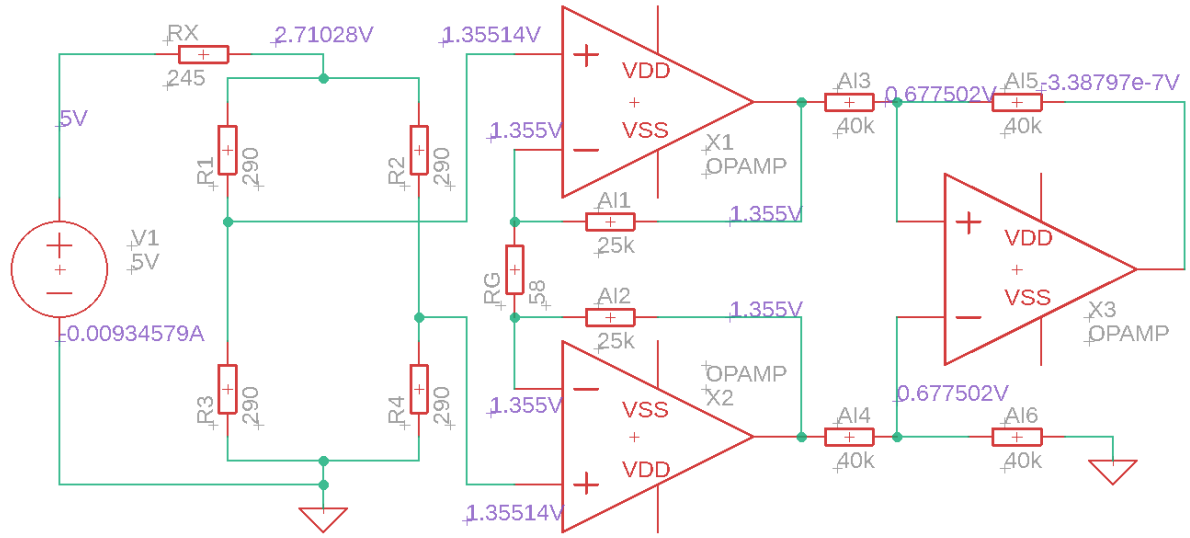


Figura 22: Simulación del puente

Los resultados que simulados de la figura 22 concuerdan con los cálculos teóricos realizados, pero no con lo que se estaba midiendo en la realidad. Un motivo de esta discrepancia pudiera ser la inestabilidad inducida por el uso de la *protoboard*. Para eliminar esta posibilidad, se optó por construir un pequeño circuito en una placa soldable de pruebas. Este circuito cubre solo la alimentación de la célula con Rx, el AI, su resistencia de ganancia (RG) y las conexiones de alimentación necesarias. El esquema de dicho circuito se proporciona en la figura 23 inferior.

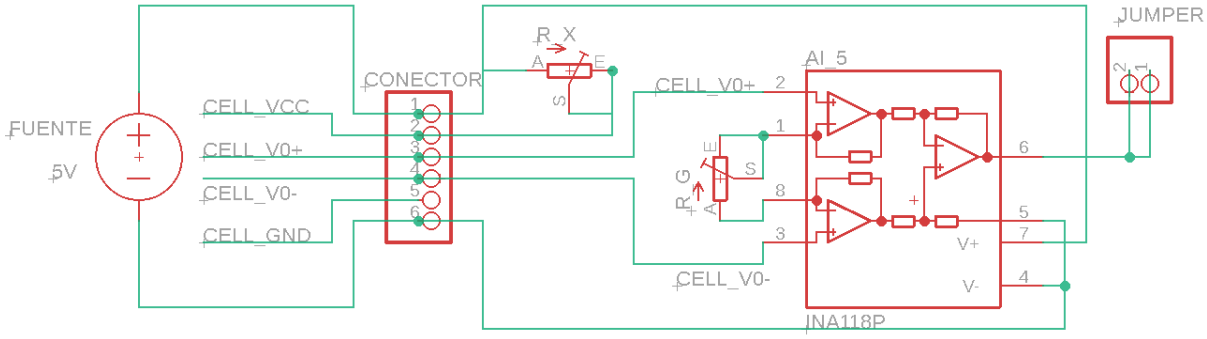


Figura 23: Esquema de conexión circuito prueba célula

Como se mencionaba, dicho circuito realizó sobre una placa perforada de cobre para prototipado de circuitos electrónicos, utilizando cables rígidos de puente para la interconexión de elementos. Con esto se pretende eliminar el posible error inducido por la protoboard. La célula de carga y la alimentación procedente de la Raspberry se conectan mediante unos bloques terminales atornillables. Se sueldan los componentes con estaño-plomo, soldadura que se aprovecha para puentear los pines 2 (salida) y 3 (tierra) de los potenciómetros para utilizarlos como resistencias variables entre dos puntos, que es todo lo que necesitamos. En la figura 24 se muestra la parte superior del circuito y en la 25 las conexiones realizadas en la zona inferior del mismo.

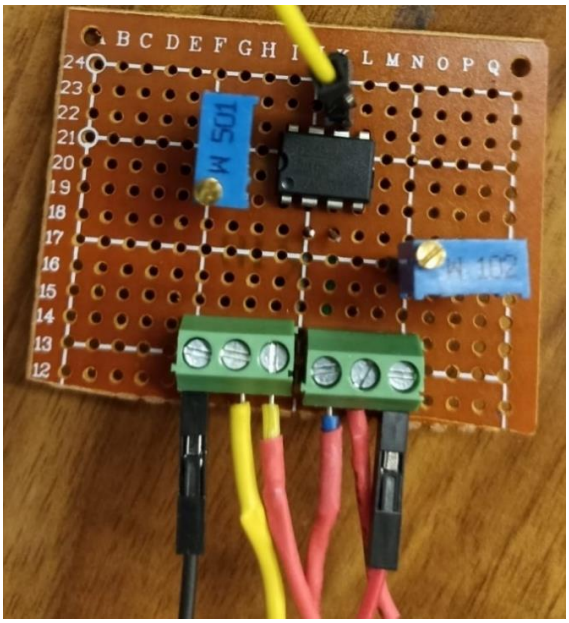


Figura 24: Circuito pruebas AI

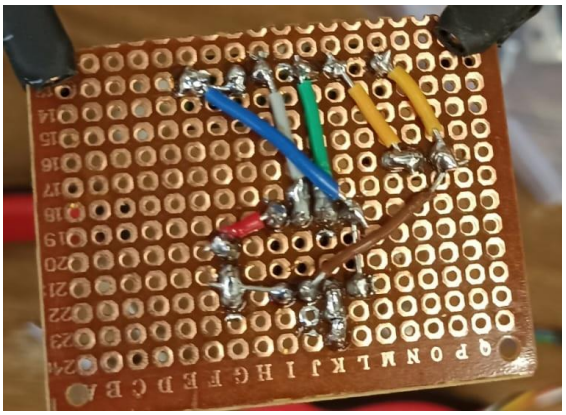


Figura 25: Circuito pruebas AI, conexiones inferiores

Se fijan los mismos valores de resistencia Rx y RG predichos por nuestros cálculos y corroborados por la simulación. Y, sin embargo, se comprueba que en la medida con el multímetro se sigue sin tener la tensión mínima de 1,2 V predicha, a pesar de que ahora el circuito debiera ser notablemente más estable. Estabilidad que se constata alimentado la señal de salida al ADS1115 en uno de los canales de lectura y viendo que fluctúa menos que en el caso del circuito construido en la *protoboard*.

Por tanto, se concluye que, si los cálculos y simulación no se ajustan a lo que se está midiendo en la realidad y el circuito ya es lo más estable que se puede construir, significa que hay algún elemento que no se está teniendo en cuenta. Y ese elemento solo puede encontrase en la parte del circuito que no en controlable, es decir, el interior del AI integrado. Probablemente se deba a que las resistencias tan elevadas de su interior se logran con configuraciones específicas de transistores, no con resistencias convencionales Situación que no solo es más difícil de modelar, sino que además es imposible deducir del integrado.

En consecuencia, se resuelve que los potenciómetros se deberán ajustar a mano para cumplir la condición de tensión mínima. Su valor podrá aproximarse por el cálculo teórico,

pero tendrá que ser últimamente ajustado a mano, girando los tornillos de ajuste de los potenciómetros mientras se mide con un multímetro las tensiones requeridas.

3.5.3.3 Problema en el desarrollo

Con el desarrollo del prototipo ya notablemente avanzado, teniendo ya en mano la primera versión de las PCB, que se verá en el apartado posterior "4.2.4. PCB", surgió un problema imprevisto con el AI. Y es que se contaba con 2 unidades del INA118 escogido para las pruebas, según se ha descrito con anterioridad. Pero para el prototipo completo son necesarios 4 (de nuevo, como se verá más adelante en el apartado "4.2.4. PCB"), y este AI quedó descatalogado del distribuidor Farnell antes de poder completar el prototipo, pero una vez diseñada y con la versión primera de la PCB en mano. Esto supuso un retraso inesperado en el desarrollo, puesto que, a priori, el INA118 aparecía clasificado como uno de los componentes "más populares" en la categoría de AI.

El problema es que ahora no solamente hay que sustituirlo por otro AI integrado que cumpla los requisitos ya descritos al inicio del apartado anterior, sino que también debe ser un AI que tenga los mismos pines y con exactamente las mismas funciones que el INA118. Requisito difícil de cumplir, pero ineludible, ya que el rutado de la PCB, que se tenía en mano lista para probar, se había realizado entorno a las funciones y posiciones de esos pines.

Tras una nueva búsqueda en el catálogo de Farnell ahora actualizado, se logró encontrar otro AI integrado que cumplía con todos estos requisitos. Se trata del AD622 de Analog Devices, que tiene la misma distribución y funciones de pines que el INA118, como se muestra comparando las huellas en las figuras 26 y 27 a continuación.

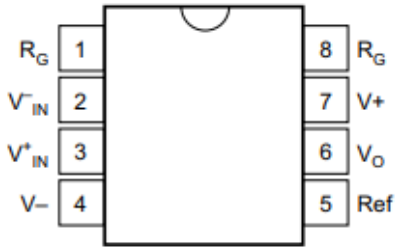


Figura 26: Pines INA118 [13]

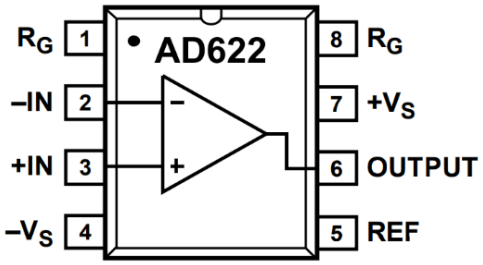


Figura 27: Pines AD622 [14]

Tiene también unas características eléctricas similares, como se puede ver comparando los puntos "*7.5 Electrical Characteristics*" para el INA118 y "*Specifications*" para el AD620, en sus respectivas *datasheets*. Así pues, no viendo discrepancias en el funcionamiento de los dos integrados teniendo en consideración a la información proporcionada por los fabricantes en las *datasheets*, seleccionamos el AD620 para las pruebas con los prototipos de las PCBs, como sustituto del INA118. Es esperable un cierto margen de error entre el comportamiento teórico y el real, que será subsanado ajustando los potenciómetros en el montaje [14].

3.5.3.3 Conclusiones y pruebas con el AI

Ya finalizada y puesta en prueba la segunda versión de la PCB (según se verá más adelante en el apartado "4.2.4.2. Segunda versión"), se comprobó en Farnell que el INA118 había sido remplazado en el tiempo transcurrido entre las pruebas con la versión 1 y la 2 de la PCB por el integrado sucesor, el INA128. Este INA128 supone una revisión del anterior INA118, que tardó en sustituirle en el mercado el tiempo que llevó completar los diseños, recepciones y pruebas de las dos PCBs que se verán más adelante (a partir del apartado "4.2.4. PCB").

Como se puede ver en los esquemas eléctricos internos del integrado, presentados en las figuras 28 (INA118) y 29 (INA128), los circuitos internos son idénticos. Punto que podemos comprobar en el apartado "*7.5 Electrical Characteristics*" de la *datsheet* de ambos. Anotar que la *datsheet* del INA128 contiene también la información correspondiente al INA129 [15].

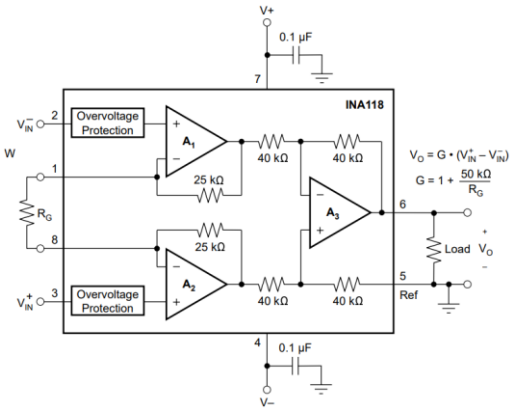


Figura 28: Esquema INA118 [13]

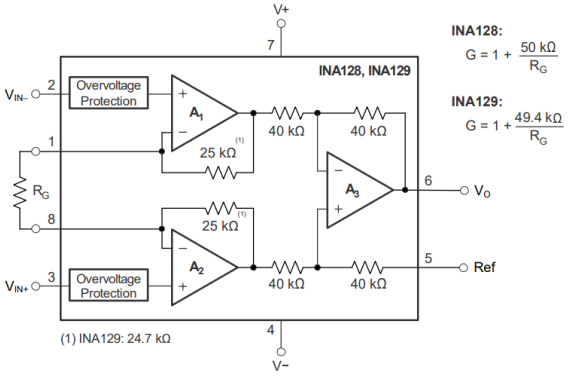


Figura 29: Esquema INA128 [15]

Desarrollando las pruebas del segundo prototipo PCB se detectó una discrepancia entre los valores de tensión esperados y los realmente proporcionados por el AD620. Se montó en el pequeño circuito de pruebas explicado en el apartado "3.5.3.2 Pruebas con el AI" y se comprobó que la tensión de salida se comportaba de forma extraña. Sucede que, con ganancia (establecida con RG) baja, el AI proporciona una tensión de salida del entorno a los 0,62 V, apenas variable a pesar de ir aumentando el peso aplicado sobre la célula.

En las siguientes figuras se muestran los resultados de estas pruebas. En la figura 30 se ve como la entrada al AI está ajustada a los valores de tensión mínimos de 1,2 V. Sin embargo, la diferencia de tensión entre la célula descargada (figura 31) y cargada (figura 32) es de apenas unos milivoltios. La carga se hizo ayudándose de una pesa de 2 kg, para tener una carga fija con la que comparar y las menos libres para medir.

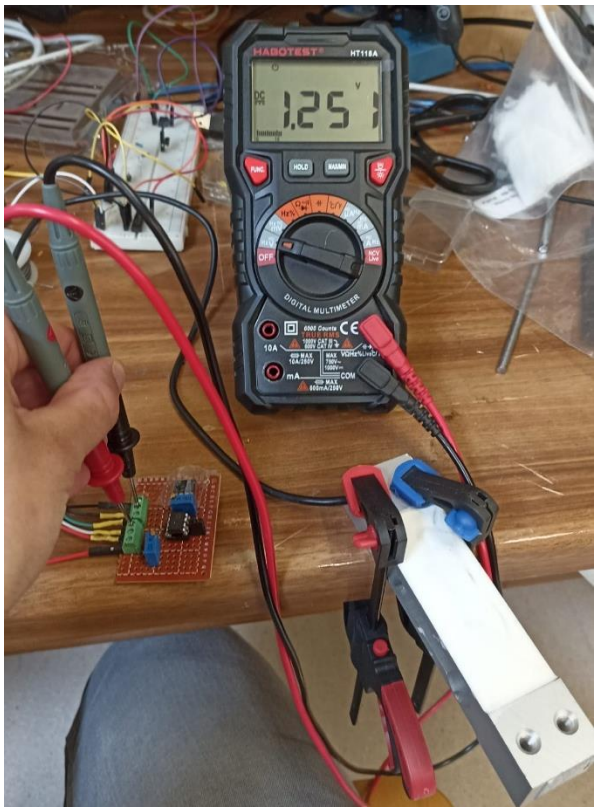


Figura 30: Ajuste del AD620 a 1,2 V de entrada diferencial

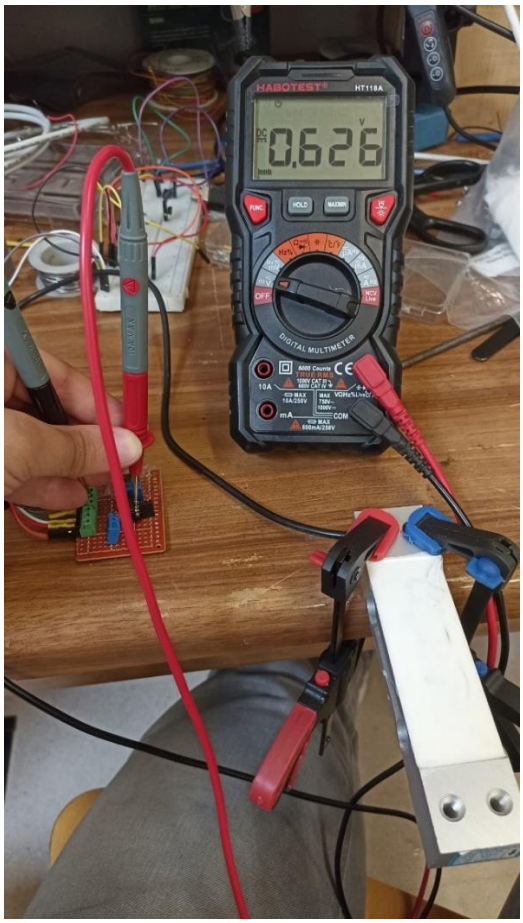


Figura 31: Célula descargada

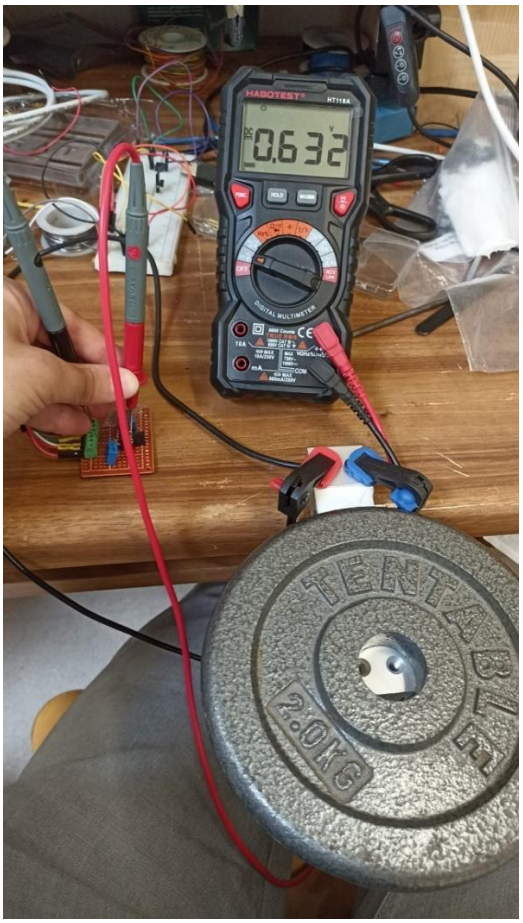


Figura 32: Célula cargada

La teoría que se alcanzó para explicar este comportamiento es que las diferencias entre el INA118 y al AD620 pueden ser más marcadas de lo esperado, sumando al hecho de que se estaba alimentando en modo común un AI (el AD620) que no explicitaba esta posibilidad en la *datasheet,* como sí lo hacía el INA118 [13] [14].

Sin embargo, probando posteriormente a aumentar la tensión a la entrada del puente (cerca del entorno de los 2,4 V), ajustando para ello el potenciómetro de Rx, y disminuyendo mucho la resistencia de ganancia (hacia el entorno de los 20 Ω), con el objetivo de conseguir una muy elevada ganancia de tensión, se alcanza finalmente a un buen comportamiento.

Probando sobre el mismo montaje del circuito del apartado "3.5.3.2 Pruebas con el AI", se encuentra que el AI responde bien, es decir, da variación de tensión de salida razonable, si se proporcionan unos márgenes de tensión de alimentación de entre 1,65 y 2,4 V a la entrada del puente. La conclusión de estas pruebas es finalmente positiva, ya que se corrobora que se puede utilizar indiferentemente el INA1X8 o el AD620.

La consecuencia negativa es que, para lograr esta indiferencia de uso de uno u otro AI, el circuito requerirá obligatoriamente de ajustes manuales. Es decir, dependiendo del AI que se utilice, se deberá medir tensión a la entrada del puente para ajustarla a los valores de funcionamiento. Teniendo en cuenta además que, en caso de que la resistencia en reposo de las galgas de las células de carga empleadas varía también de forma notable respecto al valor de 290 Ω con el que se ha trabajado, también se deberán de medir tensiones y ajustar manualmente los potenciómetros.

Los márgenes de valores de tensión y resistencia que se han comprobado que dan buenos resultados se resumen en la tabla 4 inferior.

AI	V a la entrada del puente	Resistencia de ganancia
INA1X8	1,25 – 1,55 V	50 - 100 Ω
AD620	1,65 – 2,4 V	10 - Ω

Tabla 4: Resumen valores AI

La comprobación de uso de estos valores se presenta en las tres figuras siguientes. Se prepara el circuito de pruebas con el AD620 y se ajusta el potenciómetro de Rx para tener 2,4 V a la entrada del puente, como se ve en la figura 33. midiendo la tensión de salida del AI con la célula sin cargar, como se ve en la figura 34, tenemos que la tensión de base está en el entorno de los 0,6 V que se tenía en el caso de la prueba primera presentada en las figuras 30 a 32. Con este valor de tensión de entrada de puente de 2,4 V y una resistencia de ganancia establecida entorno a los 25 Ω, cargando la célula con la pesa de 2 kg, se observa una variación de 1 V respecto a la célula descargada, como se ve en la figura 34. Esta variación de tensión de salida (figuras 34 y 35) es mucho más notable que la que teníamos en la primera prueba (figuras 31 y 32).

Se concluye por tanto que el ajuste la tensión de entrada del puente mediante Rx es el factor que permite utilizar uno (INA1X8) u otro (AD620) AI, que se resume como se ha mencionado en la table 4 superior.

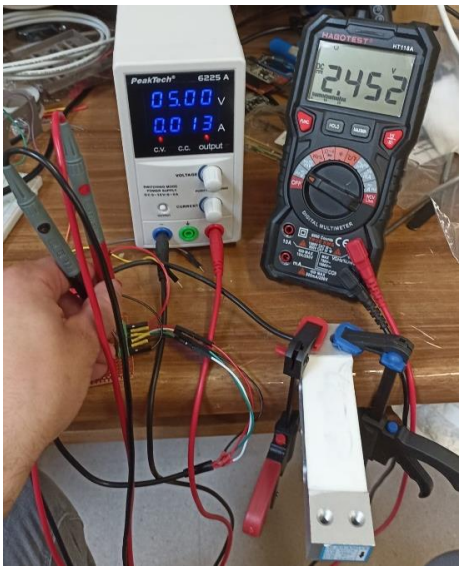


Figura 33: Ajuste del AD620 a 2'4V de entrada diferencial

Figura 34: Célula de carga descargada prueba 2

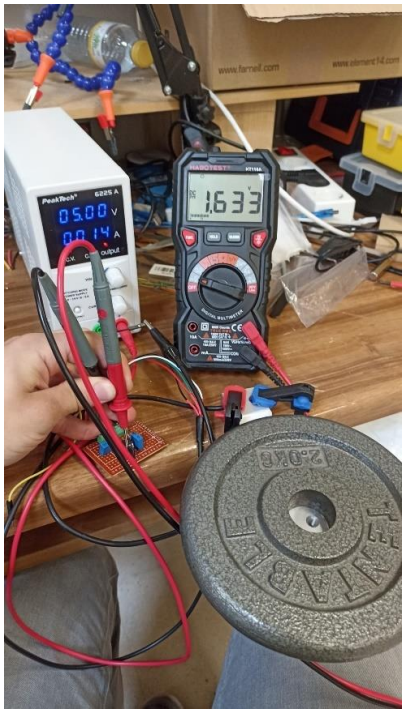


Figura 35: Célula de carga cargada prueba 2

3.5.4. Versión final de recogida de datos

Con la inclusión del AI en el circuito y el ajuste de los potenciómetros de Rx y RG, se habría alcanzado la versión final del circuito de toma de datos para las pruebas, que queda con un esquema de conexión (figura 36) y *protoboard* montada físicamente (figura 37) que se muestran a continuación.

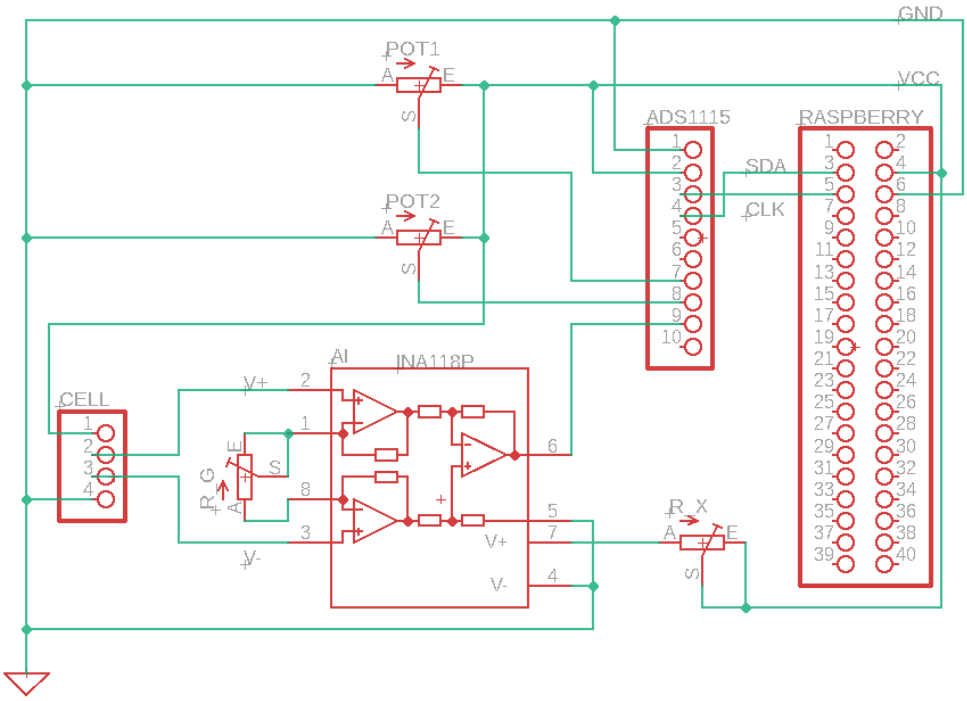


Figura 36: Esquema de conexión circuito con AI

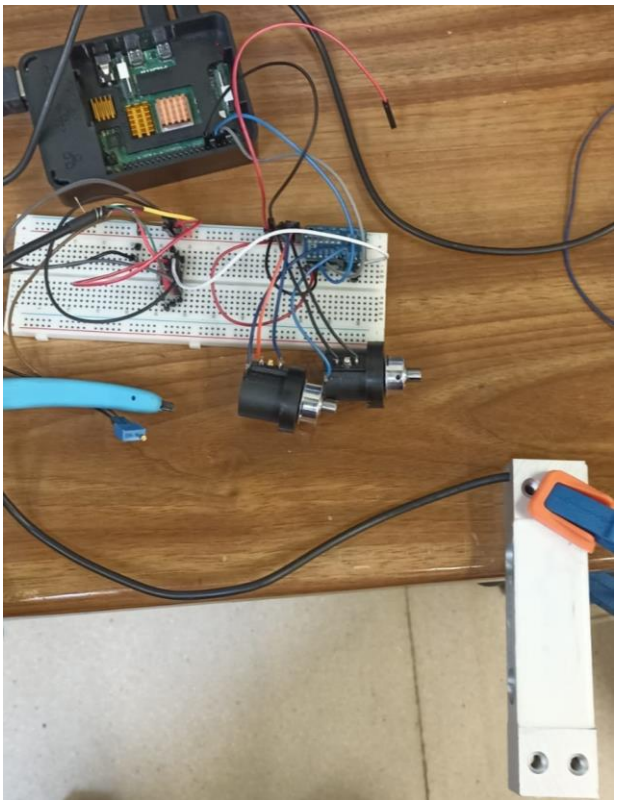


Figura 37: Protoboard circuito pruebas con AI

El cambio que se tiene que hacer al sobre el programa presentado en el apartado "3.5.2. Primer programa" es mínimo, simplemente, en lugar de usar dos canales en modo diferencial para recibir la célula de carga, se declara como un canal de entrada único, equivalente a los de los potenciómetros.

3.6. Objetivo del tiempo real

Ahora que ya se dispone de una fuente de datos fiable, se procede a considerar la siguiente parte del proyecto. El objetivo de esta sección sería aprovechar las posibilidades de la Raspberry para poder mostrar al usuario los datos en tiempo real, esto es, es un flujo constante y con información actualizada automáticamente en el momento en que el sensor registre cualquier variación. Es importante remarcar esta característica de tiempo real, no sirve que el usuario pueda ver los datos recogidos en los últimos minutos desde que se fuerce la actualización, de una u otra manera, la aplicación, tiene que ser posible ver el cambio en el momento en que suceda.

Además, se pretende que el usuario pueda acceder con facilidad a la información, y sin tener que depender de sistemas periféricos de la Raspberry como *display* o pantalla. El objetivo por tanto será que cualquiera que se conecte a la red de la Raspberry pueda ver la información desde cualquier dispositivo. La forma de lograr esto es conseguir presentarlos en una aplicación web, de forma que no sea necesario que el usuario tenga ningún software preinstalado ni exista ninguna restricción de tipo de dispositivo utilizado.

3.6.1. Alternativa descartada

Cronológicamente hablando, la primera línea de investigación fue la que se pasa a describir en este apartado. Si bien no llegó a completarse y pulirse por completo, es importante incluirla para comparar las ventajas que supone la solución finalmente adoptada (desarrollada en el apartado posterior "3.6.2. Solución escogida") frente a esta primera, así como dejar constancia de posibilidades investigadas que puedan ser útiles para futuros proyectos, como es el servidor web con Flask.

