

Sergio Gasquez Arcos

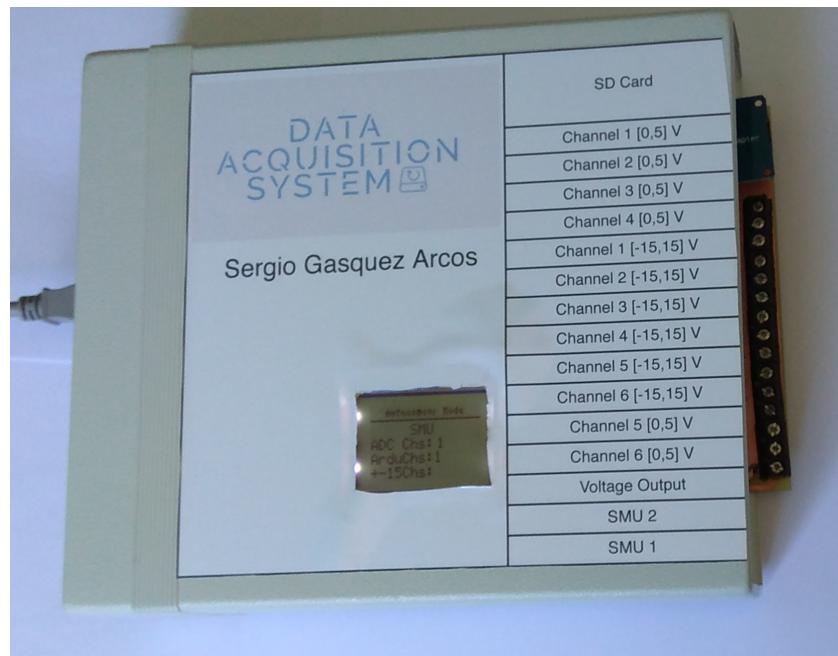
Desarrollo de una tarjeta de adquisición basada en Arduino



TRABAJO FIN DE GRADO
INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Director
Miguel Ángel Carvajal Rodríguez

Granada, Junio de 2018



Desarrollo de una tarjeta de adquisición basada en Arduino

Autor

Sergio Gasquez Arcos

Director

Miguel Ángel Carvajal Rodríguez

Resumen

En este trabajo fin de grado se pretende realizar un sistema de adquisición de datos basado en Arduino Mega 2560. El sistema dispondrá de 12 canales, 6 de los cuales serán del rango [0,5] V, mientras que el resto serán del rango [-15,15] V. La resolución será de 16 bits para 8 de los 12 canales y para los 4 restantes será de 10 bits. El sistema almacenará los datos en una tarjeta SD y los enviará, mediante GPRS, a un portal web desarrollado para visualizar en tiempo real los datos. El sistema es configurable, pudiendo elegir que canales se desean usar y cuales no. Contiene un reloj en tiempo real para almacenar la fecha y la hora en la que se realiza cada medida y una pantalla LCD para mostrar que tarea esta realizando el sistema en cada momento. También dispone de una SMU que permitirá medir intensidades. Además, el sistema será portátil y podrá trabajar de forma autónoma.

Palabras clave: Adquisición, convertidor analogico-digital, SMU, sensores, datos, hardware, DAQ, Arduino, acondicionamiento, firmware, GPRS, GSM.

Abstract

In this final degree project a Data Acquisition System based on Arduino Mega 2560 will be developed. The system has 12 channels, half of them of the range [0,5] V and the rest of the range of [-15,+15] V. The resolution, in most of the channels, is 16 bits, whereas in some channels, only 4 of 12, is 10 bits. The system will save the data in a SD card and send it, via GPRS, to a website developed to visualize the data in real time. The system can be configured to choose which channels will be on and off. It has a real time clock to keep track of the time and date when the measure was made, a LCD display to show which task the system is performing and a SMU to that will allow the system to measure currents. The system is completely portable and can function on its own in an autonomous way.

Keywords: Acquisition, analog-digital converter, SMU, sensors, data, hardware, DAQ, Arduino, conditioning, firmware, GPRS, GSM.

Yo, **Sergio Gasquez Arcos**, alumno de la titulación Grado en Ingeniería de Tecnologías de Telecomunicación. GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN. de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76438056-V, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Sergio Gasquez Arcos

Granada a 16 de Junio de 2018.

D. Miguel Ángel Carvajal Rodríguez, Profesor del Área de Electrónica y Tecnología de Computadores del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado **Desarrollo de una tarjeta de adquisición basada en Arduino**, ha sido realizado bajo su supervisión por **Sergio Gasquez Arcos**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 16 de Junio de 2018.

El director:

Miguel Ángel Carvajal Rodríguez

*Dedicado a aquellas personas que hacen del mundo un lugar
mejor.*

Índice general

1. Introducción	16
1.1. Introducción	16
1.2. Motivación	19
1.3. Objetivos	20
2. Estado del arte	22
2.1. Introducción histórica	22
2.2. Estado del arte	24
3. Descripción del sistema	26
3.1. Alimentación del sistema	27
3.1.1. Fuente de alimentación conmutada	28
3.1.2. Referencia de tensión	29
3.2. Acondicionamiento de señales	30
3.2.1. Amplificador Operacional	31

3.2.2. Acondicionamiento analógico de amplificación	35
3.3. Comutador analógico	41
3.3.1. Acondicionamiento analógico de reducción	43
3.4. Multiplexor 8 a 1	46
3.5. Source Measure Unit	47
3.6. Arduino Mega 2560	51
3.7. Convertidor Analógico-Digital	53
3.8. Convertidor Digital-Analógico	55
3.9. Pantalla LCD	57
3.10. Reloj en tiempo real	59
3.11. Módulo Tarjeta Micro SD	61
3.12. Módulo GSM/GPRS	62
4. Prototipos desarrollados del sistema	65
4.1. Prototipo V0	65
4.2. Prototipo V1	67
4.2.1. Esquemático general	69
4.2.2. Enrutado	70
4.2.3. Resultado	71
5. Firmware	73

5.1.	Entorno de programación	73
5.2.	Código del sistema	74
5.3.	Código de los módulos	79
5.3.1.	Código LCD	79
5.3.2.	Memoria EEPROM	81
5.3.3.	Configuración	81
5.3.4.	Multiplexor	83
5.3.5.	Medición de canales	83
5.3.6.	Reloj en tiempo real	84
5.3.7.	Módulo Micro SD	85
5.3.8.	Módulo GSM/GPRS	87
6.	Monitorización y visualización de los datos	89
7.	Especificaciones del sistema	102
8.	Resultados y conclusiones	104
8.1.	Resultados	104
8.1.1.	Canales	104
8.1.2.	Generador de señales	107
8.1.3.	SMU	109
8.2.	Conclusiones	112

8.3. Trabajo futuro	113
9. Apéndice	115
9.1. Presupuesto	116
9.2. Bill of Materials	117

Índice de figuras

1.1. Diagrama de un sistema de adquisición de datos	17
1.2. Resultado final del sistema de adquisición de datos	18
1.3. Diagrama de Gantt del proyecto	19
2.1. IBM 7700	23
2.2. IBM 1800	23
3.1. Diagrama general del proyecto	26
3.2. Disposición de los pines del MAX743	29
3.3. Esquemático del MAX743	29
3.4. Disposición de los pines del LM4040	30
3.5. Esquemático del LM4040	30
3.6. Diagrama de bloques del acondicionamiento analógico de reducción	31
3.7. Diagrama de bloques del acondicionamiento analógico de amplificación	31

3.8. Disposición de los pines del LM358	32
3.9. Amplificador operacional en configuración de seguidor de tensión	32
3.10. Amplificador operacional en configuración de sumador de tensión	33
3.11. Amplificador operacional en configuración de amplificador no inversor	34
3.12. Amplificador operacional en configuración de amplificador inversor	35
3.13. Esquemático del acondicionamiento analógico amplificador	36
3.14. Simulación del acondicionamiento analógico amplificador .	37
3.15. Etapa de amplificación ruido excesivo	38
3.16. Frecuencia del ruido en la etapa de amplificación	39
3.17. Etapa de amplificación tras el filtrado	40
3.18. Esquema del acondicionamiento analógico amplificador final	40
3.19. Caracterización del acondicionamiento analógico amplificador	41
3.20. Patillaje y diagrama funcional del DG419	42
3.21. Esquemático del acondicionamiento analógico reductor .	44
3.22. Simulación del acondicionamiento analógico reductor . .	45
3.23. Caracterización del acondicionamiento analógico reducción	46
3.24. Disposición de los pines del ADG408	47

3.25. Esquemático del ADG408	47
3.26. Esquemático de la SMU	48
3.27. Simulación de la SMU	49
3.28. Caracterización de la SMU	50
3.29. Esquemático de la SMU	51
3.30. Arduino Mega 2560	52
3.31. Disposición de los pines del ADS1115 de Adafruit	54
3.32. Esquemático del ADS1115 de Adafruit	54
3.33. Disposición de los pines del MCP4725 de Adafruit	56
3.34. Esquemático del MCP4725 de Adafruit	56
3.35. Caracterización del DAC	57
3.36. Disposición de los pines del Nokia5110	58
3.37. Disposición de los pines del DS3231 de Adafruit	60
3.38. Disposición de los pines del módulo Micro-SD	61
3.39. Shield SIM900	63
4.1. Prototipo V0 del sistema de adquisición de datos	66
4.2. Esquemático del ADC	68
4.3. Huella del ADC	68
4.4. Esquemático del módulo Micro SD	68
4.5. Huella del módulo Micro SD	68

4.6. Esquemático general de la PCB	69
4.7. Enrutado general de la PCB	70
4.8. Prototipo V1	72
5.1. Interfaz del entorno de desarrollo Arduino	74
5.2. Diagrama del código principal	76
5.3. Configuración del sistema a través del monitor Serie	82
6.1. Diagrama del flujo de datos entre las plataformas IoT.	90
6.2. Variables almacenadas en el canal 1	91
6.3. Variables almacenadas en el canal 2	91
6.4. Ejemplo de tablón en ThingSpeak.	92
6.5. Dispositivo de la configuración en Losant.	93
6.6. Workflow de la aplicación desarrollada en Losant.	94
6.7. Muestra de parte del código del bloque <i>Parse Fields</i>	96
6.8. Muestra de parte de una trama tras los bloques <i>funtion</i> . .	97
6.9. Resultado del <i>dashboard Configuration</i>	99
6.10. Resultado de parte del <i>dashboard</i> donde se muestran los valores de los canales	100
6.11. Resultado del <i>dashboard</i> donde se muestran los valores la SMU	101
6.12. Web desarrollada para la visualización de datos.	101

8.1.	Archivo generado en la SD para almacenar los resultados	105
8.2.	Resultados visualizados en el portal web.	106
8.3.	Resultado de la señal seno generada por el DAC	107
8.4.	Resultado de la señal seno generada por el DAC	108
8.5.	Resultado de la señal rampa generada por el DAC	108
8.6.	Resultado de la señal triangular generada por el DAC	109
8.7.	Gráfica I-V de una resistencia de 100 k Ω	110
8.8.	Gráfica I-V de un ZVP4424	111
8.9.	Gráfica I-V de un ZVP4424	112

Índice de tablas

3.1. Tabla de verdad del DG419	42
3.2. Caracterización de la SMU	49
3.3. Conexiones del ADS1115 de Adafruit	54
3.4. Conexiones del MCP4725 de Adafruit	56
3.5. Conexiones de la pantalla Nokia 5110	59
3.6. Conexiones del módulo de reloj en tiempo real DS3231 de Adafruit	61
3.7. Conexiones del módulo Micro SD	62
3.8. Conexiones del <i>shield</i> SIM900	64
7.1. Especificaciones del sistema de adquisición de datos	102
9.1. Presupuesto del proeycto	116
9.2. Bill of Materials	117

Capítulo 1

Introducción

1.1. Introducción

Todos los sistemas de procesamiento industrial, fabricas, maquinarias, vehículos, componentes hardware o software siguen las leyes de la física tal y como es entendida. En estos sistemas ocurren miles de fenómenos eléctricos y mecánicos que están continuamente sucediendo y variando. Frecuentemente se tiene interés por saber cual fue el estado de cierto sistema en un instante preciso o como progresó la magnitud de una variable en un sistema a lo largo del tiempo. En la mayoría de casos, las variables provienen de fenómenos naturales por lo que deben ser captadas por un sensor y posteriormente trasladadas al dominio digital para su adquisición.

El inmenso crecimiento de la demanda en la monitorización de datos ha supuesto nuevos retos a afrontar, de ahí la necesidad de crear dispositivos que sean capaces de captar y almacenar dichos datos, para posteriormente poder procesarlos y visualizarlos.

Un sistema de adquisición de datos [45] tiene como objetivo captar las medidas físicas o eléctricas que se deseen mediante sensores para

posteriormente almacenarlas o transmitirlas a un dispositivo para que este las almacene.



Figura 1.1: Diagrama de un sistema de adquisición de datos
Figura obtenida en: National Instrument, <http://www.ni.com/>

Dada la importancia de los sistemas de adquisición en la actualidad, se ha decidido desarrollar uno adaptado a las necesidades de la Universidad de Granada y a los proyectos en los que esta participando, por lo que algunas de las características del sistema serán: su bajo coste, su facilidad para su configuración y su fácil re-programación.

Se ha desarrollado una tarjeta de adquisición de datos basada en Arduino la cual admitirá hasta 12 entradas analógicas, de las cuales 6 serán del rango ± 15 V y otras 6 entre 0 V y 5 V. La resolución sera de 16 bits para 8 de las 16 entradas y para el resto serán 10 bits de resolución. Además tendrá la opción de usar un SMU (*Source Measuring Unit*) [40] para caracterizar dispositivos, así como se podrán generar señales del rango [-15,15] V. El sistema incluye una pantalla para poder saber la configuración del dispositivo y que tarea esta realizando en cada momento , un reloj en tiempo real, un modulo para almacenar las medidas en una tarjeta SD y un módulo de comunicación GSM/GPRS [24] [42]. Además el sistema será portátil y tendrá un modo de trabajo autónomo, es decir, sin conectar al PC en el cual podrá ser alimentado con baterías de forma que se podrá dejar

realizando las medidas deseadas en cualquier lugar, y se tendrán todos los datos disponibles.