3.6.1.1. Flask

La aplicación que resulta más práctica para desarrollar aplicaciones web en la Raspberry es Flask. El módulo Flask es un entorno programable en Python que se encarga de crear un servidor web al que el cliente puede conectarse para enviar peticiones. Peticiones que son gestionadas mediante programación con Python [16]. El programa ejemplo de todo lenguaje de programación, *"Hello World",* adaptado a Flask para escribirlo sobre una página web es [17]:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def home():
    return "Hello, World!"
if __name__ == "__main__":
    app.run(debug=True)
```

Para ver los resultados de un programa en Flask, al ser este resultado una aplicación web, tenemos que acceder a ella después de ejecutar el programa. Para acceder lo haremos desde un navegador web y escribiremos como dirección web bien la IP de la Raspberry, bien "localhost", seguido por el puerto usado por Flask (el 5000). Es decir:

http://xxx.yyyy.zzz.ttt:5000 o http://localhost:5000

Si no se conoce la dirección IP local de la Raspberry, se puede conseguir escribiendo en la línea de comandos, lo que nos dará la información de la configuración de la Raspberry para comunicaciones de red. La IP la encontramos en la segunda línea [18].

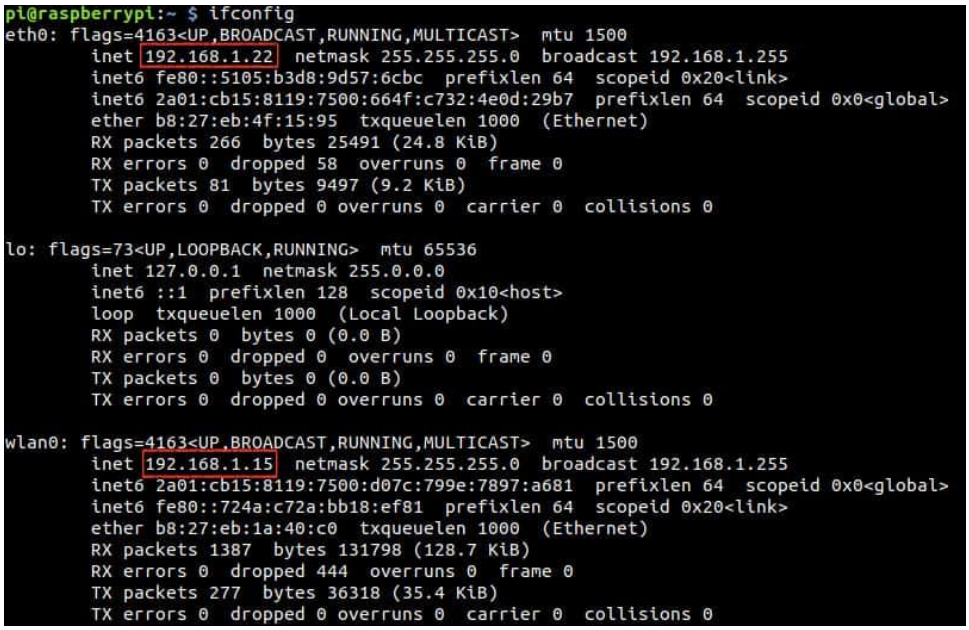


Figura 38: IP en ifconfig [18]

Con esto ya estaría cubierta la creación de un servidor para la aplicación web. Aplicación que, al ser lanzada desde Python, es compatible con el programa de recogida de datos descrito anteriormente en "3.5.2. Primer programa". Ahora se debe dar formato a la página web para que muestre los datos al usuario. Para eso serán necesarias algunas herramientas extra que se pasan a presentar en los siguientes apartados.

3.6.1.2. Diseño Web: HTML, CSS y JSChart

HTMNL (*HyperText Markup Language*) es el lenguaje que define y organiza el contenido (texto, tablas imágenes…) de cualquier página web. Utiliza un conjunto de etiquetas para diferenciar lo que hay en el interior de las mismas del resto de elementos que conforman la página, por ejemplo: <body>Texto</body> delimita el cuerpo principal de texto de la página. Se tiene entonces que escribir en este lenguaje la estructura de la página web que se vaya a mostrar al usuario usando Flask [19].

Pero si bien HTML gestiona contenido, el formato visual a la página se le da asociando un fichero de CSS (*Cascading Style Sheets*). Este es un lenguaje de diseño gráfico que se utiliza para dar un aspecto visual atractivo a páginas web escritas en HTML, así pues, siempre aparecerán estos dos lenguajes, representados por sus logotipos en la figura 39, trabajando en conjunto.

Figura 39: CSS y HTML [20]

En consecuencia, el proyecto constará de al menos tres ficheros escritos en diferentes lenguajes de programación que tienen que relacionarse entre sí. Uno es el fichero en Python del servidor web Flask, que utilizará el fichero HTML para dar estructura y contenido a la página web, que a su vez utilizará el fichero CSS para dar formato visual a su contenido. Además, si se hacen varias páginas web porque se quiera permitir que el usuario navegue por ellas, por ejemplo, para que en cada página se pueda visualizar un sensor distinto, se necesitará de varios ficheros HTML, uno para cada página. Para cubrir esta interacción, hay un esquema estándar de jerarquía entre los ficheros [21], que se muestra en la figura 40.

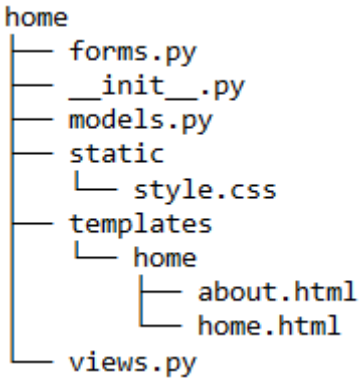


Figura 40: Esquema de proyectos en Flask [21]

Según este esquema de organización (figura 40), se guarda el fichero Python en un directorio raíz, dentro del cual estén guardados en una primera carpeta "*static*" el formato CSS de las páginas y en otra carpeta "*templates*" las páginas HTML accesibles.

Pero no es posible utilizar gráficas en nuestra página sin recurrir a una tercera herramienta: Chart.js. Esta es una librería de código libre que permite insertar gráficos de forma sencilla en una página web. Estos gráficos están programados en JavaScrip, otro lenguaje más que se apilaría sobre los ya utilizados Pero con la ventaja de que no es necesario conocer realmente el lenguaje, sino que será suficiente con tomar uno de los *templates* ofrecidos en la página del desarrollador y modificarlo [22].

Para insertar uno de estos gráficos en la web, primero hay que asignar un contenedor de *canvas* HTML mediante:

```
<div> <canvas id=”myChart”></canvas> </div>
```

El segundo paso es importar uno de los gráficos *templates* desde el CDN (*Content Delivery Network*) de Chart.js como *scrip*. Por ejemplo:

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

Dentro ya de la organización de bloques de contenido en la página mediante etiquetas HTML, se editan los ajustes del gráfico insertado desde Chart.js actuando dentro de la etiqueta “<script></script>”, de tipo HTML delimitadora. Anotar que los ajustes posibles sobre el gráfico los cuales cambian según el que sea utilizado.

3.6.1.4. Flask – SocketIO y conclusión

El problema que queda resolver es que, de crear un servidor web con solo estos elementos previos, los datos solo serían actualizados cuando el usuario recargara la página, pudiendo mostrar en ella solamente los últimos x datos recibidos, que se encontrarían almacenados en un array de longitud X. Esto se debe a que la petición de recarga de la página es lo que Flask recibe como petición de usuario que gestionar y enviar los nuevos datos.

Se podría incluir un botón que refrescase la página o bien forzar dicho refresco cada cierto intervalo de tiempo. Pero esto seguiría sin ser el tiempo real que se había marcado como objetivo. Una solución sería falsear el tiempo real por medio de forzar un refresco muy rápido de la página, pero no resultaría cómodo para ningún usuario y dependería mucho de la calidad de la conexión.

Se tiene que recurrir por tanto a otra opción para compartir mensajes (los datos) entre cliente (dispositivo usuario) y servidor (Raspberry) de manera bidireccional (para gestionar peticiones) e instantánea (tiempo real). La solución a este tipo de comunicación que se requiere es el uso de sockets. Los sockets son un concepto mediante el que dos programas situados en dispositivos diferentes (cliente-servidor) intercambian información una vez que el cliente ha abierto la comunicación. El servidor espera a esa conexión y responde en consecuencia.

El concepto de sockets tiene que ser soportado por una programación específica. En este caso, el recurso que los gestionaría sería la librería Flask-SocketIO. También de código abierto, expande sobre las funciones de Flask para permitir la comunicación por sockets. Anotar que, de forma similar a lo que sucedía con la librería del ADS1115 esta librería solamente es compatible con Python a partir de la versión 3.

En consecuencia, para lograr que los datos se actualizasen en tiempo real, lo que se debiera hacer es establecer una conexión entre cualquier número de usuarios clientes y el servidor de la Raspberry. Esta última, iría cargando los datos de los sensores en el socket creado, datos que serían alimentados a las gráficas. Se encontró un proyecto, mostrado en la figura 41, de funcional similar a lo que idealmente se planeta conseguir, que se pretendía utilizar como referencia para comenzar a adaptar las funciones [23].

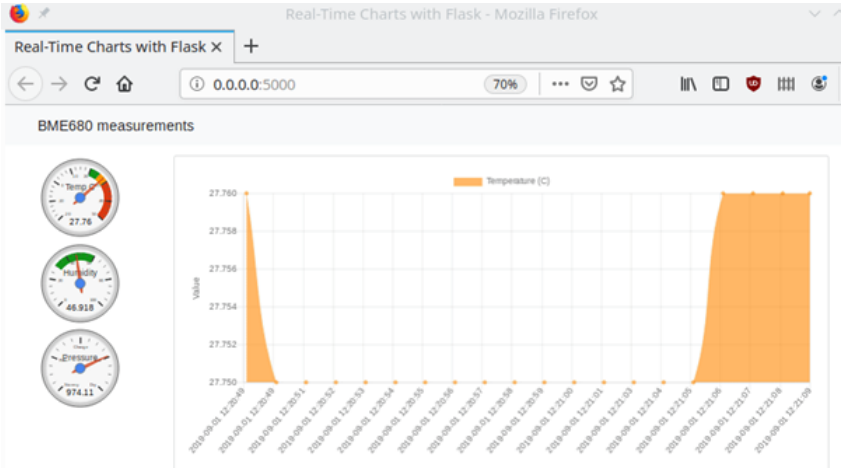


Figura 41: Ejemplo dashboard [23]

Sin embargo, durante el proceso de adaptación del mencionado programa, se encontró una forma más eficaz y sencilla de implementar una solución de tiempo real de forma más sencilla y eficaz que el conjunto de elementos explicados en este apartado. Esta nueva opción se consideró lo bastante atractiva como para abandonar por completo la línea seguida hasta este punto basada en Flask y el resto de los complementos mencionados. Se explicará de manera desarrollada en el siguiente apartado “3.6.2. Solución escogida”.

Mencionar que el programa que se desarrolló hasta este punto, con su estructura de carpetas jerárquicas y comentado, se encuentra como anexo al presente TFM. Lo que se consiguió desarrollar hasta que se abandonó la opción fue mostrar en página web una gráfica estática de los últimos X datos recibidos con un muestreo de unos milisegundos, como se muestra en la figura 42. Este muestreo se guarda en un array que se grafica en función del tiempo, usando las herramientas mencionadas anteriormente (Flask, programación HTML5 y CSS y JSChart).

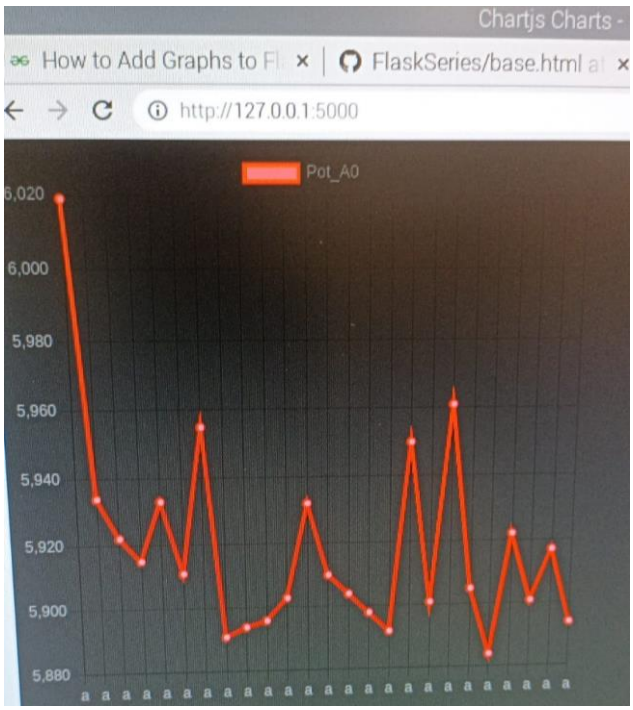


Figura 42: Gráfica con Flask

3.6.2. Solución escogida

Pasamos en este punto a explicar la solución finalmente desarrollada e implementada a nivel de software para lograr visualizar los datos en tiempo real. Esta solución pasa por utilizar un conjunto de programas prestablecidos que cumplen los roles que son necesarios para la aplicación y objetivos. En los siguientes apartados se pasa a introducir estos programas.

3.6.2.1. InfluxDB

Es una base de datos basada en serie temporal, de código libre. Una base de datos es una colección de información organizada a cuyos elementos puede acceder un usuario mediante comandos en un software gestor. Los bloques de elementos de una base de datos se apilan en ella en filas, siendo las columnas las características en común de los distintos elementos. La mayoría de las bases de datos se programan en SQL (*Structured Query Language*), utilizado para apilar, manipular, definir y controlar los elementos de la base de datos. Lo que diferencia a InfluxDB (logotipo en figura 24) es que está basada en serie temporal, lo que significa que cada fila de elementos de la base de datos tiene asociada, automáticamente, el momento (*timestamp*) en el que se añadieron a ella, y son inseparables de esa firma temporal [24].

Figura 43: InfluxDB [24]

Estas características vuelven a InfluxDB ideal para nuestra aplicación, ya que siempre podremos ver la evolución temporal de los datos de los sensores, porque va a existir en todo momento un instante de tiempo asociado a ellos. Además, como considerábamos en puntos anteriores, al ser código libre, encontramos mucha información online sobre cómo utilizarlo.

La instalación en Raspberry de InfluxDB es un poco diferente a lo que se hace generalmente con otros paquetes. Y es que hay que añadir la "llave" (*key*) del repositorio propio de InfluxDB antes de poder instalarlo. Esto se hace mediante el siguiente comando [25]:

```
curl https://repos.influxdata.com/influxdata-archive.key | gpg --
dearmor | sudo tee /usr/share/keyrings/influxdb-archive-keyring.gpg
>/dev/null
```

Una vez agregada la *key*, añadimos el repositorio de InfluxDB como fuente de descargas para el gestor de paquetes (llamado APT) de la Raspberry ejecutando el comando [25]:

```
echo "deb [signed-by=/usr/share/keyrings/influxdb-archive-
keyring.gpg] https://repos.influxdata.com/debian $(lsb_release -cs)
stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

Una vez añadida la fuente, se actualiza el gestor para asegurar que considera InfluxDB como fuente con el comando [25]:

```
sudo apt update
```

una vez hecho esto, ya se puede instalar InfluxDB normalmente con el comando [25]:

```
sudo apt install influxdb
```

Completados estos pasos, InfluxDB ya está instalado, pero no se inicia automáticamente al encender la Raspberry (*boot*) junto con el resto del sistema. Para que lo haga, es necesario ejecutar tres comandos más, el primero para asegura que pueda ser iniciado (*unmask*), el segundo para habilita InfluxDB y el tercero lo activa. La secuencia de comandos es [25]:

```
sudo systemctl unmask influxdb
```

```
sudo systemctl enable influxdb
```

```
sudo systemctl start influxdb
```

Ahora que ya está instalado y habilitado InfluxDB, es recomendable reiniciar la Raspberry. Una vez reiniciada, se puede proceder a crear una base de datos que será utilizada para almacenar los datos recibidos de los sensores. Se pueden crear varias bases de datos, pero solamente una de ellas estará en uso en cada momento (se podrá leer/escribir en ella). Para crearla, se escribe en la línea de comandos de Raspberry lo siguiente [26]:

```
influx
```

Esto inicia el servicio propio de InfluxDB, cuya línea de comandos se superpone con la de Raspberry. Para interrumpir este servicio, se debe introducir la combinación de teclado "ctrl+c". Los comandos en InfluxDB se escriben siempre en mayúsculas, para diferenciar de los normales de Raspberry. Se crea una base de datos con el siguiente comando, siendo la última palabra el nombre de la base creada [26]:

```
CREATE DATABASE mydb
```

Y para fijar que esta será la base de datos en uso se introduce [26]:

```
USE mydb
```

Ahora ya es posible escribir elementos en la base de datos. Como prueba, puede utilizarse el comando "INSERT". Otros comandos a tener en cuenta son "*DROP*", que muestra una lista con todos los elementos de la base de datos en uso y "*EXIT*", que cierra de forma segura la interacción con InfluxDB [26].

Si bien se podría hacer la prueba de escritura de datos con "INSERT", como se mencionaba, es realmente preferible escribir directamente escribir sobre ella utilizando programa de muestreo de datos adaptado, ya que es más sencillo para mantener la estructura de dichos datos [26]. Se verá cómo se hace esta escritura automática sobre la base de datos más adelante, en el apartado "3.6.2.4. Programa final".

Para ver los datos guardados en una base de datos en uso, en la que ya hayamos escrito, de forma que se pueda comprobar que se están almacenando correctamente, usamos

"*DROP*". El aspecto será el que se observa en la figura 44, con el conjunto de las medidas siempre incluyendo el *timestamp* en la primera columna izquierda, y siendo el resto de números los de la estructura que hayamos definido. Anotar que el *timestamp* está representado según el estándar de Unix como el número de segundos desde 1970.

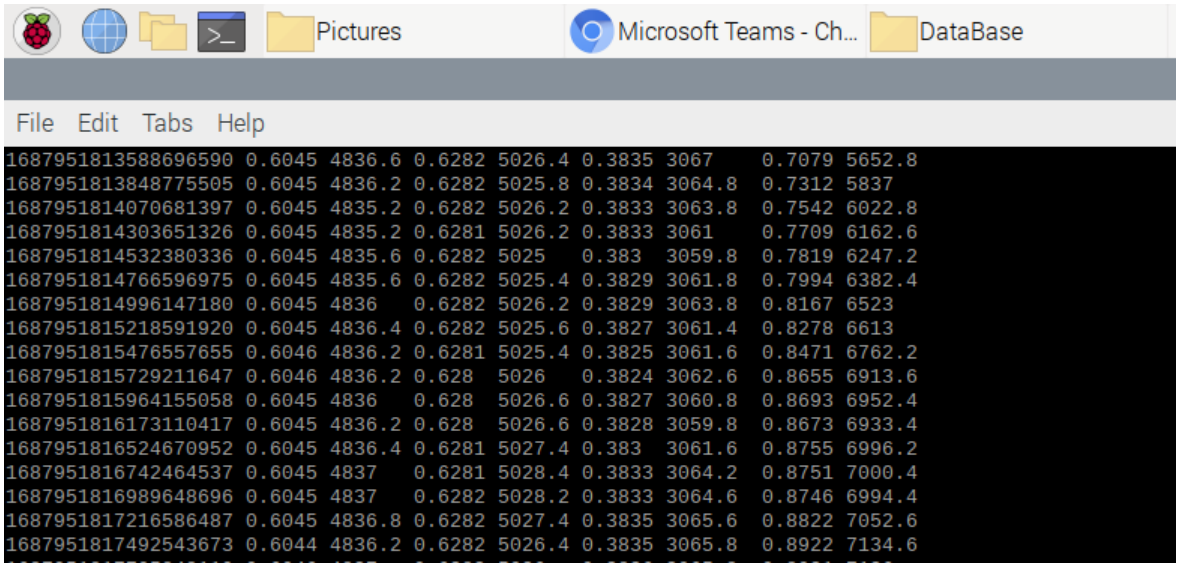


Figura 44: Base de datos en uso

3.6.2.2. Grafana

Se trata de una herramienta multiplataforma de visualización interactiva de datos basada en web, de logo en la figura 45. Es de código libre, si bien también cuenta con una versión de funcionalidades ampliadas de pago para empresas. Ofrece diversas opciones de *templates* predefinidos de gráficos, tablas, alertas y demás elementos de visualización. Su principal cometido es la creación de *dashboards* para mostrar información en tiempo real [27].

Figura 45: Grafana [27]

De nuevo, como con InfluxDB, su propósito resulta ideal para alcanzar los objetivos propuestos de visualización en tiempo real. Lo único que se necesita es alimentar a Grafana con una base de datos, y este se encargará de la representación en tiempo real. Además, utilizando InfluxDB como esta base de datos, se asegura que estos datos vayan siempre a seguir una secuencia temporal, y es compatible con Grafana.

Resulta mucho más sencillo utilizar estos dos elementos, Grafana e InfluxDB que toda la acumulación de recursos diversos necesarios para la primera propuesta que veíamos en el apartado "3.6.1. Alternativa descartada" anterior. Es mejor utilizar estas dos soluciones predefinidas, cuyo propósito se alinea directamente con los objetivos pospuestos, que intentar crear una alternativa desde cero. Estas dos soluciones están diseñadas y optimizadas para el uso que se les pretende a dar, y nos van a ofrecer unos resultados

más rápidos, reactivos, cercanos a tiempo real, así como resultar más sencillo de programar y comprender.

Para instalar Grafana y comenzar a utilizarlo en la Raspberry, se sigue un proceso muy similar al presentado anteriormente para InlfuxDB. Se añade la llave (*key*) utilizando [28]:

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

Y se agrega el repositorio de Grafana al gestor APT, actualizando al tiempo el gestor [28]:

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt-get update
```

Y con esto ya se puede instalar Grafana normalmente [28]:

```
sudo apt-get install -y grafana
```

Como sucedía también con InlfuxDB, aunque esté instalado es necesario habilitar el inicio de Grafana en el *boot* de la Raspberry así como activar su servidor usando los comandos [28]:

```
sudo /bin/systemctl enable grafana-server
sudo /bin/systemctl start grafana-server
```

Se reinicia para asegurar la instalación y ya se puede comenzar a trabajar con Grafana. Como se mencionaba anteriormente, es una aplicación basada en web, así el acceso a Grafana se hace desde el navegador de Raspberry (Chromium, por defecto en la instalación). El puerto que utiliza Grafana es el 3000, así que se escribe en la barra de dirección la IP de la Raspberry (que se obtiene según se veía en el apartado "3.6.1.1. Flask") seguida de este puerto de acceso, según se muestra a continuación [28]:

```
http://<Dirección IP de la Raspberry>:3000
http://localhost:3000
```

Aparecerá entonces la pantalla inicial de *login* de Grafana, mostrada en la figura 46. Se accede escribiendo como usuario "*admin*" y contraseña por defecto "*admin*". A continuación, pedirá que introduzcamos una nueva contraseña para el usuario "*admin*". Es posible escribir una nueva contraseña segura o continuar utilizando la predeterminada. Esta contraseña puede ser cambiada en el futuro, así como añadir nuevos usuarios [28].

Más adelante (apartado "3.6.2.5. Interfaz en Graphana") veremos cómo crear una *dashboard* con base de datos asociada, por el momento queda configurado Graphana para poder comenzar a trabar con él.

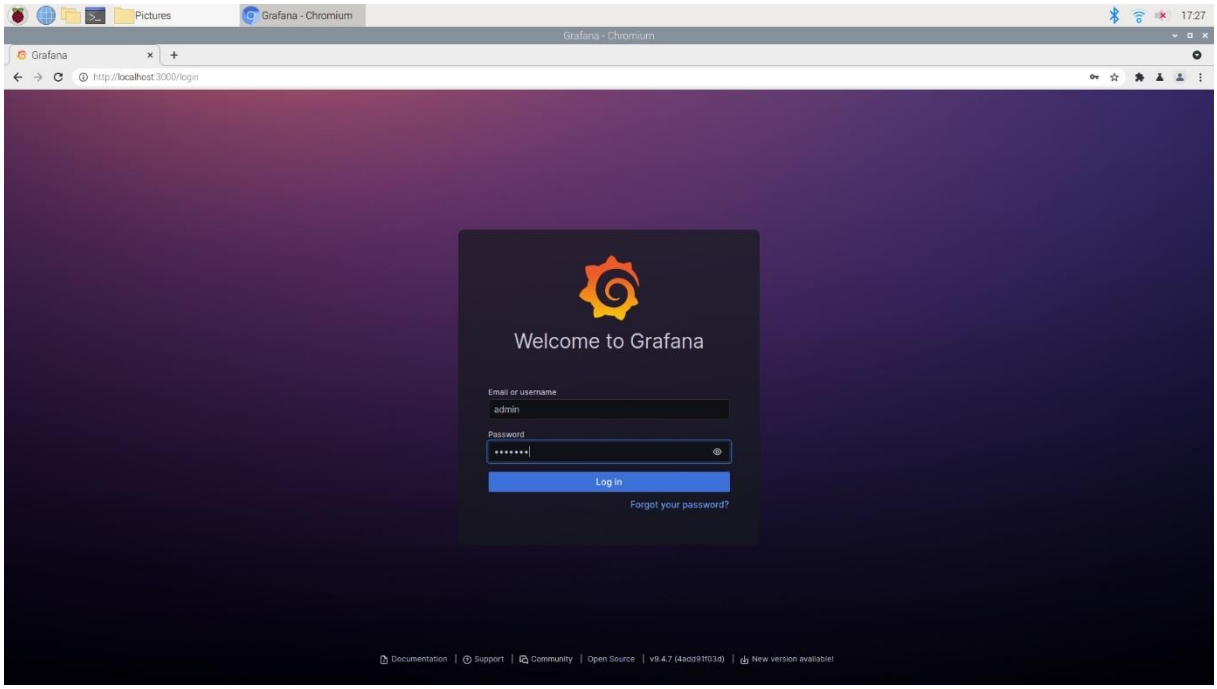


Figura 46: Login Grafana

3.6.2.3. Librerías auxiliares

Se han incluido en el proyecto, además de InfluxDB y Graphana, algunas librerías auxiliares de Python para mejorar la ejecución y simplificar la programación, siendo:

- **Time**: Con esta librería de Python, se introduce un pequeño retardo (*delay*) al final del bucle de ejecución del programa dando tiempo a que se actualicen los datos en todos los bloques posteriores del conjunto del proyecto. Téngase en cuenta que primero se lee por comunicación I2C del ADS1115, se aplican operaciones en el programa de Python, se escriben los datos sobre InfluxDB y de ahí, por último, lee Grafana para representar el gráfico tiempo real. Operaciones que conllevan cierto tiempo [29].
- **Board y Busio**: Ya se podían ver utilizados en la primera versión del programa con solo la lectura de datos, pero se aprovecha este apartado para explicarlos. Board sirve simplemente para acceder a los pines GPIO por sus nombres de función. Busio, por su parte, es una mejora del manejo base de las comunicaciones cableadas de Raspberry, y se utiliza para gestionar la conexión I2C con el ADS. Trabajan ambos en conjunto con la librería de Adafruit para el control del ADS1115 [30].
- **Numpy**: Es una librería matemática de código libre, inspirada por Matlab, que se utiliza principalmente para simplificar las cuentas con los *arrays* de datos. Si bien Python cuenta con *arrays* de forma predeterminada, para poder operar sobre ellos haciendo un sumatorio de sus elementos guardados (como se verá que es preciso), hay que recorrerlos con un bucle *for*. Usando Numpy, que cuenta con más opciones de operación, se puede hacer un sumatorio directo. Se debe tener en cuenta que para poder operar con ellos, hay que declarar los *arrays* como los pertenecientes a la librería de Numpy, no del tipo estándar de Python [31].

3.6.2.4. Programa final

Agregando todos los elementos anteriormente descritos, se puede ya pasar a escribir un programa, que, utilizando como base el de recopilación de datos que ya se ha visto ("3.5.2. Primer programa"), que escriba sobre la base de datos creada en InfluxDB para después alimentar con ella a Grafana.

Además de la lectura-escritura, se incluye un filtrado digital, con objetivo de estabilizar las medidas, conocido como media móvil. Esto consiste en tomar las n primeras medidas recibidas y devolver la media de las mismas como resultado. Cuando se recibe la medida n+1, se elimina la primera medida recibida y la medida n+1 pasa a la posición n, siguiendo una suerte de estructura FIFO, y se recalcula la media. Esto asegura una mejor estabilidad de las medidas finales frente a pequeñas variaciones.

Para escribir sobre la base de datos, es necesario organizar las lecturas en un diccionario. Un diccionario es un tipo de dato de Python que será lo que realmente se escriba como nuevo elemento sobre la base de datos. Estas estructuras se basan en pares llave (*key*) - valor, siendo la *key* una referencia para acceder al valor. En el presente programa, la *key* será la referencia al canal y el valor, el que se haya recibido en el mismo. Esta estructura pasará a ser directamente los campos de la base de datos.

El programa se divide en una primera sección en que se importan y declaran los elementos necesarios y una segunda sección que es un bucle *while* infinito en donde se lee, se calcula la media y se escribe sobre InfluxDB en un bucle constante. Para detener la ejecución del programa, debe ser interrumpirlo con la combinación "ctrl+c".

Pasando al programa en sí, se presenta directamente aquí con comentarios para mejorar la comprensión del mismo, y preparado para recibir de los 4 canales posibles:

```
#Se importan las librerías necesarias para el programa
import time
import board
import busio
import numpy as np
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from influxdb import InfluxDBClient

# Se declara la vía de comunicación I2C
i2c = busio.I2C(board.SCL, board.SDA)
ads = ADS.ADS1115(i2c)

#Se declaran los canales en uso
A0 = AnalogIn(ads, ADS.P0)
A1 = AnalogIn(ads, ADS.P1)
A2 = AnalogIn(ads, ADS.P2)
A3 = AnalogIn(ads, ADS.P3)
```

```
#Se declaran arrays de 5 posiciones para la media móvil
A0_Xarr = np.array([0, 0, 0, 0, 0])
A0_Varr = np.array([0, 0, 0, 0, 0])
A1_Xarr = np.array([0, 0, 0, 0, 0])
A1_Varr = np.array([0, 0, 0, 0, 0])
A2_Xarr = np.array([0, 0, 0, 0, 0])
A2_Varr = np.array([0, 0, 0, 0, 0])
A3_Xarr = np.array([0, 0, 0, 0, 0])
A3_Varr = np.array([0, 0, 0, 0, 0])

#Instancia del cliente de InfluxDB
client = InfluxDBClient(host = 'localhost', port = 8086)

#DB en que se guardarán las medidas, creada previamente
client.switch_database('mydb')

#Se escribe por pantalla la parte superior de la tabla
print("| A0_val |   A0_V | A1_val |   A1_V | A2_val |   A2_V |
A3_val |   A3_V |".format(*range(6)))
print("-"*37)

#Bucle de ejecución del programa
while (True):
    #Inserción de nuevo elemento en array de la media móvil
    A0_Xarr = np.delete(A0_Xarr, 0)
    A0_Xarr = np.append(A0_Xarr, A0.value)
    A0_Varr = np.delete(A0_Varr, 0)
    A0_Varr = np.append(A0_Varr, round(A0.voltage,4))
    A1_Xarr = np.delete(A1_Xarr, 0)
    A1_Xarr = np.append(A1_Xarr, A1.value)
    A1_Varr = np.delete(A1_Varr, 0)
    A1_Varr = np.append(A1_Varr, round(A1.voltage,4))
    A2_Xarr = np.delete(A2_Xarr, 0)
    A2_Xarr = np.append(A2_Xarr, A2.value)
    A2_Varr = np.delete(A2_Varr, 0)
    A2_Varr = np.append(A2_Varr, round(A2.voltage,4))
    A3_Xarr = np.delete(A3_Xarr, 0)
    A3_Xarr = np.append(A3_Xarr, A3.value)
    A3_Varr = np.delete(A3_Varr, 0)
    A3_Varr = np.append(A3_Varr, round(A3.voltage,4))

    #Cálculo de media y escritura en la variable
    #Nota: Hay que redondear a tensión o puede desbordar
    A0_Xavg = np.sum(A0_Xarr)/5
    A0_Vavg = round((np.sum(A0_Varr)/5),4)
    A1_Xavg = np.sum(A1_Xarr)/5
    A1_Vavg = round((np.sum(A1_Varr)/5),4)
    A2_Xavg = np.sum(A2_Xarr)/5
    A2_Vavg = round((np.sum(A2_Varr)/5),4)
    A3_Xavg = np.sum(A3_Xarr)/5
    A3_Vavg = round((np.sum(A3_Varr)/5),4)
```

```
#Se usa un diccionario para guardar las medidas resultantes
#El nombre de la medida en la base de datos es "ADS1115"
#El resto de valores son campos dentro de la misma
pt = [{"measurement":"ADS1115",
 "fields":{"A0_X":A0_Xavg,"A0_V":A0_Vavg,"A1_X":A1_Xavg,"A1_V
 ":A1_Vavg,"A2_X":A2_Xavg,"A2_V":A2_Vavg,"A3_X":A3_Xavg,"A3_V
 ":A3_Vavg}}]

#Se escribe el diccionario en la base de datos
client.write_points(pt)

#Se muestran los valores por pantalla y damos un retardo
print(pt)
time.sleep(0.1)
```

3.6.2.5. Interfaz en Graphana

Ahora que ya hay información guardada en la base de datos y un programa que puede seguir escribiendo sobre ella, podemos asociar esta base de datos como fuente en Grafana y crear una *dashboard* para visualizar los datos.

Antes de preparar la *dashboard* en sí, es conveniente mejorar el intervalo de refresco de Grafana, es decir, el tiempo entre actualizaciones del *dashboard*, para tener una mejor precisión en la visualización. Para ello, se accede a la carpeta de instalación de Grafana desde la línea de comandos de la Raspberry escribiendo [32]:

```
cd /etc/grafana/
```

Y se abre el fichero" grafana.ini", que contiene la configuración de Grafana, usando [32]:

```
sudo nano grafana.ini
```

Abierto el archivo, se navega por él, hasta encontrar una línea que declara "min_refresh_interval" y se borran los "5 s" definidos por defecto y se escribe en su lugar "100 ms", según se muestra en la figura 47. Guardando y cerrando el archivo, se habrá cambiado el intervalo de refresco [32].

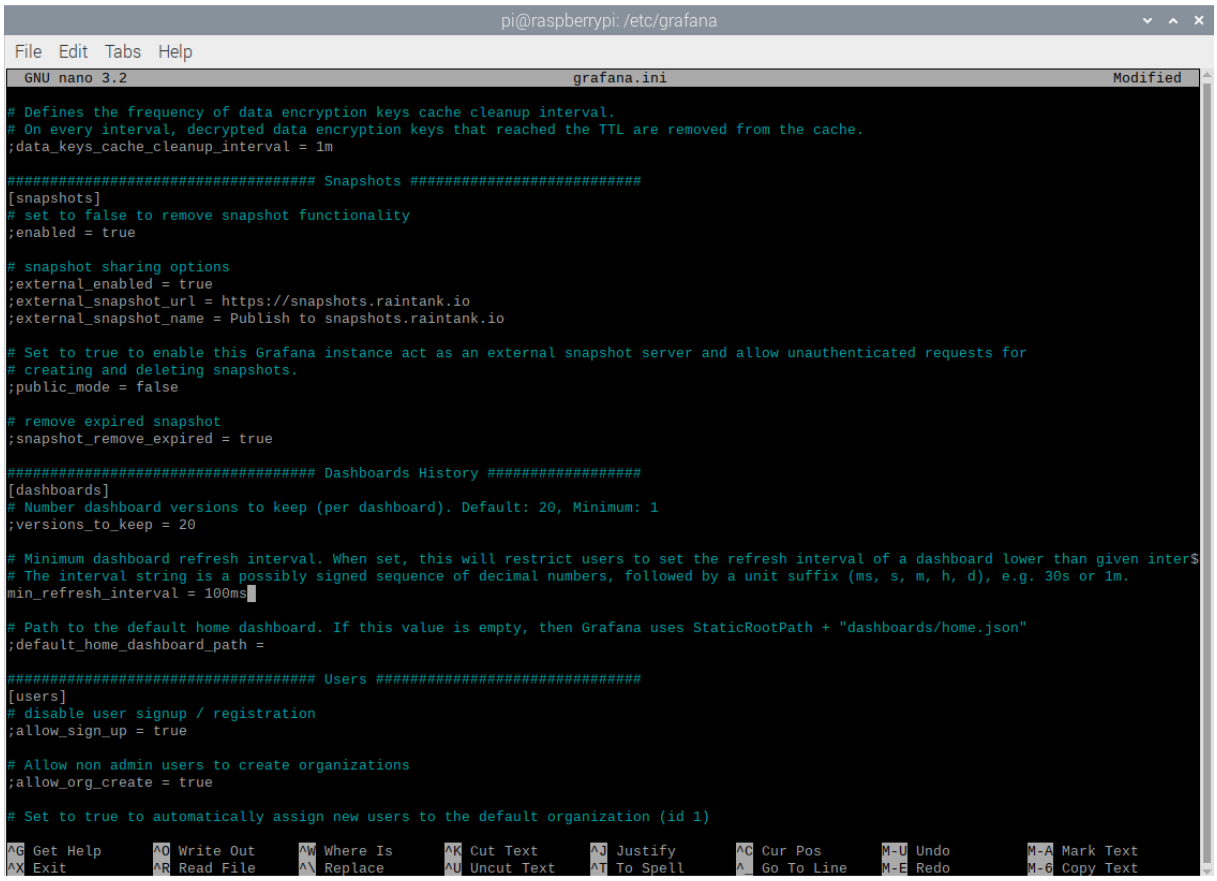


Figura 47: Edición de grafana.ini

Ahora se puede añadir ya la base de datos de InfluxDB como fuente de datos (*Datasource*). Para ello, se selecciona "*Configuration*" en el icono de engranaje que se encuentra en la parte inferior izquierda de la pantalla inicial de Grafana. Se abrirá una ventana con el aspecto que se muestra en la figura 48. Después se pulsa en el botón azul "*Add new data source*", que se ve en la figura 48 situado sobre las bases de datos ya añadidas, en la zona superior derecha. De entre las opciones ofrecidas que aparecen, se selecciona InfluxDB como tipo de base de datos fuente [33].

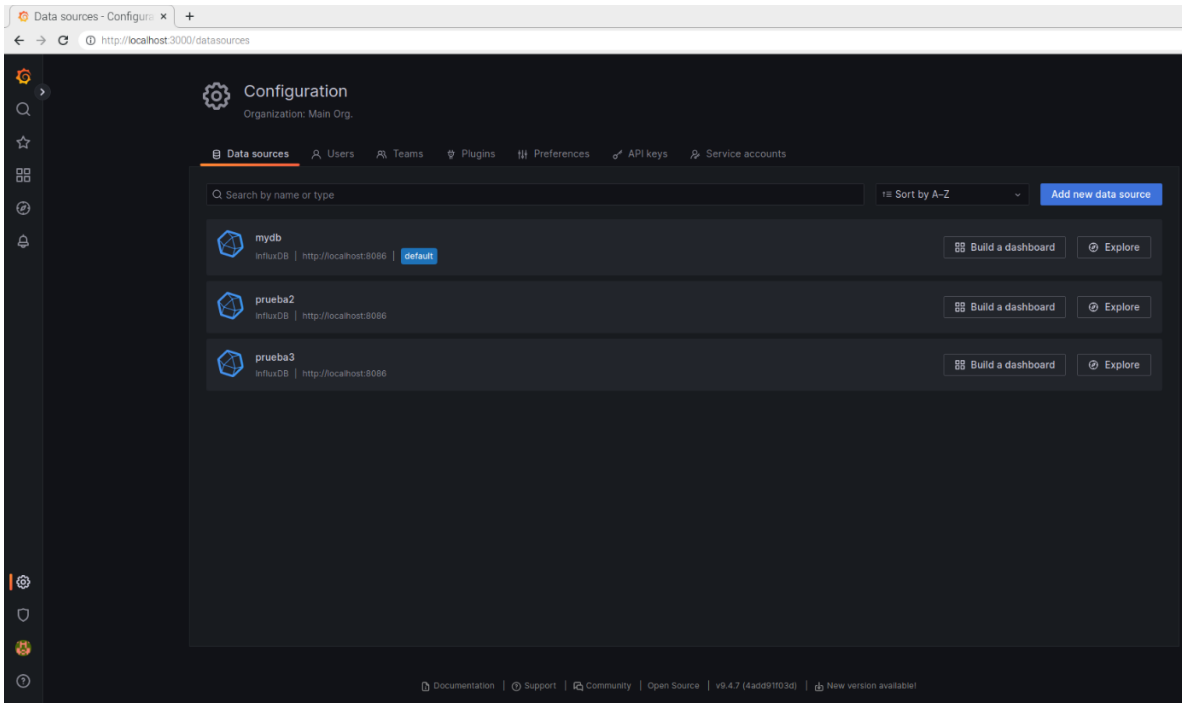


Figura 48: Configuration Grafana

En la configuración, se mantiene todo como parece de forma predeterminada, a excepción de la URL, en la que tenemos que escribir "http://localhost:8086", que se refiere a la dirección IP local y el puerto utilizado por InfluxDB. También cambiaremos "*Min time interval*" a 100 ms para poder tener mayor margen de tiempo de refresco, según hemos configurado antes [32]. Se deberá concluir con una ventana como la que se representa en la figura 49.

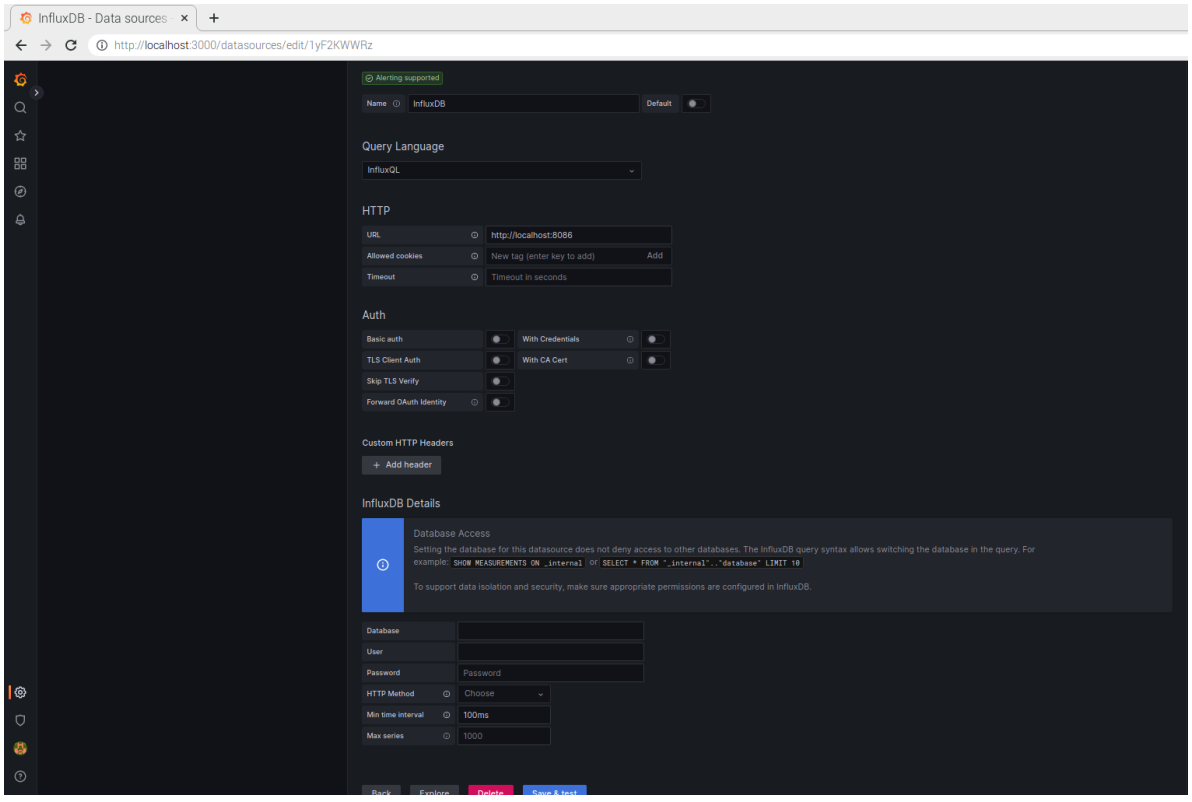


Figura 49: Datasource Grafana

Ahora ya se puede pasar a la creación de un *dashboard* para mostrar los datos recogidos en la base de datos. Se selecciona "*New dashboard*" desde su icono en la barra izquierda de herramientas y se agregan elementos en esta nueva *dashboard* usando el botón situado a la izquierda del de guardar, según se ve en la figura 50. En el caso particular de la presente aplicación, se añadirán 4 gráficas que muestren la evolución temporal de los valores analógicos de las células y potenciómetros y otras tantas "*Gauge*" que muestren el valor instantáneo de las tensiones de dichos sensores. Para poder guardar el aspecto y proporciones finales que se muestra en la figura 50, es necesario añadir cuadros de texto vacíos que rellenen el espacio sobrante entre los *gauges* y las gráficas y límites de la pantalla.

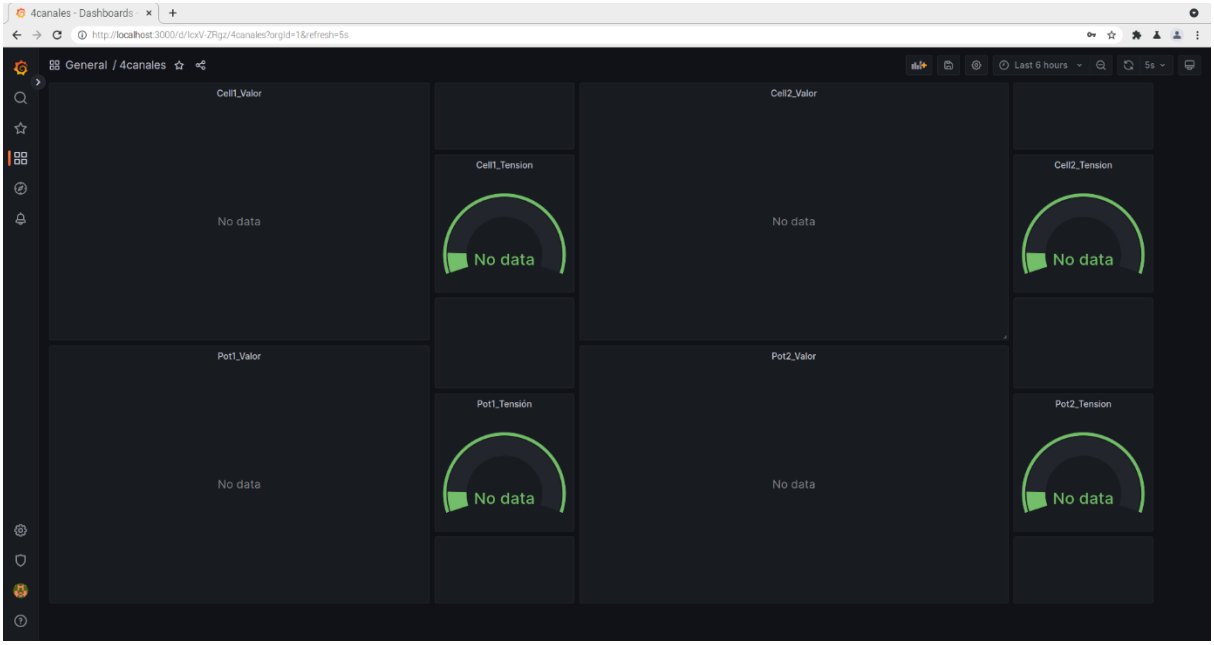


Figura 50: Diseño dashboard en Grafana

En los "*Settings*" generales del *dashboard*, accesibles desde el botón de engranaje a la derecha del de guardar, según se ve en la figura 50, se accede a los ajustes del *dashboard* concreto en edición. Se accede a la pantalla que se muestra en la figura 51, en dónde se editará el apartado "*Auto refresh*" para poder establecer tiempos menores de los 5 s por defecto. Se escribirá en esta sección 100 y 500 ms, según se ve en la menciona figura 51. También es posible cambiar el nombre del *dashboard* desde aquí.

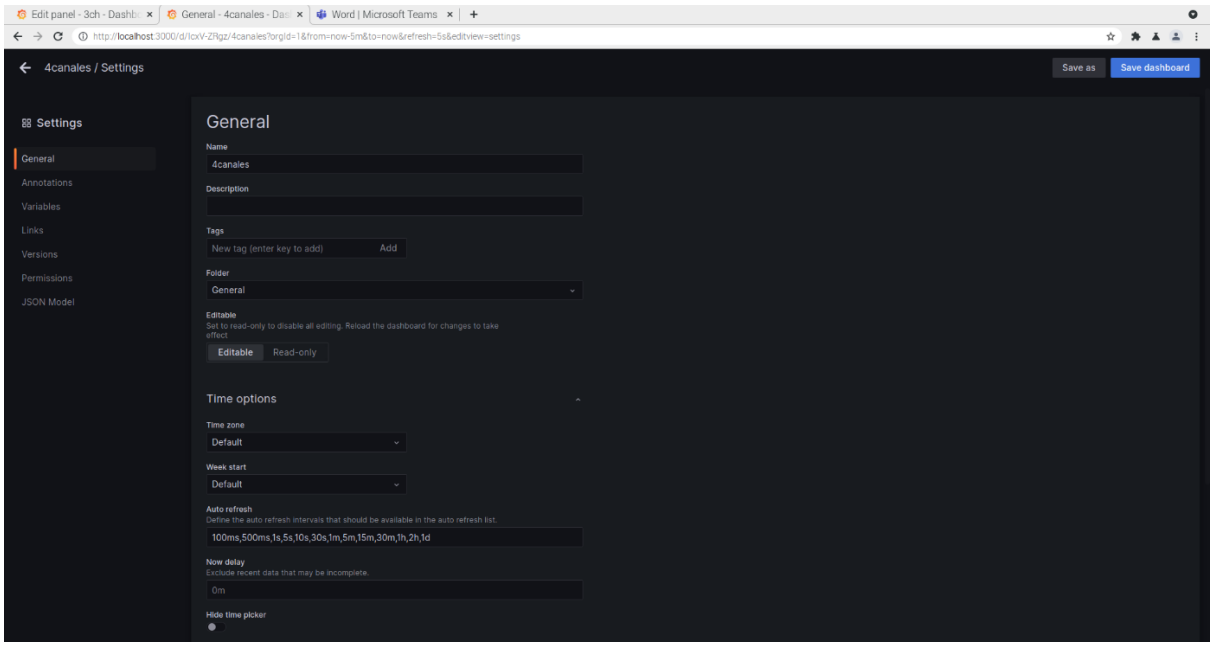


Figura 51: Auto refresh Dashboard settings

Se pasa ahora a editar los orígenes de datos de las gráficas para que estas muestren los datos recopilados por el programa. Para acceder a estas opciones, se pulsa sobre el botón "*Edit*" de la parte superior de las gráficas vistas en la figura 50, entrando en la pantalla de edición mostrada en la siguiente figura 52. En el apartado "*FROM*" se selecciona la medida de la base de datos en la que se guardan los datos, "ADS1115". En el apartado "*SELECT*", se pulsa sobre el recuadro adyacente para que se abra un desplegable con cada uno de los campos de la medida "ADS1115", que son los datos analógicos y de tensión de cada canal. Se selecciona el analógico que corresponda a la gráfica, titulamos dicha gráfica en correspondencia al dato seleccionado y se guarda. Debe quedar una pantalla editada de forma similar a la mostrada en la figura 52.

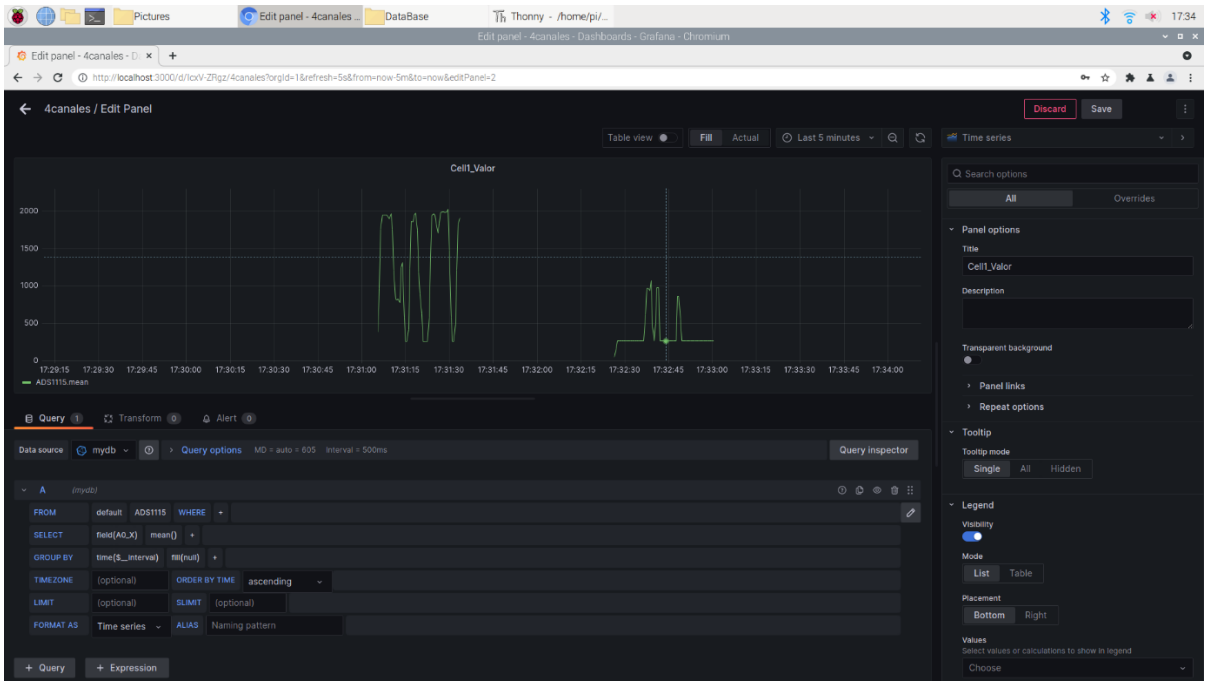


Figura 52: Settings para gráfica

Para la selección de dato en los gráficos tipo *Gauge*, se actúa de forma paralela a lo visto para la gráfica para seleccionar el origen de los datos, pero agregando un paso extra. Se busca en la columna derecha las opciones "*Standard options*", en dónde se cambiará el mínimo (Min) y máximo (Max) de la escala del gauge, de forma que visualmente en el gráfico, permanezca fija entre -0,5 y 5 V, que es el recorrido máximo esperado de tensión. Se debe alcanzar una edición similar a al mostrada en la figura 53.

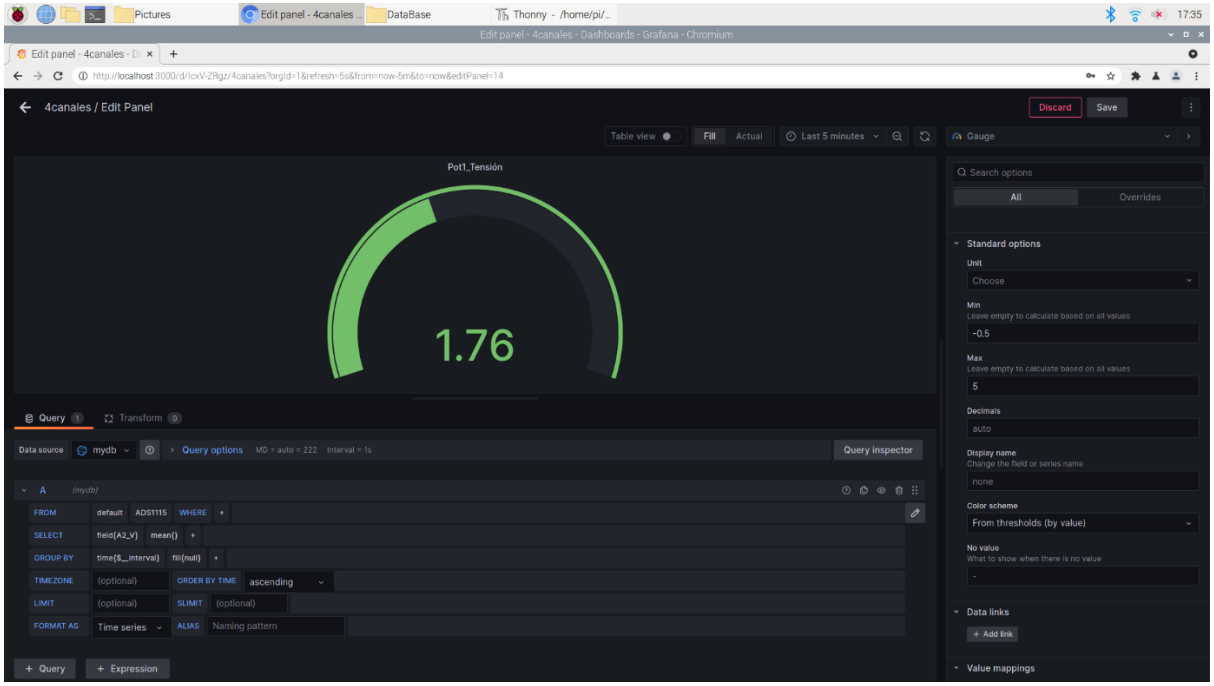


Figura 53: Settings para gauge

Con este último paso, queda el *dashboard* listo para recibir datos de los sensores desde la base de datos en que escribe el programa funcional. Cualquier usuario que comparta red WiFi con la Raspberry podrá ver desde un navegador el *dashboard* diseñado. Para ello, tendrá que escribir en la barra de direcciones la dirección IP de la Raspberry y el puerto correspondiente a Grafana (el 3000) tal que:

https://[Dirección IP de la Raspberry]:3000

4. Diseño electrónico

Ahora que ya está listo la parte del software, el siguiente paso será diseñar una PCB que haga el primer tratamiento de la señal de los sensores. Es decir, que adapte la señal de las células de carga, permita la conexión de los sensores (compatible con Sirius) y envíe la información al ADS1115 y de él a la Raspberry. Esto sustituirá al circuito de pruebas el protoboard utilizado para los programas de recogida de datos que se veían en el apartado "3.3.1. Circuito de pruebas" y evolucionará hasta convertirse en la versión definitiva utilizada en utilizada en el diseño final del equipo.

4.2. Eagle

Como programa para el diseño electrónico del prototipo, se utilizará el programa Autocad Eagle. Esta decisión se toma en base a que se trata de un software bastante completo e intuitivo que permite realizar tanto simulaciones sencillas (mediante ngspice) de funcionamiento de circuitos electrónicos (es el que utilizábamos en el apartado "3.3.3. Amplificador de instrumentación") como diseño electrónico general. El procedimiento general consiste en importar diferentes componentes al esquema eléctrico y unirlos con líneas de conexión. Todos los elementos pueden ser renombrados para claridad del esquema. Tiene el programa una opción que convierte el esquema trazado en diseño de PCB, pudiendo disponer en la superficie las huellas de los componentes seleccionados en el esquema y respetando las conexiones entre los mismos declaradas en el esquema eléctrico. Los cambios en esquema o PCB se actualizan automáticamente entre una y otra parte del programa. Además, Eagle tiene integración con Fusion360, que se utilizará posteriormente para el diseño de la envolvente del prototipo.

4.2.1. Restricciones

Nos planteamos una serie de restricciones o condiciones que debe de cumplir el circuito en PCB que vamos a diseñar. Estas condicionarán el diseño general del circuito y la forma y estructura de la PCB, por lo que se describen antes de pasar al diseño en sí.

4.2.1.1. Asociados a células de carga

Ya hemos hablado sobre el proceso de decisión de uso, selección y ajuste de los AI (apartado "3.5.3. Amplificador de instrumentación"), que son el componente que debemos utilizar para emplear las células de carga. Recordamos, necesita un AI por cada célula de carga y cada AI tendrá asociado dos potenciómetros para ajustar tensión de entrada al puente (Rx) y ganancia del AI (RG).

4.2.1.2. Compatibilidad con Sirious

Los sensores que conectemos al prototipo también deben de poder ser admitidos por el DAQ profesional Sirius. Esto obliga a utilizar conectores DB9, el principal de entre los posibles admitidos por Sirius. No sería estrictamente necesario usar este conector para nuestros sensores, que tienen unos requisitos de conexionado relativamente sencillos (3 cables para potenciómetros lineales, 4 para células de carga), pero se tiene que asegurar que los sensores se puedan utilizar tanto en nuestro diseño como en Sirius.

Además del propio conector, el problema añadido que conlleva esta compatibilidad es que Sirius exige un conexionado específico dentro de los DB9 de los sensores para admitirlos. Conexionado que obliga a una serie de puentes y pines del DB9 de uso específico que dependen del sensor utilizado. Este conexionado podemos verlo dentro del propio Sirius seleccionado un nuevo tipo de sensor a conectar.

Comprobando en Sirius el necesario para sensores de tipo potenciómetro y para sensores con puente de Wheatstone completo (células de carga) y encontramos que el conector que utilicemos debe cumplir, resumido en el esquema de figura 54 [2]:

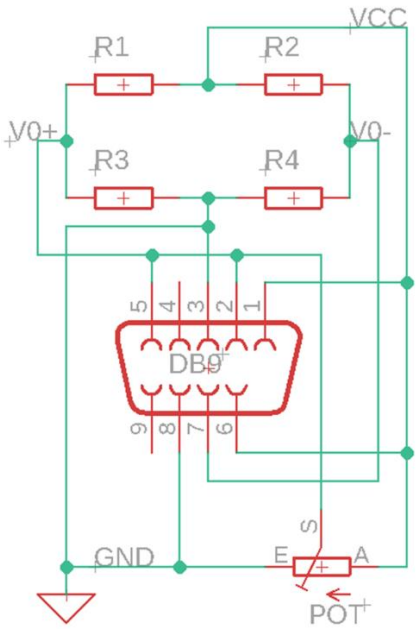


Figura 54: Esquema de conexión DB9 para Sirius

Los DB9 que se utilicen en nuestro dispositivo tendrán que ser compatibles con este, y por tanto se emplearán los mismos pines de alimentación y de salida del sensor. Anotar también que los DB9 de entrada tendrán que ser hembra (como en Sirius) y que es recomendable buscar un DB9 en ángulo recto para facilitar la soldadura en la PCB.

4.2.1.3. Admisión de puentes y potenciómetros

Estrechamente relacionado con el apartado anterior, de la misma forma que el Sirius es capaz de admitir tanto potenciómetros como células de carga de puente completo, se plantea que lograr que el dispositivo desarrollado también tenga esa capacidad.

Para ello, se pretende aprovechar alguno de los pines que quedaban libres como señal digital *Enable* para conocer qué (célula, potenciómetro o nada) hay conectado en concreto en el DB9. Se utilizarán los pines 5 y 9 para esta función, ya que no se contará con una resistencia de Shunt ni el chip TEDS en ninguno de los sensores utilizados en el taller. Uno de estos pines en activo significará "potenciómetro conectado" y el otro "célula de carga conectada" [2].

Es necesario distinguir entre dispositivos conectados porque la alimentación es distinta en uno y en otro. Los potenciómetros van alimentados directamente a 5 V mientras que la alimentación de las células de carga debe de pasar antes por la resistencia Rx, previa al puente, que asegura la tensión mínima de entrada necesaria al AI.

Así que tenemos que intercambiar la ruta de alimentación según haya uno u otro sensor conectado. Este cambio supone abrir o cerrar una ruta de alimentación, como si fuese un interruptor, en función de la mencionada señal de tensión *Enable*. Esa señal *Enable* la se conseguirá puenteando los 5 V de alimentación de potenciómetro o de célula de carga, según corresponda, a uno de los pines libres. Se seguirá, por tanto, en el cableado de los DB9 macho soldados a los sensores utilizados para la conexión con la PCB el esquema que se muestra en la figura 55.

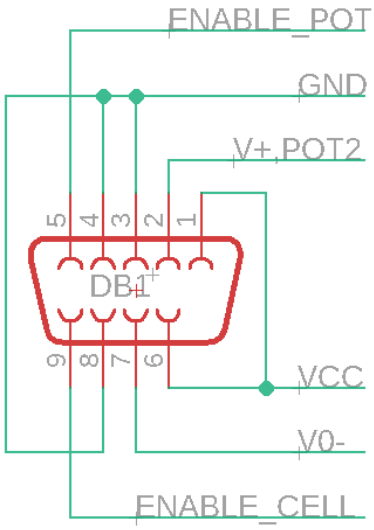


Figura 55: Señales "enable"

Esta función de interruptor suele hacerse mediante transistores, que abren o cierran el camino colector-emisor en función de si reciben alimentación o no en la base, actuando, así como interruptores electrónicos. Sin embargo, el problema es que introducen en el circuito que abre una tensión colector-emisor de paso, según la figura 56 [34].

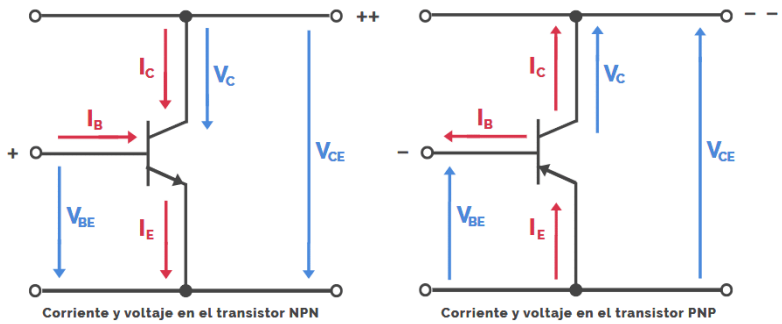
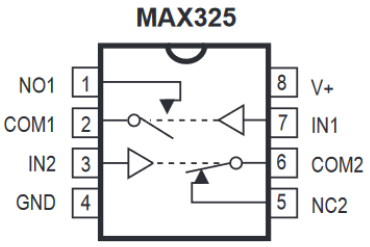


Figura 56: Tensiones en transistores [34]

Esta tensión que introducen es muy pequeña y se podría, por lo general, simular. Pero se considera preferible no alterar la alimentación de dispositivos sensibles como son los sensores. Así que, en lugar de transistores, se opta por utilizar un interruptor analógico integrado. Esto son dispositivos que, de forma similar al transistor base, abren o cierran el paso de corriente en un circuito según una señal digital recibida.

Se busca uno que se posible alimentar con los 5 V de la Raspberry que se emplea como alimentación de todo el circuito, encontrando que el modelo MAX325, representado en la figura 57, se adapta teóricamente a estas necesidades. Cuenta con dos interruptores internos, por lo que, en el diseño del circuito, se podrían emparejar las conexiones de los DB9 para tratar de que no se crucen las pistas de las PCB [35].



MAX325		
LOGIC	SWITCH	SWITCH 2
0	OFF	ON
1	ON	OFF

Figura 57: Switch analógico MAX325 [35]

4.2.1.4. Stackeable

Se quiere conseguir que el área de la PCB sea lo bastante reducida como para colocarse sobre la Raspberry, sobresaliendo lo mínimo posible de ella. La referencia tomada para este punto son los llamados HAT, ejemplo de los cuales se muestra en la figura 58, placas que se colocan sobre la Raspberry para ampliar sus funcionalidades o ejercer como interfaz con algún sistema. Estos HAT y la PCB diseñada por ende también, se fijan y alinean con los 40 pines GPIO de la Raspberry.

Apuntar a un diseño tan reducido restará espacio para los conectores DB9, bastante grandes en comparación con las dimensiones de la Raspberry. Para poder aspirar a un número elevado de conexiones, en consecuencia, será necesario que, además de apilarse

sobre la Raspberry, también puedan apilarse unos HATs/PCBs sobre otros, para incrementar el número de sensores conectados.

Dado que cada ADS1115 admite 4 canales de entrada y se pueden conectar 4 ADS1115 máximo a la Raspberry por I2C, recordando del apartado "3.4.1. El ADS1115" que a cada ADS1115 se le puede dar 1 de 4 direcciones I2C posibles. Se pretende entonces que a una PCB con 4 conectores DB9 que se coloque sobre la PCB como HAT, a la que se le puedan sumar otras 3 idénticas para un total de 4 PCB con 16 conectores DB9 para otros tantos sensores [10].

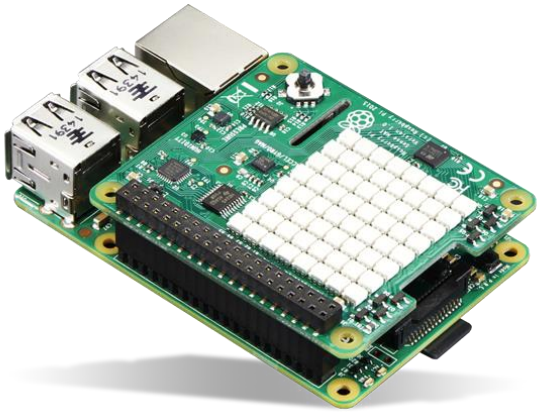


Figura 58: Ejemplo de HAT en Raspberry [36]

4.2.1.5. Agujero pasante

Hay dos tecnologías posibles de soldadura en PCBs: de superficie y de agujero pasante, ambas se ven representadas en la figura 59 inferior. En las de superficie los componentes son más pequeños y tienen unas superficies planas de soldadura. Se adhieren a las marcas de soldadura con una pasta de estaño que se funde al aplicarle calor en un hornillo. En el agujero pasante los componentes son más grandes y tienen unos pines metálicos. Estos pines se introducen por unos *pads*, agujeros metalizados en la PCB, y se fijan por soldadura con un estañador.

Se utilizará para el presente diseño componentes de este segundo tipo, de agujero pasante. Los motivos son que estos son más cómodos para el prototipado, pudiendo manipularse con mayor facilidad y permitiendo un desoldado, de ser necesario, relativamente sencillo. Además, no se dispone del equipamiento necesario para soldar correctamente componentes de superficie. El tamaño de la PCB no debiera verse demasiado afectado, ya que la mayor restricción de tamaño es el espacio necesario para los DB9, que ocuparán lo mismo independientemente del tipo de tecnología.

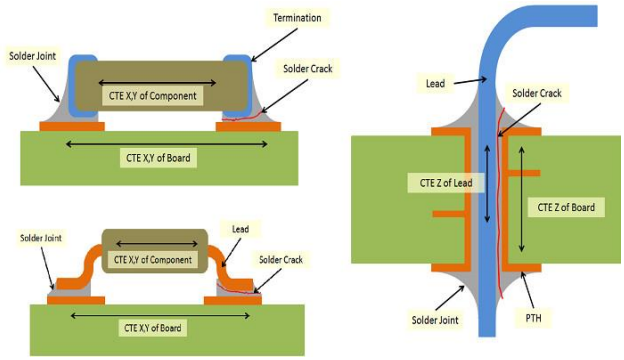


Figura 59: Tecnologías de soldadura [37]

4.2.3. Circuito eléctrico

Se comenzará trazando el circuito que incluya los componentes necesarios seleccionados, prestando especial atención a la correcta conexión entre dichos componentes. En el caso de los circuitos integrados (switches y AI) se debe tener presente la hoja de datos de los componentes para no errar en el uso de los pines.

Para el proceso de diseño, se opta por tomar como base uno de los puertos DB9 e ir trazando sus conexiones hasta llegar a los pines I2C de la Raspberry. Después, y ya comprobadas las conexiones, se hará uso de la herramienta copiar para tratar de reducir las posibilidades de error.

Así, se comienza el trazado de las conexiones en el DB9 que aseguran la compatibilidad con Sirius. Seguidamente, se marcan los dos pines (5 y 9) que señalizarán si hay conectada una célula o un potenciómetro utilizando LEDs. Estos servirán para dar *feedback* al usuario, confirmando la conexión. Es necesario limitar la corriente (no puede ser mayor de 15 mA) a los LED con resistencias de 330 Ω.