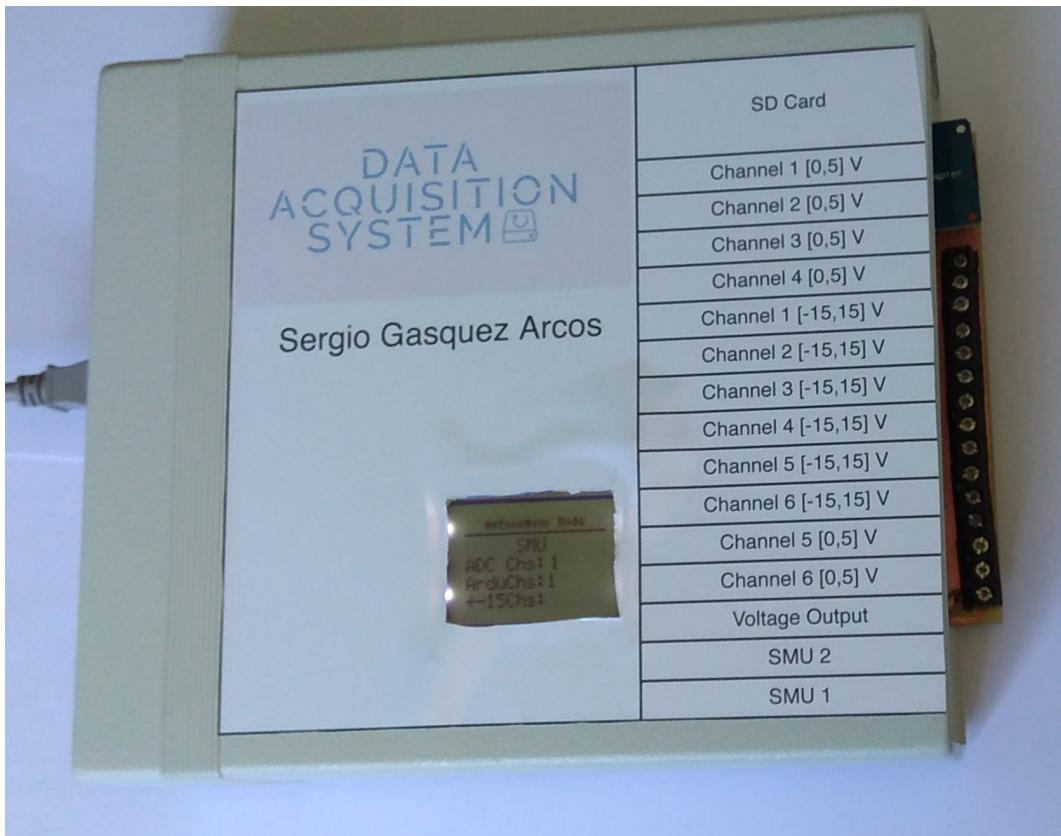


Figura 1.2: Resultado final del sistema de adquisición de datos

El proyecto se ha desarrollado usando, en medida de lo posible, *software* de código abierto por lo que se ha usado el entorno de programación de Arduino para el desarrollo del código y KiCad [7] para el diseño de la PCB (*Printed Circuit Board*). Mientras que para la simulación de circuitos se ha usado OrCad PSpice [29] ya que dispone de una versión gratuita para estudiantes. Además, se ha usado control de versiones con GitHub [12] en el que se pueden encontrar todos los cambios realizados en el proyecto así como esquemas, códigos y diseños. Dicho repositorio se puede encontrar aquí

Para mejorar la organización de objetivos y tareas se desarrolló un diagrama de Gantt [21] en el que se detalla el tiempo asignado a cada tarea:

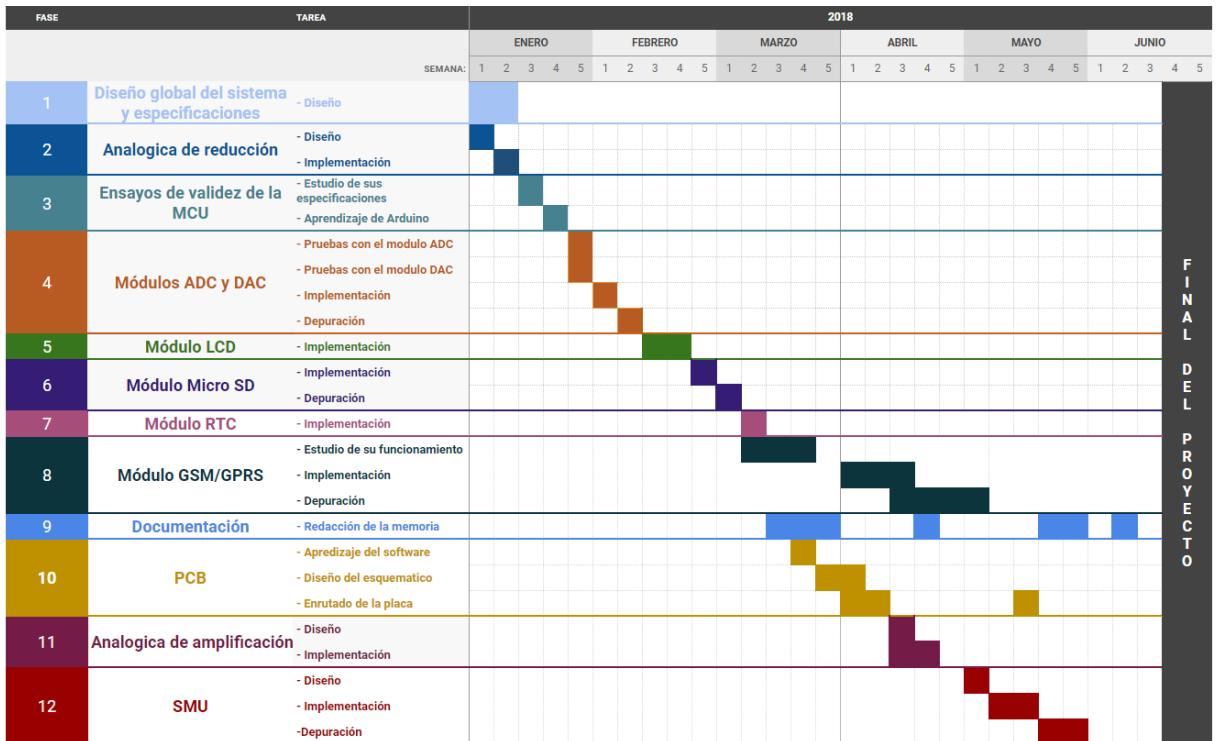


Figura 1.3: Diagrama de Gantt del proyecto

1.2. Motivación

El proyecto ofrece una gran variedad de desafíos y oportunidades para el desarrollo de un ingeniero de telecomunicaciones, ya que abarca diferentes ámbitos y los une. Desde el diseño de partes analógicas hasta la programación en C [30] de un Arduino [17] pasando por el diseño de PCB, comunicaciones a través de GPRS y muchos más retos.

En los últimos años se han comenzado a valorar cada vez más y más el valor de los datos ya que con ellos se pueden optimizar procesos, evitar situaciones no deseadas o simplemente visualizar estados. Por lo que el sistema de adquisición de datos que se desarrollará será una manera fácil, portátil y autónoma de adquirir datos para su posterior envío a la nube para su visualización y procesamiento.

Por ultimo, el proyecto se ha desarrollado para suplir las necesidades de la Universidad de Granada y se ha intentado que su coste sea lo mas bajo posible, ya que en el mercado existen gran variedad de sistemas de adquisición de datos, pero todos ellos comparten la características de tener un precio elevado. El sistema de adquisición de datos será sencillo, eficiente y muy barato.

1.3. Objetivos

El principal objetivo será desarrollar un sistema de adquisición de datos económico, funcional, portátil y con capacidad de enviar los datos a un portal. Se desarrollará un sistema con 12 entradas analógicas de dos rangos diferentes, de [0,5] V y de [-15,+15] V cada rango dispondrá de 6 entradas. Además, se le añadirá una SMU para tener la opción de medir intensidades y caracterizar dispositivos mediante sus curvas I/V.

Se desea conseguir que el sistema sea portátil y se pueda alimentar mediante una *power bank* ya que esto le aportaría gran versatilidad y atractivo debido a la facilidad que daría para medir variables en cualquier tipo de entorno sin la necesidad de tener fuente de alimentación externa o conexión a la red eléctrica. El consumo del sistema se intentará reducir, para que su vida autónoma sea mayor, durmiendo el microcontrolador siempre que sea posible.

Otro de los grande objetivos a conseguir del proyecto es añadirle un

módulo GSM/GPRS con el que poder enviar datos. Esto le aportaría una gran autenticidad al sistema ya que casi ninguno de los sistemas actuales posee tal característica.

Por lo que el sistema puede funcionar de diferentes modos:

- Conectado a un PC.
- De forma autónoma con conexión a la red.
- De forma autónoma sin conexión a la red.

En los casos en los que el sistema disponga de conexión a la red, el módulo GSM/GPRS permitirá visualizar los datos en tiempo real (se debe tener en cuenta que es necesaria que en la localización del dispositivo haya cobertura GPRS), mientras en los casos mas extremos en los que no se disponga de conexión a la red o de cobertura GPRS, los datos quedarán solamente almacenados en la tarjeta de memoria.

Capítulo 2

Estado del arte

2.1. Introducción histórica

El primer sistema de adquisición de datos se remonta a 1963, cuando IBM presentó un ordenador cuya especialidad era la adquisición de datos, denominado *IBM 7700 Data Acquisition System* [16]. Podía almacenar datos de hasta 32 fuentes diferentes simultáneamente y transmitir los resultados hasta 16 impresoras o monitores.

Fue sucedido un año después por, de nuevo, un ordenador desarrollado por IBM, el *IBM 1800 Data Acquisition and Control System* [16] que era una variante del *IBM 1130* y que presentaba mejoras significativas frente a su anterior.

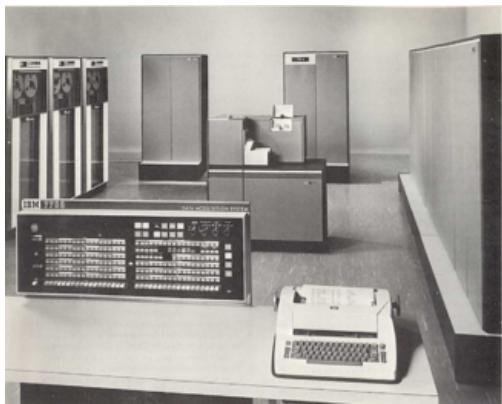


Figura 2.1: IBM 7700



Figura 2.2: IBM 1800

Figuras obtenidas en: IBM Archives, <http://www-03.ibm.com/ibm/history/>

El siguiente sistema en revolucionar el campo de los sistemas de adquisición de datos vino de la mano de *Tecmar/Scientific Solutions Inc* [33] y su producto *S-100* el cual era un ordenador que tenía una tarjeta de adquisición de datos.

Finalmente, en 1981 con la invención de los ordenadores comenzaron a surgir sistemas de adquisición por ordenador siendo algo más similares a los de hoy en día. A partir de esta fecha, se fueron consiguiendo avances hacia su portabilidad usando microcontroladores [36] [41] y FPGAs [5], que hacían posible que el sistema fuese portátil y mucho más eficiente.

Cabe destacar dos grandes hitos que permitieron grandes avances en los sistemas de adquisición: LabVIEW y Arduino. LabVIEW [47] es un software de ingeniería el cual permite, de forma visual, monitorizar y automatizar

labores, mientras que Arduino [4] es una familia de placas de portotipado las cuales incluyen un microprocesador, convertidores analógico-digital y otros componentes de gran utilidad.

Los últimos avances en el campo han sido relacionados con el almacenamiento y visualización de los datos procesados por el sistema [34], los cuales, desde la invención del *Internet of Things (IoT)* [31] [50] se pueden visualizar en cualquier lugar del mundo gracias a módulos GPRS, WiFi o Bluetooth [8]. Por eso, uno de los grandes objetivos es integrar un módulo GSM/GPRS en el sistema.

2.2. Estado del arte

En la actualidad, hay muchas empresas las cuales tienen desarrollados avanzados sistemas de adquisición de datos. Desde la adquisición de datos para sensores específicos a casos más generales. Marcas líderes en el sector como National Instruments, Agilent Technologies y Tektronix, entre otras, ofrecen gran variedad de sistemas de adquisición cuyas características se adaptan a las necesidades del consumidor. Algunos de ellos incluso incluyen un *software* para poder visualizar los datos en tiempo real.

Para adaptarse a las necesidades del consumidor, los sistemas de adquisición de datos se pueden clasificar en dos grandes grupos:

- **Sistemas DAQ (Data Acquisition)**[38]: Están conectados a un ordenador o a internet de forma que se puede procesar y visualizar los datos.
- **Tarjetas modulares** [1]: Se trata de módulos con funciones específicas que se pueden añadir a ordenadores para que estos desarrollen su función de medición y almacenamiento. La gran ventaja de estos es que aprovechan los recursos de almacenamiento y procesamiento

de los ordenadores.

Actualmente, hay gran variedad en el mercado de sistemas de adquisición de datos, por lo que las características entre ellos varían mucho. Se pueden encontrar sistemas que alcancen hasta los 100 kHz de frecuencia de muestreo [37] [20], 24 bits de resolución y más de 60 entradas analógicas [10].

Como ejemplo similar, se puede tomar el *USB-6002* de *National Instrument* del cual se pueden destacar las siguientes características:

- Número de canales monopolares: 8
- Número de canales bipolares: 8
- Resolución: 16 bits
- Frecuencia máxima de muestre: 50 kS/s
- Acepta entradas entre ± 10 V
- Tiene 2 salidas analógicas del rango ± 10 V y generadas por un DAC cuya resolución es de 16 bits.

El precio del *USB-6002* ronda los 430€ y no incluye ningún tipo de módulo para almacenar los datos ni para enviarlos.

Capítulo 3

Descripción del sistema

En este apartado se describirán todos los componentes usados, citando y describiendo sus principales características así como el motivo por el cual se han elegido dichos componentes en lugar de otros.

A continuación se podrá ver un diagrama general del proyecto, con cada uno de sus módulos principales:

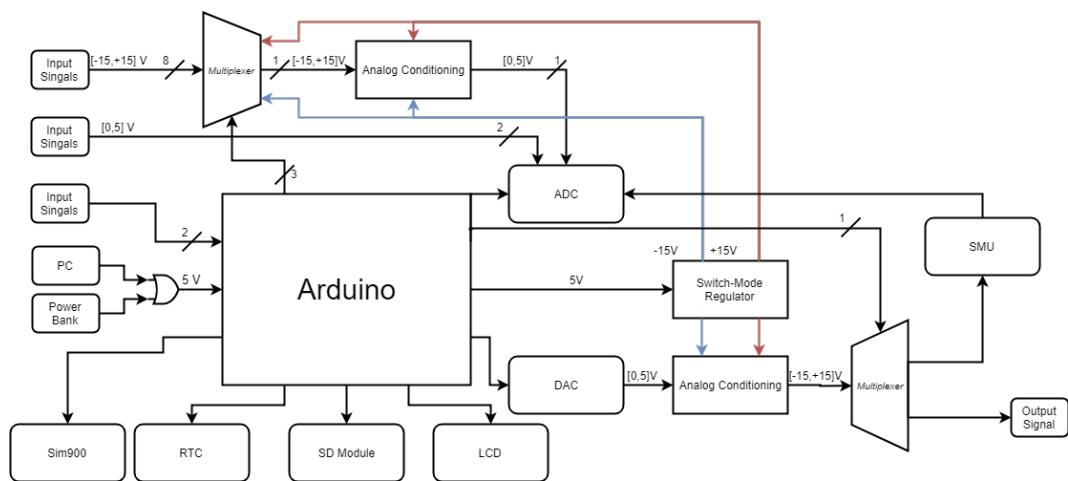


Figura 3.1: Diagrama general del proyecto

El sistema esta formado por 13 módulos interconectados y con una unidad de procesamiento, el Arduino Mega 2560. Habrá dos acondicionamientos analógicos, uno para reducir señales de [-15,+15] V a [0,5] V y un acondicionamiento analógico para realizar el proceso inverso, amplificar señales de [0,5] V a [-15,+15] V. También se dispone una SMU para tener la opción de medir corrientes. Además de un RTC para obtener la hora en tiempo real, una pantalla para visualizar los datos, un módulo con tarjeta SD para el almacenamiento de los datos y, para el envío de datos se usara un módulo GSM/GPRS el cual se basa en el SIM900.

3.1. Alimentación del sistema

El sistema se puede alimentar de varias formas, algunas de ellas incluso hacen que el sistema sea potable y autónomo. A continuación se explicarán las diferentes formas de alimentarlo.

Se podrá conectar vía USB tipo B, el cual puede estar conectado a un PC o a una batería portátil. El cable USB [51] esta formado por cuatro hilos, dos de alimentación (positivo y negativo) y 2 de datos, por lo que, estando conectado al PC se podrán visualizar los datos en tiempo real a través de la comunicación serie. El USB porpociona 5V y una corriente maxima de 500mA si es USB 2.0 y 900mA en caso de ser USB 3.0. Ambos serán suficientes para alimentar el sistema.

Otra de las maneras de alimentar el sistema es a través de un conector DC Barrel Jack hembra [46], el cual soporta tensiones desde 7 V a 12 V, ideal para conectarlo a la red eléctrica usando un transformador.

Por ultimo, también se podrá conectar el sistema usando una los terminales positivo y negativo de una batería cuyo voltaje de salida sea de entre 5 V y 12 V.

El módulo de GSM/GPRS se alimentará de forma independiente mediante un conector *Jack* a una tensión de 5 V.

A continuación, se describirá el convertidor de continua-continua (DC-DC) [52] que permite obtener ± 15 V a partir de los 5 V del USB lo cual permite que el sistema acepte entradas de ± 15 V y que la salida del convertidor Digital-Analógico sea en dicho rango. Posteriormente, se comentarán las características del LM4040 [28] el cual se usará como referencia de tensión.

3.1.1. Fuente de alimentación conmutada

El MAX743 es un convertidor DC-DC capaz de conseguir ± 15 V ó ± 12 V a partir de una entrada de 5V. Esta construido de forma compacta en un embalaje de 16 pines de orificio pasante. Se basa en la inducción de 2 terminales en lugar de en transformadores y consigue una precisión de $\pm 4\%$ en función de las condiciones en las que se encuentre (voltaje, temperatura, corriente de la carga...).

La eficiencia del convertidor esta en torno al 80 % en la mayor parte del rango de cargas, opera a 200 kHz lo que hace que el ruido que pueda producir sea fácil de filtrar.

El MAX743 proporciona ± 100 mA o ± 125 mA en función de si se encuentra en modo ± 15 V o ± 12 V respectivamente. También incluye un sistema de protección de apagado a altas temperaturas (el dispositivo opera entre los 0°C y los 70°C).