Se toman los dos pines de salida de la célula de carga y se conectan al AI. El INA118 tiene modelo incluido en Eagle, que se utilizará para asegurar las conexiones. La salida del AI irá junto con la salida directa del potenciómetro al primer canal del ADS1115. Se modela el ADS1115 en el esquema como una tira de pines, así como la Raspberry será una serie 2x20 de pines. Se deja una tercera tira de dos pines para conectar el ventilador para refrigeración y una cuarta de cuatro para seleccionar la dirección I2C del ADS1115.

En cuanto a la alimentación de sensores, hay que incluir en ella el MAX325, que no se encuentra en la librería de Eagle. Se utilizará en su lugar una suerte de *placeholder* DIP8, que permite y mantiene conexiones, pero no indica si hay errores en las mismas, como sí sucede con los otros modelos incluidos en librería. Habrá que tener especial cautela al conectar este dispositivo.

Se conecta la resistencia Rx previa el puente en el pin 2 de alimentación del DB9. Esta será la alimentación por defecto. Del mismo pin 2, se pasa una conexión por el interruptor analógico. Esta se abrirá solamente con una señal del *Enable* del potenciómetro, alimentado entonces a 5 V y sin resistencia Rx al potenciómetro, porque la corriente siempre seguirá en una malla el camino de menor resistencia.

Se utilizarán para las resistencias de ganancia y pre-puente Rx potenciómetros con huella tipo de multivuelta de precisión compactos, para asegurar la posibilidad de ajuste fino a las resistencias que sean necesarias.

Se copian y ajustan las conexiones para las otras 3 entradas DB9 que se corresponden con los 4 canales restantes del ADS1115. El MAX325 se conecta por parejas, uno para cada 2 DB9 del mismo lado. Se nombran todos los nudos de conexión para facilitar la comprobación de conexiones en la PCB. El resultado final del esquema, cumplidos estos pasos, queda como se muestra en la figura 60 inferior.

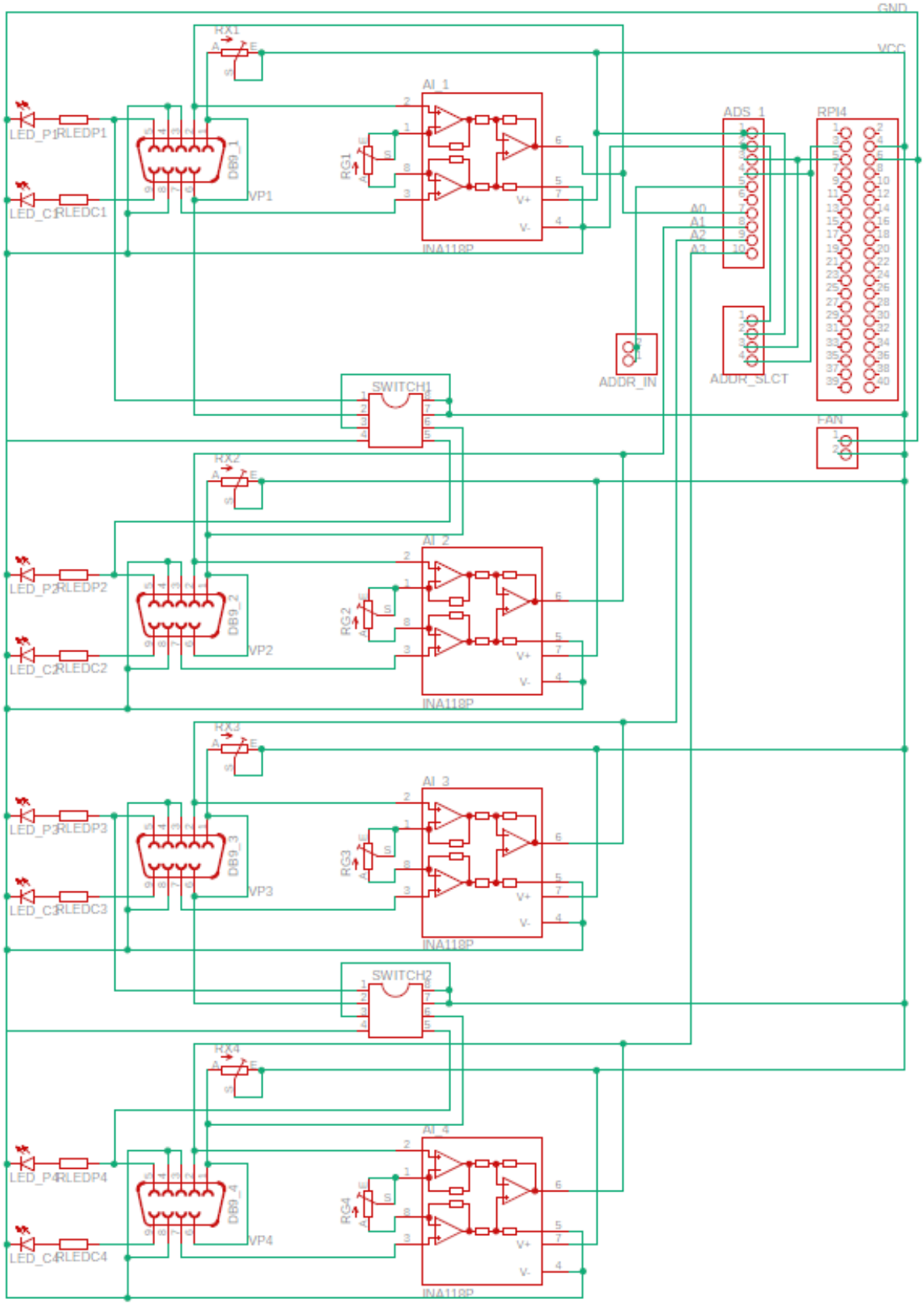


Figura 60: Esquema del circuito

4.2.4. PCB

Con el circuito definido, se puede pasar a convertirlo en PCB. Cuando se cambia de espacio de trabajo, aparecerán las huellas de los componentes, una PCB de tamaño predefinido y los *ratnest*, o líneas que indican qué pines comparten nudo de tensiones.

Lo primero que se recomienda hacer es ajustar una rejilla de tamaño 0,5 mm sobre la que guiar el trabajo. Las medidas de componentes, área de PCB y rutas están, por defecto en pulgadas, que es la unidad general de circuitos impresos, pero es necesario ajustarse al tamaño de la Raspberry, que viene dado en milímetros.

Para ello, se pulsa el botón de rejilla, inmediatamente debajo del de guardar en la interfaz de Eagle, según se ve en la figura 61. En los ajustes de *Grid*, se establece el tamaño de 0,5 mm deseado de espacio de rejilla y el aspecto de líneas, para tener una mejor guía, tal como se muestra en la figura 61. Es necesario asegurarse que el *display* está activo (en "*ON*") antes de aceptar.

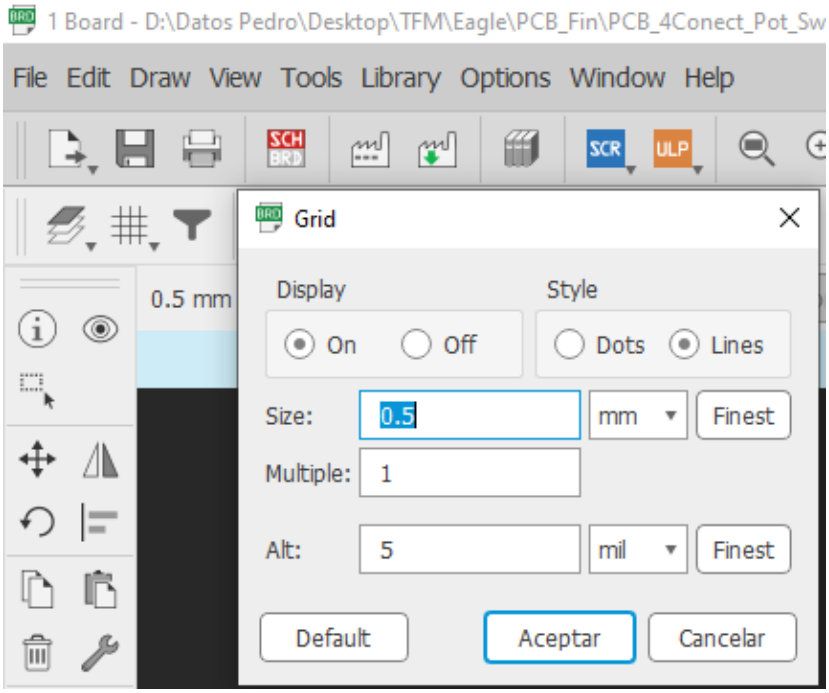


Figura 61: Selección rejilla

Con la rejilla ajustada, lo primero es colocar los pines de 2x20, que son los que deben conectarse a la Raspberry para actuar como HAT y servirán de guía. Se ajustará la medida del área de la PCB entorno a ellos, con objetivo de que la PCB ocupe la misma área que la Raspberry. Se colocarán los pines que representan el ADS1115 lo más cerca posible de los pines I2C de la Raspberry, midiendo para dejar suficiente área entre los unos y los otros como para que quepa el circuito impreso del ADS1115.

Ahora, se pasa a colocar en los bordes las huellas de los DB9. Se selecciona para ello los bordes más largos, lo que dejará más espacio para añadir junto a los conectores los LED indicadores. Seguidamente, se asocian las resistencias de ganancia con sus respectivos AI, y así como las resistencias pre-puente se asocian a los *switches* analógicos. Se colocan el resto de los componentes en el área interior libre.

Anotar que, a pesar de las medidas tomas, después de tratar de disponer todos los componentes por primera vez, se vio que sería necesario extender ligeramente en el ancho por fuera del área de la Raspberry. En la figura 62 se muestra los compo9nentes repartidos por el espacio de la PCB junto con las líneas amarillas de los "*Ratnest*" que unen los *pads* al mismo nudo de tensión pero que no han sido conectados entre sí todavía por una ruta de cobre.

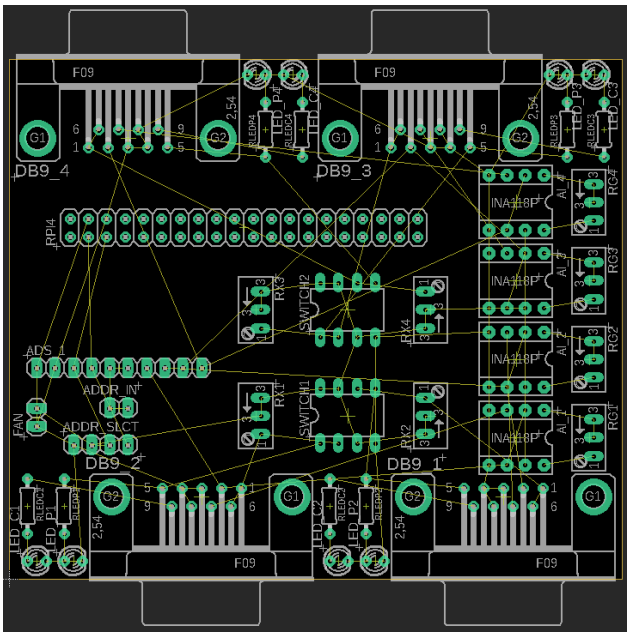


Figura 62: Distribución de componentes

Una vez dispuestos los componentes, se optó por utilizar la función de rutado automático de Eagle para simplificar la tarea. Dado que la producción de la placa la iba a llevar a cabo una empresa especializada, las rutas complejas o a dos caras no debieran suponer un problema. Si bien se puede anotar que se trató de realizar a mano en una sola cara, pero no se logró completar todas las conexiones de los componentes en ninguno de los intentos, por lo que se optó por facilitar el trabajo usando este autorrutado

El autorruter es una opción de Eagle que analiza varias opciones para unir con pistas los *pads* pertenecientes al mismo nudo de tensiones, según las conexiones definidas en el esquema. Se encuentra en la barra izquierda vertical de herramientas, como se puede ver en la figura 63. Seleccionado el "*Autorouter*", simplemente se escogen las opciones de esfuerzo alto, como se ve en la figura 64, y se pulsa "*Continue*". Le tomará al programa un tiempo probar diversas configuraciones hasta ofrecer una que considere sea la más óptima. Una vez concluido, pulsando "*End Job*" se aplicará esta configuración óptima de forma definitiva sobre la PDB. Anotar que en esta versión no se ajustaron ninguna de las características de las rutas de cobre antes de lanzar el autorrutado.

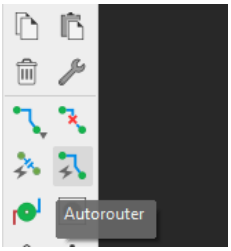


Figura 63: Botón autorrutado

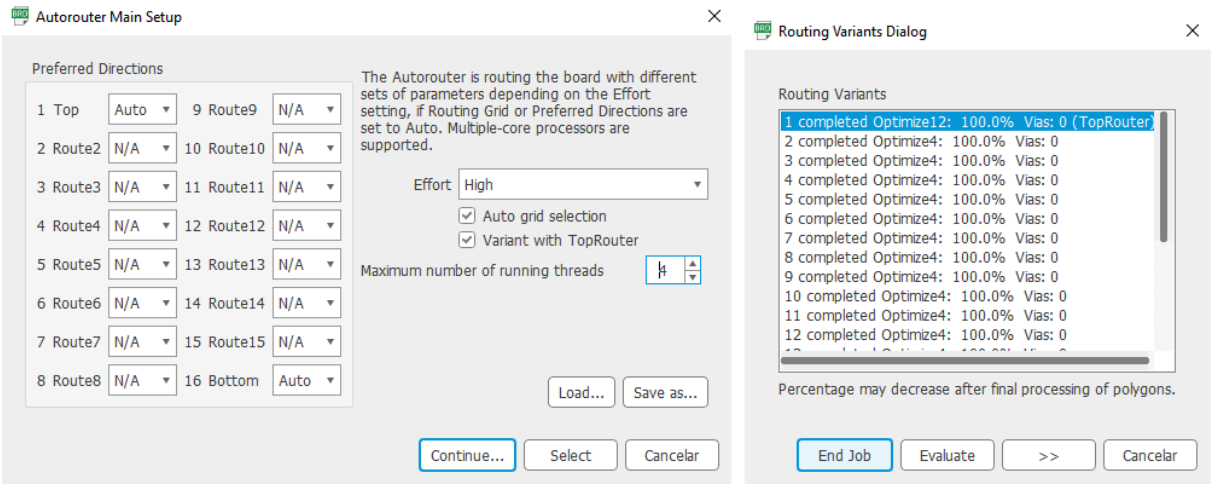


Figura 64: Proceso autorrutado

Una vez finalizado el autorrutado, se obtuvo el rutado que se muestra a continuación en la figura 65 inferior.

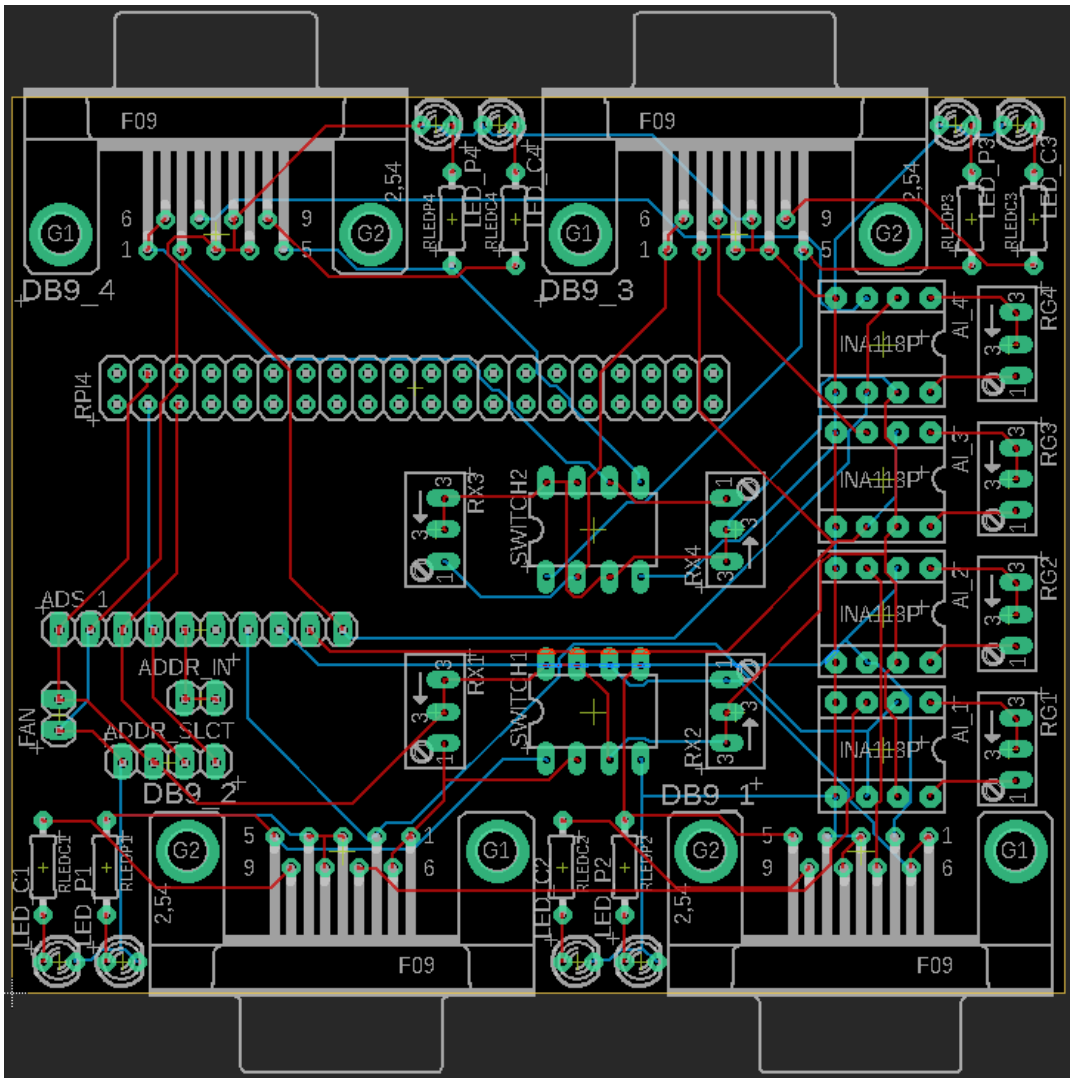


Figura 65: Rutado primera versión

4.2.4.1. Fabricación PCB

Para fabricar el prototipo de la PCB, hay que obtener lo primero los archivos de fabricación, que describen los lugares y anchos de taladro y los *prints* de *pads* y rutas de ambas caras de la PCB. Es sencillo obtenerlos en Eagle, simplemente se pulsa el botón "*Generate CAM Data*" de la barra de herramientas, localizado según se muestra en la figura 66. Esto generará automáticamente una carpeta comprimida con todos los archivos, que se muestran en la figura 67, necesarios para fabricar la PCB (cotas, taladros, capas de cobre y serigrafías).

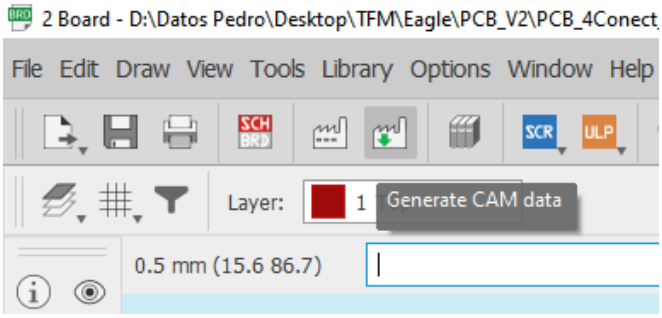


Figura 66: Botón "Generate CAM data"

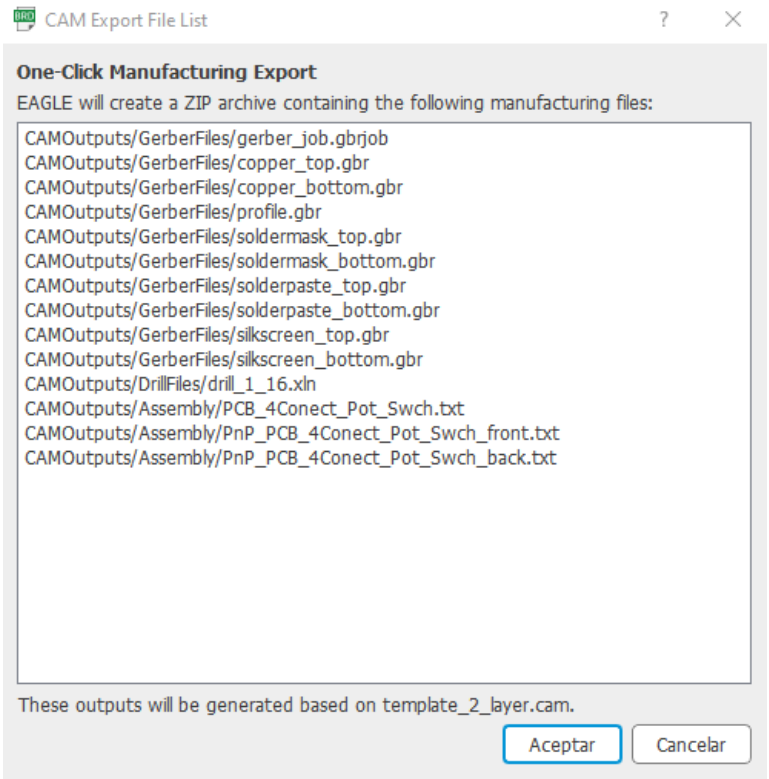


Figura 67: Archivos generados en la carpeta

Como no se dispone de la tecnología necesaria para fabricarla en el taller, se optó por buscar empresas que ofreciera este servicio en Internet. Se optó por JLCPCB, una empresa china especializada en el prototipado de PCB y que se conoce por ser utilizada tanto por aficionados como por empresas que buscan prototipos rápidos [38].

Para pedir la PCB desde la página de JLCPCB, hay que acceder a la sección de prototipado simple, mostrada en la figura 68, y subir la carpeta de fabricación que se ha exportado

desde Eagle. Pulsando el botón en azul "*Add gerber file*", aparecerá una ventana del explorador de archivos de Windows, similar a lo que se ve en la figura 69, dese la que seleccionar la carpeta comprima a subir a la página.

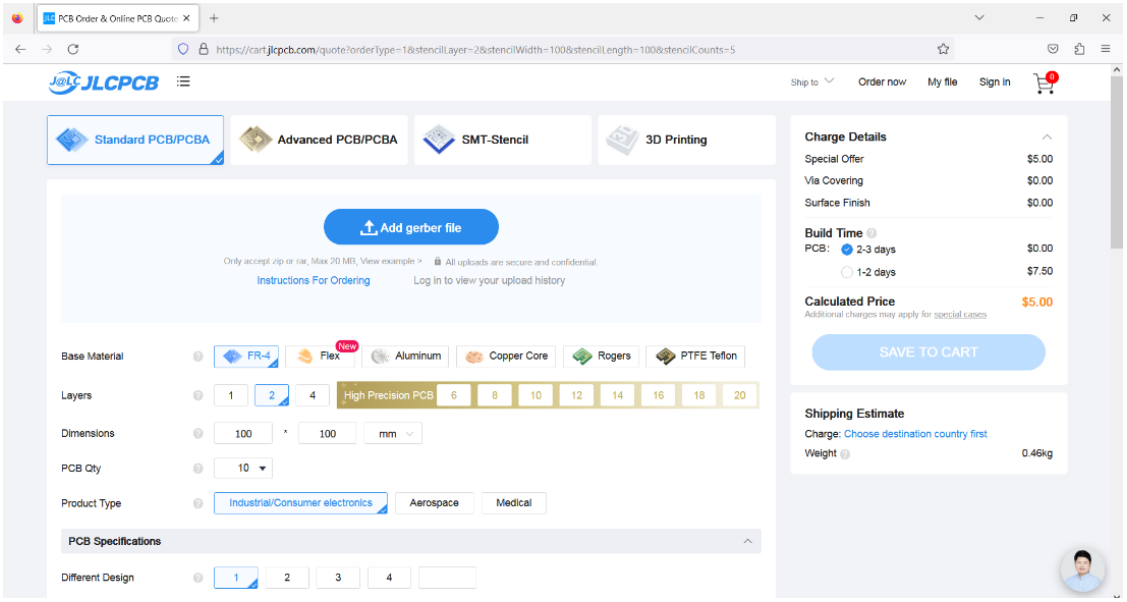


Figura 68: Página principal JLCPCB

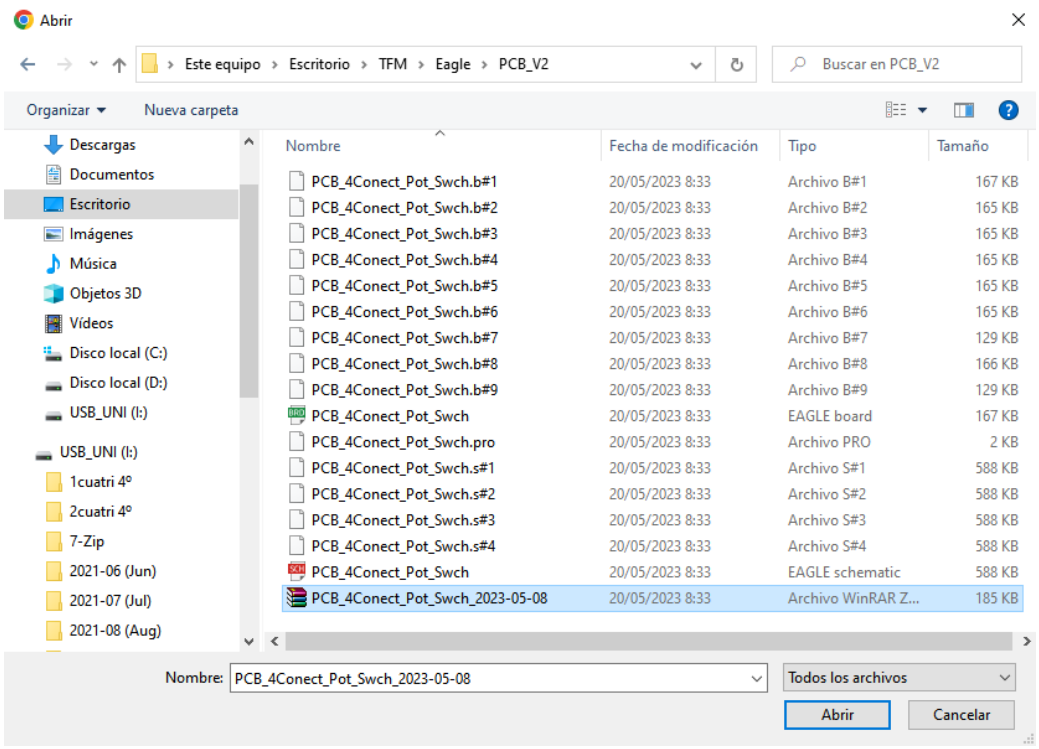


Figura 69: Selección carpeta comprimida

Una vez subida, se generará un modelo 3D que representa la PCB terminada. Se recomienda aprovechar este momento para comprobar que se corresponda correctamente con nuestro diseño en Eagle. Aparecerán una serie de opciones seleccionables debajo, como se muestra en la figura 70. Entre ellas, por ejemplo, se pueden ajustar características de la placa como el color, el ancho o el material base de la

placa. Solamente se cambiará el color y el número de placas. Se piden 10 copias de PCB para asegurar que haya margen si se cometen errores de soldadura o de pruebas.

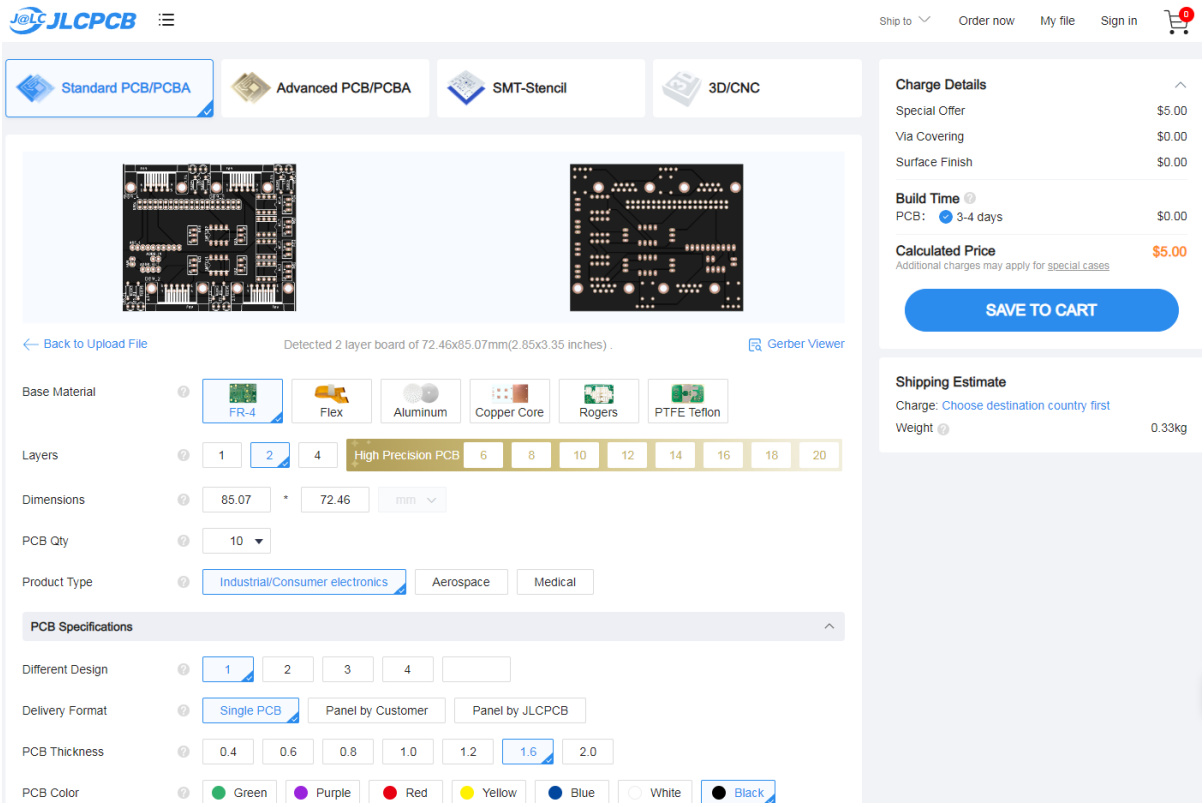


Figura 70: Selección opciones JLCPCB

Una vez seleccionadas las opciones, se añade al carrito la PCB y se siguen los pasos que irán apareciendo para el pago y selección de dirección de envío. Anotar que es posible pedir varios diseños de PCB distintas para que sean enviadas en el mismo pedido, añadiendo los distintos modelos como otro elemento del carrito.

El único problema que presenta esta solución es que el precio de los costes de envío supera por mucho al de las propias PCBs, y que el tiempo de envío de las mimas es bastante elevado. Pero se considera con suficiente previsión temporal resulta una muy buena opción para prototipado. Mencionar que se ofrecen también otras opciones en la página como PCB multicapa o PCB de base flexible, opciones que no han sido exploradas por no ser necesarias para el presente trabajo.

4.2.4.2. Primera versión

Sirva este apartado para reseñar un problema que obligó a retrasar el desarrollo del prototipo varias semanas. Y es que cuando se fue a probar la placa (con los componentes ya soldado), al conectarse a los pines 2x20, forzaba a la Raspberry a apagarse. Esto podría suceder porque existiera un cortocircuito en la PCB o bien porque la Raspberry no la estaba proporcionando suficiente potencia. Lo primero era la circunstancia más probable, pero se decidió conectar la PCB a una fuente de alimentación externa para descartar por seguro la segunda opción.

Al hacerlo, se comprobó que la placa comenzaba a calentarse muy rápidamente, y es que estaba absorbiendo más de 3 A de la fuente. Este nivel de potencia era imposible con los componentes utilizados, así que se pasó a comprobar continuidades con la función propia del multímetro, a fin de encontrar el cortocircuito que debía encontrarse en algún punto de la placa. Se acabó por localizar en los pines de uno de los *switches*. Y es que, en el primer autorrutado realizado, según se explicaba en la primera parte de este apartado "4.2.4. PCB", el programa había puenteado cuatro de los pines del componente *switch* y dado la solución como válida. Estas placas fueron pedidas antes de haberse percato de la existencia este problema. El cortocircuito en cuestión se muestra ampliado sobre la zona de trabajo de PCB en Eagle en la figura 71.

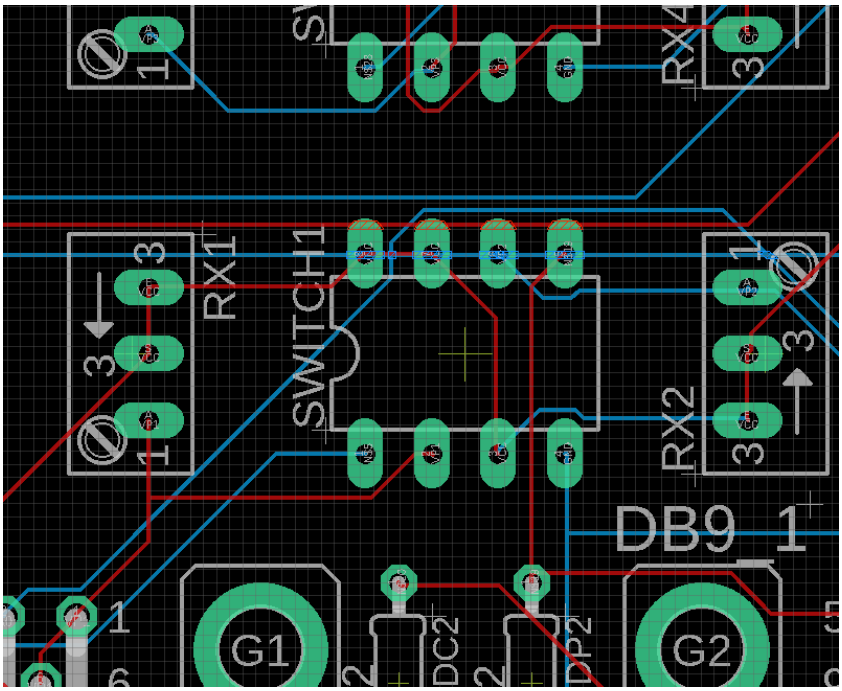


Figura 71: Detalle cortocircuito

Señalar además que, una vez se colocó la PCB sobre la Raspberry, se comprobó que había habido un fallo a la hora de tomar medidas y que la PCB se encontraba ligeramente desviada (con respecto a la que había sido intención de diseño) hacia uno de los laterales de la Raspberry. Este problema se solucionó con la segunda versión de diseño de la PCB, pero es la causante de que la versión 2 del prototipo de la caja, que se explica en el apartado "5.3.2. Prototipo 2" tenga una de las paredes desviadas para facilitar el acceso a los puertos.

4.2.5. Inventario de componentes

Resumiendo en un inventario general los componentes electrónicos necesarios para producir una unidad completa funcional de la PCB diseñada (también conocido como BOM o *Bill Of Materials*) y presentando dicho inventario en un formato de tabla (tabla 5 inferior), en el que se han tomado los costes de componentes según lo ofertado por Farnell, que es proveedor principal del laboratorio de estructuras de la EII. Se incluyen también las referencias en Farnell de dichos componentes, para facilitar la posible búsqueda, consulta y adquisición de los mismos [39].

Referencia PCB	Componente	Modelo	Huella	Fabricante	Refer. Farnell	Nº ud	Coste/ud (€)	Cte/10 ud (€)
AI_1, AI_2, AI_3, AI_4	Amplificador de instrumentación	AD622ANZ	DIP8	Analog Devices	4019190	4	10,73	9,69
SWITCH1, SWITCH2	Interruptor analógico	MAX325 CPA+	DIP8	Analog Devices	2518644	2	5,14	4,64
ADS_1	ADC 16 bits, 4 canales	ADS1115	17,27x 27,94mm	Adafruit	No disp.	1	14,95	13,46
RG1, RG2, RG3, RG4	Potenciómetro multivuelta 200Ω	T93YA201 KT20	RJ9W	Vishay	1141398	4	2,09	1,83
RX1, RX2, RX3, RX4	Potenciómetro multivuelta 500Ω	T93YB501 KT20	RJ9W	Vishay	1141414	4	1,88	1,65
RLEDP1, RLEDP2, RLEDP3, RLEDP4, RLEDC1, RLEDC2, RLEDC3, RLEDC4	Resistencia 330Ω	MRS2500 0C3300FC T00	R-EU_ 0204/7	Vishay	9467327	8	0,227	0,227
LEDP1, LEDP2, LEDP3, LEDP4	LED rojo 3mm	TLHR4405	LED3MM	Vishay	1045457	4	0,338	0,241
LEDC1, LEDC2, LEDC3, LEDC4	LED verde 3mm	TLHG4400	LED3MM	Vishay	1652496	4	0,536	0,378
DB9_1, DB9_2, DB9_3, DB9_4	Conector DB9 en ángulo recto	5747 844-6	F09HP	TE Connectivity	1338746	4	2,97	2,61
RPI4	Zócalo pines hembra 2x20	M20-6102045	PINHD-2X20	Harwin	1569230	1	5,72	5,20
SWITCH1, SWITCH2, AI_1, AI_2, AI_3, AI_4	Zócalo componentes DIP8	2227MC-08-03-18-F1	DIP8	Multicomp Pro	1103844	6	0,258	0,182
ADDR_IN, ADDR_SLCT, FAN	Pines macho conexión	826629-8	PINHD-1X2, PINHD-1X4	TE Connectivity	3418364	1	1,71	1,50
						Tot.	110,20	97,94
						+IVA	133,34	118,51

Tabla 5: Listado de componentes

A los costes anteriores de suma de adquisición de los componentes electrónicos, de un total de unos 120 €, se le tendrían que añadir a parte los costes de adquisición de una Raspberry, que es el componente realmente más caro del proyecto, el ventilador de 5 V para refrigeración y el coste de la placa base de la PCB en sí. Se podrían considerar

también otros costes, como son una estimación teórica de costes de impresión y por mano de obra, tanto de soldadura y como de montaje; estimado por horas. Se considerará posteriormente en un apartado final de “7.3.2. Aspectos económicos”

4.2.4.2. Segunda versión

Habiendo sido identificado el problema de cortocircuito anteriormente descrito, se vuelve a reincidir sobre el diseño para subsanarlo. Se retoma por tanto la PCB trazada, con el área ajustada y se eliminan las pistas de cobre trazadas y para repetir el autorrutado. En este nuevo caso, se incrementará el ancho de las rutas a 12 mil. De la misma forma, se incrementarán también tanto la distancia entre tanto entre rutas y *pads* como entre ruta y ruta a 12 mil. La razón es evitar interferencias entre las rutas y reforzar su estructura, que tenía un aspecto frágil a la vista en las PCB recibidas anteriormente. Para hacer este cambio, se accede a las reglas de diseño (*Design Rules*) en el desplegable de edición (*Edit*) de Eagle, según se muestra en la figura 72, y una vez en él se editan los valores según lo establecido, como se muestra en la figura 73.

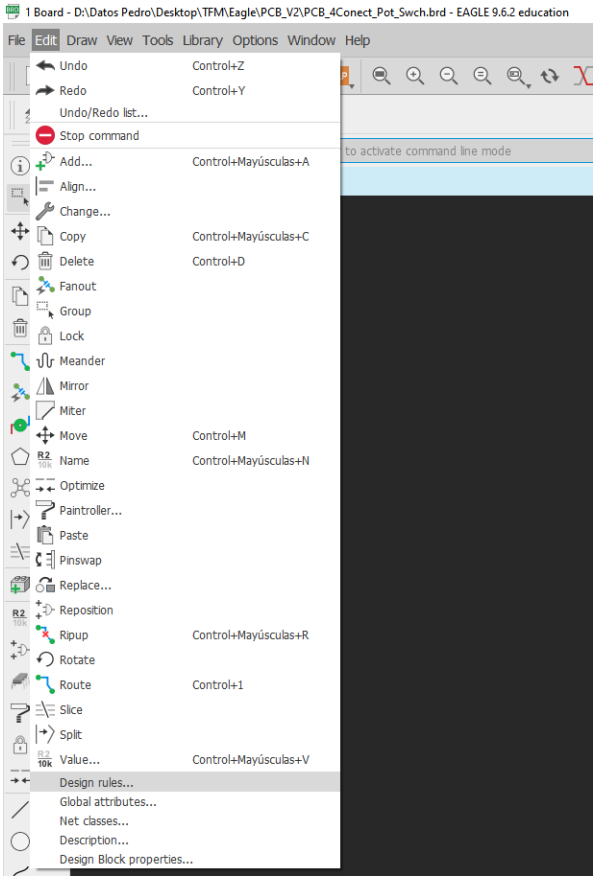


Figura 72: Design rules

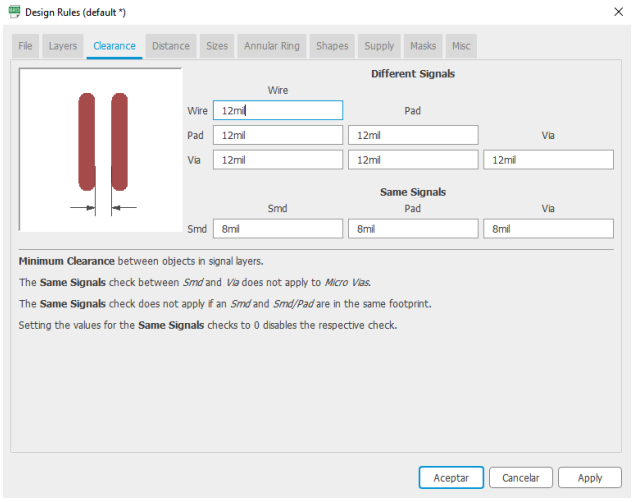


Figura 73: Clearance

Se somete al resultado del autorrutado proporcionado por el programa a un escrutinio atento de las rutas establecidas para evitar que se repita el mismo error. Se obtiene el resultado mostrado en la figura 74 y se realiza un nuevo pedido de PCBs siguiendo los mismos pasas que se explicaron en el apartado “4.2.4.1. Fabricación PCB”.

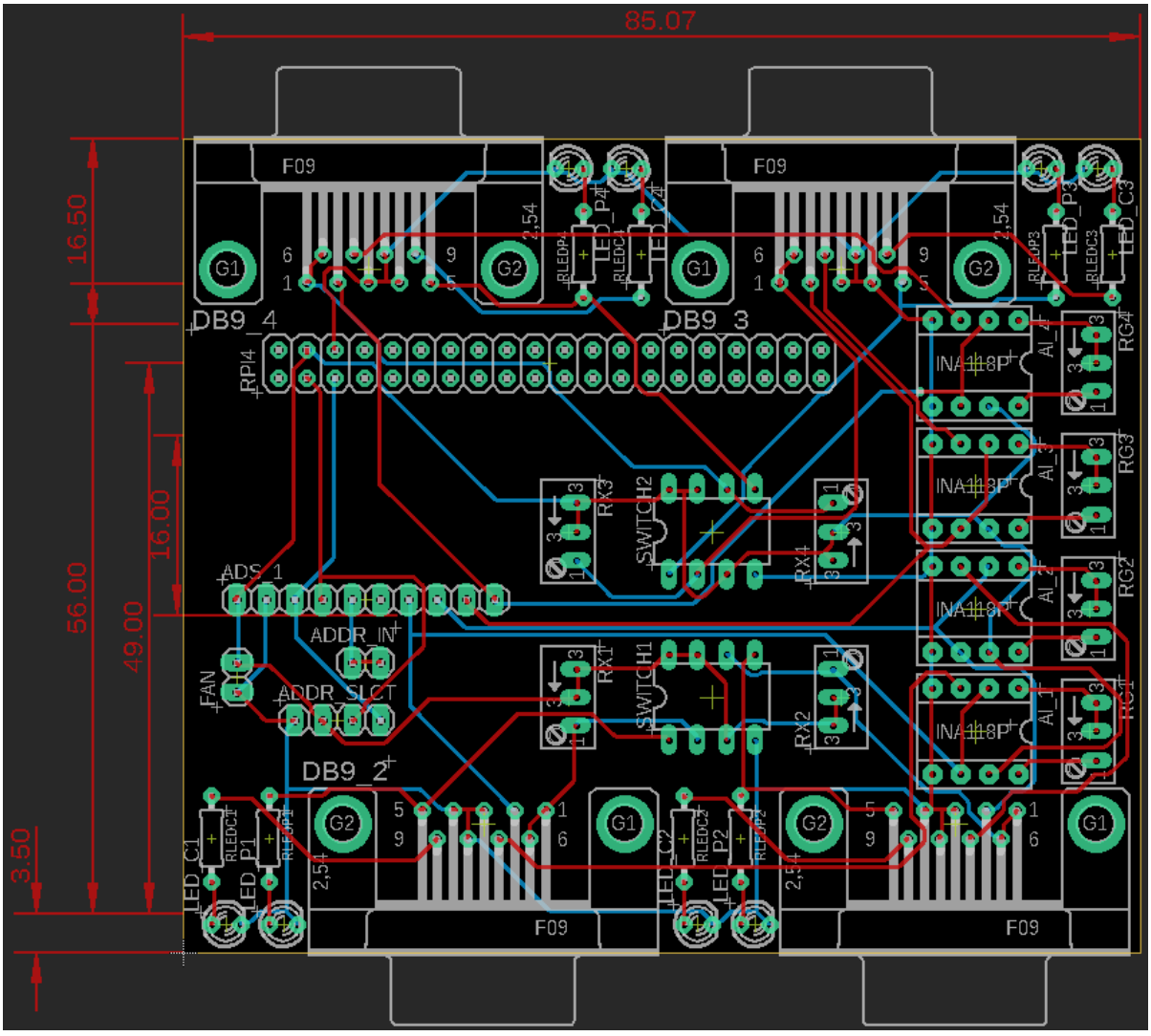


Figura 74: Diseño segunda versión

Para prevenir que se debiera de repetir el proceso de soldadura de todos los componentes y para después descubrir un nuevo problema con las rutas de cobre, como sucedió con la versión anterior, antes de soldar ningún componente se comete a la PCB a una prueba de continuidad. Esto es, haciendo uso del multímetro se comprueba que cada uno de los *pads* para cada componente de la PCB esté correctamente conectados con los pertenecientes a su mismo nudo, según se definió en el esquema eléctrico.

Se encuentra que, de hecho, existe un cortocircuito en alguna parte entre los canales de entrada del ADS1115. Se acota la zona hasta llegar a la tira de pines que ejerce de conector del ADS1115 y se descubre que la causa del cortocircuito es una pista de cobre que no debiera existir. Buscando en el modelo de la PCB en Eagle, se deduce que el programa ha tomado una de las líneas de cota que se ven en la figura 74 como pista de cobre. En concreto, la línea que nos servía para ajustar la distancia entre el ADS1115 y los pines para la Raspberry, según se muestra ampliado en la figura 75.

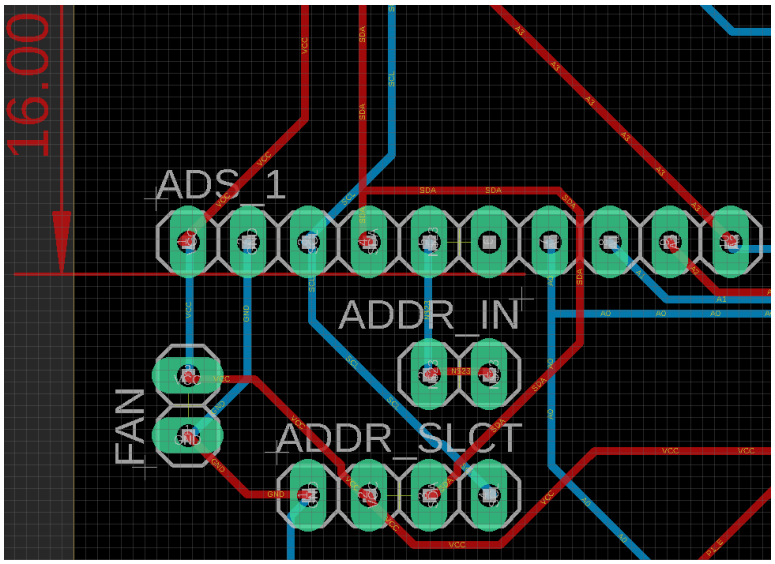


Figura 75: Cota convertida en ruta de cobre

Afortunadamente, la pista es muy delgada y resulta fácil de destruir con un cúter, como se ve en la figura 75, eliminando primero la capa aislante protectora y después, aplicando más fuerza, la pista cobre en sí. Se comprueba una vez eliminada que ya no haya continuidad entre los *pads* del ADS1115.

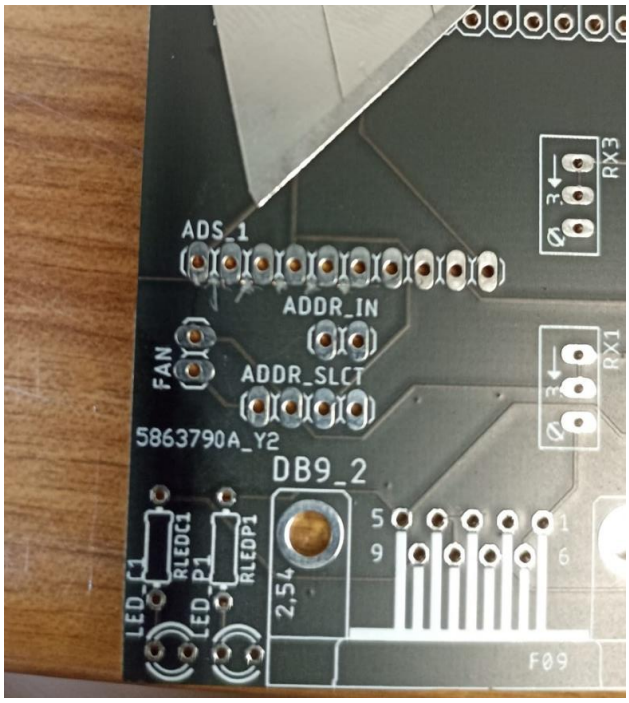
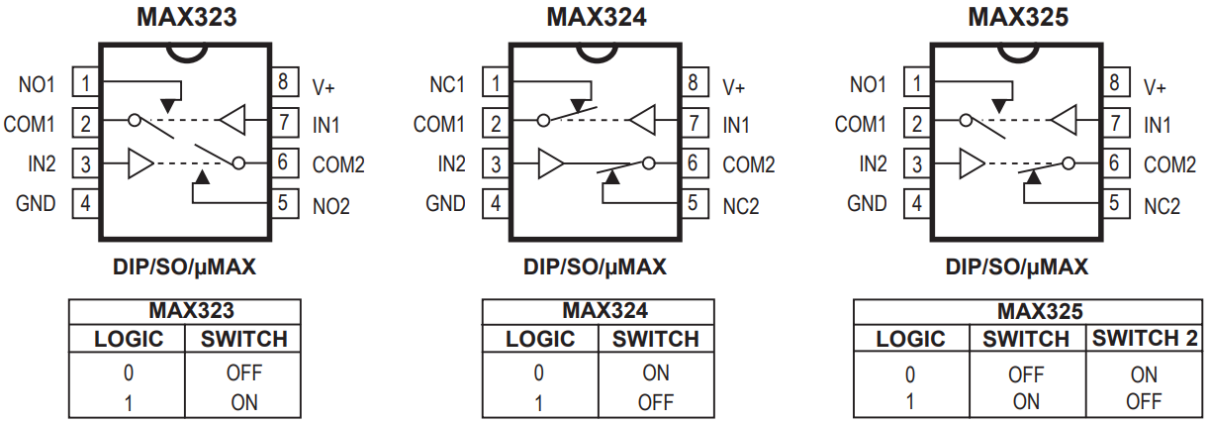


Figura 76: Ruta destruida

Hecho este ajuste, se sueldan todos los componentes en su sitio para poder realizar las pruebas que permitan corroborar el correcto funcionamiento del diseño. En estas pruebas, lo primero que llama la atención es que, en cuanto se alimenta la placa, hay dos LEDs encendidos a pesar de que no hay ningún dispositivo conectado. La causa se encuentra en el interruptor utilizado, y es que el MAX325 tiene dos interruptores internos distintos, uno normalmente abierto y otro normalmente cerrado. Para nuestro propósito deberíamos de haber utilizado el MAX323. Se aportan la comparación entre interruptores y tablas de la verdad de la familia MAX en la figura 77 Para las siguientes pruebas nos servirá, pero en el futuro deberá ser cambiado por otro más adecuado, como veremos en el apartado siguiente “4.2.5.2. Versión avanzada – justificación” [35].



MAX323	
LOGIC	SWITCH
0	OFF
1	ON

MAX324	
LOGIC	SWITCH
0	ON
1	OFF

MAX325		
LOGIC	SWITCH	SWITCH 2
0	OFF	ON
1	ON	OFF

Figura 77: Interruptores analógicos serie MAX32X [35]

Se conectan una célula de carga y un potenciómetro, ambos ya preparados con su DB9 y se lanza el programa de muestreo. Se comprueba que se reciben correctamente los datos

del potenciómetro, pero la célula de carga mantiene un valor estático, independientemente de la carga aplicada sobre ella.

Inicialmente, se presupone que es un problema de ajuste de los valores de Rx y RG por diferencias entre el AD622 y el INA118 (desarrollado en el apartado "3.5.3.3 Conclusiones y pruebas con el AI"), ya que se dejaron inicialmente los valores de resistencia comprobados para este segundo. Sin embargo, al usar el voltímetro para medir las tensiones de AI, lo que se encuentra es que, sin peso de carga en la célula, las tensiones en la entrada diferencial son diferentes entre sí, cosa que no debería suceder en ningún caso. Esto implica que el puente está desequilibrado y, por tanto, no funciona.

Buscando el motivo, se coloca el AD622 en el circuito de prueba presentado en el apartado "3.5.1. Circuito de pruebas", pero aquí no aparece ningún problema de desequilibrio en el puente. Así que el problema ha de encontrarse en el diseño de la PCB. Repasando el esquema de conexiones, se descubre que lo que está causando el desequilibrio en el puente es que la salida del AI se encuentra conectada directamente con la entrada diferencial positiva del mismo. Y es que esta entrada diferencial lleva al pin 2 del DB9, que se utiliza también para la salida única del potenciómetro lineal.

A la hora de diseñar el circuito, se tuvo en cuenta el cambio de alimentación entre célula y potenciómetro, pero no se observó que se había cometido este error al querer utilizar el pin 2 como salida de datos de ambos dispositivos, de forma equivalente al uso que se hacía en el Sirius y que se veía en "4.2.1.2. Compatibilidad con Sirious". Pero al tener dos señales diferentes de salida, serían necesarios también dos rutas distintas para estas dos señales, idea que se desarrolla y explica mejor más adelante en el apartado "4.2.5.2. Versión avanzada – justificación". Tal y como se encuentra el diseño actualmente, como solo se cambia la alimentación, se fuerza un puente que no debería haber entre entrada y salida del AI. Puede verse marcado este puente en azul en la siguiente figura 78.

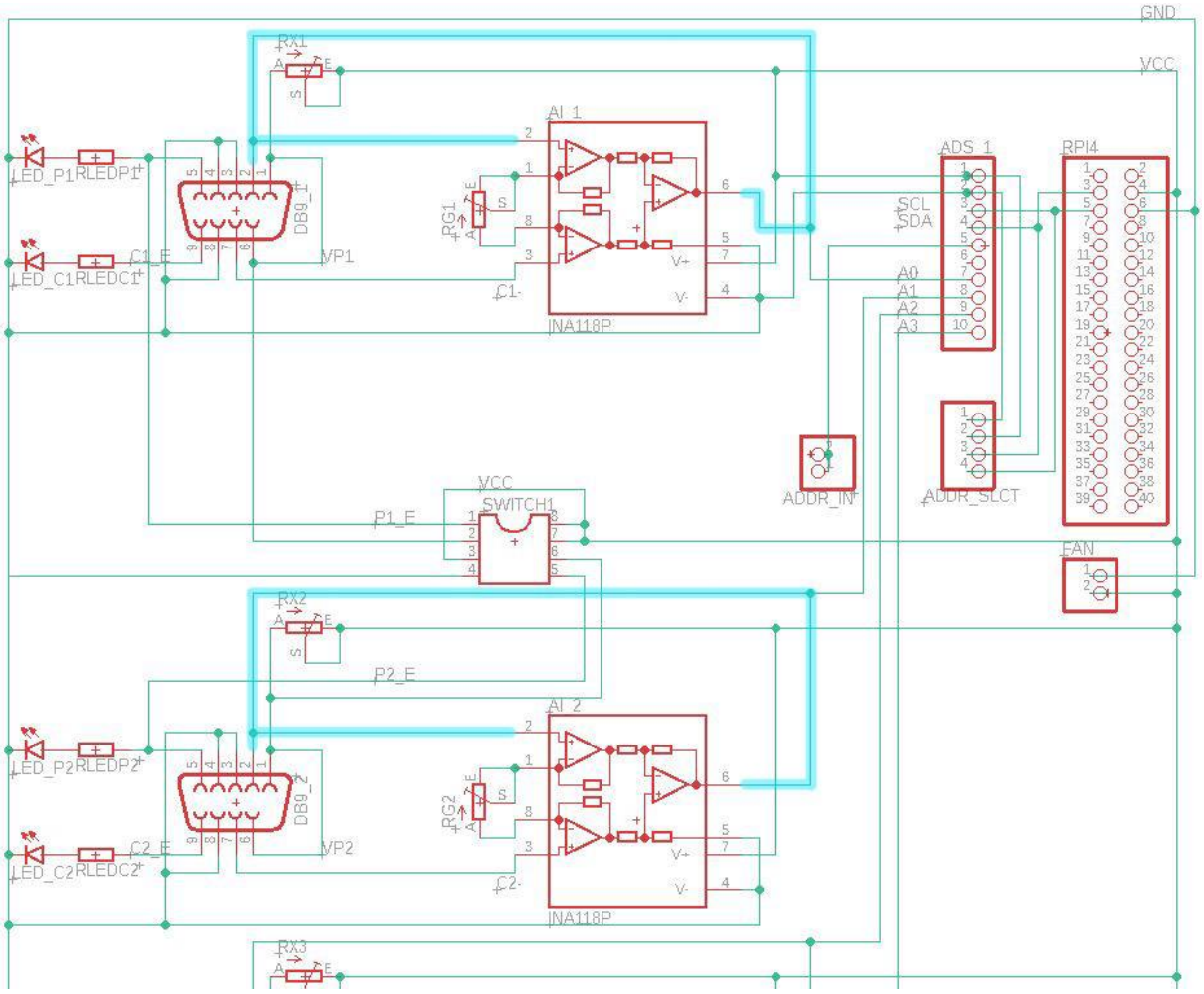


Figura 78: Conexión errónea

Se opta por destruir esta ruta y ajustar el circuito para que sea posible seguir utilizándolo, aunque sea a una capacidad reducida. Es decir, aceptando dos potenciómetros en los conectores de un lado y dos células de carga en el opuesto, sin que sea intercambiable qué sensor se utiliza en cada conector. Además de estas rutas, se tienen que seccionar otras y empalmar algunas conexiones con cables externos. Los cortes que se tienen que realizar se representan en la figura 79 inferiores en color morado.

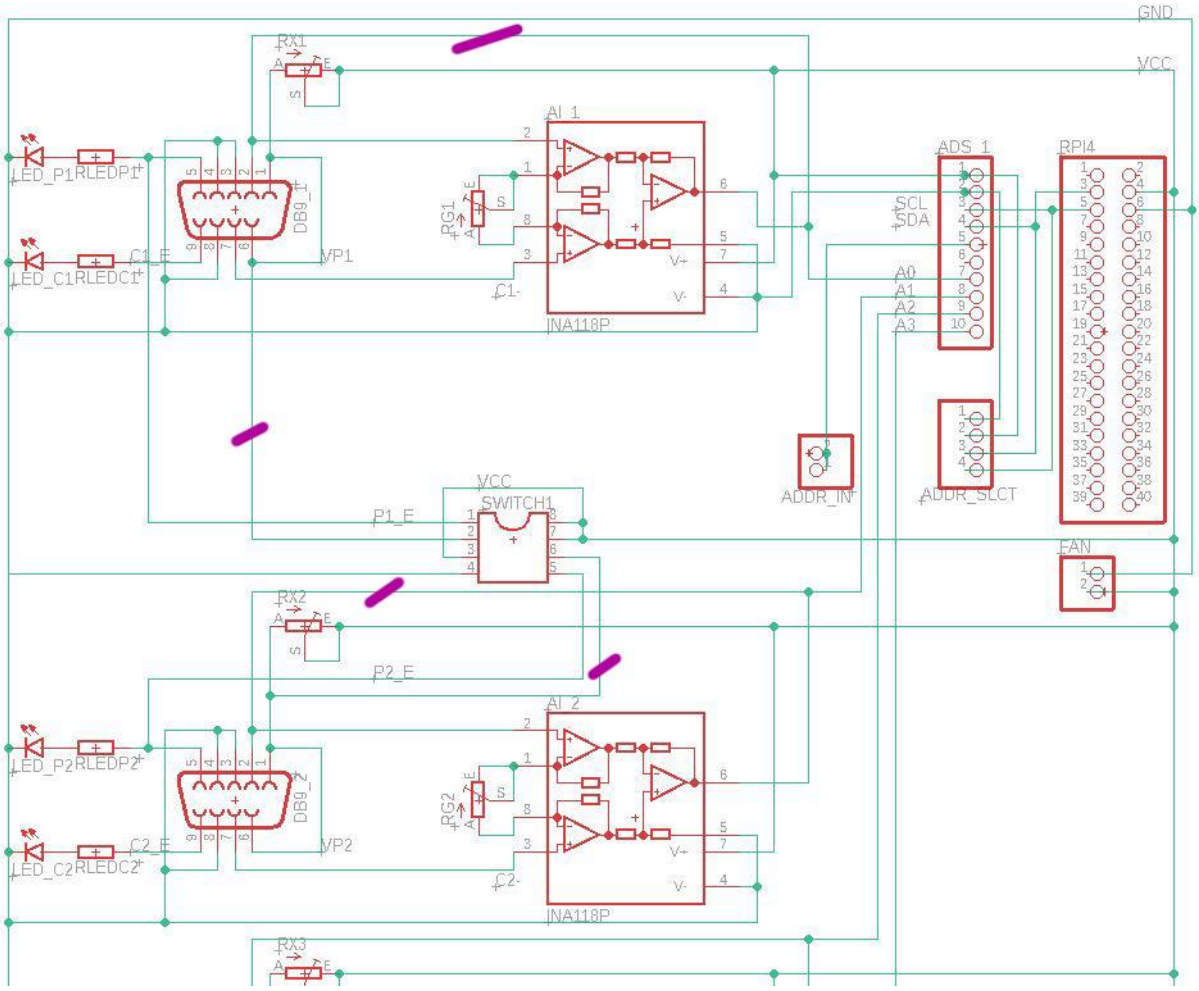


Figura 79: Cortes en las conexiones

Estos cortes de conexiones no pueden trasladarse directamente a las rutas de la PCB, por la forma en que han sido trazadas teniendo en cuenta los nudos de tensión. Hay que identificar las distintas rutas en la PCB que equivalen a los nudos identificados y marcados en la figura 79. Además, se tendrá que volver a conectar la tensión positiva de la célula de carga con la entrada positiva del AI, que se ve afectado por los cortes que se deben efectuar. Estos cortes y empalmes se representan sobre la PCB en la figura 80, representando los cortes en morado y los empalmes en naranja.

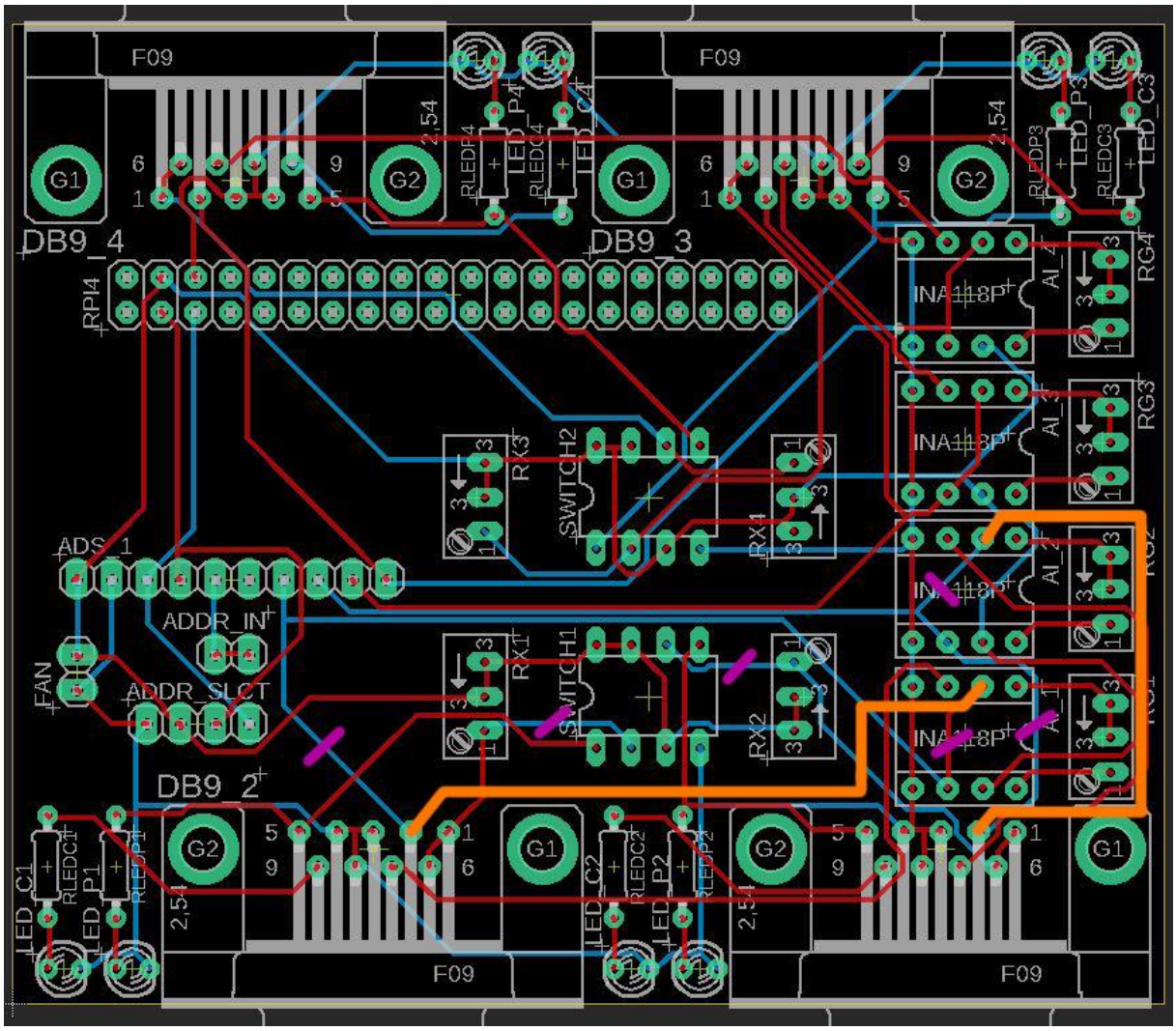


Figura 80: Cortes en las rutas

Se destruyen estas rutas de cobre, de nuevo haciendo uso de un cúter, con el resultado que se muestra en la figura 81. A continuación, se realizan los empalmes con cables rígidos y soldadura, quedando la parte inferior de la placa finalmente como se muestra en la figura 82. Señalar que es importante que los cables añadidos estén lo más pegados posible a la PCB, para no impedir que la placa se pueda conectar sobre la Raspberry a causa de la altura de los puertos USB de esta última.

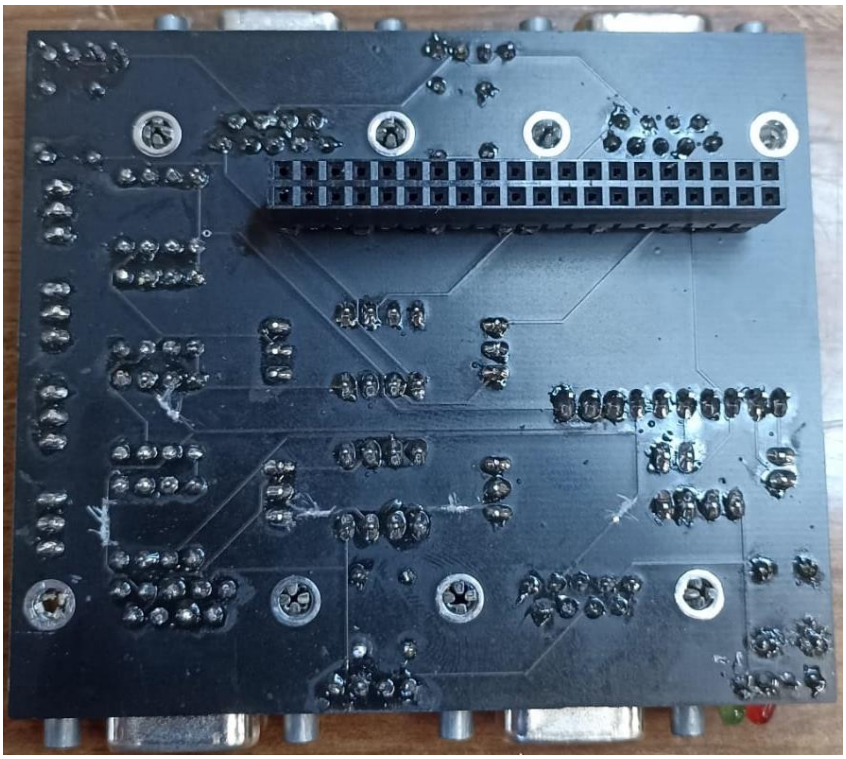


Figura 81: Rutas cortadas

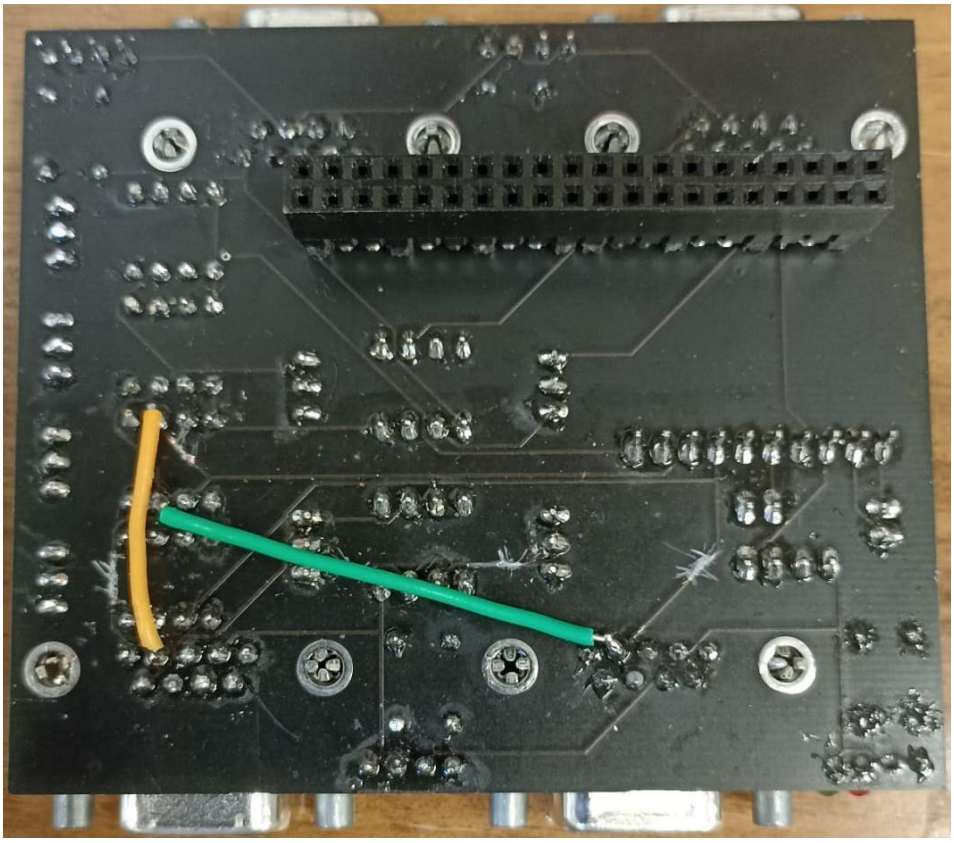


Figura 82: Conexiones por cable

Realizadas las pruebas con esta placa, se encuentra que ya hay una recepción de datos normal y esperada. Estos resultados finales se muestran el apartado “6. Resultados”.

4.2.5. Versiones futuras

A la vista de los resultados de las pruebas y errores explicados en el apartado anterior "4.2.4.2. Segunda versión", se proponen dos nuevas versiones de PCB para el proyecto, a considerar para un posible futuro. Cada una de ellas cubre un aspecto diferente posible respecto a la continuación o puesta en uso del presente trabajo.

4.2.5.1. Versión reducida

La primera versión es la traducción directa de los arreglos que se tuvieron que hacer a la placa en el aparatado anterior, es decir, los cortes y empalmes realizados para adaptar el circuito. El resultado es una conexión directa de dos células de carga a sendos AI y posteriormente al ADS1115, al que se conecta así mismo dos potenciómetros. Tiene, por tanto, capacidad para dos potenciómetros y dos células de carga, en conectores fijos, no adaptables, que operan simultáneamente, pero en lados opuestos de la placa. Su esquema eléctrico correspondiente se muestra en la figura 83.

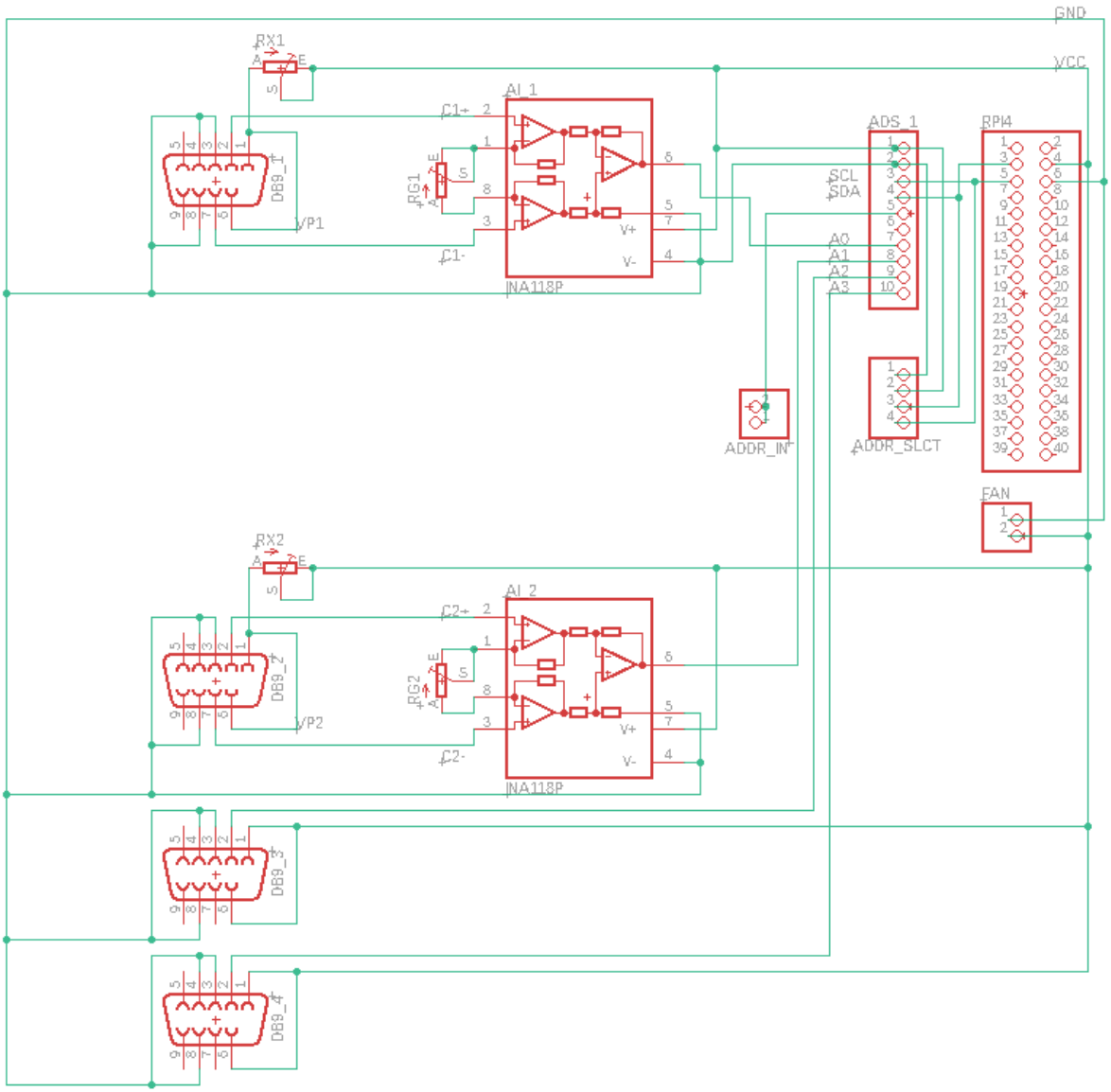


Figura 83: Esquema versión reducida

El esquema que resulta es más sencillo, prescindimos de la midad de los AI que en el caso visto en el apartado “4.2.4.2. Segunda versión”, así como de los interruptores. Como puede verse en la siguiente figura 84, la duperficie de la pCB queda mucho menos densamente poblada, si bien el tamaño se mantiene igual, disctaminado principalemnet por el ancho de los DB9 y para permitir intercambiabilidad entre cualesquiera de las propuesta spresnetadas en este trabajo.

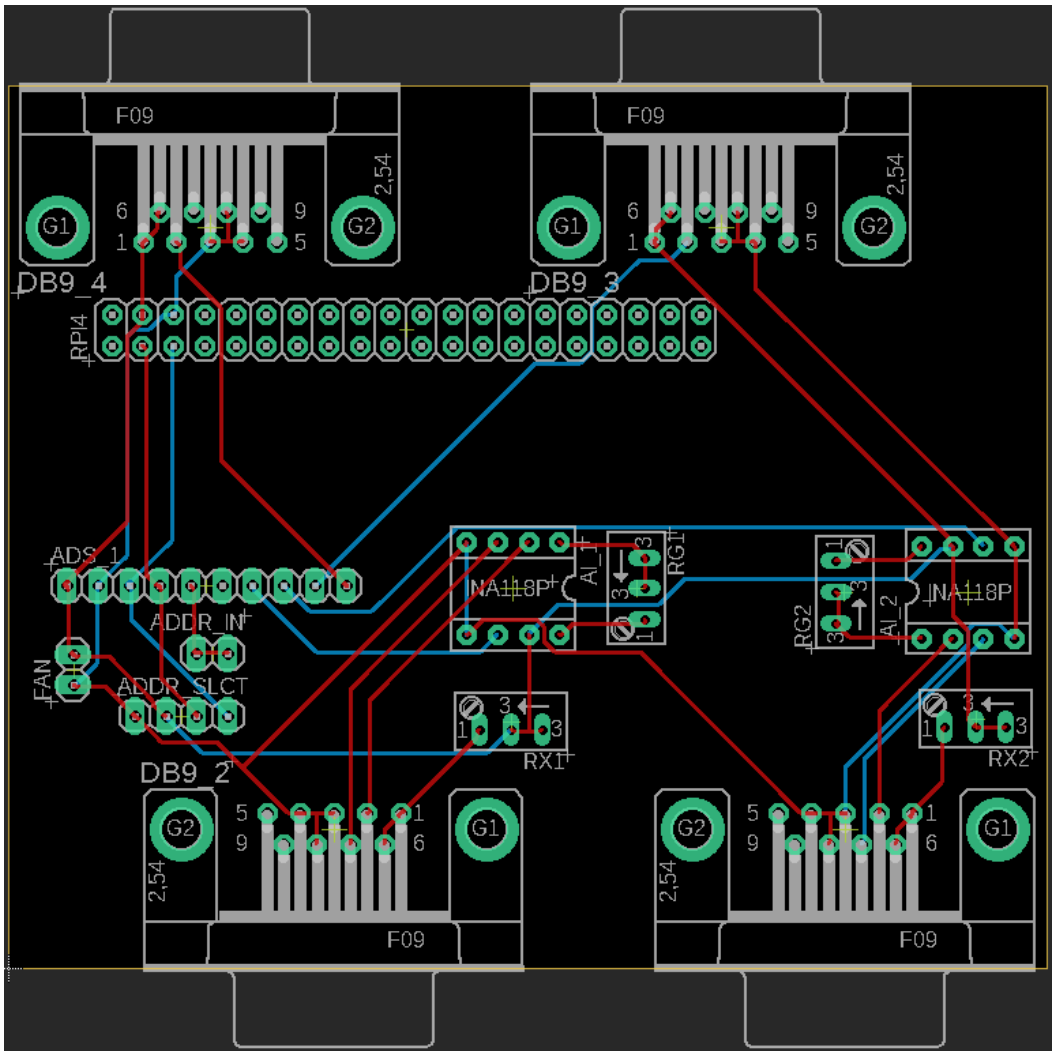


Figura 84: PCB versión reducida

En cuanto al inventario de componentes, queda un resultado algo más económico que en la versión más completa por la falta de estos componentes mencionados. Se presenta el resultado en la tabla 6 siguiente.

Referencia PCB	Componente	Modelo	Huella	Fabricante	Refer. Farnell	Nº ud	Coste/ud (€)	Cte/10 ud (€)
AI_1, AI_2	Amplificador de instrumentación	AD622ANZ	DIP8	Analog Devices	4019190	2	~ 12	~ 11
		INA128P		Texas Instruments	3004493			
ADS_1	ADC 16 bits, 4 canales	ADS1115	17,27x 27,94mm	Adafruit	No disp.	1	14,95	13,46
RG1, RG2	Potenciómetro multivuelta 200Ω	T93YA201 KT20	RJ9W	Vishay	1141398	2	2,09	1,83
RX1, RX2	Potenciómetro multivuelta 500Ω	T93YB501 KT20	RJ9W	Vishay	1141414	2	1,88	1,65
DB9_1, DB9_2, DB9_3, DB9_4	Conector DB9 en ángulo recto	5747 844-6	F09HP	TE Connectivity	1338746	4	2,97	2,61
RPI4	Zócalo pines hembra 2x20	M20-6102045	PINHD-2X20	Harwin	1569230	1	5,72	5,20
AI_1, AI_2	Zócalo componentes DIP8	2227MC-08-03-18-F1	DIP8	Multicomp Pro	1103844	2	0,258	0,182
ADDR_IN, ADDR_SLCT, FAN	Pines macho conexión	826629-8	PINHD-1X2, PINHD-1X4	TE Connectivity	3418364	1	1,71	1,50
						Tot.	66,72	59,92
						+IVA	80,73	71,91

Tabla 6: Coste versión reducida

4.2.5.2. Versión avanzada - justificación

En esta versión se mantiene la idea inicial de permitir la conexión indiferente de potenciómetros y células de carga en cada DB9. Se mantiene también la lógica de uso de interruptores analógicos frente a transistores, teniendo en cuenta ahora la necesidad de utilizar dos canales de cambio por cada DB9, frente al canal único considerado en inicio.

Esta necesidad proviene del error detectado en el prototipo segundo de la PCB, y es que no solamente es necesario cambiar la alimentación según se tenga célula de carga o potenciómetro conectado en el DB9, sino que también hay que asegurar que la señal del pin 2 del DB9, compartida tanto por la señal de datos del potenciómetro como por el "V-" de la célula de carga, no interfiera con el funcionamiento del AI.

Según lo que había establecido en el prototipo desarrollado, según se ha visto en el apartado "4.2.4.2. Segunda versión", solamente se estaba utilizando el interruptor para intercambiar la alimentación de 5 V a paso por Rx. Este cambio de tensión resulta sencillo de plantear, porque al cerrarse el interruptor, el nudo formado conecta directamente a la alimentación de 5 V, ignorando el camino de mayor resistencia, la parte de la malla que contiene Rx. Así, la alimentación es obligada a atravesar Rx para la célula de carga de forma predeterminada, pero fluye libre sin resistencia si se conecta un potenciómetro, abriendo el circuito. Representado esquemáticamente en la figura 85:

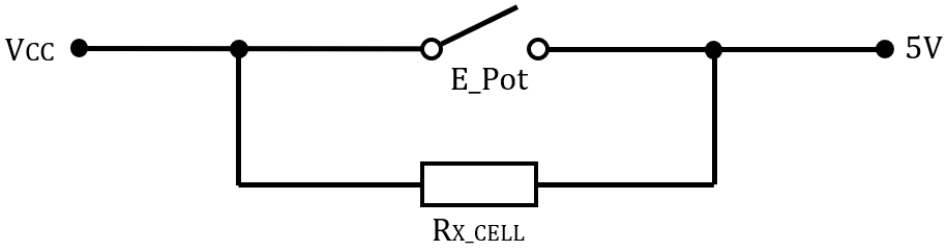


Figura 85: Alimentación con Enable

Pero se había obviado que también hay que intercambiar la conexión entre sensor y canal del ADS1115, o se estará forzando inintencionadamente el cortocircuito entre V+ y tensión de salida del AI, que causa que las entradas estén desequilibradas y, por tanto, no funcione el puente.

Para hacer este intercambio, se volverá a recurrir a la señal de *Enable.* Señal que finalmente se ha agregado solamente al DB9 del potenciómetro, considerando la célula de carga como modo "por defecto". Es decir, para simplificar el diseño, el circuito estará preparado en su configuración normal para aceptar células de carga, y cambiará de configuración si detecta un potenciómetro a través de esta señal *Enable*. Señal que vendrá dada desde un pin libre, según el esquema que compatibilidad con Sirious que veíamos en el apartado "4.2.1.3. Admisión de puentes y potenciómetros". Esta seña se logra puenteando en el DB9 macho del potenciómetro, al que va directamente el cableado, no al hembra de la PCB. Se logra puenteando directamente la alimentación con el pin libre 5, como se muestra en el esquema de la figura 86.

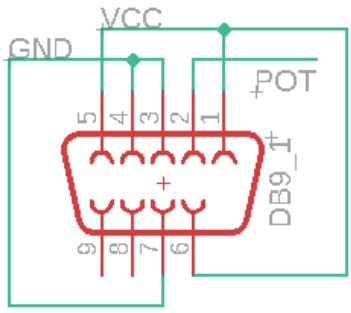


Figura 86: Conexión DB9 macho potenciómetro

Como se mencionaba, al igual que con la alimentación, la conexión predeterminada de entrada de señal es la de la célula de carga, con su V+ (pin 2 del DB9) y V- (pin 7 del DB9) dirigidos a las entradas correspondientes del AI. El cambio en el circuito se producirá a la señal alta de *Enable* del potenciómetro, que cerrará el nuevo interruptor que se deba agregar, comunicando potenciómetro y canal de entrada al ADS1115. De esta forma, el esquema de representación sería el de la figura 87.

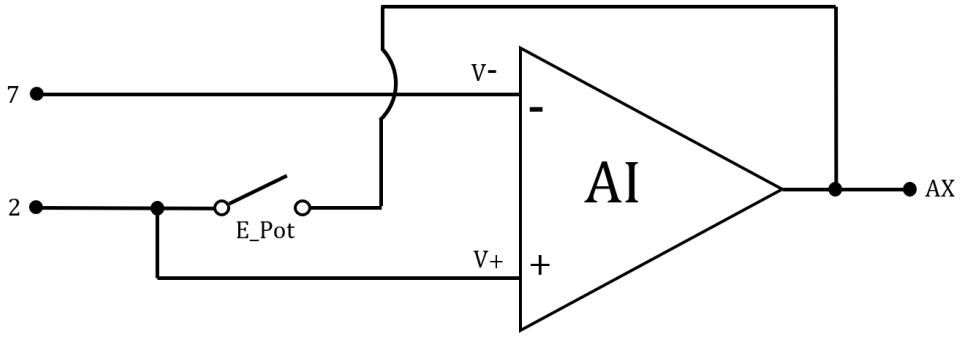


Figura 87: Esquema interruptor con AI

Así, sin señal de *Enable* (E_Pot: *OFF*), el interruptor permanece abierto y la célula de carga conecta directamente con el AI, sin interferencias con la tensión de salida, que se envía directa a uno de los canales del ADS1115. El estado por defecto del circuito representado en la figura 87 sería el que se muestra a continuación en la figura 88.

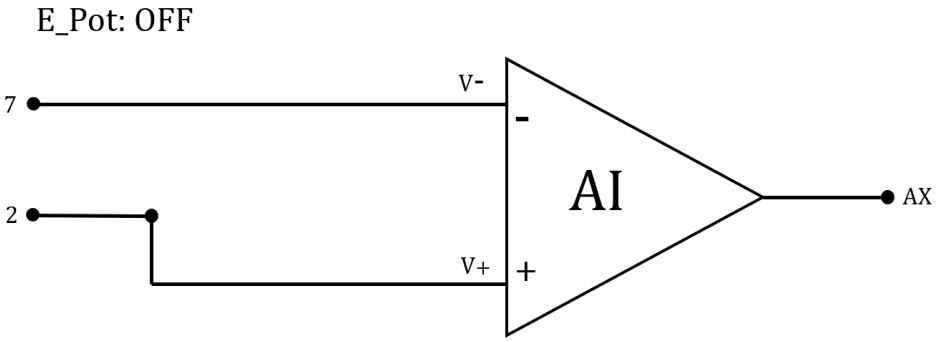


Figura 88: Interruptor AI abierto

Y con la señal de *Enable* activa (E_Pot: *ON*), el interruptor se cierra. La configuración del circuito entonces cambia, enviando directamente la señal de datos del potenciómetro del pin 2 del DB9 a uno de los canales del ADS1115. El AI por su parte queda inactivo, al no tener una de las entradas diferenciales alimentada, resultando indiferente la tensión de la secundaria. El cambio de estado del circuito de la figura 87 resulta, por tanto, en este caso, en el circuito que se muestra en la figura 89

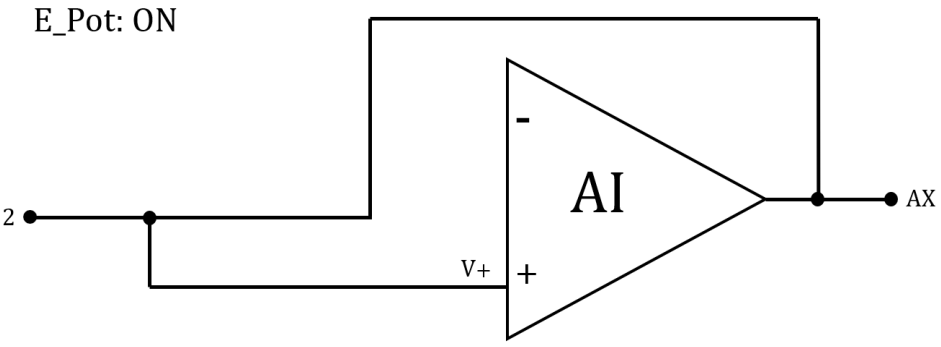


Figura 89: Interruptor AI cerrado

Para encontrar un interruptor que mejor sirva para este propósito mejorado, se buscan una vez más en Farnell candidatos que puedan cubrir las necesidades de entre los disponibles. Se necesita un interruptor analógico de envolvente tipo DIP, de agujero pasante, y con 4 canales. Se prevé la utilización dos de este tipo porque sería lo que mejor se ajuste al espacio que queda disponible en la PCB.

4.2.5.2.1. CD4066B

El primer candidato estudiado ha sido el CD4066B de Texas Instruments. El motivo es que se disponía ya de dos unidades del mismo, ya que se consideró en su momento como sustituto barato (su precio ronda los 0,6 €) al MAX325, si es que el espacio de la PCB y características del integrado lo permitían. Como se ve en su esquema de la figura 90, cumple con las características mínimas que se necesitaban y puede ser alimentado a 5 V según su *datasheet*. El comportamiento lógico, que se muestra en la tabla de la verdad de la figura 91, también es el que se necesita, activándose teóricamente el interruptor al recibir señal de control alta, el *Enable* del potenciómetro [40].

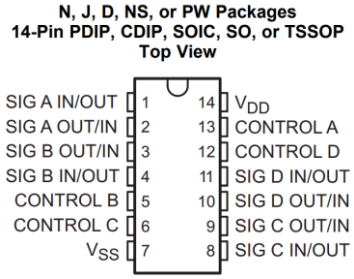


Figura 90: Interruptor CD4066B [40]

Table 1. Function Table

INPUTS		OUTPUT
SIG IN/OUT	CONTROL	SIG OUT/IN
H	H	H
L	H	L
X	L	Hi-Z

Figura 91: Tabla interruptor CDB4066B [40]

Sin embargo, al realizar pruebas con él se encontró que, si bien para los estados de conexión a alto a 5 V (H) y bajo a 0 V o tierra (L) funciona como dicta la tabla lógica, para el pin de control desconectado aparece un inconveniente para el uso que se le pretendía dar. Y es que, si el pin de control se deja al aire, el interruptor permanece abierto. Esto obliga a descartarlo directamente para su uso en el cambio de configuración del circuito, ya que es necesario que el interruptor se active al recibir la señal de 5 V del *Enable*.

Podemos ver en las ilustraciones inferiores este comportamiento, se conectó como prueba un LED a la salida del interruptor y utilizando el cable jumper amarillo para alternar en las conexiones entre 5 V (línea roja), tierra (línea azul) y al aire. El la ilustración 92 se ve que el interruptor se conecta con el cable naranja a 5 V, dando como resultado el estado de ON (LED encendido), y que si como en la figura 93, el cable se conecta a GND (0 V), da estado OFF (LED apagado). Sin embargo, dejando el cable naranja al aire como en la figura 94, sin conectar a nada, prevale el estado de ON (LED encendido), característica en principio contraria a la vista en la *datasheet*.