A continuación, se mostrará la disposición de los pines y esquemático del circuito usado:

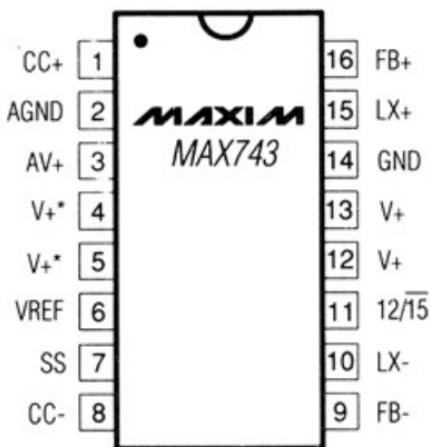


Figura 3.2: Disposición de los pines del MAX743

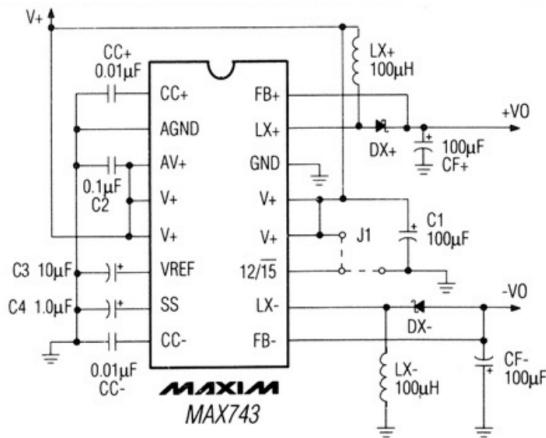


Figura 3.3: Esquemático del MAX743

Como anteriormente se ha comentado, se usará el modo de ± 15 V, por lo que la patilla 11 estará conectada a tierra. V_+ serán 5 V y $\pm V_0$ serán los ± 15 V.

3.1.2. Referencia de tensión

Como referencia de tensión se usara el LM4040, se trata de un elemento principalmente usado para dicha función, no requiere de condensadores (aunque si de un buffer) y acepta cargas capacitativas . Para el sistema se ha usado la versión que da una salida de 5 V. En su versión de orificio pasante, que será la usada, viene encapsulado en un TO-92 y opera entre los -40ºC y los 85ºC.

En la siguiente figura se mostrara su patillaje y el esquemático del circuito que se usará:

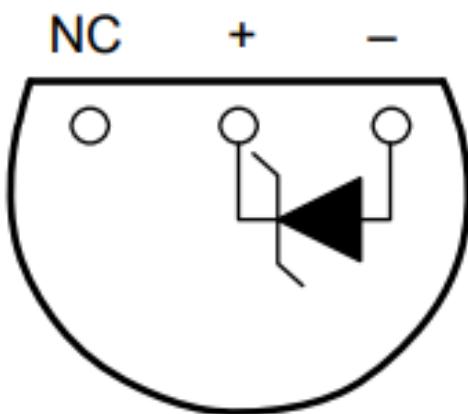


Figura 3.4: Disposición de los pines del LM4040

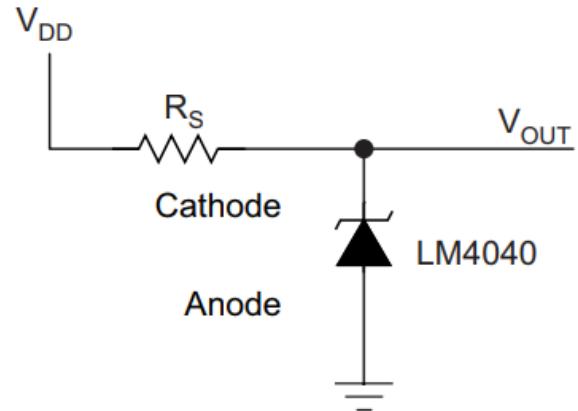


Figura 3.5: Esquemático del LM4040

V_{DD} serán 15 V y la resistencia R_S será de 22 k Ω ó 10 k Ω , para el uso que se le ha dado en el sistema. Con dicha configuración, se obtienen 5 V de manera estable gracias al LM4040.

3.2. Acondicionamiento de señales

En esta sección se comentarán los dos acondicionamientos analógicos realizados, uno se encargara de reducir señales de ± 15 V al rango de 0 V a 5 V, mientras que el otro hará lo opuesto, pasar de [0,5] V a [-15,+15] V. El diagrama de bloques de ambos será:

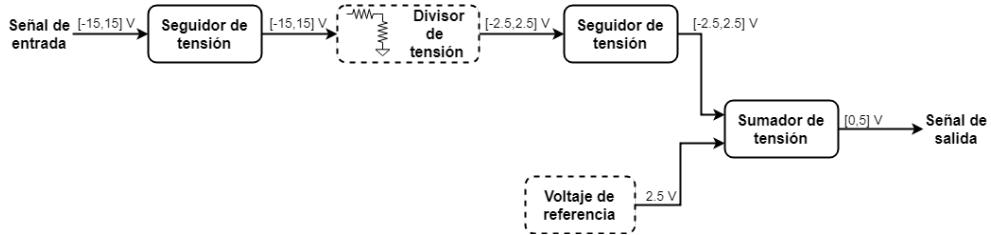


Figura 3.6: Diagrama de bloques del acondicionamiento analógico de reducción

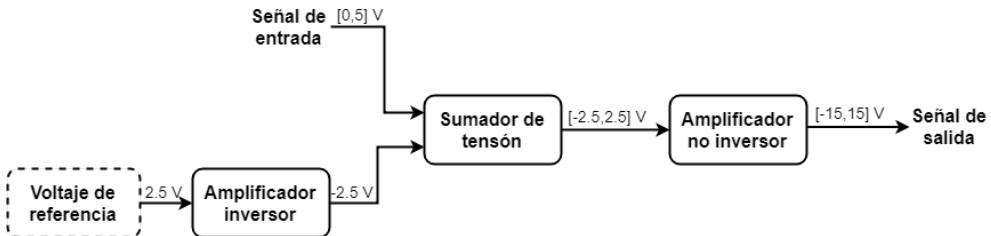


Figura 3.7: Diagrama de bloques del acondicionamiento analógico de amplificación

Ambos siguen una estructura similar basada en amplificadores operacionales [26] (LM358 [27]), el cual se explicará con detalle a continuación, y usando el LM4040 en la configuración explicada anteriormente en el apartado 3.1.2.

3.2.1. Amplificador Operacional

Todos los amplificadores operacionales usados en el sistema son LM358, debido a su relación calidad/precio. Se trata de un amplificador operacional doble, de bajo consumo que permite alimentación monopolar o bipolar, en la versión usada viene embalado en un DIP8. Se puede alimentar entre 3 V y 32 V en caso de alimentación simple y entre ± 1.5 V y ± 16 V en caso de alimentación doble además, incluye compensación de temperatura.

A continuación se mostrara un diagrama general y su disposición de pines:

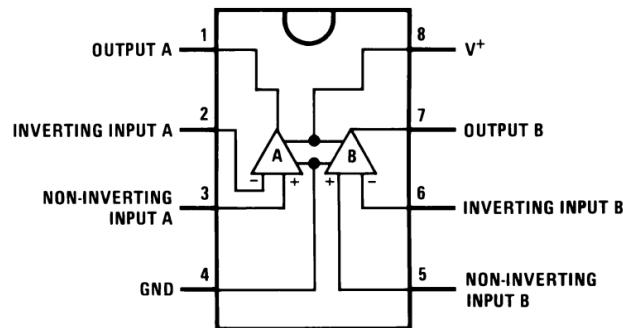


Figura 3.8: Disposición de los pines del LM358

En los siguientes párrafos se detallarán cada una de las diferentes configuraciones [23] usadas:

Configuración de seguidor de tensión

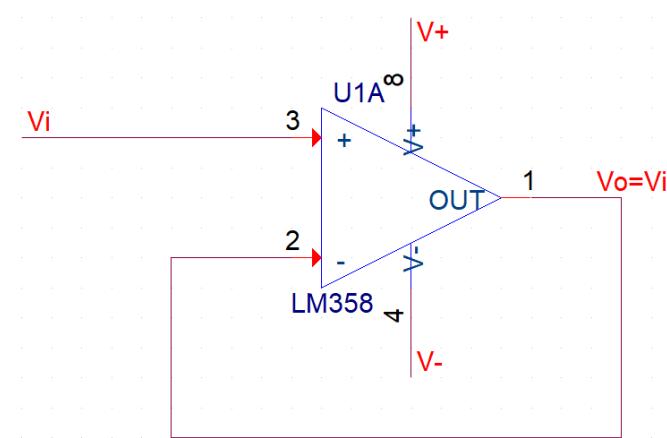


Figura 3.9: Amplificador operacional en configuración de seguidor de tensión

El principal uso de esta configuración es para aislar ambos extremos, ya que por la entrada no inversora del operacional la corriente es 0 A pero se sigue consigue transportar el voltaje al otro extremo. Es decir, se tendría una ganancia de 1.

$$V_o = V_i \quad (3.1)$$

Configuración de sumador de tensión

La configuración de sumador de tensión produce una señal invertida cuyo valor en tensión es proporcional a la suma de ambos voltajes que se desean sumar (se pueden sumar mas de 2 señales)

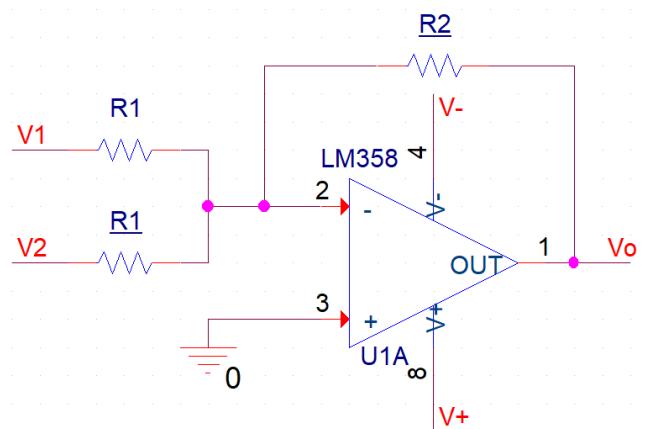


Figura 3.10: Amplificador operacional en configuración de sumador de tensión

En este caso, la salida será la siguiente:

$$V_o = -\frac{R2}{R1} \cdot (V1 + V2) \quad (3.2)$$

Configuración de amplificador no inversor

Esta configuración permite amplificar la señal de entrada sin invertirla, de la siguiente manera:

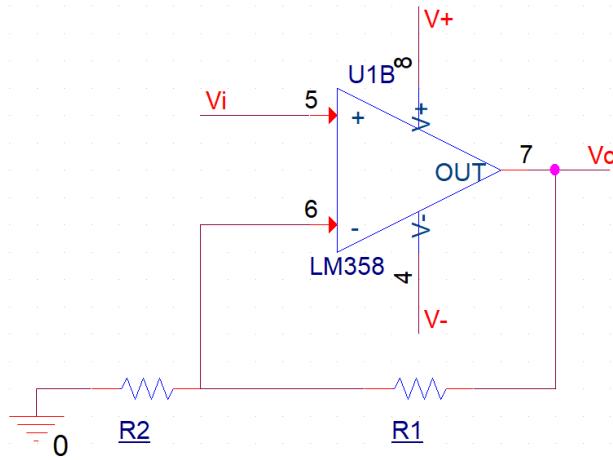


Figura 3.11: Amplificador operacional en configuración de amplificador no inversor

$$\frac{V_o}{V_i} = 1 + \frac{R_1}{R_2} \quad (3.3)$$

Configuración de amplificador inversor

Esta configuración es muy similar a la anterior (Sección 3.2.1), con la diferencia que la salida de esta si estará invertida y el ratio de amplificación varia:

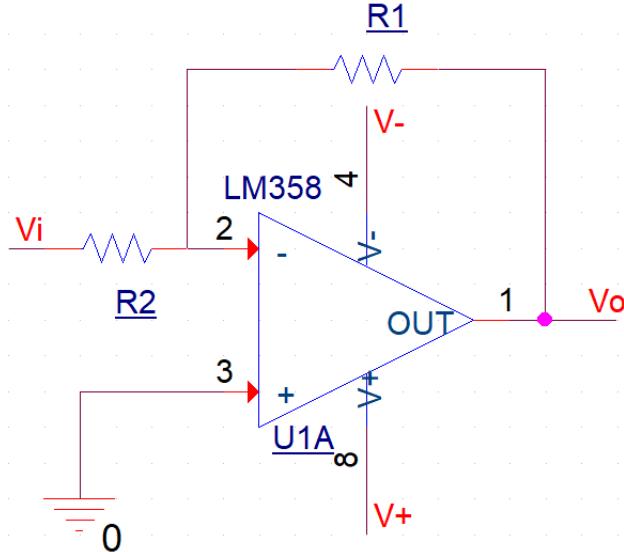


Figura 3.12: Amplificador operacional en configuración de amplificador inversor

$$\frac{V_o}{V_i} = -\frac{R_1}{R_2} \quad (3.4)$$

3.2.2. Acondicionamiento analógico de amplificación

Esta analógica tiene el objetivo conseguir [-15,+15] V a partir de señales del rango [0,5] . La amplificación se basará en amplificadores operacionales, de los cuales se usarán 3: Uno en configuración inversora, uno en configuración sumador y el ultimo en configuración de amplificador. Se usará el amplificador operacional en inversora para convertir en -5 V los 5 V obtenidos con la configuración del LM4040 citada en Fig.3.5, la señal generada será sumada a la señal que se desea amplificar en el segundo amplificador operacional, obteniendo así una señal del rango [-2.5,+2.5] V que posteriormente será amplificada en un factor 6 en el tercer operacional.

Se han realizado simulaciones, de nuevo con el software OrCAD PSpice. Para la simulación, se han asumido algunas simplificaciones:

- No se usará referencia de tensión, se usara una fuente de tensión continua de -5 V.
- La alimentación de los amplificadores será con fuentes de tensión continua en lugar de con la configuración del MAX743 citada en Fig.3.3 .
- No se pondrán condensadores a en la alimentación de los operacionales.

Teniendo en cuenta estas tres simplificaciones, el esquemático resultante es:

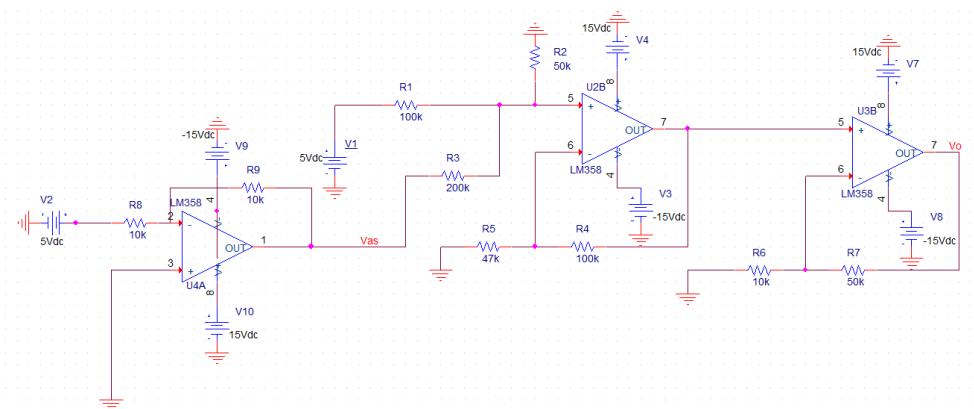


Figura 3.13: Esquemático del acondicionamiento analógico amplificador

Para la simulación, se variará la fuente de tensión V1 y se representará tanto la tensión en V1 como la tensión de salida Vo:

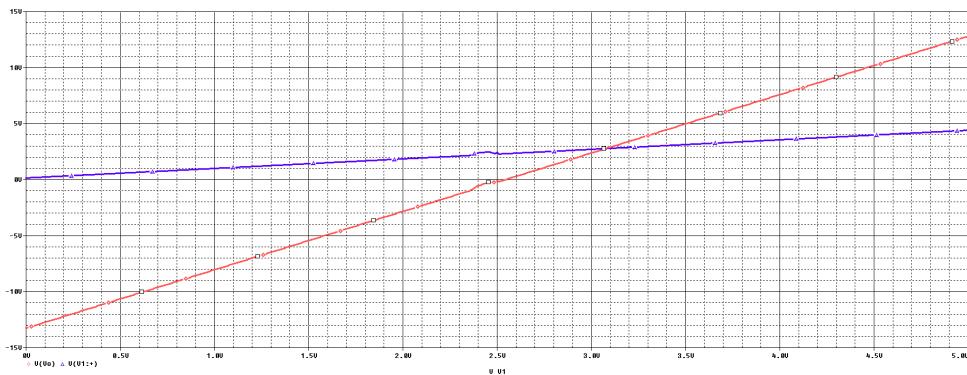


Figura 3.14: Simulación del acondicionamiento analógico amplificador

Se puede comprobar que se cumple el objetivo, ya que mientras que la fuente V1 hace un barrido de tensión de 0 V a 5 V la salida varía entre -15 V y +15 V aproximadamente.