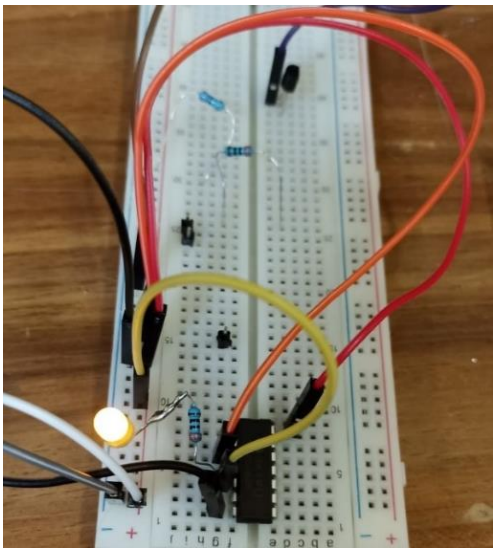


Figura 92: Interruptor estado ON

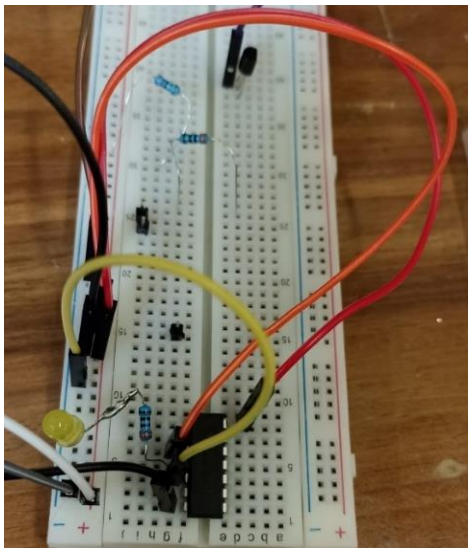


Figura 93: Interruptor estado OFF

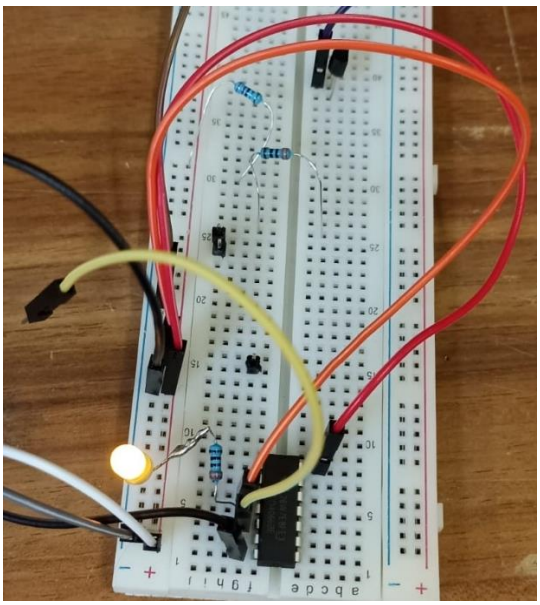


Figura 94: Interruptor estado indeterminado

4.2.5.2.2. DG302

El segundo candidato considerado fue el DG302 de Analog Devices. Tiene un encapsulado DIP14, que ahorra un poco de espacio en la PCB, si bien no es una característica troncal para tomar la decisión. Lo que sí resulta atractivo es la configuración interna del switch, que activa dos canales con una sola señal de entrada. Señal que sería el *Enable* del potenciómetro. Esta configuración se denomina DPST (*Dual-Pole/Single-Throw*), y es el único de este tipo que se encontró disponible en Farnell. Esto, junto a configuración simétrica de pines que se muestra en la figura 96, nos ayuda a simplificar el rutado [41].

Sin embargo, analizando la *datasheet*, se encontró que, en caso de alimentarlo en modo común, la tensión mínima requerida es de 10 V. Esto viene especificado en la tabla

presentada aquí como figura 95. Pudiendo, por diseño solamente proporcionar 5 V, los proporcionados por la Raspberry, se descarta este integrado [41].

Table 1. Typical Single Supply Parameters

PARAMETER		V+ SUPPLY VOLTAGE (V- = 0V)			
		+10V	+15V	+20V	+30V
Switching Time (R_L = 1kΩ)	t_{ON}	190ns	150ns	110ns	70ns
	t_{OFF}	40ns	40ns	40ns	40ns
On-Resistance	V_{SIGNAL} = +1V	71Ω	51Ω	42Ω	31Ω
	V_{SIGNAL} = V+ / 2	77Ω	54Ω	43Ω	30Ω
	V_{SIGNAL} = V+	84Ω	63Ω	54Ω	43Ω
Input Logic Levels		0.8V, 4.0V	0.8V, 4.0V	0.8V, 4.0V	0.8V, 4.5V

Figura 95: Tabla tensiones DG302 [41]

4.2.5.2.3. MAX4614

El tercero y finalmente seleccionado como solución para la aplicación es el MAX4614, también de Analog Devices. La configuración de los pines es algo más compleja respecto a los anteriores, como se ve en la figura 97 y puede comparar con la 96 adyacente. y al ser el tipo de interruptor SPST (*Single-Pole/Single-Throw*), el rutado va a resultar más complejo, además de que fuerza a duplicar la señal de *Enable* para darla como entrada a dos interruptores distintos. Sin embargo, es un circuito, según el fabricante y comprobable en la *datsheet*, especialmente diseñado para aplicaciones de baja tensión, del entorno a los 5 V, con los que trabaja el prototipo [42].

Además, asegura una bajísima resistencia (1 a 10 Ω) a estas tensiones bajas, lo que hace que su efecto sobre el circuito sea despreciable. Vemos en la tabla de la verdad del integrado que funciona como requerimos, activándose con una señal lógica alta. Además, los interruptores tienen un estado NA (Normalmente Abierto), lo que asegura que, si no se aporta la señal de *Enable*, el circuito permanecerá abierto sin necesidad de forzar el cambio de estado del interruptor, es decir, no es necesario en teoría agregar una resistencia de *Pull-Down* [42].

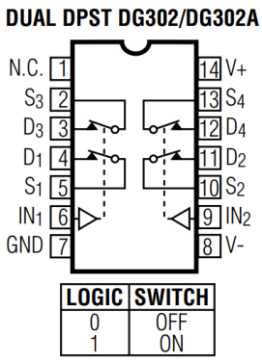


Figura 96: Interruptor DG302 [41]

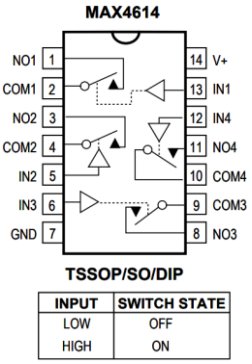


Figura 97: Interruptor MAX4614 [42]

4.2.5.3. Versión avanzada - diseño y BOM

Vista la justificación y selección del nuevo interruptor, se puede pasar a ver el diseño de esta versión, que no difiere mucho del prototipo 2 fabricado y probado en el apartado "4.2.4.2. Segunda versión", a excepción del uso del interruptor diferente y mejor adaptado. Anotar que para el diseño del esquema eléctrico, mostrado en la figura 98, se ha utilizado un componente genérico DIP de 14 pines, conectando los mismos según corresponde a la justificación del apartado anterior "4.2.5.2. Versión avanzada – justificación" y usando como referencia la *datasheet* del MAX4614.

Comentar también que, como se mencionaba en el apartado "4.2.5.2. Versión avanzada – justificación", el *Enable* funcional pasa a ser solamente el del potenciómetro. Esto es, solamente la señal de *Enable* del potenciómetro cambiará la configuración del circuito. Se mantiene la conexión del *Enable* de la célula de carga, pero su propósito es solamente informativo, es decir, solamente sirve para encender el LED correspondiente, no da ninguna otra señal. El *Enable* del potenciómetro sí que, además de activar su LED correspondiente, da la señal a los *switches*, como se puede seguir en el esquema de la figura 98 inferior.

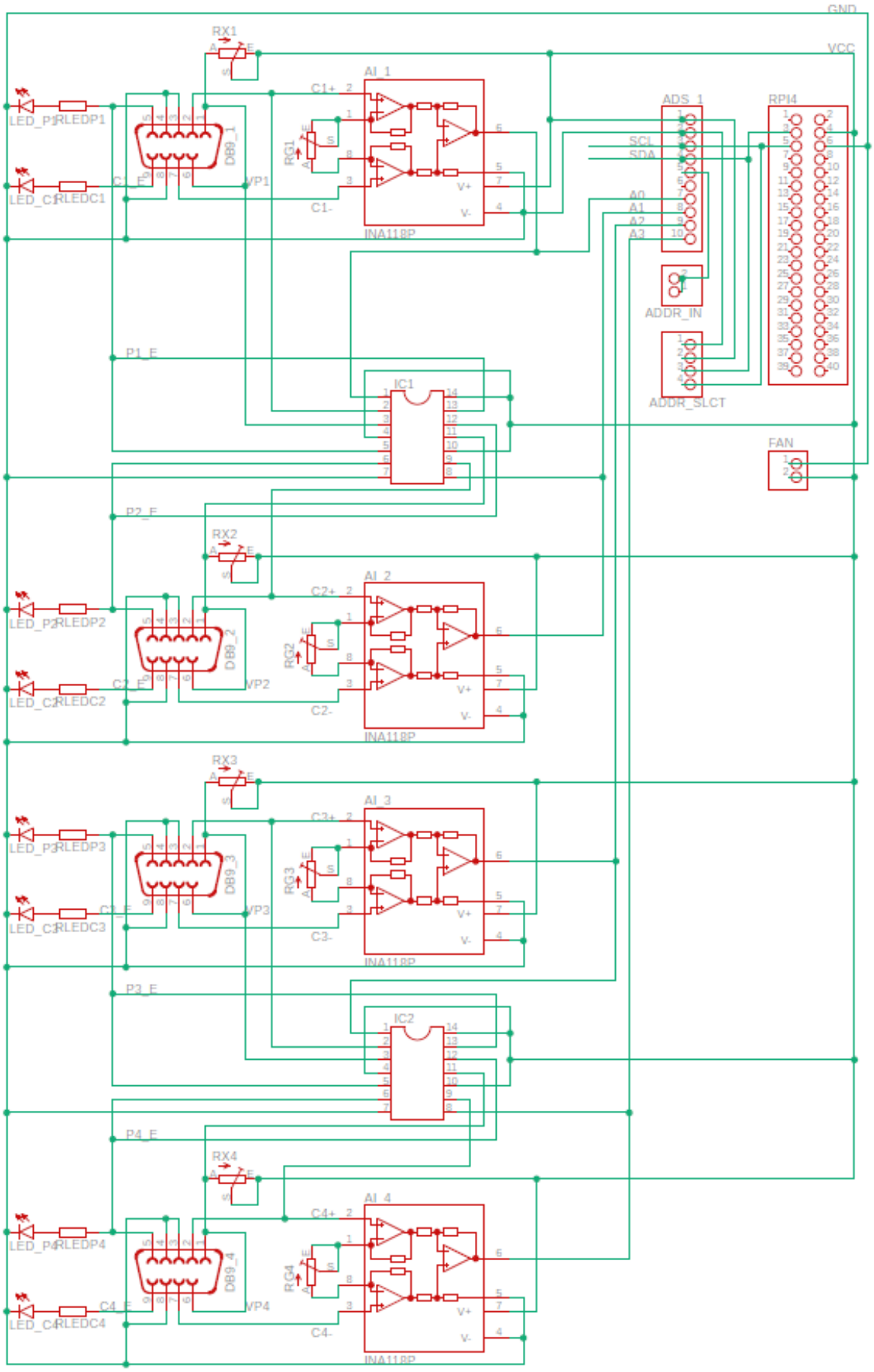


Figura 98: Esquema versión avanzada

El rutado resulta un poco más complejo que en el caso del Prototipo 2 por las nuevas rutas necesarias para el interruptor, por lo que el ancho de pista y *clearance* se han reducido de los 12 mil a 10 mil, que sigue siendo más que suficiente para la corriente que circulará por el circuito, pero da mejores resultados hechas varias pruebas con el autorouter de Eagle. El espacio de la PCB es suficiente para alojar los interruptores DIP14 con un pequeño desplazamiento de los potenciómetros de Rx. El resultado de esta operación se muestra en la figura 99 inferior.

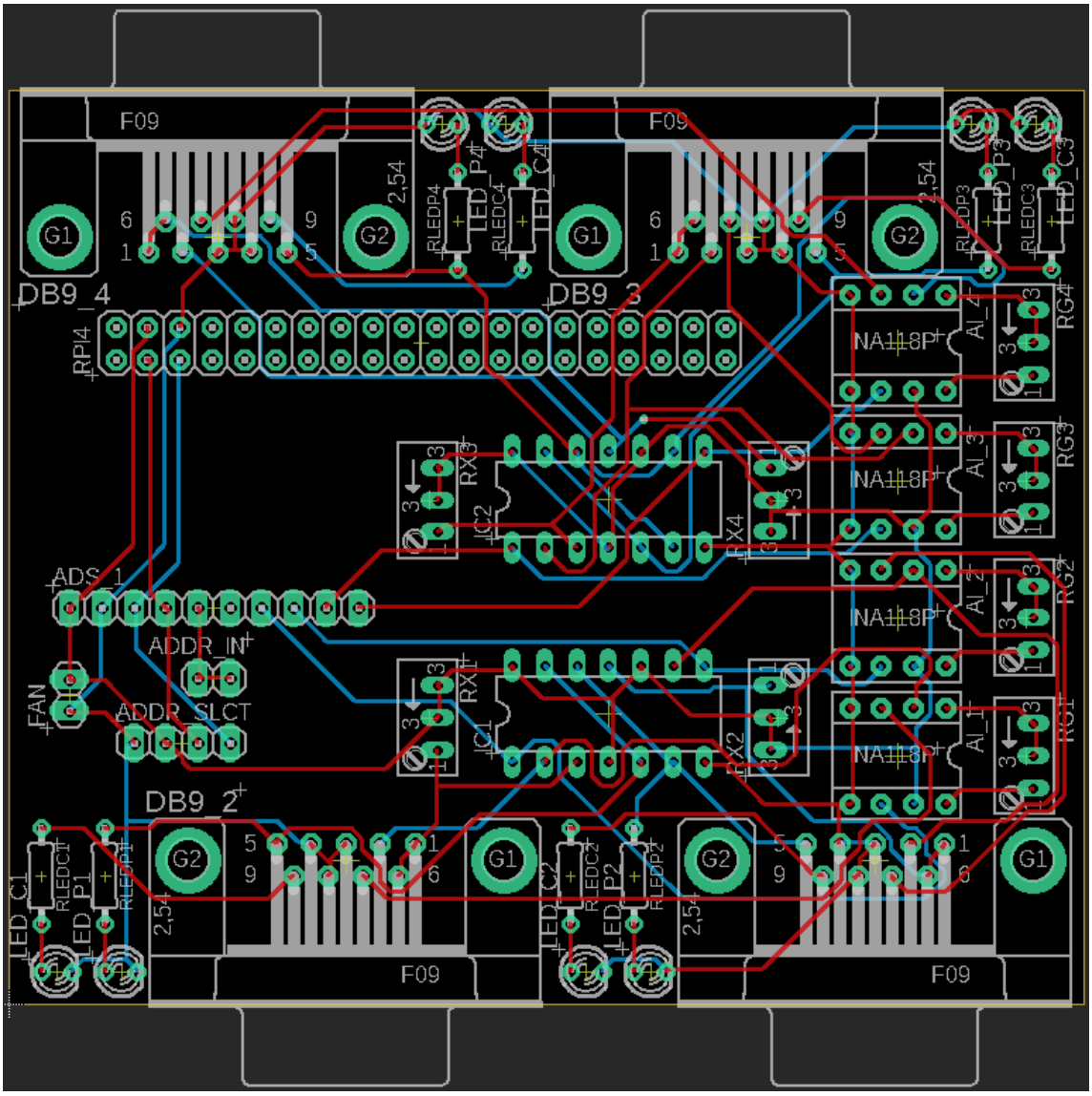


Figura 99: PCB versión avanzada

Por último, el coste es, nuevamente representado en una tabla (en este caso la tabla 7), en componentes es muy similar al del Prototipo 2. Se recomienda realizar pruebas con este diseño en el futuro para asegurar su viabilidad.

Referencia PCB	Componente	Modelo	Huella	Fabricante	Refer. Farnell	Nº ud	Coste/ud (€)	Cte/10 ud (€)
AI_1, AI_2, AI_3, AI_4	Amplificador de instrumentación	AD622ANZ	DIP8	Analog Devices	4019190	4	~ 12	~ 11
		INA128P		Texas Instruments	3004493			
ADS_1	ADC 16 bits, 4 canales	ADS1115	17,27x 27,94mm	Adafruit	No disp.	1	14,95	13,46
IC1, IC2	Interrupto analógico 4 canales	MAX4614 CPD	DIP8	Analog Devices	2513144	2	3’99	3’58
RG1, RG2, RG3, RG4	Potenciómetro multivuelta 200Ω	T93YA201 KT20	RJ9W	Vishay	1141398	4	2,09	1,83
RX1, RX2, RX3, RX4	Potenciómetro multivuelta 500Ω	T93YB501 KT20	RJ9W	Vishay	1141414	4	1,88	1,65
RLEDP1, RLEDP2, RLEDP3, RLEDP4, RLEDC1, RLEDC2, RLEDC3, RLEDC4	Resistencia 330Ω	MRS2500 0C3300FC T00	R-EU_ 0204/7	Vishay	9467327	8	0,227	0,227
LEDP1, LEDP2, LEDP3, LEDP4	LED rojo 3mm	TLHR4405	LED3MM	Vishay	1045457	4	0,338	0,241
LEDC1, LEDC2, LEDC3, LEDC4	LED verde 3mm	TLHG4400	LED3MM	Vishay	1652496	4	0,536	0,378
DB9_1, DB9_2, DB9_3, DB9_4	Conector DB9 en ángulo recto	5747 844-6	F09HP	TE Connectivity	1338746	4	2,97	2,61
RPI4	Zócalo pines hembra 2x20	M20-6102045	PINHD-2X20	Harwin	1569230	1	5,72	5,20
IC1, IC2, AI_1, AI_2, AI_3, AI_4	Zócalo componentes DIP8	2227MC-08-03-18-F1	DIP8	Multicomp Pro	1103844	6	0,258	0,182
ADDR_IN, ADDR_SLCT, FAN	Pines macho conexión	826629-8	PINHD-1X2, PINHD-1X4	TE Connectivity	3418364	1	1,71	1,50
						Tot.	112,98	101,06
						+IVA	136,71	122,29

Tabla 7: Coste versión avanzada

5. Diseño de la envolvente

Para contener la Raspberry y electrónica asociadas al proyecto se ha optado por diseñar una envolvente personalizada e imprimible en 3D. Esto es porque las características geométricas del proyecto son muy específicas como para encontrar una caja normalizada que se adapte a ellas y permita acceder a los puertos de conexionado correctamente sin incurrir en importantes modificaciones. Así pues, aprovechando la disponibilidad en el taller de estructuras de una impresora 3D, se diseñará una envolvente propia.

La impresora que se utilizará es una Ender 3, fotografiada en la figura 100, modificada para tener extrusión directa (un motor paso a paso empuja directamente el filamento a la boca del extrusor), además de otras modificaciones menores como tensor manual y mejoras de estabilidad. Utiliza Marlin como firmware controlador y tiene un BLTouch incluido como sensor de nivelación, el cual tomando 9 puntos de la cama caliente ajusta automáticamente su altura.

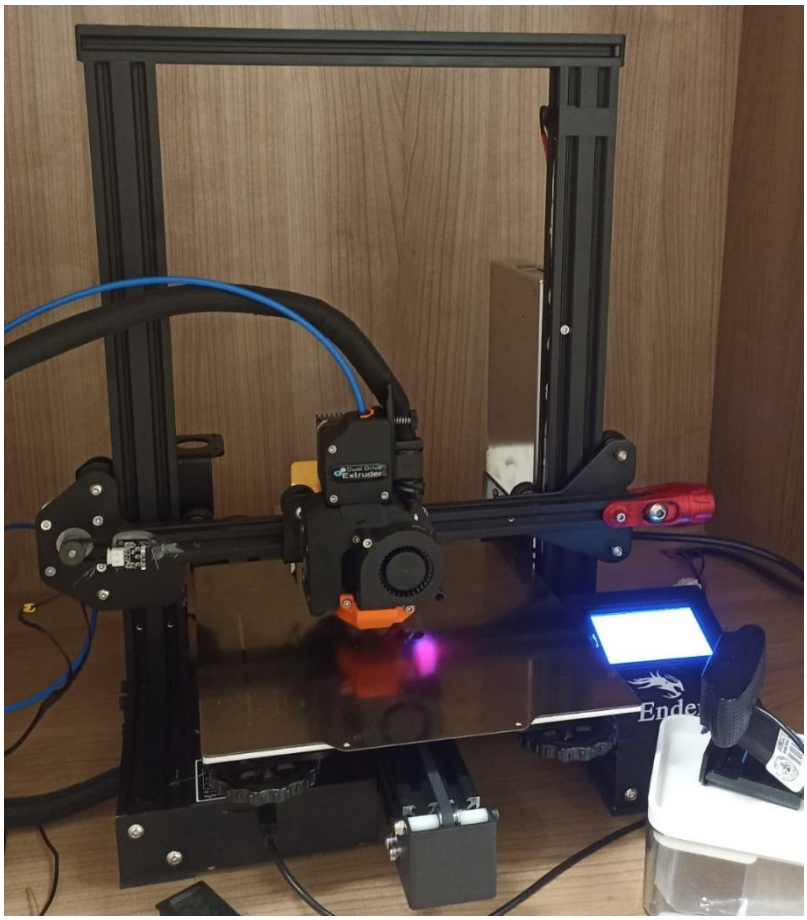


Figura 100: Impresora Ender 3

5.1. Fusion360

La evolución del diseño ha seguido un proceso iterativo que ha constado de 3 versiones, hasta alcanzar la que se ha considerado como definitiva. De cada una de las versiones prototípicas anteriores se ha extraído una mejora aplicable a la geometría de la envolvente o al proceso de modelado de esta. El programa escogido para este proceso de diseño ha sido el Fusion360, de Autodesk.

La elección de dicho programa se ha basado en tres puntos principales. El primero la compatibilidad de este software con Eagle, ambos propiedad de Autodesk, que permite importar la PCB diseñada en el apartado anterior para tomar referencias y medidas con facilidad. El segundo es que es un programa de diseño 3D bastante sencillo e intuitivo, en comparación con otras propuestas como Catia, o incluso de la misma compañía, cómo pudiera ser Inventor; además de que permite incluir en una misma escena varias piezas diferentes para tomar y proyectar medidas. Y el tercero es la facilidad con que se pueden exportar los diseños para imprimir en 3D, motivo por el que se puede encontrar fácilmente en internet piezas libres de otros creadores que tomar como referencia o directamente incluir en el diseño. El inicio de Fusion se puede ver en la figura 101.

La forma más básica de diseño geométrico en Fusion360 es crear un boceto 2D sobre un plano y después extruirlo la cantidad necesaria. Todas las caras de una geometría creada de esta forma pasarán a contar como planos sobre los que se pueden crear nuevos bocetos e ir sumando de esta forma complejidad al diseño. Fusion360 ofrece varias herramientas propias de los programas CAD para el trazado de los bocetos, como pudieran ser referencias automáticas al centro de las geometrías o restricciones para métricas de tangencia o paralelismo; así como de refinado de la geometría tridimensional, como chaflanes y redondeos o taladros. Con la suma de estas herramientas, se irá esculpiendo la envolvente, apilando secuencialmente operaciones. En las últimas versiones, Fusion360 incluye una línea temporal que permite volver atrás para corregir errores o ajustar medidas en esta secuencia, por lo que es recomendable que todos los bocetos estén suficientemente acotados y parametrizados.

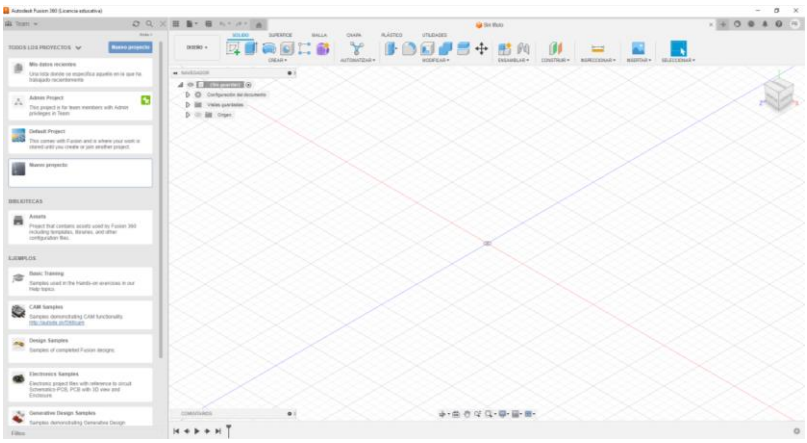


Figura 101: Pantalla inicial Fusion360

5.2. Impresión 3D

Además de diseñar una envolvente que cubra todos los elementos electrónicos (Raspberry y PCB), también se debe de que vigilar que se siga permitiendo el acceso a los puertos varios (DB9 y conectores de la Raspberry). Además, puesto que la pieza se va a imprimir, se debe de tener en cuenta este aspecto en el diseño, para poder permitir una impresión lo más fluida posible. Se deben de tener en cuenta los siguientes puntos:

- Permitir cierta tolerancia en las medidas, puesto que el extrusor de la impresora no es capaz de hacer una capa menor de 0,1 mm;
- Evitar los voladizos en medida de lo posible, ya que tendrían que generarse material de soporte, lo que retrasaría el tiempo de impresión;
- Generar tantos espacios abiertos como sea posible, para reducir la cantidad de material utilizado y mejorar la ventilación, con el objetivo de evitar el sobrecalentamiento, especialmente del procesador de la Raspberry.

Una vez modelada la pieza en Fusión360, la se guardará como archivo de malla con extensión ".stl". Este archivo se importará al programa Cura, un *slicer* que generará el código G de instrucciones para la impresora a partir de esta malla. En él pueden ajustarse la cara sobre la que se imprimirá, el relleno de los volúmenes, la altura de las capas, la temperará del extrusor y la cama caliente y la posición de las mallas sobre esta, que se posicionarán de forma que ocupen el menor espacio posible. Por último, una vez generado el código G, se cargará en la impresora a través del programa Octoprint, según el *setup* que hay preparado en el taller y se imprimirá.

5.3. Evolución del diseño

Lo que se hará para diseñar nuestra envolvente es colocar todos los elementos electrónicos y puertos que componen nuestro proyecto, situados en el espacio de trabajo de un proyecto en Fusion360 según sus medidas previstas. De esta forma se podrá trabajar esculpiendo la geometría del envolvente entorno a ellos, utilizándolos como referencias fijas en todo momento.

Lo primero por tanto será importar la PCB que se ha diseñado en el apartado anterior "4. Diseño electrónico". Para ello, se exporta la PCB desde Eagle usando la pestaña a la derecha de la interfaz de diseño de PCB, según se ve en la figura 102; y lo guardamos siguiendo los pasos indicados. Después podremos abrirla desde Fusion como otro componente del proyecto [43].

Se agregarán como componentes en el proyecto de Fusion esta PCB exportada, una placa de Raspberry 4 modelada siguiendo las medidas reales de puertos y talados, una representación de los 4 puertos DB9 necesarios, una representación del ventilador, y un bloque que represente la distancia existente entre la Raspberry y la PCB, medido del bloque de pine 2x40 utilizado. Estas partes constituyentes se alinearán según las distancias que han de guardar en la realidad, tal y como se muestra en la figura 103 inferior. Esta será la referencia sobre la que trabajar la envolvente.

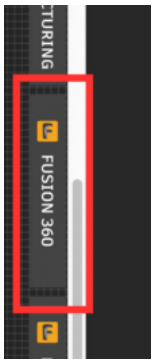


Figura 102: Pestaña exportar a Fusion [43]

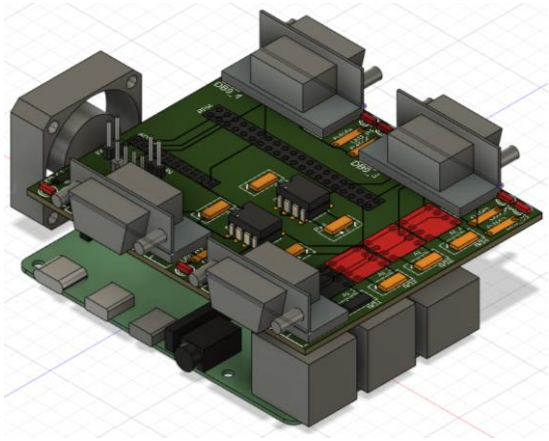


Figura 103: Componentes posicionados en Fusion360

5.3.1. Prototipo 1

Para trazar este primer prototipo, se proyectan los contornos exteriores de las PCBs en un plano inferior y se boceta la base, dando un desfase a los bordes para dejar un espacio libre entre las paredes y las placas. Desde la base se extruyen las paredes, resultando en una caja sencilla cuadrada. En las paredes se proyectan entonces los contornos de los puertos de las Raspberry, los DB9, el hueco del ventilador y el futuro hueco que necesitarán los LED y se extruye un corte para dejar ese espacio. Se deja un margen de error alrededor de estos cortes de 0,4 mm. De la misma forma, se crean unas ranuras verticales en la pared opuesta al conector 2x20 para permitir la salida del flujo de aire del ventilador. Se incluyen además unos salientes para que la PCB pueda reposar sobre las paredes de la caja. Para finalizar, se toman las medidas de la parte superior de la caja y se modela una tapa con ranuras para aliviar la cantidad de material utilizado. El resultado del modelo se muestra en las figuras 104 y 105.

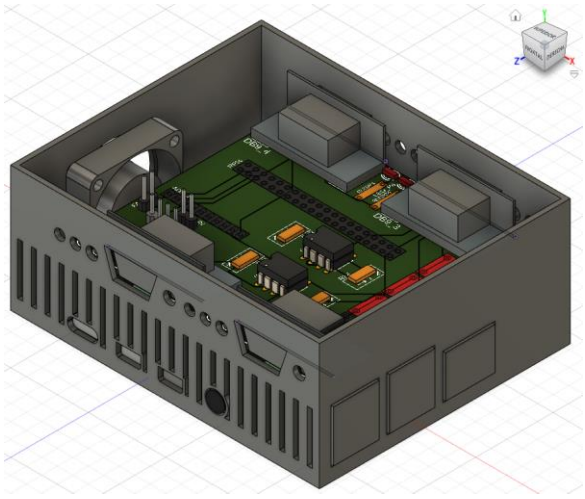


Figura 104: Prototipo 1 vista 1

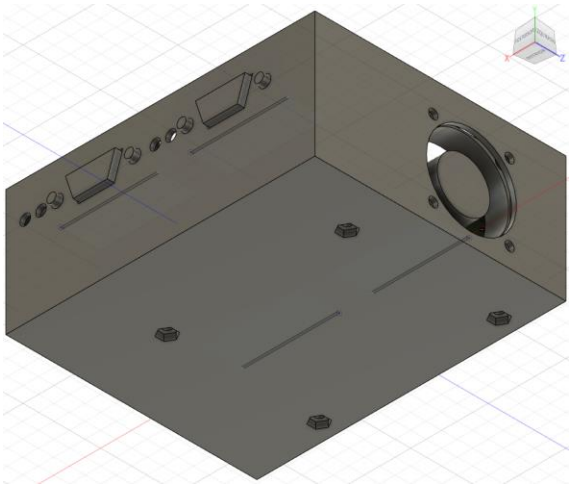


Figura 105: Prototipo 1 vista 2

Una vez impreso, según se muestra en la figura 106, y una vez colocados los componentes electrónicos, se identificaron una serie de problemas a mejorar:

- Las paredes, de un ancho de 1,5 mm en esta versión, se separaron de la base cuando se enfrió. Se reforzarán las paredes hasta un ancho de 2,5 mm, y si fuera necesario, se incluirá un chaflán interior a 45º como refuerzo.

- Se tomó como referencia las medidas de los puertos DB9 hembra de la PCB, no de los macho, los de los sensores, por lo que no hay espacio suficiente para conectarse.
- Se ha comprobado que hay un desfase mayor del estimado entre los puertos de la Raspberry y los DB9 de la PCB. Están estos segundos más hacia el interior de lo esperado. Habrá que modelar la pared correspondiente en el siguiente prototipo para conseguir un mejor ajuste.
- La tapa se modeló por separado del cuerpo principal de la envolvente, las medidas trasladadas de un archivo a otro. Resulta más sencillo crear dos cuerpos diferentes en el mismo archivo de Fusion360 y proyectar directamente las medidas.
- Solo se han tenido en cuenta los puertos de la Raspberry 4, el siguiente prototipo se hará compatible con los puertos de los modelos 3 y 4.

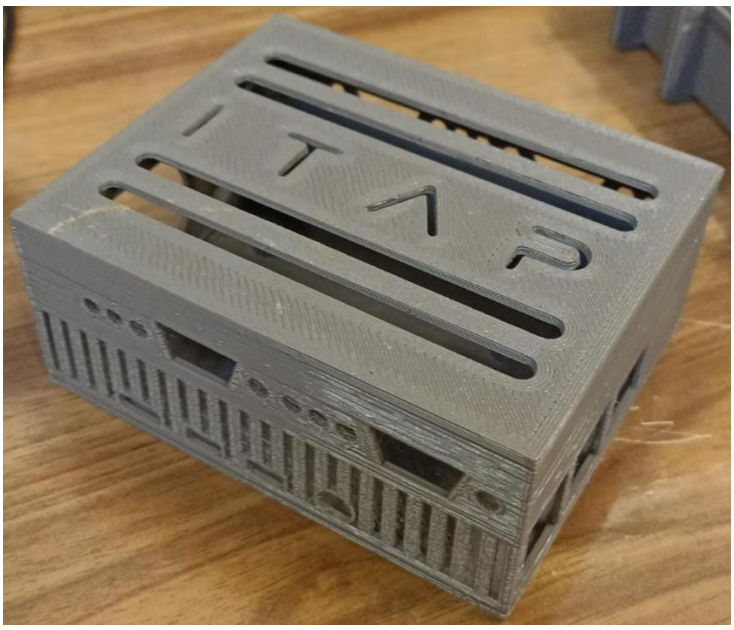


Figura 106: Prototipo 1 impreso

5.3.2. Prototipo 2