Se realizaron pruebas en el laboratorio para comprobar que la etapa realizaba su cometido, y mediante el uso del osciloscopio se detectó un ruido excesivo a la salida de esta etapa.

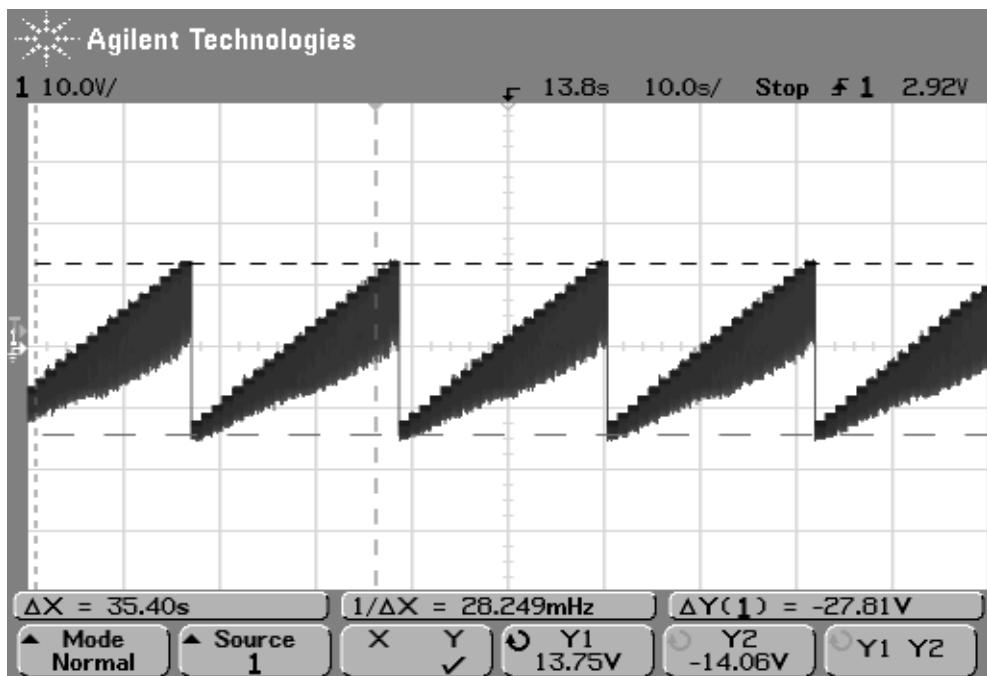


Figura 3.15: Etapa de amplificación ruido excesivo

Este ruido haría que todas las señales generadas por el DAC y posteriormente amplificadas en esta etapa fuesen de muy poca utilidad, por lo que se buscó la frecuencia del ruido para posteriormente lidiar con el:



Figura 3.16: Frecuencia del ruido en la etapa de amplificación

Se obtuvo su frecuencia y se procedió a realizar un filtrado RC [39] cuya frecuencia de corte esta en torno a 6 Hz (más de dos décadas por debajo) para reducir dicho ruido. Tras el filtrado, la señal es mucho más nítida:

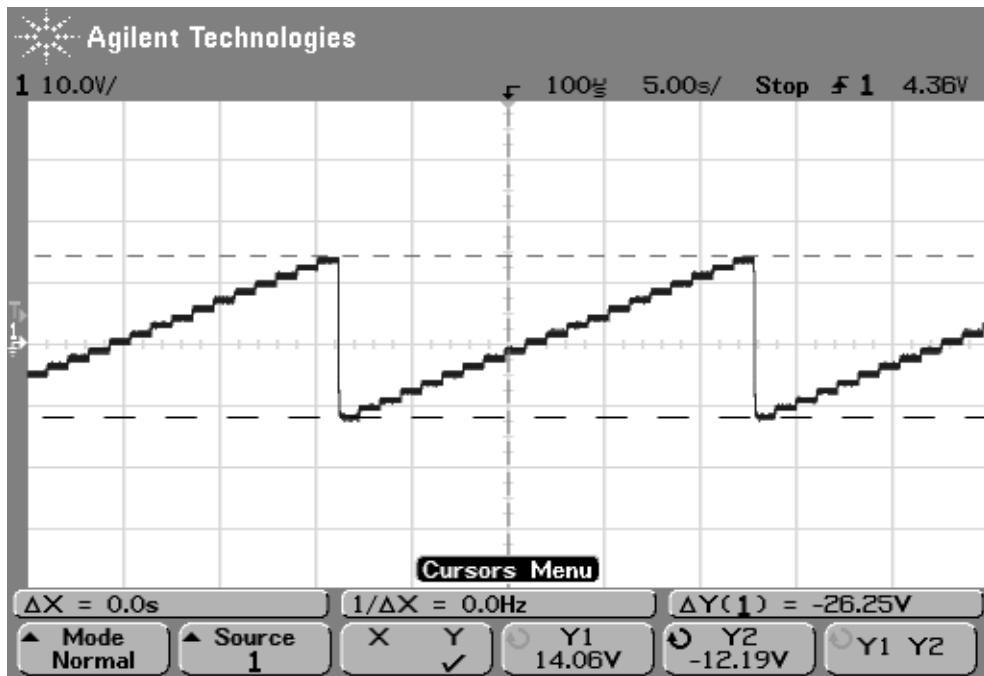


Figura 3.17: Etapa de amplificación tras el filtrado

Por lo que el esquema tras la introducción del filtro es así:

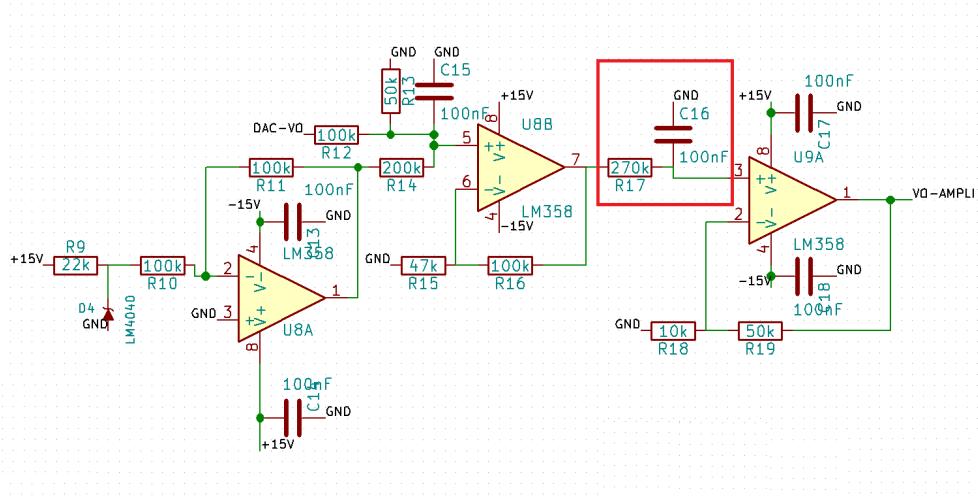


Figura 3.18: Esquema del acondicionamiento analógico amplificador final

El filtro introducido se encuentra en la zona remarcada.

Con esta configuración se midieron 20 valores para comprobar que su funcionamiento era el deseado, a continuación se muestra una gráfica de los valores experimentales obtenidos.

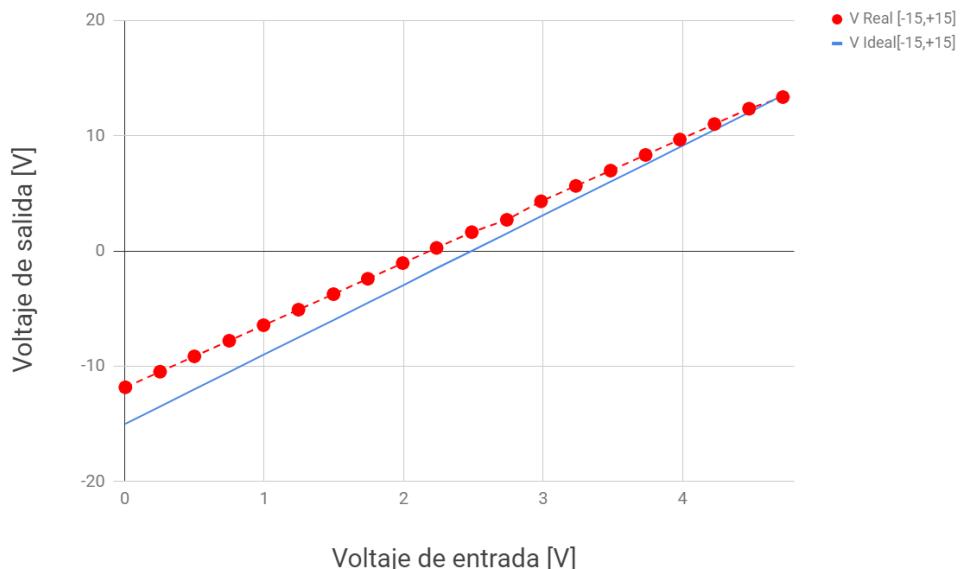


Figura 3.19: Caracterización del acondicionamiento analógico amplificador

Se puede observar que tanto a -15 V como a +15 V hay mayor diferencia entre la medida y el valor ideal, esto se debe a usar amplificadores operacional los cuales no son *rail-to-rail*.

3.3. Conmutador analógico

Para conmutar entre la salida por terminal de tornillo de la señal amplificada y el uso de la SMU se usará el DG419, un conmutador analógico [44]. Admite alimentación monopolar o bipolar, cuando se alimenta de forma monopolar admite señales de +10 V hasta +30 V mientras que cuando lo

hace de forma bipolar admite señales de entre ± 4 V y ± 20 V. El conmutador esta encapsulado en un DIP8. A continuación se detallarán los pines con su diagrama funcional.

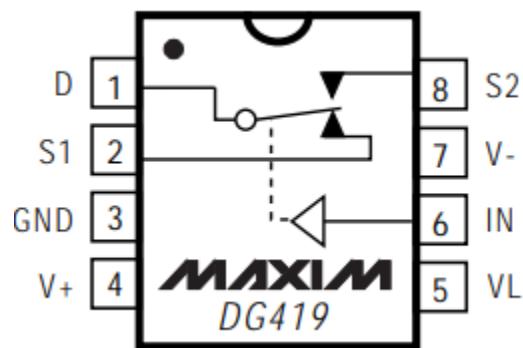


Figura 3.20: Patillaje y diagrama funcional del DG419

Su tabla de verdad se podría resumir en:

Entrada	Switch 1	Switch 2
0	ON	OFF
1	OFF	ON

Tabla 3.1: Tabla de verdad del DG419

Como se puede observar pose un conmutador el cual esta por defecto cerrado y otro que esta por defecto abierto, y se conmutará usando la señal de entrada, en el caso del sistema, señal de entrada será generada mediante un pin del Arduino. A continuación se detallarán los pines:

- **D:** Conexión común entre los dos *switches*, para el sistema desarrollado se conectaría la salida de la analógica de amplificación.
- **S1:** Se trata del conmutador que esta cerrado por defecto. Se conectaría a la entrada del SMU.

- **S2:** Se trata del conmutador que esta normalmente abierto. Se conectará a un terminal de tornillo, para poder generar señales de ± 15 V.
- **GND:** Tierra del dispositivo.
- V_+ : Estará conectado a +15 V y es la alimentación positiva del dispositivo.
- V_- : Estará conectado a -15 V y es la alimentación negativa del dispositivo.
- **VL:** Se trata del *Voltage Level* , el cual indica que nivel de voltaje será considerado como un 1 para cambiar entre estados según su tabla de verdad comentada anteriormente (Tab.3.3).
- **IN:** Es la señal de entrada que seleccionara que salida se desea y estará controlada por el pin 31 del Arduino.

3.3.1. Acondicionamiento analógico de reducción

Este módulo de reducción será encargado de transformar señales del rango de ± 15 V a señales que vayan entre 0 V y 5 V, esta formado por 3 amplificadores operacionales: Dos en configuración de seguidor de tensión y el ultimo en configuración de sumador.

Se usan los seguidores de tensión para no absorber corriente de la tensión a medir, de manera que se tendrán aislados los dos extremos. Entre el primer y el segundo amplificador operacional se usará un simple divisor de tensión para reducir la señal de entrada en torno a 1/6 de su tensión inicial, por lo que se obtendría una señal cuyo rango estaría entre [-2.5,+2.5] V, para obtener una señal de salida de [0,5] V se usa un operacional en configuración de sumador. Se sumará dicha señal a una generada con la

referencia de tensión generada con el LM4040 con la configuración de la Fig.3.5

Se simulará dicho circuito mediante el software OrCAD PSpice teniendo en cuenta algunas simplificaciones a la hora de la simulación: No se usaran condensadores en la alimentación (tanto positiva como negativa) de los amplificadores operacionales, la alimentación de los amplificadores se hará con fuentes de tensión continua en lugar de con un MAX743 como en el sistema y en lugar de usar una tensión de referencia, se usara una fuente de alimentación continua al voltaje deseado. He aquí el esquemático resultante y la simulación:

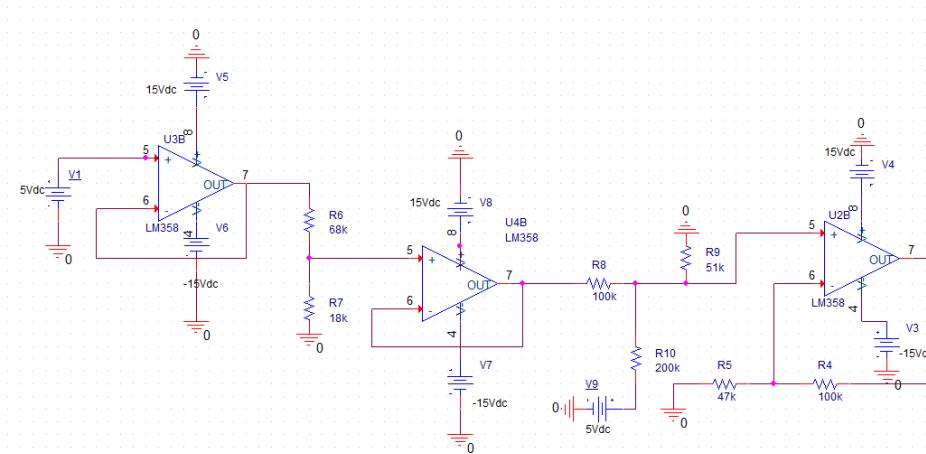


Figura 3.21: Esquemático del acondicionamiento analógico reductor

Se hará variar la fuente de alimentación V1 y en la simulación se representaran las tensiones de V1 y Vo.

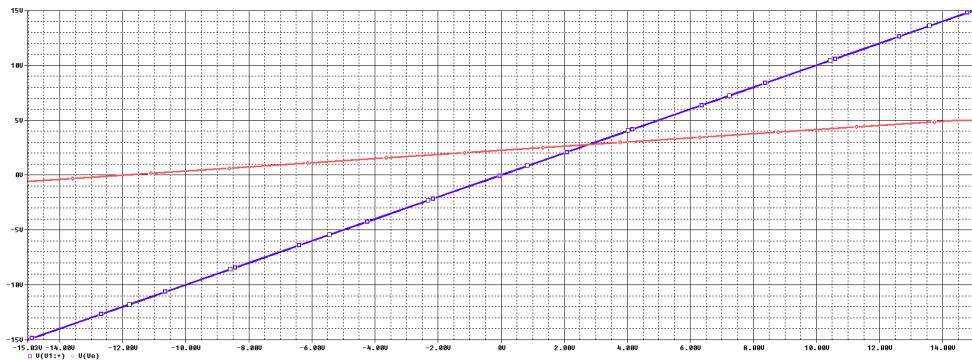


Figura 3.22: Simulación del acondicionamiento analógico reductor

Se puede observar que la salida es aproximadamente la deseada, convierte señales del rango [-15,+15] V a [0,5] V.

Esta analógica permitirá que el sistema mida elementos cuya entrada sea del rango [-15,+15] V dándole aun mayor versatilidad y viabilidad.