El prototipo 2 tuvo que ser directamente descartado por el evento que se señalaba en el apartado “4.2.4.2 Primera versión”, el cambio de la PCB. Con el modelo corregido, ya no resulta necesario adaptar la forma de la pared al entrante de los puertos. Sin embargo, sí se pueden reseñar algunos puntos a tener en cuenta vistos a partir de este prototipo:

- El voladizo dejado para los puertos es demasiado largo para la impresora. Se colocarán soportes fijos intermedios que no interfieran con los puertos de la versión 3 ni 4 de Raspberry , además de material de soporte en el *slicer*.
- La impresora hace con mayor calidad de impresión las rejillas de ventilación de tipo hexagonal que las rectangulares. Se utilizarán de este tipo a partir de ahora.
- No es posible introducir la PCB en la caja porque los espacios abiertos para los DB9 están cubiertos por encima, y el espacio alrededor no es suficiente para maniobrar. En la siguiente versión se dejará la parte superior de los puertos libre en el cuerpo de la caja, acabando de cubrirlos con la tapa.

- Los chaflanes a los lados del ventilador complican el diseño simétrico y no aportan realmente en cuestión de ahorro de material ni tiempo de impresión.
- Para modelar caja y tapa como cuerpos separados en el mismo proyecto, se puede generar un nuevo cuerpo desde la extrusión de un plano desfasado y la separación se mantendrá, siempre y cuando las geometrías no se toquen.

Se muestran a continuación las vistas de este modelo en Fusion en las figuras 107 y 108 y el resultado final impreso en la figura 109.

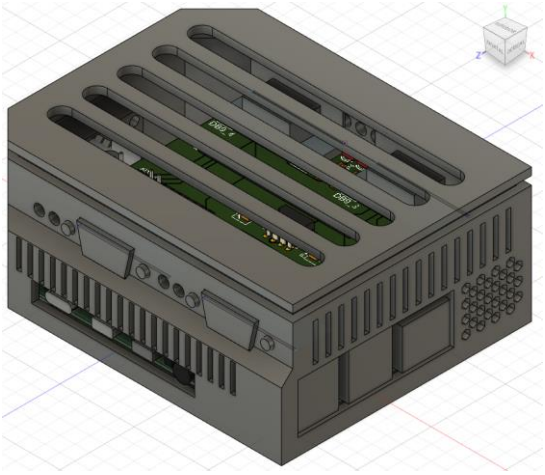


Figura 107: Prototipo 2 vista 1

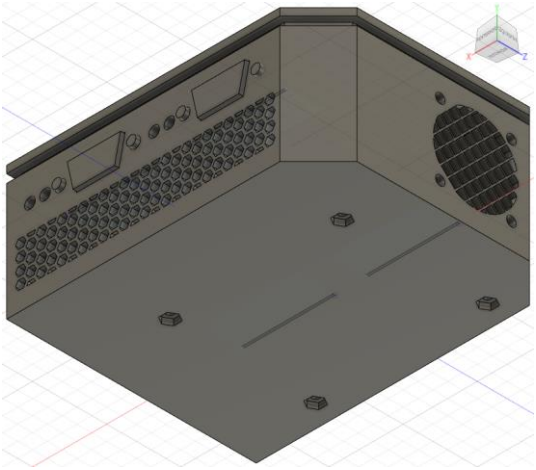


Figura 108: Prototipo 2 vista 2

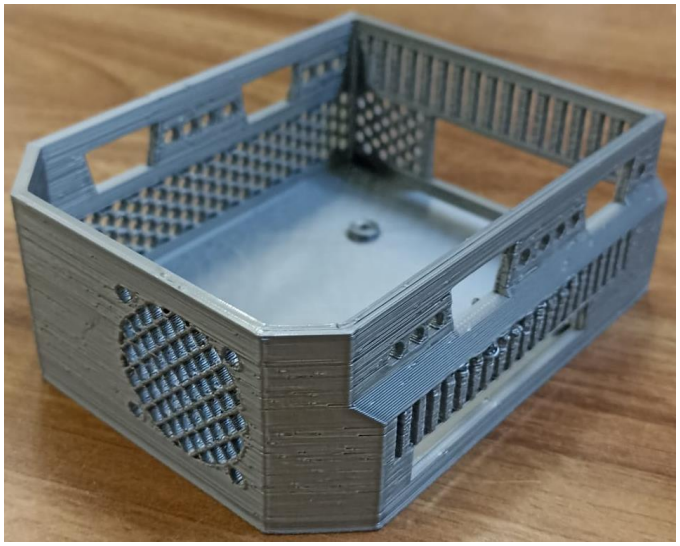


Figura 109: Prototipo 2 impreso

5.3.3. Diseño final

En esta última versión se ha unido lo aprendido de los defectos de las anteriores junto con la mejora del alineamiento entre la PCB y la Raspberry en la versión corregida del apartado "4.2.4.2. Segunda versión". Se ha respetado además que las medidas de geometrías y márgenes de error entorno a los componentes fijos se mantuvieran en medidas múltiplo de 0,4 mm, que es el diámetro del extrusor de la impresora. Por último, se ha priorizado

poner el mayor número posible de rejillas hexagonales para refrigerar la Raspberry. Los hexágonos son un elemento estructural muy fuerte que la impresora es capaz de gestionar sin inconveniente. La única pared de la caja libre de ellos es la del ventilador, asegurando un solo flujo de entrada de aire.

En cuanto a la tapa de la caja, se ha modelado de forma que complete el cierre por encima de los puertos DB9. El cierre de la tapa se ha optado por hacerlo a presión, dejando un margen de error ajustado y una solapa inferior en la tapa que la mantenga en su sitio. Dicha solapa se modela lo suficientemente larga como para que se pueda colocar entorno a la misma una cinta adhesiva, a fin de ajustar aún más el cierre. El modelo de esta versión se muestra en las figuras 110 y 111.

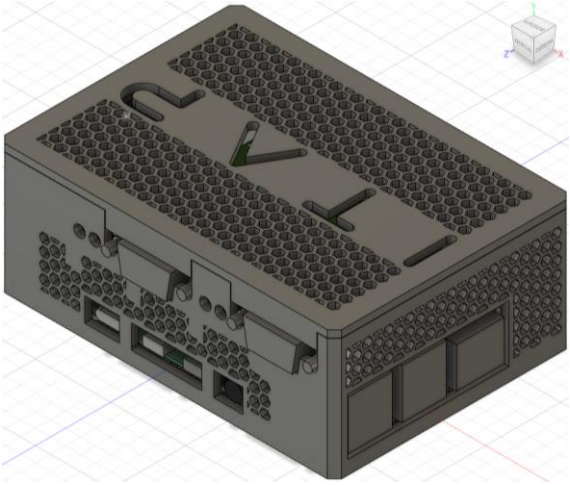


Figura 110: Diseño final vista 1

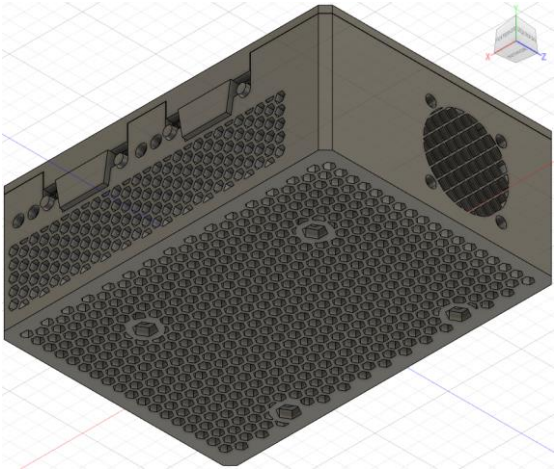


Figura 111: Diseño final vista 2

Que una vez impreso, queda el siguiente resultado, mostrado en las figuras 112 y 113, al que no se le observan errores notables que se considere se deban corregir:

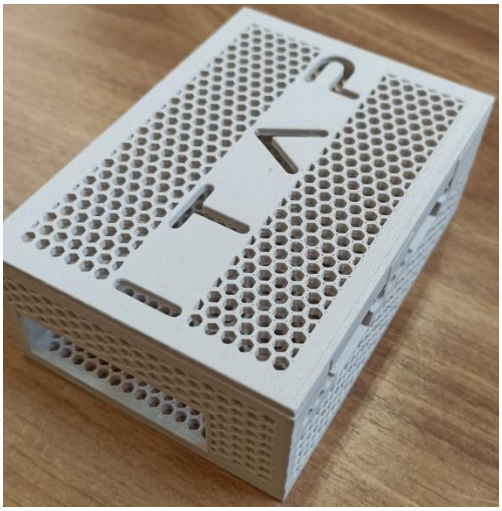


Figura 112: Diseño final impreso vista 1

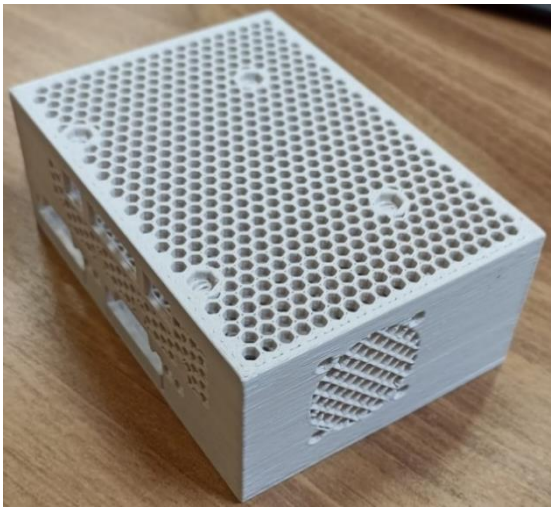


Figura 113: Diseño final impreso vista 2

6. Resultados

Finalmente, recogemos los resultados conseguidos en el desarrollo del TFM y el estado final del prototipo desarrollado.

Lo primero que se puede comprobar es que la PCB y Raspberry entran sin problema en la envolvente final diseñada, como se muestra en las siguientes figuras 114 y 115. De la misma forma se puede comprobar que queda sucediente espacio para las conexiones con los puertos y las placas pueden mantenerse fijas en su interior.

Figura 114: Caja y PCB con tapa

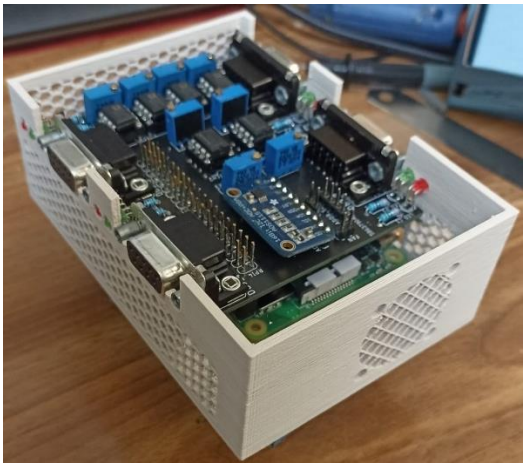


Figura 115: Caja y PCB sin tapa

Ahora se pasa a conectar al conjunto Raspberry-PCB un potenciómetro y una célula de carga, según se muestra en la figura 116. Ambos sensores cuentan con el DB9 adaptado para su compatibilidad con Sirius y con la señal de *Enable* correspondiente puenteada en el interior de los DB9. La PCB utilizada aquí se corresponde con el estado final en que se dejó adaptada (rutas seccionada y cables agregados) la placa descrita en el apartado “4.2.4.2. Segunda versión”. Es decir, tiene capacidad para 2 potenciómetros en un lado y dos células en el opuesto. Los potenciómetros multivuelta de ganancia (RG) y alimentación (Rx) se encuentran ajustados según lo descrito en el apartado “3.3.3.3 Conclusiones y pruebas con el AI”.

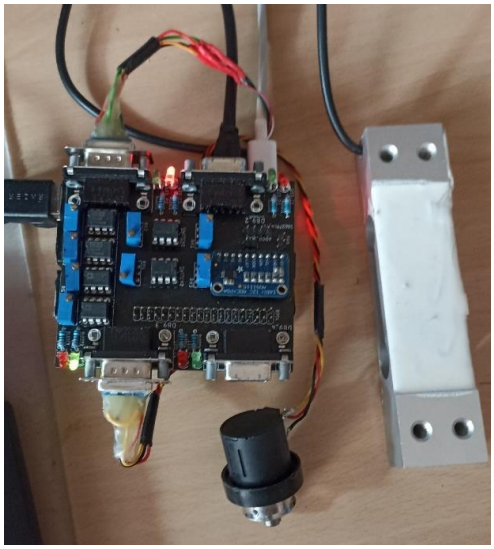


Figura 116: Conexionado final

Se enciende ahora la Raspberry y se inicia el programa final preparado para los 4 canales del ADS1115, descrito en el apartado “3.4.2.4. Programa final”. Se comprueba con la salida por pantalla, que se muestra en la figura 117, que se están escribiendo medidas en la base de datos y que estas varían si se alteran los sensores conectados.

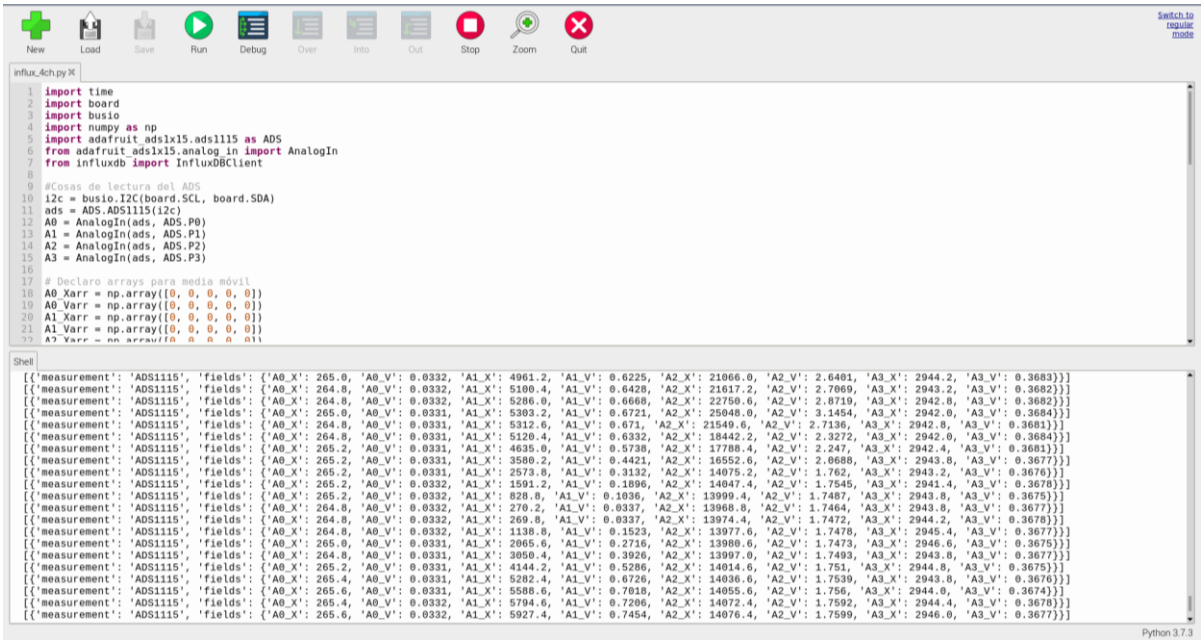


Figura 117: Programa final

Por último, se abre el *dashboard* de Grafana descrito en el apartado “3.4.2.5. Interfaz en Graphana” y se comprueba que se está recibiendo la información de la base de datos correctamente y en tiempo real, como se muestra en la figura 118. Los canales en uso son la superior derecha con célula y el inferior izquierdo con potenciómetro. Lo que vemos en los otros canales, libres de sensores en esta prueba, son pequeños errores amplificados por la escala adaptable de la gráfica.

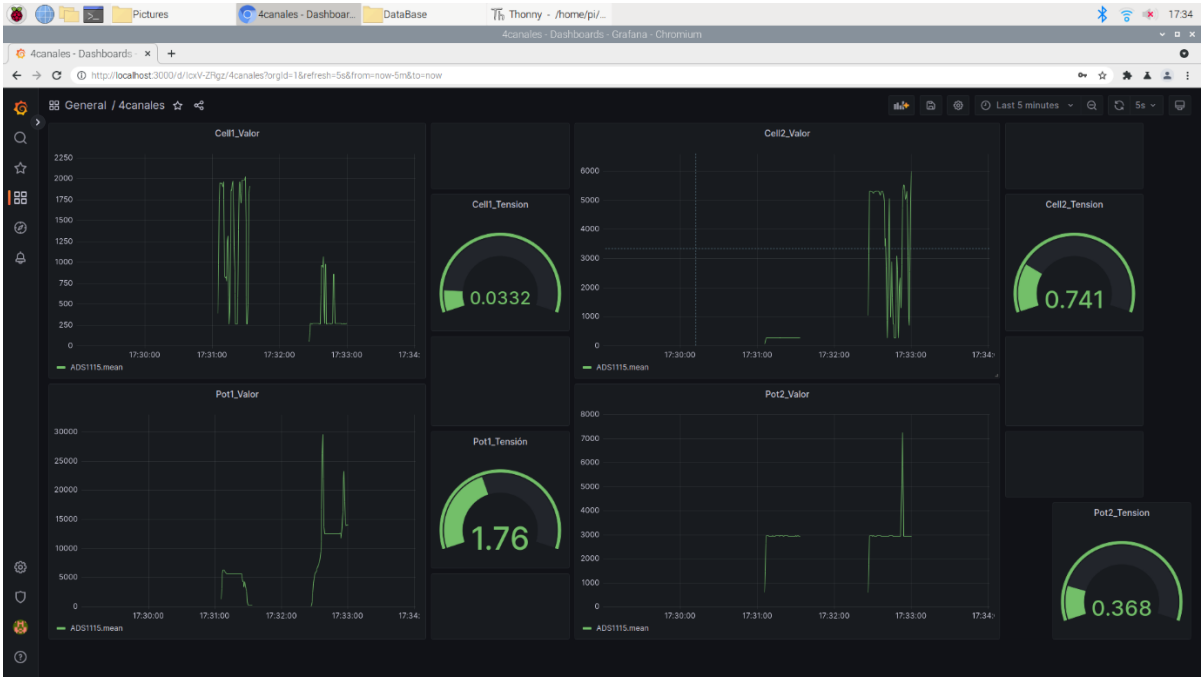


Figura 118: Dashboard final

7. Conclusiones

Se considera que, por lo general, se han cumplido con los objetivos propuestos para este trabajo. Se ha conseguido diseñar un sistema accesible, adaptable, escalable y reproducible, que muestra datos de los sensores en tiempo real a cualquier usuario.

El tiempo real se consigue con el uso combinado de InfluxDB y Grafana para la recogida y muestra de datos instantánea y dependiente del tiempo.

La accesibilidad se alcanza gracias al uso de Grafana, por el que cualquier usuario que comparta red WiFi con la Raspberry que realice el procesado puede conectarse con cualquier dispositivo que tenga un navegador web y visualizar los datos.

La adaptabilidad se logra con los potenciómetros integrados en el diseño, que, si bien no es la opción más cómoda, puesto se tienen que ajustar a mano; sí que resulta una solución sólida y da el margen de uso de varios sensores.

La escalabilidad se obtiene gracias a la filosofía seguida de PCBs que pueden ser colocadas una sobre otra hasta conseguir un máximo de 16 entradas distintas totales, siendo la única limitación las 4 direcciones posibles del ADS1115.

La reproducibilidad del prototipo se gana gracias al uso de software de código abierto, al uso de componentes comerciales y a otros elementos, como es la impresión 3D y pedidos online, que pueden ser variablemente accesibles por cualquier persona.

Si bien no se ha conseguido que el último prototipo desarrollado realmente alcance todas las prestaciones que le serían deseables, en el desarrollo del proyecto sí que se han identificado los problemas que pueden aparecer y los errores en que se puede caer. Y se ha podido desarrollar un prototipo físico funcional de características limitadas, pero que ya cubre varios escenarios de uso. Además, se ha podido utilizar lo visto para proponer dos posibles vías de desarrollo para nuevos prototipos que cubren necesidades distintas.

En el presente documento se deja constancia de los problemas que pueden aparecer en el proceso de desarrollo de un prototipo de estas características, incluyendo vías enteras que deben ser abandonadas y adaptaciones a las nuevas situaciones imprevistas que surgen en el desarrollo.

En resumen, se concluye que se han conseguido una parte importante de los objetivos propuestos, y por el camino desarrollado se ha descubierto como enfrentarse y adaptarse a los problemas y circunstancias cambiantes de un proyecto de prototipado, se ha aprendido a conjugar campos distintos de la ingeniería, así como ampliar en lo general conocimientos de las plataformas utilizadas y se ha terminado con un prototipo funcional que utilizar o expandir sobre la base que ha asentado.

7.1. Avances conseguidos

Se ha conseguido desarrollar un prototipo de comprobada funcionalidad que muestra las capacidades y limitaciones de la propuesta de sistema desarrollado. Esto se ha logrado con una combinación de software de código libre, adaptado a las necesidades del prototipo y con el diseño propio de PCB y caja.

Así mismo, también se ha logrado la compatibilidad de sensores entre este sistema y Sirius, que era uno de los requisitos principales a cumplir. Esto se consigue con el esquema de alimentación de la PCB y el cableado de los DB9.

Si bien las capacidades del sistema comercial Sirius que se tomaba como referencia siguen quedando muy alejadas, en términos de precisión, capacidad de aceptar distintos sensores, capacidad de trabajar con la señal recibida, etcétera; si que se ha conseguido un prototipo que sigue esa idea de toma de señales en tiempo real y que resulta mucho más económico, que puede ser replicado por cualquiera con los medios e instrucciones disponibles, y que puede tener aplicaciones en campo de la educación o en pequeñas aplicaciones de control específicas.

Por último, todos los archivos de programa, PCB y modelado se aportan adjuntos a la presente memoria de TFM, a fin de poder trabajar sobre ellos en el futuro o crear nuevos prototipos a partir de ellos.

7.2. Sugerencias a futuro

Como ya se mencionaba, el sistema tal cuál ha sido concebido es ya funcional y cubre varios casos de uso. Se identificaron sin embargo durante el proceso de desarrollo unas propuestas para ampliar sus posibilidades y mejorar su compatibilidad de forma automática con un mayor rango de sensores. Propuestas que no se llevaron a cabo por ser demasiado extensas de implementar para el presente TFM, pero que se considera importante reseñar para posibles trabajos futuros.

7.2.1. Multiplexor y tabla de la verdad

Con el diseño actual, el dispositivo puede alimentar automáticamente con la tensión indicada a potenciómetros y células de carga. Sin embargo, la Raspberry desconoce qué tipo de sensor en concreto hay en cada puerto, información que podría utilizarse para escalar correctamente las tensiones recibidas respecto medidas del SI.

De lo que se puede disponer para esto es de los pines *Enable* que ya han sido implementados en los DB9. Esa señal correspondería a un 0 o 1 digital que señale a la Raspberry lo que hay conectado. No se puede hacer una conexión directa, pues no habría GPIO libres suficientes para los 16 sensores máximo que se pueden conectar.

La solución pasa por utilizar dos multiplexores de 8 entradas, siendo las señales de Enable de los 4 DB9 de cada placa estas entradas multiplexadas. La salida sería un número que ya sí podría ser enviado directamente a un GPIO de la Raspberry que lo lea. Cada número

de salida significaría una combinación de entradas, que se distinguen mediante una tabla de la verdad, presentada aquí dividida en dos (tablas 8 y 9).

ADS1115_1								ON/OFF								Nº en Deci-mal
DB9_1		DB9_2		DB9_3		DB9_4		DB9_1		DB9_2		DB9_3		DB9_4		
EP1	EC1	EP2	EC2	EP3	EC3	EP4	EC4	EP1	EC1	EP2	EC2	EP3	EC3	EP4	EC4	
0	0	0	0	0	0	0	0									0
0	0	0	0	0	0	0	1									1
0	0	0	0	0	0	1	0									2
0	0	0	0	0	1	0	0									4
0	0	0	0	0	1	0	1									5
0	0	0	0	0	1	1	0									6
0	0	0	0	1	0	0	0									8
0	0	0	0	1	0	0	1									9
0	0	0	0	1	0	1	0									10
0	0	0	1	0	0	0	0									16
0	0	0	1	0	0	0	1									17
0	0	0	1	0	0	1	0									18
0	0	0	1	0	1	0	0									20
0	0	0	1	0	1	0	1									21
0	0	0	1	0	1	1	0									22
0	0	0	1	1	0	0	0									24
0	0	0	1	1	0	0	1									25
0	0	0	1	1	0	1	0									26
0	0	1	0	0	0	0	0									32
0	0	1	0	0	0	0	1									33
0	0	1	0	0	0	1	0									34
0	0	1	0	0	1	0	0									36
0	0	1	0	0	1	0	1									37
0	0	1	0	0	1	1	0									38
0	0	1	0	1	0	0	0									40
0	0	1	0	1	0	0	1									41
0	0	1	0	1	0	1	0									42
0	1	0	0	0	0	0	0									64
0	1	0	0	0	0	0	1									65
0	1	0	0	0	0	1	0									66
0	1	0	0	0	1	0	0									68
0	1	0	0	0	1	0	1									69
0	1	0	0	0	1	1	0									70
0	1	0	0	1	0	0	0									72
0	1	0	0	1	0	0	1									73
0	1	0	0	1	0	1	0									74
0	1	0	1	0	0	0	0									80
0	1	0	1	0	0	0	1									81
0	1	0	1	0	0	1	0									82

Tabla 8: Tabla de la verdad multiplexor parte 1

DB9_1		DB9_2		DB9_3		DB9_4		DB9_1		DB9_2		DB9_3		DB9_4		
EP1	EC1	EP2	EC2	EP3	EC3	EP4	EC4	EP1	EC1	EP2	EC2	EP3	EC3	EP4	EC4	
0	1	0	1	0	1	0	0									84
0	1	0	1	0	1	0	1									85
0	1	0	1	0	1	1	0									86
0	1	0	1	1	0	0	0									88
0	1	0	1	1	0	0	1									89
0	1	0	1	1	0	1	0									90
0	1	1	0	0	0	0	0									96
0	1	1	0	0	0	0	1									97
0	1	1	0	0	0	1	0									98
0	1	1	0	0	1	0	0									100
0	1	1	0	0	1	0	1									101
0	1	1	0	0	1	1	0									102
0	1	1	0	1	0	0	0									104
0	1	1	0	1	0	0	1									105
0	1	1	0	1	0	1	0									106
1	0	0	0	0	0	0	0									128
1	0	0	0	0	0	0	1									129
1	0	0	0	0	0	1	0									130
1	0	0	0	0	1	0	0									132
1	0	0	0	0	1	0	1									133
1	0	0	0	0	1	1	0									134
1	0	0	0	1	0	0	0									136
1	0	0	0	1	0	0	1									137
1	0	0	0	1	0	1	0									138
1	0	0	1	0	0	0	0									144
1	0	0	1	0	0	0	1									145
1	0	0	1	0	0	1	0									146
1	0	0	1	0	1	0	0									148
1	0	0	1	0	1	0	1									149
1	0	0	1	0	1	1	0									150
1	0	0	1	1	0	0	0									152
1	0	0	1	1	0	0	1									153
1	0	0	1	1	0	1	0									154
1	0	1	0	0	0	0	0									160
1	0	1	0	0	0	0	1									161
1	0	1	0	0	0	1	0									162
1	0	1	0	0	1	0	0									164
1	0	1	0	0	1	0	1									165
1	0	1	0	0	1	1	0									166
1	0	1	0	1	0	0	0									168
1	0	1	0	1	0	0	1									169
1	0	1	0	1	0	1	0									170

Tabla 9: Tabla de la verdad multiplexor parte 2

De esta forma se podría conocer en todo momento el tipo de dispositivo que haya conectado en cada puerto DB9, y ajustar de forma automática el tratamiento software de la señal en consecuencia. Sería ir un paso más allá sobre la propuesta actual del interruptor analógico.

Cada combinación de tipos de sensores conectada, correspondiente a un número de la tabla, debería tratarse a posteriori para abrir o cerrar los circuitos de alimentación y envío de datos. Es decir, sustituir la actual función de regulación de los interruptores analógicos. Esto podría hacerse con un demultiplexor (como el que se muestra en la figura 119), mosfets u otra aplicación de interruptores analógicos [44].

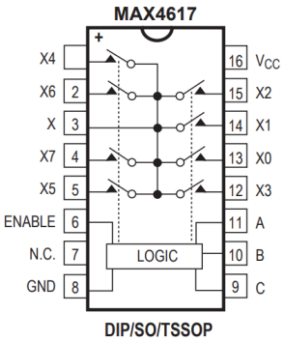


Figura 119: Ejemplo multiplexor, MAX4617 [44]

7.2.2. Uso de potenciómetros digitales

Actualmente, el ajuste de alimentación de las células de carga, si bien cubre cierto margen de error, ha de hacerse a mano, midiendo con un multímetro y girando el potenciómetro correspondiente. Esto conlleva un tiempo cada vez que se quiera cambiar de tipo de células de carga leídas, siempre que sus resistencias sean distintas.

Este cambio podría automatizarse sustituyendo el potenciómetro analógico estándar por uno digital, como el que se muestra en la figura 120, que sea controlado por la Raspberry. Este tipo de circuitos integrados contienen internamente una serie de resistencias que conectan o desconectan para conseguir el efecto de un potenciómetro. Son menos precisos que los potenciómetros analógicos, ya que la suma de resistencias se hace en saltos en lugar de continuo. Cuántas resistencias internas se conecten al circuito viene regido por una comunicación I2C con el dispositivo controlador. Existen potenciómetros digitales de distintos rangos de resistencia total, resistencia por salto y número de potenciómetros internos [45].