Para comprobar el correcto funcionamiento de la etapa montada en el prototipo, se han tomado 30 valores en el rango de [-15,15] V para observar y comparar los resultados obtenidos con los de la simulación realizada anteriormente:

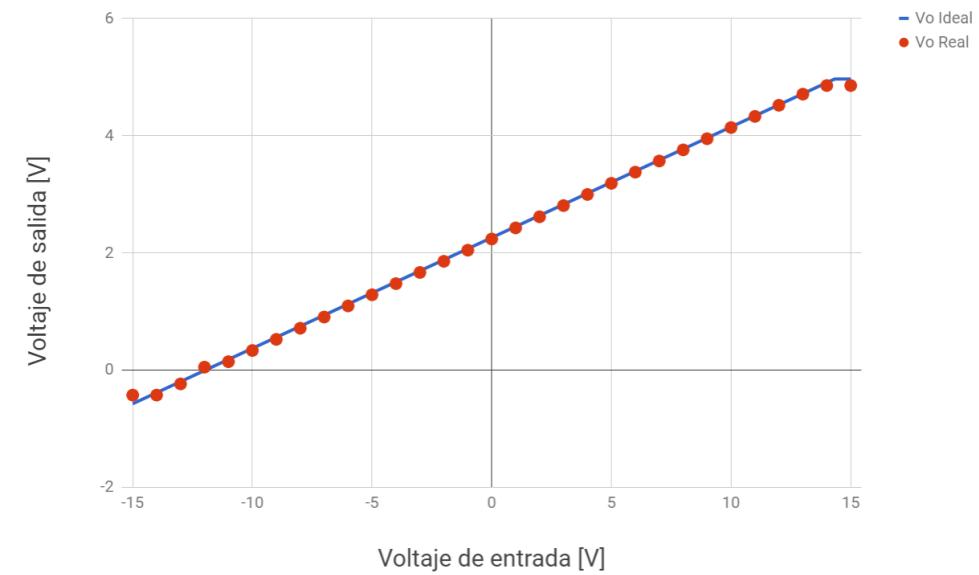


Figura 3.23: Caracterización del acondicionamiento analógico reducción

Se puede observar el efecto de no usar amplificadores operacionales *rail-to-rail* en ambos extremos. Salvo por esta deficiencia conocida, la etapa funcional tal y como se esperaba.

3.4. Multiplexor 8 a 1

Para ampliar el numero de señales a medir del rango [-15,15] V se usará un multiplexor cuyos canales de dirección serán controlados mediante el Arduino. Se trata de un multiplexor de 8 a 1, es decir, ocho entradas y una salida, la cual sera seleccionada a través de tres bit de dirección. Su consumo máximo es de $75 \mu\text{A}$ y se alimenta en el rango de la señal analógica que vaya a procesar, en el caso del sistema se alimentará con ± 15 V. También dispone de un pin el cual habilitará o deshabilitará el multiplexor por completo en función de su estado. Para la versión usada en el sistema, vendrá encapsulado en un DIP16. A continuación se mostrará

el diagrama general y la disposición de los pines:

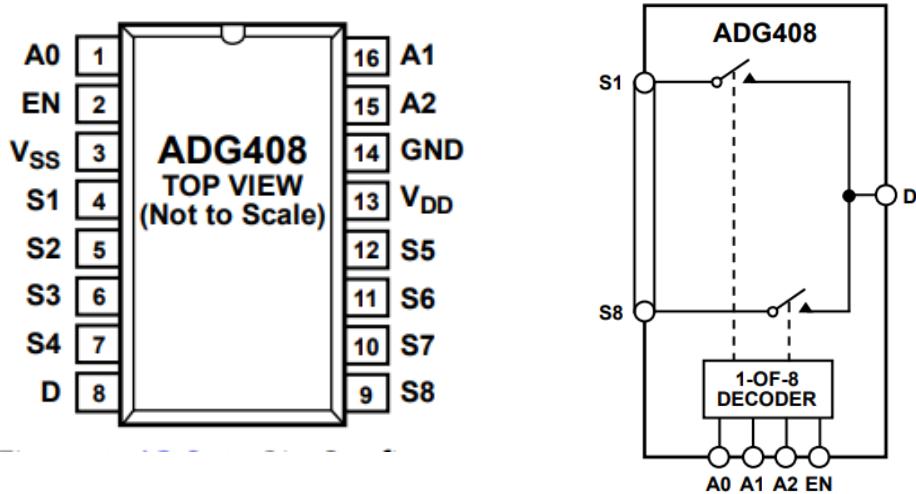


Figura 3.24: Disposición de los pines del ADG408

Figura 3.25: Esquemático del ADG408

La principal función del multiplexor en el sistema será tener la opción de medir 6 entradas en el rango [-15,15] V en lugar de solo una, por lo que las 6 entradas estarán conectadas a terminales de tornillo visibles en el exterior y la salida pasará por la analógica de reducción mencionada en la Sección 3.3.1 para finalmente ser medida por el ADC.

3.5. Source Measure Unit

La SMU permitirá medir corrientes, dándole aun mayor versatilidad al proyecto. Esta medición se hará de forma indirecta, es decir, se medirán tensiones mediante una entrada analógica al convertidor analógico-digital, que será detallado posteriormente, y se realizará una transformación conocida para obtener la corriente.

Para el diseño de la *Source Measure Unit* se ha usado un amplificador operacional, de nuevo, el LM358 así como una referencia de tensión gene-

rada con un LM4040 en la configuración mostrada en Fig.3.5 y un divisor de tensión. Tras su diseño se realizó una simulación usando el *software* OrCAD PSpice para la cual se usó el siguiente esquemático:

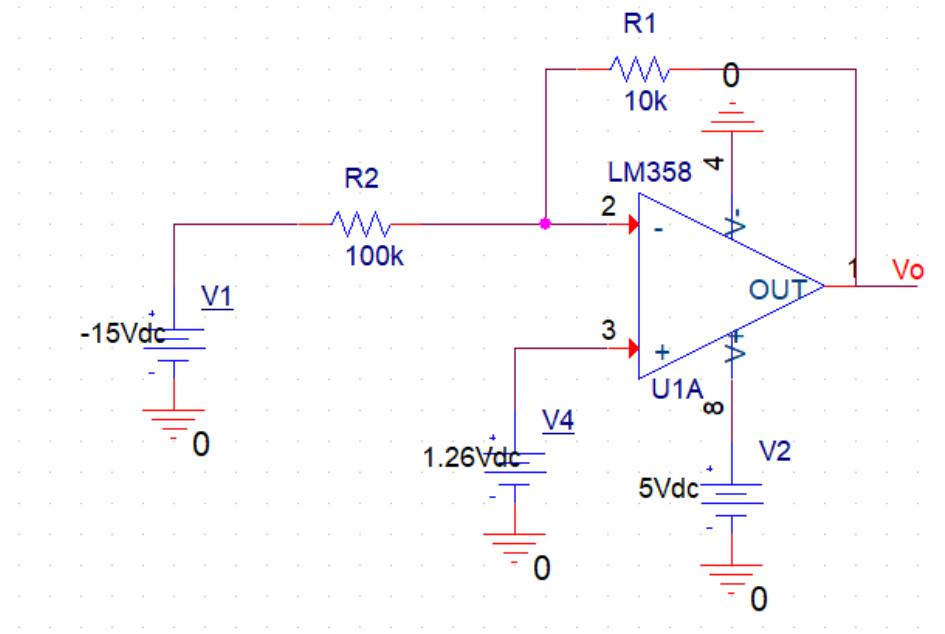


Figura 3.26: Esquemático de la SMU

Para la simulación se ha usado una fuente de tensión continua en lugar de el LM4040 y un divisor de tensión. También, para la simulación se ha usado una resistencia como elemento a estudiar, R2, pero en el sistema, los dos terminales a los que esta conectados R2 estarán conectados a terminales de tornillo para poder conectar con facilidad el elemento que se desee.

La razón por la que se ha puesto la entrada no inversora a 1.26 V es para no tener la necesidad de usar un amplificador operacional *rail-to-rail*. A continuación, se mostrará una simulación en la que se hace variar la fuente V1 entre -15 V y 15 V:

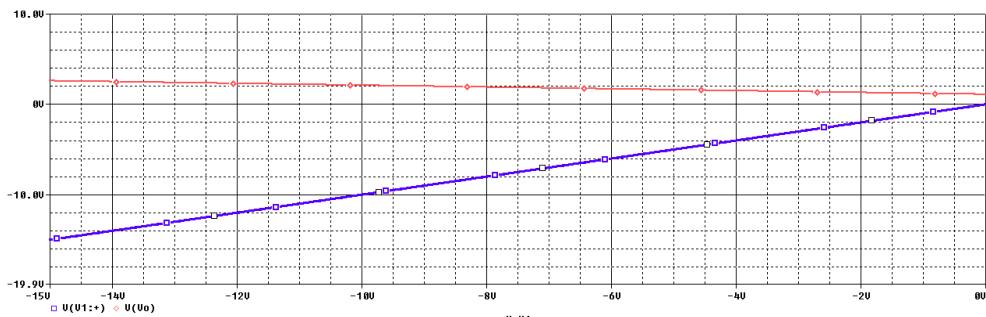


Figura 3.27: Simulación de la SMU

Para comprobar su correcto funcionamiento, se realizó una caracterización en el laboratorio. En la que se fijando una tensión mediante el convertidor digital-analógico, el cual se comentará en un futuro, posteriormente se hacia pasar por la analógica de amplificación detallada en la Sec.3.2.2 y estos fueron los valores obtenidos comparados con los obtenidos en las simulaciones.

Muestra	Código del DAC	Salida analógica de amplificación (V)	Salida SMU (V)
0	0	-13.08	2.6611875
1	205	-11.71	2.6191875
2	410	-10.37	2.48475
3	615	-9.01	2.350125
4	820	-7.66	2.21025
5	1025	-6.3	2.072625
6	1230	-4.93	1.9340625
7	1435	-3.59	1.79325
8	1640	-2.26	1.6640625
9	1845	-0.96	1.531875

Tabla 3.2: Caracterización de la SMU

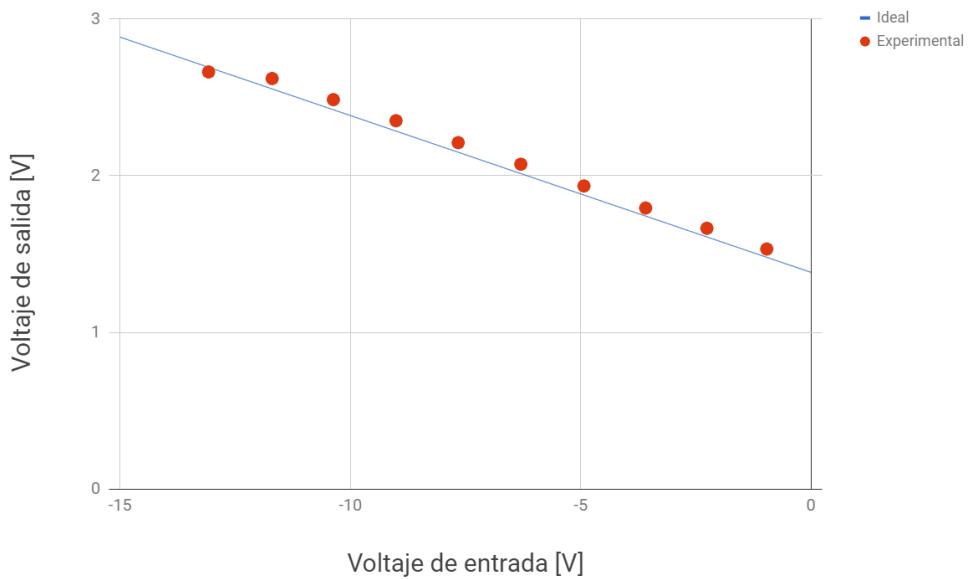


Figura 3.28: Caracterización de la SMU

Se puede apreciar su gran similitud, por lo que el bloque de la SMU queda validado experimentalmente. Para su desarrollo físico, este fue el circuito realizado:

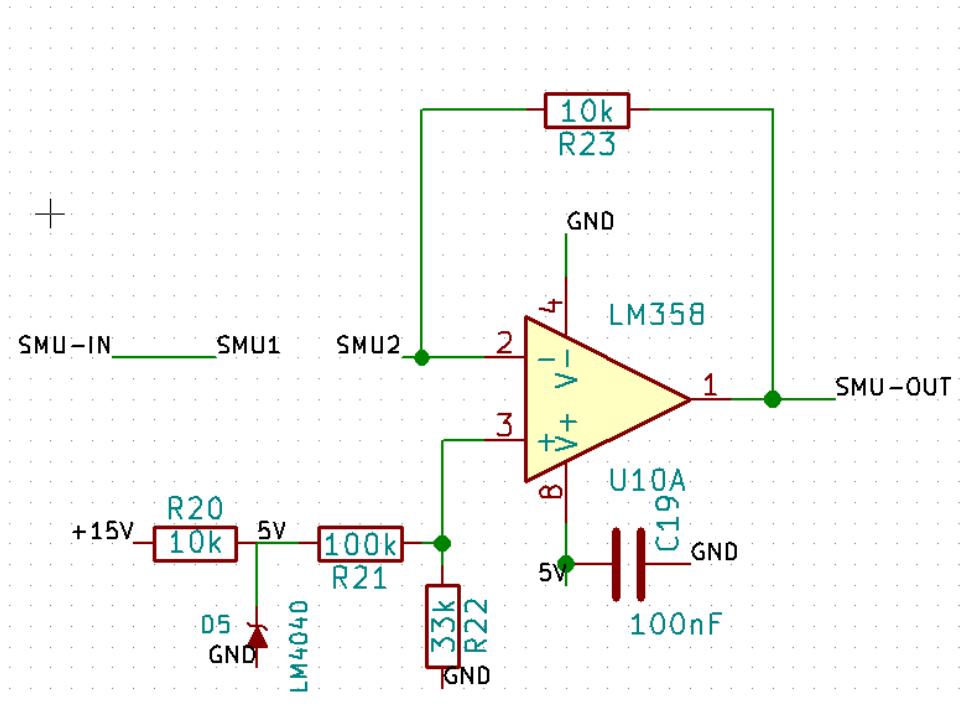


Figura 3.29: Esquemático de la SMU

SMU1 y SMU2 son salidas de un terminal de tornillo y al igual que para su simulación, entre SMU1 y SMU2 se conectó, para su validación, una resistencia de $100\text{ k}\Omega$.

3.6. Arduino Mega 2560

Arduino Mega 2560 es un microcontrolador cuyo procesador es el ATmega2560. Pertenece a la familia de controladores de Arduino y fue diseñado en 2010.

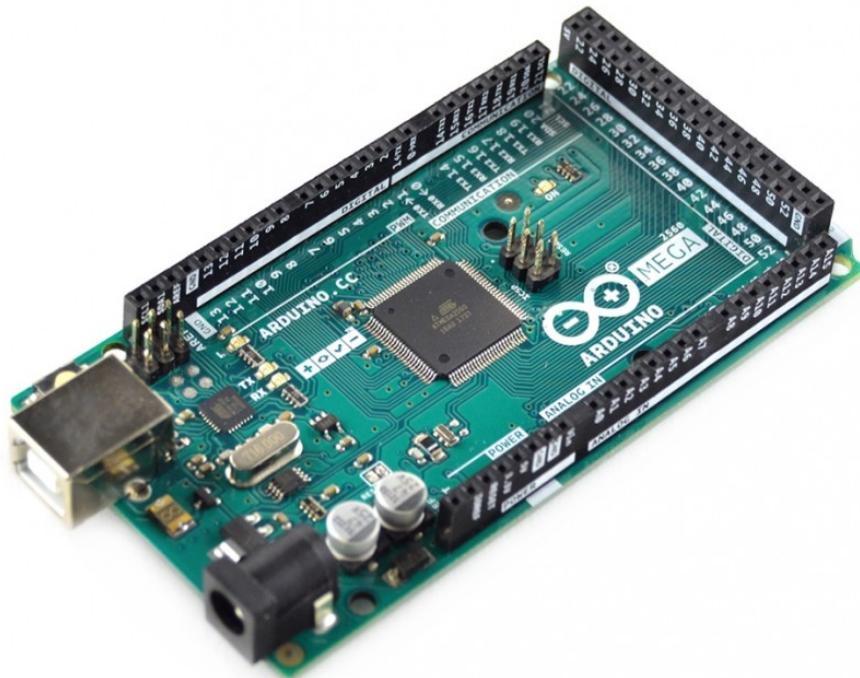


Figura 3.30: Arduino Mega 2560

Como principales características del Arduino Mega 2560 cabe destacar:

- 54 pines digitales de entrada o salida de los cuales, 15 son capaces de generar señales PWM.
- 16 entradas analógicas.
- Un cristal oscilador de 16 MHz.
- Conector USB tipo B.

- Conector tipo jack, a través del cual puede ser alimentado entre 7 y 12 V
- Cabecera ICSP.
- Botón de reinicio.
- 256 KB de memoria Flash
- 8 KB de SRAM
- 4 KB de EEPROM

Arduino es una compañía que apoya completamente el hardware y software abierto, por lo que tiene de forma pública todos los esquemáticos en su página web.

Para programar el Arduino Mega 2560 se usará la IDE de Arduino [3], la cual se puede obtener de forma gratuita en la web de Arduino.

3.7. Convertidor Analógico-Digital

El ADS1115 es un convertidor Analógico-Digital [32] de bajo consumo y de 16 bits de precisión, se comunica a través de I²C [43] y dispone de 4 canales que pueden ser configurados también en modo diferencial o comparador.

Como se mencionó en la Sección 3.6, el Arduino Mega dispone de 16 entradas analógicas conectadas al convertidor analógico propio del Arduino, cuya resolución es 10 bits. La razón de usar un convertidor analógico externo es la mejora de resolución.

La resolución efectiva es de 15 bits ya que la salida del convertidor es de tipo *signed integer* por lo que el primer bit designará el signo del número,

Por lo que habrá 32768 posibles valores, sabiendo que su fondo de escala es de 6.144V , se puede obtener el incremento en voltaje entre dos valores consecutivos, el cual es 0.1875 mV. El convertidor ADC interno del Arduino Mega tiene una resolución de aproximadamente 5 mV.

En la siguiente figura se puede ver el diagrama general del ADC y su patillaje:

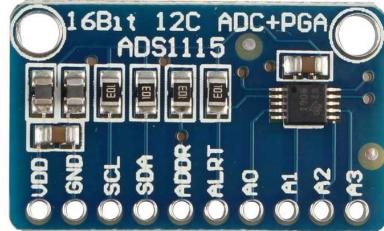


Figura 3.31: Disposición de los pines del ADS1115 de Adafruit

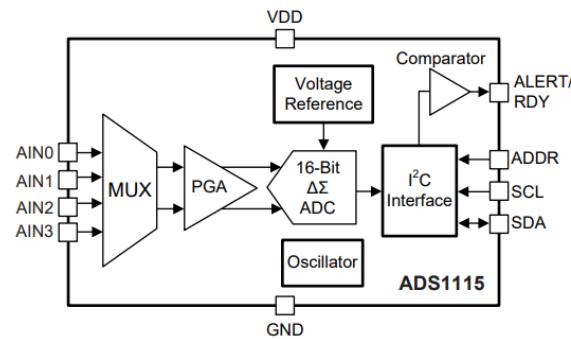


Figura 3.32: Esquemático del ADS1115 de Adafruit

Las conexiones que se realizaran serán las siguientes:

ADS1115	Arduino Mega 2560
VDD	5 V
GND	GND
SCL	SCL
SDA	SDA
ADDR	Sin conectar
ALRT	Sin conectar

Tabla 3.3: Conexiones del ADS1115 de Adafruit

Los pines SCL y SDA son los que permiten la interfaz I²C entre el Arduino y el ADC. Los pines A0,A1,A2 y A3 son los 4 canales del ADC e

irán conectados a los cuatro elementos que se deseen medir, es decir, 2 estarán conectados a un terminal de tornillo en la parte visible de la caja , otro estará conectado a la salida de la analógica de reducción de la Sección 3.3.1 la cual reducía la señal introducida en alguna de las 6 entradas del multiplexor ADG408 comentado en la Sección 3.4 y el ultimo canal medirá el valor de la SMU comentada en 3.5

3.8. Convertidor Digital-Analógico

El MCP4725 es un DAC (Digital-Analog Converter) [48] el cual tiene una resolución de 12 bits, es de bajo consumo y tiene incluida una EEPROM de 14 bits. Se comunica a través de una interfaz I²C y permite conectar hasta ocho MCP4725 en paralelo ya que tiene un pin de dirección. Funciona en el rango de temperaturas [-40, +125]ºC.

Al tener 12 bits, es capaz de establecer 4096 (2^{12}) estados los cuales estarán entre 0 V y 5 V, es decir, el paso entre dos estados consecutivos será de 1.22 mV. Su error INL(Integral nonlinearity) [2] típico es de ±2 %, el INL mide la desviación entre la salida ideal y la salida proporcionada por el DAC para algún estado. Su DNL(Differential nonlinearity)[25] de ±0.2 %, el DNL mide la desviación entre 2 estados correspondientes a valores de estados adyacentes.

A continuación se presentara la disposición de pines y el diagrama del MCP4725 de Adafruit:



Figura 3.33: Disposición de los pines del MCP4725 de Adafruit

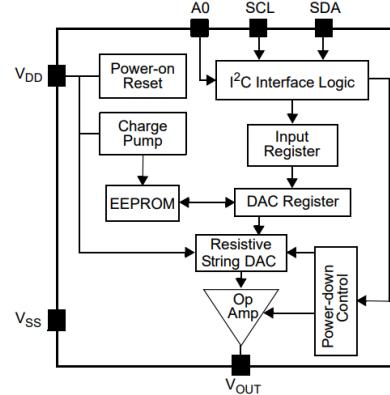


Figura 3.34: Esquemático del MCP4725 de Adafruit

Las conexiones realizadas con el DAC son las siguientes:

MCP4725	Arduino Mega 2560
VDD	5 V
GND	GND
SCL	SCL
SDA	SDA

Tabla 3.4: Conexiones del MCP4725 de Adafruit

De nuevo los pines SCL y SDA permitirán la comunicación I²C entre el Arduino y el DAC. Ya que solo se usara un DAC, el pin A0 estará sin conectar y su dirección será la asignada por defecto (0x62), el pin VOUT será el que de el voltaje generado por el DAC y sera conectado al acondicionamiento analógico de amplificación diseñado y explicado en la Sección 3.2.2.

Para conocer mejor el convertidor digital-analógico usado se ha estudiado en el laboratorio, se han generado 20 muestras entre 0 y 4096 y se ha medido con un multímetro de alta precisión el voltaje generado por el DAC. Estos son los resultados representados:

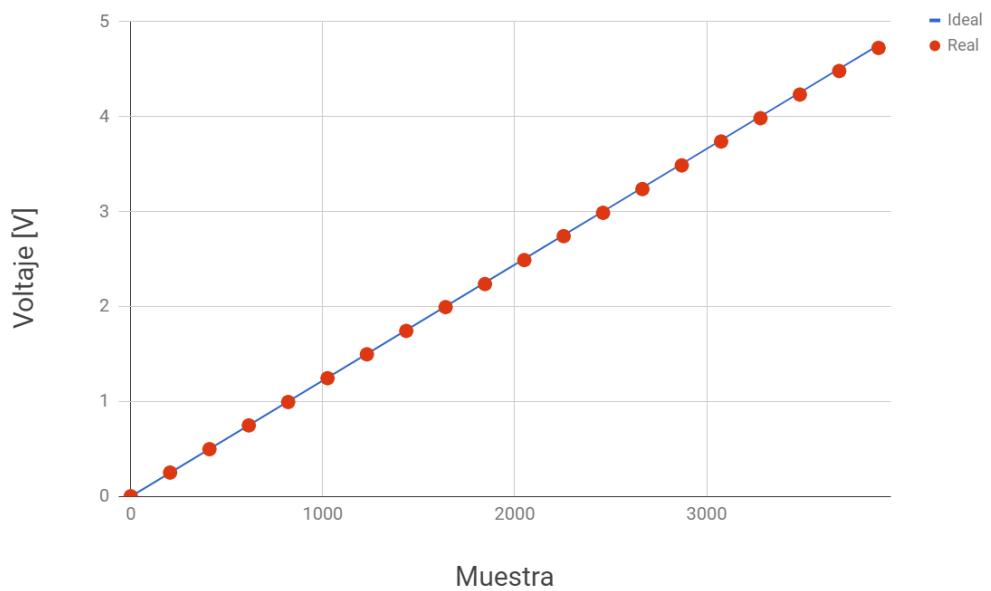


Figura 3.35: Caracterización del DAC

3.9. Pantalla LCD

Se eligió usar una pantalla LCD Nokia 5110 en lugar de la usual TC1602 ya que permite mucha mas versatilidad y opciones diseño sobre que se desea mostrar, la manera, forma y tamaño de hacerlo.

Nokia 5110 es una pantalla LCD de 84x48 (84 columnas y 48 filas) que se alimenta a un voltaje de 3.3 V, tiene un pin de Reset , es de bajo consumo y opera entre los -25ºC a 70ºC.

En la siguiente figura se muestra su disposición de pines y posteriormente se comentará en detalle cada pin:

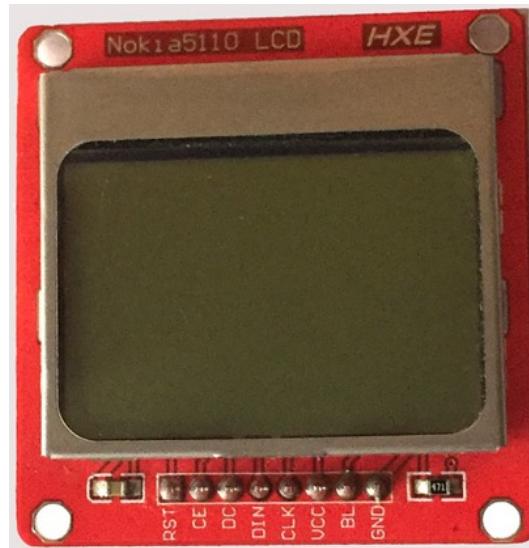


Figura 3.36: Disposición de los pines del Nokia5110

- **Reset(RST):** Pin que reinicia la pantalla cuando se pone a 0 V.
- **Chip Enable(CE):** Se usa para elegir el dispositivo deseado cuando hay mas de un periferico conectado por SPI.
- **Data/Command (DC):** Se usa para cambiar entre modo datos (cuando el pin esta en alto) a modo comandos (cuando el pin esta en bajo).
- **Serial Input(DIN):** Por este pin recibe las comunicaciones serie.
- **Clock(CLK):** Al ser una comunicación SPI requiere de un pin de reloj.
- **Power(VCC):** Pin de alimentación, acepta tensiones desde los 2.7 V hasta los 3.3 V.
- **Back Light(BL):** En función del voltaje aplicado a este pin se obtendrá una iluminación mayor o menor en la pantalla.
- **Ground(GND):** Conecta la tierra del circuito.

Las conexiones que se realizaran con el Arduino serán las siguientes:

Nokia 5110	Arduino Mega 2560
RST	Pin 3
CE	Pin 4
DC	Pin 5
DIN	Pin 6
CLK	Pin 7
VCC	3.3 V
BL	5 V
GND	GND

Tabla 3.5: Conexiones de la pantalla Nokia 5110

El pin BL, como se vio en la descripción de pines podría estar conectado a tierra o a cualquier voltaje entre 0 V y 5 V y en función de esto se obtendrá menor o mayor brillo de pantalla, en el caso del sistema estará conectado por defecto a 5 V para una mejor visualización de la pantalla.

3.10. Reloj en tiempo real

El DS3231 [19] es un reloj en tiempo real de bajo coste, de gran precisión y con un oscilador que tiene en cuenta la temperatura a la que se encuentra. El dispositivo incluye una batería de litio de 3V para mantener la hora cuando el sistema se desconecta. Se comunica a través de I²C y se alimenta a 3.3 V, aunque se puede alimentar a 5 V ya que tiene integrado un regulador.

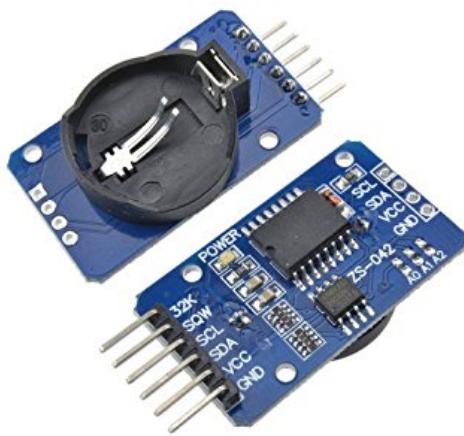


Figura 3.37: Disposición de los pines del DS3231 de Adafruit

Como se puede observar, el módulo dispone de 6 pines:

- **32K:** Da una señal de 32 kHz de salida. Se puede dejar en abierto en caso de no usarse.
- **SQW:** Este pin permite seleccionar entre dos modos: Interruptor activado por pulso bajo o generador de una señal cuadrada. Se puede dejar en abierto en caso de no usarse.
- **SCL:** (Serial Clock Input) este pin es el reloj para la interfaz serie I²C.
- **SDA:** (Serial Data Input/Ouput) este pin es la entrada o salida de datos para la interfaz I²C.
- **VCC:** Alimentación del módulo.
- **GND:** Tierra del módulo.

Se ha decidido usar la interfaz I²C para conectar el módulo, por lo que no se usarán los pines 32K y SQW y se dejaran en abierto, las conexiones que se realizan son las siguientes:

DS3231	Arduino Mega 2560
32K	Sin conectar
SQW	Sin conectar
SCL	SCL
SDA	SDA
VCC	5 V
GND	GND

Tabla 3.6: Conexiones del módulo de reloj en tiempo real DS3231 de Adafruit

3.11. Módulo Tarjeta Micro SD

Este módulo permite almacenar de forma sencilla datos en una tarjeta externa MicroSD, se alimenta entre 3.3 V y 5 V, lo cual es ideal para alimentar con el Arduino Mega 2560, y se comunica a través de interfaz SPI (Serial Peripheral Interface) [14].

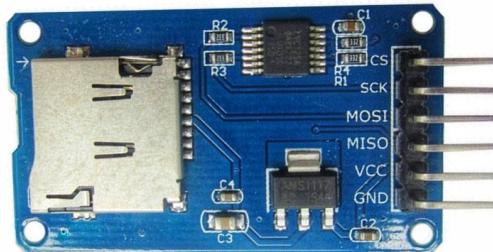


Figura 3.38: Disposición de los pines del módulo Micro-SD

Como se puede observar, el módulo dispone de 6 pines los cuales son:

- **CS (Choose Slave):** Pin usado para que el maestro pueda activar y desactivar todos los dispositivos.

- **SCK (Serial Clock):** Reloj usado para la sincronización de la transmisión de datos generada por el maestro.
- **MOSI (Master Out Slave In):** Línea usada para que el esclavo le envíe datos al maestro.
- **MISO (Master In Slave Out):** Línea usada para que el maestro le envíe datos al esclavo.

Se consultó la documentación del Arduino Mega 2560 para saber que pines tiene asignados a la interfaz SPI y estas fueron las conexiones realizadas:

Micro SD	Arduino Mega 2560
CS	Pin 53
SCK	Pin 52
MOSI	Pin 51
MISO	Pin 50
VCC	5 V
GND	GND

Tabla 3.7: Conexiones del módulo Micro SD

3.12. Módulo GSM/GPRS

Se trata *shield* ideado para Arduino Uno que usa como procesador el SIM900 de SIMCom [49], ofrece las bandas de 850/900/1800/1900 MHz como posibles de operación, tiene conectores para micrófono y auriculares para poder comunicarse en caso de realizar llamadas. También permite el envío de SMS, fax y datos, que será su utilidad en el sistema, funciona a 3 V para lo que incluye un regulador que adaptará los 5 V de entrada al voltaje de operación. Además, el *shield* incluye un RTC, aunque no será

usado en el proyecto ya que la deriva es bastante mayor que la del DS3231. También incluye su propia antena.

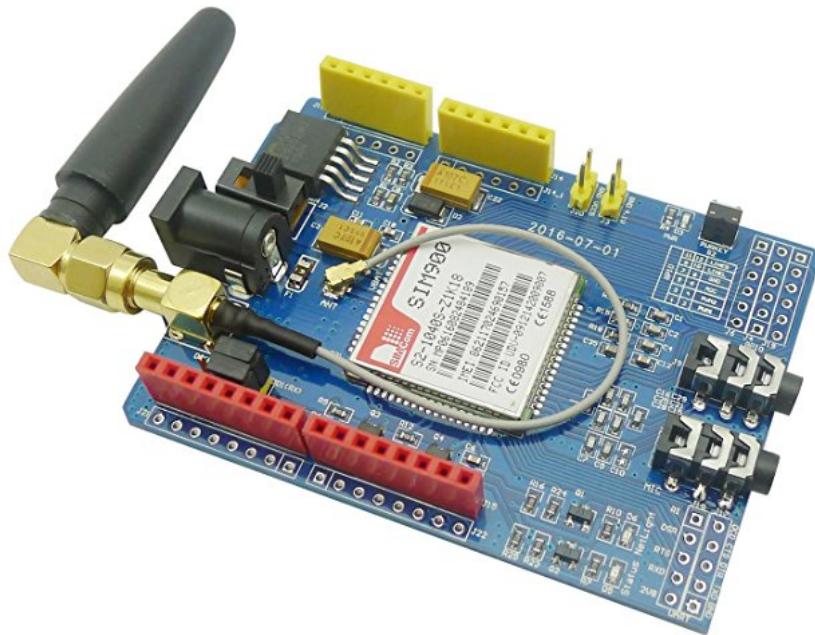


Figura 3.39: Shield SIM900

Su uso se basa en comandos AT [13], los comandos AT son instrucciones para controlar un *módem*, proviene de *ATtention* en apartados futuros se detallarán y explicarán los comandos AT usados.