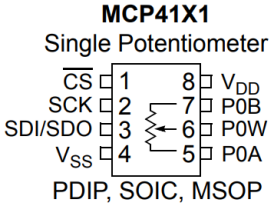


Figura 120: Ejemplo potenciómetro digital, MCP41X1 [45]

7.2.3. Pantalla de ajustes

Si bien Flask resultó no ser la ruta idónea para mostrar datos en tiempo real, sí que sigue siendo una opción interesante con la que crear un servidor web que permite a un usuario comunicarse en remoto con la Raspberry. Esta facultad podría ser utilizada para crear una aplicación en que el usuario pudiera ajustar los rangos de escala de los diferentes sensores conectados.

Esto requeriría crear un servidor web tipo aplicación que permitiera introducir al usuario remoto los valores de escalas. Estos valores serían almacenados en la Raspberry y utilizados para calcular el valor real de la magnitud (escalarlo) a partir de las cifras recibidas del ADS1115. Por tanto, el servidor web tendría que comunicarse con el programa de muestreo, el cual se comenzaría a ejecutar después de finalizarse el ajuste de valores, pasando después a guardar ya los valores reales en la base de datos.

Así mismo, podría crearse una rutina de ajuste a cero antes de lanzar el programa de muestreo. Es decir, obtener la diferencia entre el cero real y el que capta el sensor en el estado teórico de cero. Esto ayudaría a reducir el error en la medida. Seguiría la misma secuencia de ejecución mencionada en el párrafo previo.

7.3. Consideraciones adicionales

7.3.1. Planificación y diagrama de Gantt

En este apartado se aporta un diagrama de Gantt, el representado en la tabla 10, a manera de mostrar de forma más sencilla y eficaz el proceso temporal seguido en el desarrollo del trabajo visto.

	Febrero		Marzo		Abril		Mayo		Junio		Julio		Agosto	
Planificación y reuniones														
Trabajos iniciales														
Desarrollo software fase 1														
Desarrollo software fase 2														
Diseño electrónico														
Montaje y soldadura														
Diseño 3D														
Pruebas del prototipo														
Espera componentes														
Redacción de la memoria														

Tabla 10: Diagrama de Gantt

Las fechas son estimativas, la idea principal a transmitir es la siguiente:

Se tuvo una primera fase de trabajo bastante lineal en la que se plantearon posibilidades y objetivos, algunos de los cuales fueron rápidamente descartados por volver el trabajo excesivamente complejo ya en esas primeras semanas, y un proceso de investigación y puesta a punto, también lineal, de los componentes que se encontraban a disposición.

Después, se comenzó con el desarrollo software del programa que finalmente sería descartado, que llevó más tiempo que el de la solución finalmente adoptada, porque requería sumergirse más profundamente en los lenguajes de programación (HTML5 y CSS), para finalmente cambiar el punto de vista y comenzar con InfluxDB y Grafana.

Hacia el final de esa fase, se comenzó a plantear el primer prototipo de PCB y pedir los componentes electrónicos necesarios para las primeras pruebas, en esta ocasión primera, sobre una protoboard, para poder probar los AI.

Comprobado esto, se procedió a pedir la primera PCB, que tardaba bastante tiempo en llegar y se comenzó mientras tanto con el diseño 3D de la caja para seguir avanzando en el proyecto por otro de sus frentes. Señalar además que fue necesario desoldar los componentes varias veces para cambiar de PCB y hacer pruebas.

Aquí es donde se ve reflejado el proceso iterativo y de trabajos en paralelo que se ha comentado anteriormente. Se siguió esta técnica hasta que se dio por finalizado el prototipo y se procedió a comprobar y tomar los últimos datos para esta memoria, para finalmente redactarla, ya sin necesidad de acudir al laboratorio.

7.3.2. Aspectos económicos

A los inventarios presentados en sus apartados correspondientes (tablas 5, 6 y 7), que nos dan una cifra de costes fija de los componentes electrónicos, podemos acompañarla de otros costes de naturaleza más estimativa, formando en conjunto la tabla 11 final de aspectos económicos del prototipo.

Primeramente, la Raspberry Pi. La Raspberry es el componente más caro con diferencia del prototipo, pero es imprescindible para la funcionalidad del mismo. El problema que encontramos con ella es que coste de la misma puede variar, y de hecho lleva un tiempo variando, fuertemente en el mercado. Puede fluctuar de los 80 hasta los 120 € el modelo 4 y de los 50 a los 80 € el modelo 3. Y no siempre está disponible a la venta.

Por otra parte, también hay un tiempo y material gastado en la impresión 3D. El coste de esta impresión puede estimarse por el consumo de electricidad y el material consumido, sabiendo que el rollo de PLA ronda los 20-30 €.

La PCB (referida en este punto a la placa en sí sobre la que soldar los componentes) por su parte, en coste directo de producción (según la página utilizada), era de solamente 4,5 € por 10 PCBs. Sin embargo, el problema es que el envío no descendía de los 25 € y al menos 15 días de espera. Además, al venir del extranjero, podían estar sujetos a aranceles en la frontera, que supusieron 30 € de media entre los dos envíos.

Por último, se pueden considerar dos costes de manos de obra diferentes. Por un lado el de soldadura y montaje de los componentes es la PCB, que podemos estimar por el salario medio de un Operario de Producción (17000 €/año) por la hora y media aproximada que lleva el proceso. Y, por otro lado, el de Ingeniero Industrial (27500 €/año) para el proceso de diseño general y programación del proyecto, calculando por una estimación de las horas trabajadas en este aspecto del proyecto de entorno a las 4h diarias durante toda la extensión del mismo. Esta mano de obra estimada será finalmente el aspecto más caro del proyecto, lo habitual tratándose de un prototipado.

Elemento de coste	Horas estimadas	Coste por hora/Coste total
Componentes electrónicos	-	120
Raspberry Pi	-	80
Ventilador refrigeración	-	5
PCB	-	0,45
Transporte y aranceles	-	55
Impresión	6	3,5
Montaje/Soldadura	1,5	13,28
Diseño y desarrollo	128	1833,33
	Total	2110,56

Tabla 11: Aspectos económicos

7.3.3. Seguridad y medio ambiente

La mayor problemática de seguridad y medio ambiente que presenta el diseño es aquello que gira en torno a la PCB y componentes electrónicos. En el uso del prototipo en sí, no existe ningún riesgo de seguridad notable. Sin embargo, en su fabricación hay un punto al que se debe de prestar atención, que es la soldadura de componentes de agujero pasante, en dónde se encuentran tres posibles amenazas a la seguridad:

- **El estaño-plomo:** Esta aleación de baja temperatura de fusión (232 ºC) es la que se utiliza para soldar componentes. La baja temperatura y maleabilidad se la da el plomo, que es un metal altamente tóxico. Es por ello que es importante lavarse las manos en profundidad después de manipularlo.
- **El flux:** Es un fluido que se puede, bien aplicar con anterioridad a la soldadura, o que viene embebido en los hilos de estaño-plomo comerciales actualmente. Sirve para limpiar y prevenir la oxidación de los metales en contacto, pero inhalarlo durante la soldadura produce irritación en las vías respiratorias. Es importante realizar la soldadura en un espacio bien ventilado.
- **El estañador:** Es la herramienta utilizada para fundir y conducir el estaño a los metales, por lo que su punta se calienta al entorno de los 300 ºC. Se debe dejar reposar la herramienta en una zona segura alejada de brazos y manos mientras no se esté utilizando directamente.

Por otra parte, la PCB en sí es fabricada por una empresa localizada en Hong Kong, región administrativa especial de la República Popular China, por lo que la huella de carbono del transporte hasta Valladolid, y teniendo en cuenta además que son productos personalizados, es bastante elevada.

Por su lado, los componentes electrónicos utilizados, por su naturaleza, contienen semiconductores, metales pesados, piezas de plástico, etcétera; que los vuelven de difícil y específico reciclaje. En ningún caso se deben de desechar junto con otros residuos urbanos. Para su reciclaje, llegado el caso, se deberán reunir y clasificar y ser portados a un punto limpio municipal o a una empresa especializada.

Por último, el plástico PLA utilizado en la caja diseñada no se trata de un material tóxico, pero sí de un plástico, que como tal se debe reciclar por sus vías normales urbanas correspondientes. Si bien se puede anotar que existen formas de refundir este PLA para ser utilizado en nuevos rollos para impresora 3D.

Bibliografía

[1] Dewesoft, «Sirius Modular». [En línea]. Disponible en: https://dewesoft.com/es/productos/sirius. Acceso: julio 2023.

[2] Dewesoft, «Nuevo SIRIUS UNI universal, galga extensiométrica, IEPE, voltaje, RTD, resistencia y amplificador de corriente». [En línea]. Disponible en: https://dewesoft.com/es/blog/amplificador-sirius-uni-universal. Acceso: marzo 2023.

[3] Arduino, «Arduino». [En línea]. Disponible en: https://www.arduino.cc/: Acceso: marzo 2023

[4] Espressif, «ESP32». [En línea]. Disponible en: https://www.espressif.com/en/products/socs/esp32. Acceso: marzo 2023.

[5] RaspberryPi, «Raspberry Pi». [En línea]. Disponible en: https://www.raspberrypi.com/. Acceso: marzo 2023.

[6] RaspberryPi, «Raspberry Pi 4». [En línea]. Disponible en: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[7] RaspberryPi, «Raspberry Pi hardware». [En línea]. Disponible en: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html. Acceso: julio 2023.

[8] RaspberryPi, «Raspberry Pi OS». [En línea]. Disponible en: https://www.raspberrypi.com/software/. Acceso: marzo 2023.

[9] Adafruit, «ADS1115». [En línea]. Disponible en: https://www.adafruit.com/product/1085. Acceso: marzo 2023

[10] Texas Instruments, «ADS111x Ultra-Small, Low-Power, I2C-Compatible, 860-SPS, 16-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator». [En línea]. Disponible en: https://www.ti.com/document-viewer/ads1115/datasheet. Acceso: marzo 2023

[11] PyMyLifeUp, «Configuring I2C on the Raspberry Pi». [En línea]. Disponible en: https://pimylifeup.com/raspberry-pi-i2c/. Acceso: marzo 2023

[12] Adafruit Circuit Python, «Support for the ADS1x15 series of analog-to-digital converters». [En línea]. Disponible en: https://docs.circuitpython.org/projects/ads1x15/en/latest/index.html. Acceso: abril 2023

[13] Texas Instruments, «INA118 Precision, Low-Power Instrumentation Amplifier.» [En línea]. Disponible en: https://www.ti.com/lit/ds/symlink/ina118.pdf?ts=1689150404080&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FINA118. Acceso: julio 2023

[14] Analog Devices, «Low-Cost Low Power Instrumentation Amplifier AD620». [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/AD620.pdf. Acceso: julio 2023

[15] Texas Instruments, «INA12x Precision, Low-Power Instrumentation Amplifiers». [En línea]. Disponible en: https://www.ti.com/lit/ds/symlink/ina128.pdf?ts=1689130082583&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fes-mx%252FINA128. Acceso: julio 2023

[16] Flask, «User's guide». [En línea]. Disponible en: https://flask.palletsprojects.com/en/2.3.x/. Acceso: marzo 2023

[17] FreeCodeCamp, «How to build a web application using Flask and deploy it to the cloud». [En línea]. Disponible en: https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/. Acceso: abril 2023

[18] PyMyLifeUp, «Finding the IP Address of your Raspberry Pi». [En línea]. Disponible en: https://pimylifeup.com/raspberry-pi-ip-address/. Acceso: abril 2023

[19] MDN web docs, «HTML basics». [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. Acceso: abril 2023

[20] Wikimedia commons, «File:CSS3 and HTML5 logos and wordmarks.svg». [En línea]. Disponible en: https://commons.wikimedia.org/wiki/File:CSS3_and_HTML5_logos_and_wordmarks.svg. Acceso: agosto 2023

[21] Software Testing Help, «Flask App And Flask Project Layout With Blueprint & Bootstrap». [En línea]. Disponible en: https://www.softwaretestinghelp.com/flask-app-with-blueprint-and-bootstrap/. Acceso: abril 2023

[22] ChartJS, «ChartJS». [En línea]. Disponible en: https://www.chartjs.org/docs/latest/. Acceso: abril 2023

[23] Embedded Linux, «Show Real Time Data From NodeMCU With Flask and SocketIO». [En línea]. Disponible en: https://adrianalin.gitlab.io/popsblog.me/posts/show-real-time-data-from-nodemcu-with-flask-and-socketio/

[24] Database of Databases, «InfluxDB». [En línea]. Disponible en: https://dbdb.io/db/influxdb/revisions/8. Acceso: abril 2023

[25] PyMyLifeUp, «Installing InfluxDB to the Raspberry Pi». [En línea]. Disponible en: https://pimylifeup.com/raspberry-pi-influxdb/. Acceso: abril 2023

[26] InfluxDB, «Getting Started with Python and InfluxDB». [En línea]. Disponible en: https://www.influxdata.com/blog/getting-started-python-influxdb/. Acceso: abril 2023

[27] Grafana Labs, «Grafana». [En línea]. Disponible en: https://grafana.com/. Acceso: mayo 2023

[28] Grafana Labs, «Install Grafana on Raspberry Pi». [En línea]. Disponible en: https://grafana.com/tutorials/install-grafana-on-raspberry-pi/. Acceso: abril 2023

[29] Python Docs, «Time access and conversions». [En línea]. Disponible en: https://docs.python.org/3/library/time.html. Acceso: mayo 2023

[30] CircuitPython, «Hardware accelerated external bus access». [En línea]. Disponible en: https://docs.circuitpython.org/en/latest/shared-bindings/busio/index.html. Acceso: mayo 2023

[31] NumPy, «NumPy». [En línea]. Disponible en: https://numpy.org/. Acceso: mayo 2023

[32] Grafana Labs, «How to change refresh-rate from 5s to 1s». [En línea]. Disponible en: https://community.grafana.com/t/how-to-change-refresh-rate-from-5s-to-1s/39008. Acceso: junio 2023

[33] Grafana Labs, «Data source management». [En línea]. Disponible en: https://grafana.com/docs/grafana/latest/administration/data-source-management/. Acceso: mayo 2023

[34] MiElectrónicaFácil, «Qué es y cómo funciona un Transistor». [En línea]. Disponible en: https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/ads1015-slash-ads1115. Acceso: abril 2023

[35] Analog Devices, «Precision, Single-Supply, SPST Analog Switches MAX323/MAX324/MAX325». [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX323-MAX325.pdf. Acceso: junio 2023. Acceso: julio 2023

[36] Mouser, «Raspberry Pi Sense HAT - Adafruit». [En línea]. Disponible en: https://www.mouser.es/new/adafruit/adafruit-raspberry-pi-sense-hat/. Acceso: abril 2023

[37] Linkedin, «Through hole vs. Surface mount solder fatigue». [En línea]. Disponible en: https://www.linkedin.com/pulse/through-hole-vs-surface-mount-solder-fatigue-gil-sharon/. Acceso: abril 2023

[38] JLCPCB, «JLCPCB». [En línea]. Disponible en: https://jlcpcb.com/. Acceso: junio 2023

[39] Farnell, «Farnell». [En línea]. Disponible en: https://es.farnell.com/. Acceso: agosto 2023

[40] Texas Instruments, «CD4066B CMOS Quad Bilateral Switch». [En línea]. Disponible en: https://www.ti.com/lit/ds/symlink/cd4066b.pdf. Acceso: julio 2023

[41] Analog Devices, «TTL Compatible CMOS Analog Switches». [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/DG300A-DG303A.pdf?ADICID=SYND_WW_P682800_PF-spglobal. Acceso: julio 2023

[42] Analog Devices, «Low-Voltage, High-Speed, Quad, SPST CMOS Analog Switches MAX4614/MAX4615/MAX4616». [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX323-MAX325.pdf. Acceso: julio 2023

[43] Adafruit, «How to convert Eagle PCBs to 3D Models in Fusion 360». [En línea]. Disponible en: https://learn.adafruit.com/how-to-convert-eagle-pcbs-to-3d-models-in-fusion-360?view=all. Acceso: junio 2023

[44] Analog Devices, «High-Speed, Low-Voltage, CMOS Analog Multiplexers/Switches». [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/AD620.pdf. Acceso: agosto 2023

[45] Microchip, «7/8-Bit Single/Dual SPI Digital POT with Non-Volatile Memory». [En línea]. Disponible en: https://ww1.microchip.com/downloads/en/DeviceDoc/22059b.pdf. Acceso: agosto 2023