Ya que usa interrupciones, se conectarán los pines TX y RX a pines que dispongan interrupciones del Arduino Mega para posteriormente emular, mediante una librería, un puerto serie. El pin 9 se encargará de encender el módulo. Estas son las conexiones realizadas:

SIM900	Arduino Mega 2560
Pin 9	Pin 9
TX	Pin 10
RX	Pin 11
GND	GND

Tabla 3.8: Conexiones del *shield* SIM900

Capítulo 4

Prototipos desarrollados del sistema

4.1. Prototipo V0

En primer lugar, para desarrollar el sistema se utilizó una placa de baquelita y una *protoboard* en las que se iban montando y conectando los diferentes módulos para su validación experimental.

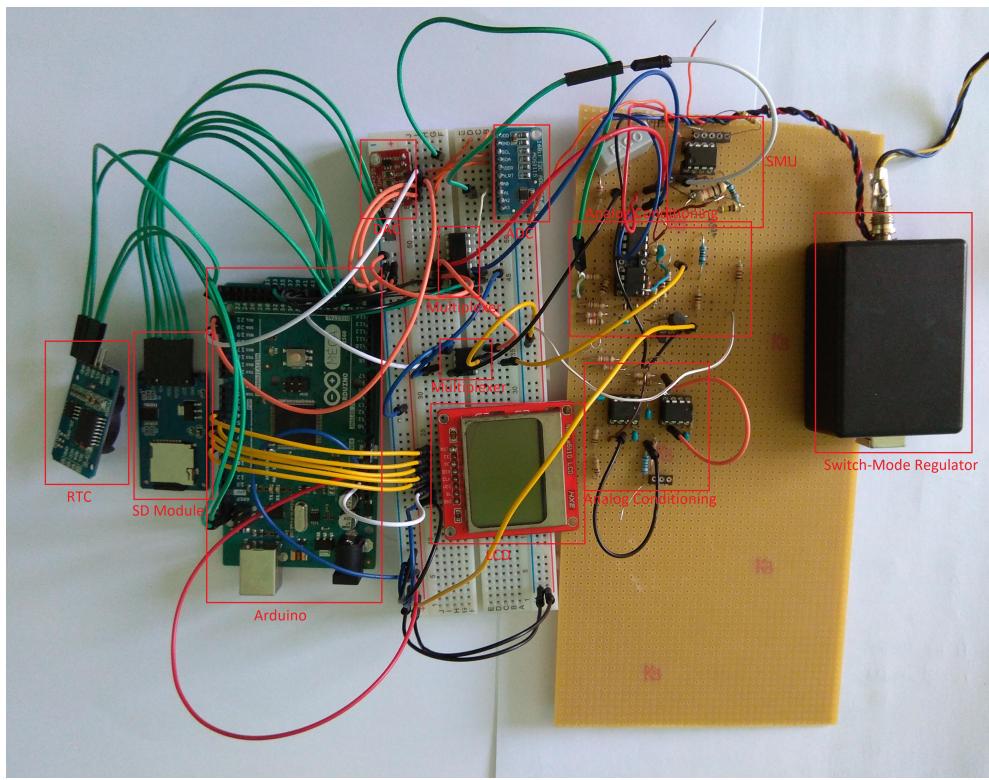


Figura 4.1: Prototipo V0 del sistema de adquisición de datos

En la imagen se pueden apreciar las diferentes partes del sistema, comenzando de izquierda a derecha están: módulo RTC, modulo micro SD, Arduino Mega, *protoboard* (en la que se encuentra el ADC, el DAC, ambos multiplexores y la pantalla LCD), placa de baquelita donde se encuentran los acondicionamientos analógicos así como la SMU y por ultimo, en la caja negra esta desarrollada la configuración explicada en la Fig.3.3 del MAX743.

Este prototipo ha sido de gran utilidad debido a la facilidad que aportaba para realizar cambios, así como depurar errores de conexiones. Pero también se puede observar que la organización no era la más adecuada, los cables de elevada longitud resultaron, como era de esperar, muy poco inmunes a la entrada de ruido externo, algunos cables no realizaban bien las conexiones y algunos otros problemas destacables fueron encontrados

en el prototipo, que no serviría para ser un sistema portátil debido a sus frágiles conexiones.

Al ser un sistema de medición analógico se debe minimizar el ruido lo mayor posible ya que afectará de manera directa a la medición.

Posteriormente, y debido a su complejidad, se probó de forma aislada el módulo GPRS.

4.2. Prototipo V1

Con fin de mejorar los fallos de la versión comentada en la sección anterior se propuso realizar una *Printed Circuit Board (PCB)*, para el diseño [22] de esta se barajaron diferentes *softwares* como Altium, KiCad y Eagle. Finalmente se optó por KiCad [9] debido a su enfoque, se trata de programa de código abierto y gratuito, con una gran comunidad que aporta y ayuda de forma activa en sus foros oficiales.

Antes del diseño del sistema final se realizaron varios proyectos más sencillos como toma de contacto con el *software*, posteriormente se procedió a realizar el esquemático del sistema final y su enrulado, en cuanto al enrulado, también se realizaron dos versiones debido a las dimensiones del embajale final del proyecto. Solo se comentará la versión final ya que la primera quedó descartada. El proyecto será embalado en una caja cuyas dimensiones son 18x15.5x5.2 cm, en concreto se usará una Hammond 1598C por lo que la PCB se desarrolló teniendo en cuenta las dimensiones físicas de esta.

Para conocer mejor el *software* y ganar experiencia en el desarrollo y diseño de PCB se diseñaron las siguientes huellas y esquemáticos de módulos del proyecto:

- Terminales de tornillo.
- Reloj en tiempo real.
- Modulo Micro SD.
- Pantalla LCD.
- Multiplexor 8 a 1.
- Convertidor Analógico-Digital.
- Convertidor Digital-Analógico.

A continuación se mostraran los esquemáticos y huellas de algunos de ellos como ejemplo:

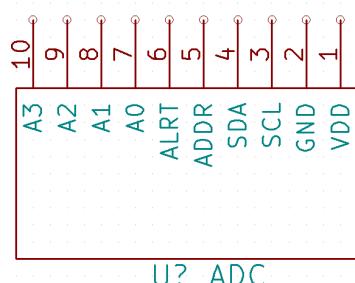


Figura 4.2: Esquemático del ADC

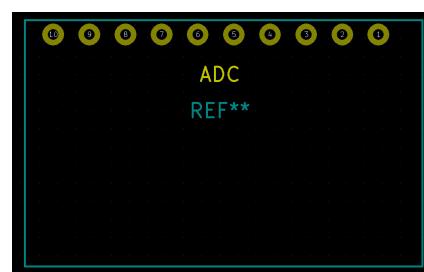


Figura 4.3: Huella del ADC

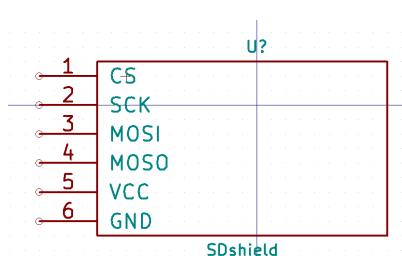


Figura 4.4: Esquemático del módulo Micro SD



Figura 4.5: Huella del módulo Micro SD

Mientras que las huellas y esquemáticos del resto de componentes fueron usados de librerías, o bien incluidas por defecto en el programa, o bien aportados por la comunidad en los foros oficiales.

4.2.1. Esquemático general

El primer paso para el diseño de la placa de circuito impreso fue diseñar su esquemático en KiCad, para lo cual se tomaron como referencia, las conexiones usadas en el prototipo y fueron replicadas.

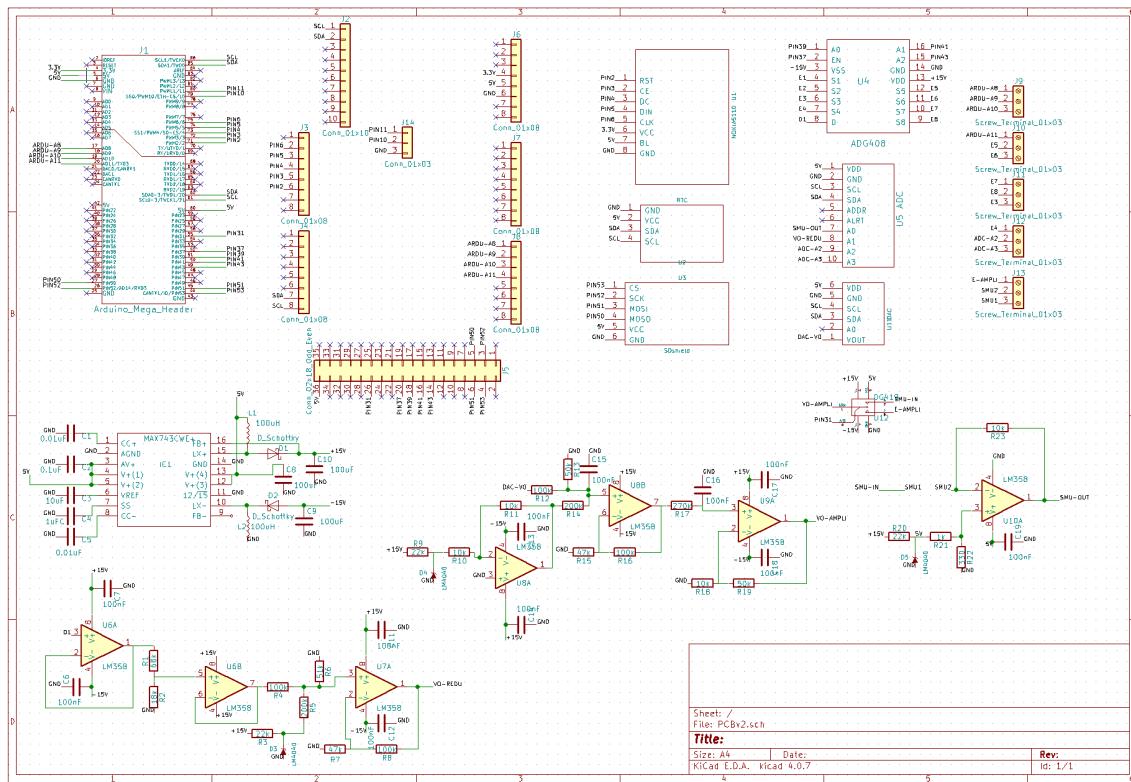


Figura 4.6: Esquemático general de la PCB

Se pueden identificar fácilmente los módulos y los acondicionamientos analógicos, la mayor parte de conexiones fueron realizadas mediante

etiquetas (*labels*) para no sobrecargar el esquema de cables y que dificulte su comprensión. La razón de los conectores J2, J3, J4, J5, J6, J7 y J8 será explicada en el apartado de enrutamiento.

4.2.2. Enrutado

La primera decisión a tomar en el enrutado fue de que forma colocar el microcontrolador debido al espacio que ocupa, se decidió colocarlo volteado, de forma que se pondrían tantos pines macho como pines del Arduino Mega, y este se colocaría boca abajo conectándose con tales pines macho, esa es la razón de los diferentes conectores J2, J3, etc.

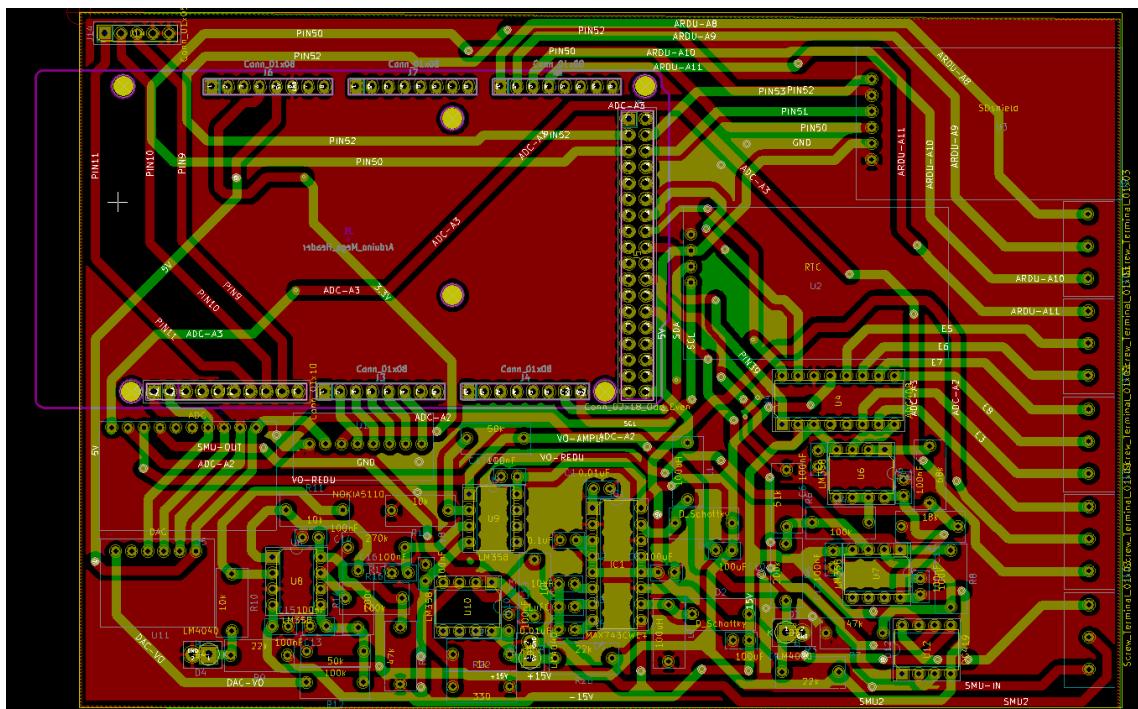


Figura 4.7: Enrutado general de la PCB

Se ha usado como plano de tierra tanto la cara frontal (color rojo) como

la cara posterior (color verde) de la placa mientras que se han intentado realizar, en medida de lo posible, las conexiones por la cara trasera. También se han tenido en cuenta algunas otras consideraciones:

- Se han intentado realizar pista lo mas anchas posibles para evitar fallos. Sobretodo en las pistas de alimentación del sistema.
- Se ha procurado dejar siempre que ha sido posible, al menos 1 mm entre dos pistas.
- No se han generado ángulos agudos en los giros de las pistas.
- Se ha intentado distanciar lo máximo posible los componentes lineales de los conmutados, para evitar el ruido.
- Realizar el mínimo de vías, ya que son una fuente de error al soldar.

Debido a su gran tamaño y la falta de espacio en la placa de circuito impreso, se decidió que el módulo de GSM/GPRS fuese pegado a una pared de la caja en lugar de incluirlo en el circuito impreso, pero si se han tenido en cuenta sus conexiones y se han puesto unos pines en la esquina superior izquierda para facilitar su conexión.

4.2.3. Resultado

Tras fabricar la PCB, se fueron soldando y validando cada uno de sus bloques por separado. El resultado final de la PCB es el siguiente:

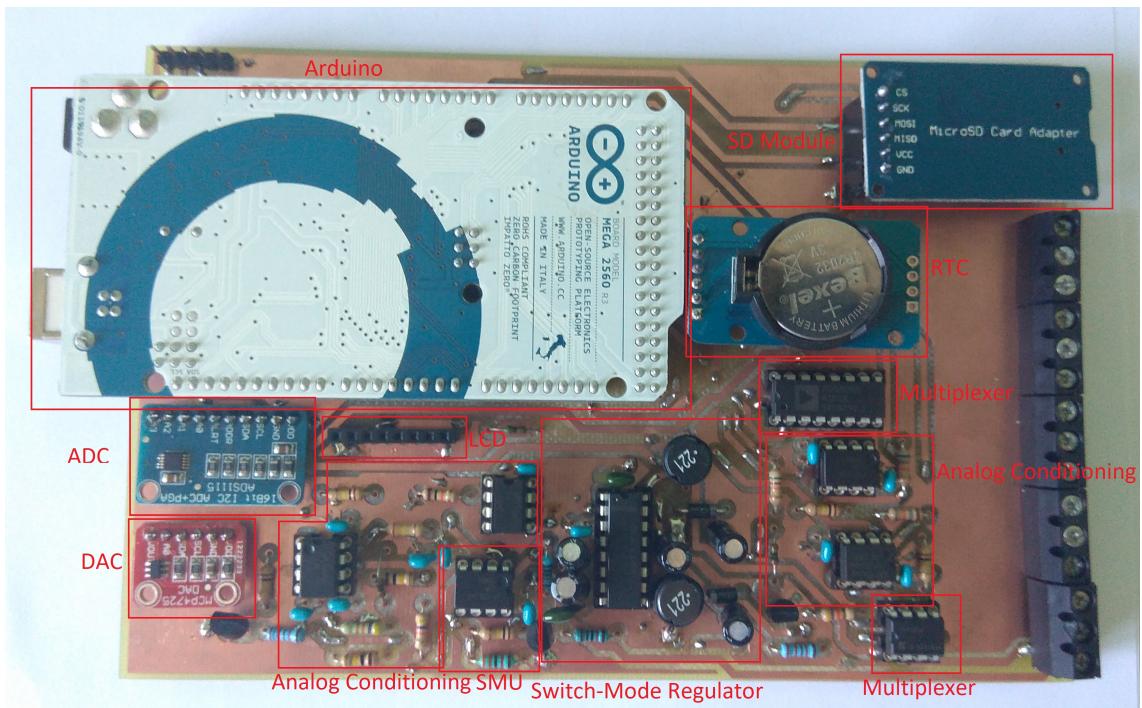


Figura 4.8: Prototipo V1

La pantalla LCD no se encuentra conectada ya que irá en el exterior del embalaje para su visualización, por otro lado, el modulo GMS se conectará en los pines situados en la parte superior izquierda en los casos en los que se vaya a usar. Tanto los terminales de tornillo como la tarjeta SD serán accesibles desde el exterior de la caja, así como los conectores del Arduino y del módulo GSM/GPRS para facilitar su alimentación

Capítulo 5

Firmware

5.1. Entorno de programación

Para el desarrollo del código que se ejecutará en el Arduino Mega 2560 se ha usado el entorno de desarrollo propio de Arduino, el Arduino IDE, el cual se puede obtener de forma gratuita y es de código abierto.

El entorno de desarrollo Arduino esta escrito en Java, aunque para programar sus micro controladores se usa el lenguaje de programación Arduino, el cual se basa en C por lo que tiene con el un alto grado de similitud, compartiendo sintaxis y lógica de programación.

Arduino es un lenguaje de alto nivel y dispone de una gran documentación en su web oficial. El lenguaje se puede dividir en tres bloques:

- Variables y constantes.
- Estructuras.
- Funciones.

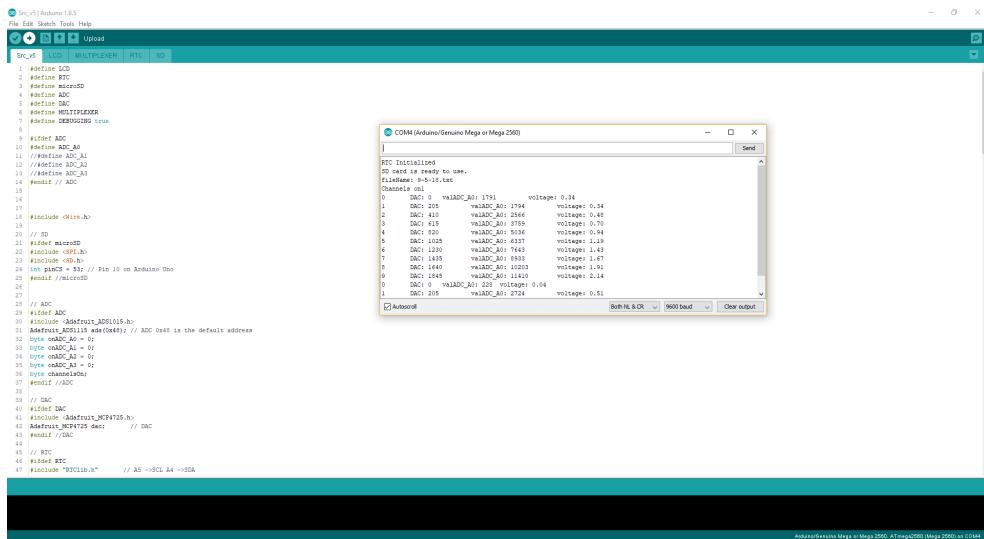


Figura 5.1: Interfaz del entorno de desarrollo Arduino

5.2. Código del sistema

El código del proyecto, así como el resto de partes ha sido desarrollado usando control de versiones git mediante *GitHub*. Se trata de un código modular en el que se puede desactivar el uso de cualquier módulo o canal con tan solo la edición de una linea, ya que el resto del código se adaptará a este cambio mediante condicionales. Además, contiene la opción de depuración de errores mediante mensajes a través del *Serial Monitor* con una directiva del lenguaje Arduino.

En los controladores Arduino, el código tiene siempre al menos dos funciones:

- **Setup:** Se ejecuta una vez al iniciar o reiniciar el dispositivo.
- **Loop:** Se ejecuta por primera vez tras el *Setup* y se repite mientras este el dispositivo encendido.

Para tener de manera mas organizada el código, se han realizado varios archivos *.ino* para los diferentes módulos o bloques que requerían de diversas funciones. Como estilo se ha usado *camelCase* para la definición de variables y funciones [15].

A continuación se describirá el código principal del sistema, y, posteriormente se detallará el funcionamiento de los módulos por separado, de forma muy simplificada, el diagrama de flujo del código es el siguiente.

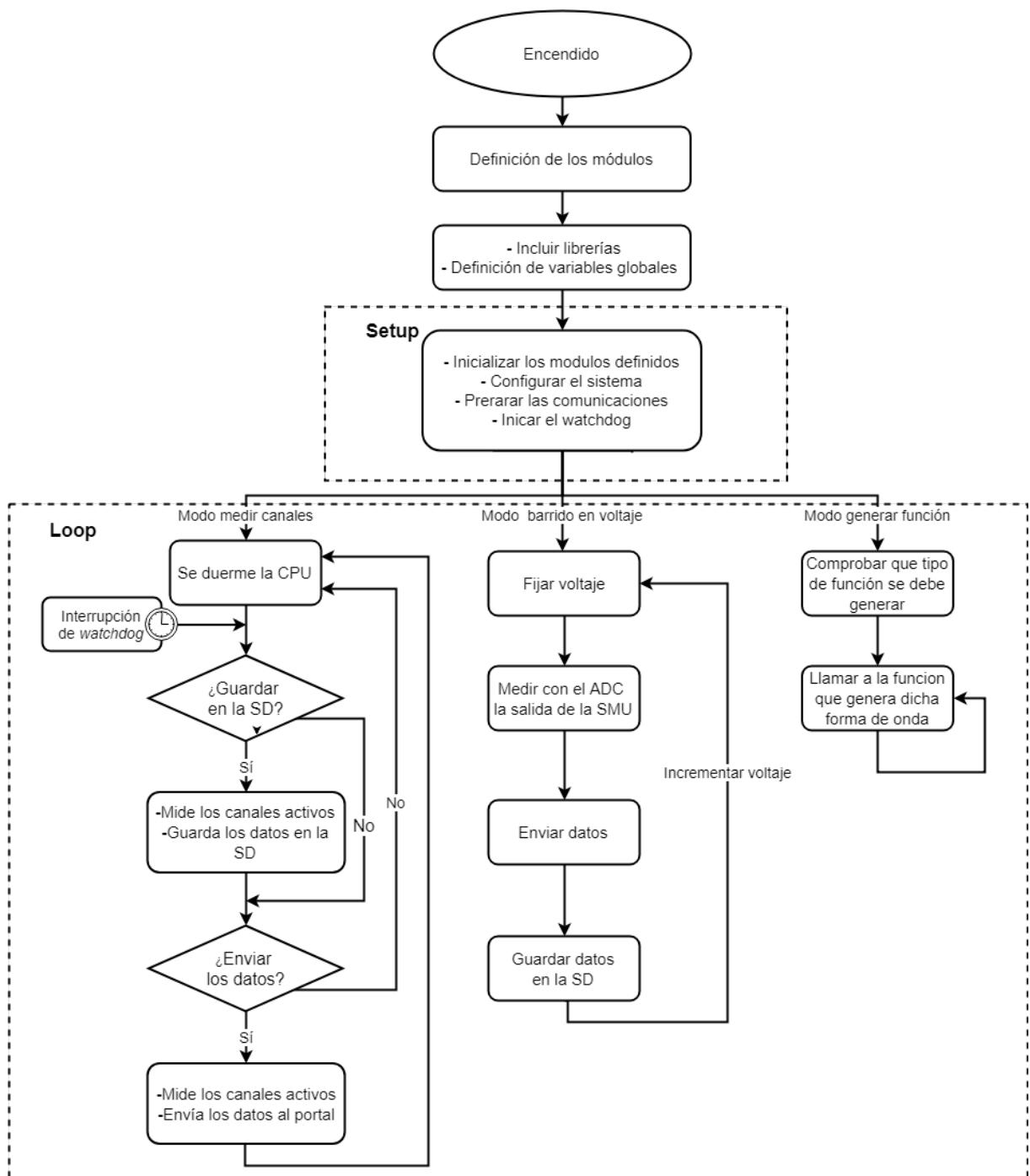


Figura 5.2: Diagrama del código principal

Los módulos definidos son los siguientes:

```
1 #define LCD
2 #define RTC
3 #define microSD
4 #define ADC
5 #define DAC
6 #define MULTIPLEXER
7 #define MEM_EEPROM
8 #define COMGPRS
9 #define DEBUGGING true
```

Si se desea dejar de usar algún módulo tan solo habría que comentar su define y el resto del código seguiría funcionando. La directiva *DEBUGGING* activará o desactivará los mensajes de depuración por pantalla segun este en *true* o *false*

El código íntegra un perro guardián (*watchdog*) [6], el cual reiniciará el sistema en caso de que se produzca algún error que lo tenga demasiado tiempo pausado en alguna parte del código y el cual controlará los tiempos para realizar algunas acciones como el guardar los datos en la SD o el enviarlos, estos tiempos serán definidos al principio del código y posteriormente se traducirán a ciclos de *watchdog* (de 8 segundos), una vez completados los ciclos de *watchdog* deseados para realizar ambas acciones, activarán una bandera.

En el *setup* se inician todos los módulos, se declaran las direcciones del DAC y del ADC y se procede a la configuración del sistema, la cual será cargada de la EEPROM si no es introducida por el Monitor Serie, por ultimo, si la configuración ha sido cambiada, se enviará a la plataforma de visualización y se actualizará en la EEPROM.

En el *loop* se comprobará que modo de trabajo ha sido introducido en la configuración. Hay tres modos de trabajo diferentes, cada uno opera de

una manera:

- **Medición de canales:** Se duerme el dispositivo, y cada 8 segundos el *watchdog* lo despertará, entonces comprueba el estado de las banderas de enviar datos y de guardar los datos en la SD, si alguna de las banderas esta activada, se mide el valor de todos los canales activos, se toma la hora del RTC y se realiza la tarea adecuada (enviarlos o guardarlos) y posteriormente se vuelve a dormir. El microcontrolador estará dormido la mayor parte del tiempo para así reducir su consumo, la mayoría de iteraciones del *loop* se despertará y se irá dormir ya que no habrá banderas activas. Se usara el modo *SLEEP_MODE_PWR_DOWN* que es el modo de dormir mas profundo que se puede usar con Arduino, además se deshabilitará el convertidor Digital-Analógico interno del Arduino para reducir aun más el consumo.

Cada 8 segundos el *watchdog* hará una interrupción, e incrementará en uno todos los contadores de ciclos *watchdog* que hay definidos, es decir, el de *reset* por *watchdog*, el de guardar en la SD y el de enviar datos. En el caso del contador de *reset*, si llega a 10 ciclos de 8 segundos, reiniciará el sistema. Por lo que se han incluido en las partes criticas del código en las que podrían producirse fallos llamadas a una función que reseteará el contador de *reset*, de manera que si el código esta funcionando de manera correcta nunca se reinicie, pero si por cualquier razón se bloquease, se auto-reiniciaría. Los contadores de la SD y del envío de datos, solo activarán la bandera correspondiente y se reiniciarán cuando llegan al máximo definido mediante sus tiempos declarados al principio.

- **Modo barrido en voltaje:** Este modo realizará un barrido de voltaje desde la tensión que se le defina (como máximo -15 V) hasta 0 V. El DAC irá fijando tensiones, que pasarán por la etapa analógica de amplificación y posteriormente el ADC externo medirá el valor de la tensión a la salida de la SMU. Conociendo esta tensión se puede

obtener la corriente. Una vez medida la tensión de la iteración, los datos serán guardados en la SD y enviados al portal. Finalmente, se incrementa la tensión fijada por el DAC y se repite el proceso hasta terminar el barrido.

- **Modo generar función:** Este modo generará una señal sinusoidal o una función diente de sierra, ambas del rango [-15,15] V. La señal a generar será especificada en la configuración.

5.3. Código de los módulos

En esta sección se hará una explicación sobre el código de los diferentes módulos usados en el sistema, explicando con detalle sus principales funciones, con fin de comprender mejor su funcionamiento y entender de mejor manera el código principal.

**Nota: Se ha de tener en cuenta que las funciones serán de tipo void y no recibirán parámetros de entrada a no ser que se especifique lo contrario en la explicación de la función.*

5.3.1. Código LCD

Se ha usado la librería *LCD5110_Graph* para facilitar el desarrollo del código necesario para la pantalla, aunque se explicarán las principales funciones de dicha librería también para comprender mejor su funcionamiento:

- **initLCD:** Inicializa la pantalla y fija una orientación, por defecto la orientación sera apaisada. También fija el tipo de letra en *ninguno*.
- **setFont:** Fija el tipo de letra, hay diferentes tipos de letra los cuales se deben definir previamente a usarlos.

- **clrScr:** Limpia todo lo escrito en la pantalla y pone el fondo en negro.
- **print:** Imprime una cadena de caracteres en las coordenadas especificadas.
- **printNuml:** Imprime un numero entero almacenado en una variable en las coordenadas especificadas.
- **update:** Actualiza la pantalla con las impresiones realizadas.

La pantalla se usará para conocer en que estado se encuentra el sistema, es decir, mostrará los canales activos mientras el sistema está dormido y cuando este realizando una tarea como la medición de canales, guardarlos en la SD o enviarlos, mostrará un mensaje anunciándolo. Además, en la parte superior de la pantalla se podrá observar un mensaje que indica si el sistema está conectado por USB al ordenador, o si se encuentra alimentado por batería y está trabajando de forma autónoma.

Basándose en las funciones incluidas en la librería de la pantalla LCD, se han desarrollado tres funciones:

- **updateModeLCD:** Muestra en la parte superior el modo en el que está operando el sistema, ya sea modo conexión por USB a un PC o modo autónomo.
- **updateTaskLCD:** Muestra en la parte central de la pantalla la tarea que está realizando el dispositivo. Tiene como parámetro de entrada un *char* que distinguirá entre las diferentes tareas.
- **updateConfigLCD:** Muestra la configuración del sistema. Usará 3 líneas en la que se mostrarán solo los canales activos. Cada una de las líneas corresponderá con: canales del ADC, canales del Arduino y canales de $\pm 15V$.

5.3.2. Memoria EEPROM

Con el fin de tener la capacidad de poder acceder a configuraciones previas incluso habiendo apagado el sistema, se implementó un código para guardar y leer diferentes registros de la memoria EEPROM asignados mediante *defines* en los que se registrará el ultimo valor de forma booleana (1 ó 0) que indicará si el canal se usará o no. Esto se ha conseguido mediante las siguientes funciones:

- **readConfiguration:** Lee las direcciones de memoria correspondientes a todos los canales y los almacena en las variables booleanas globales que han sido definidas en el código principal que indican si el canal esta activo o no. En caso de que el *define* de depuración este activo, de muestran por el monitor serie el estado de cada canal.
- **saveConfigEEPROM:** Esta función almacena en la EEPROM la configuración definida por el usuario, el proceso de configuración que será mostrado y comentado en el siguiente apartado.

5.3.3. Configuración

Este bloque se ha desarrollado y separado del resto debido a su extensión en código. Su fin es realizar la configuración del sistema cuando se inicializa. La configuración seguirá la siguiente dinámica: Primero comproba si esta alguien quiere introducir la configuración o si esta en modo autónomo, en caso de que este en modo autónomo cargará la configuración previa de la EEPROM y ya estará el sistema configurado, en caso de que un usuario desee cambiar la configuración, este introducirá de manera guiada los diferentes parámetros para activar los canales que desee y tras finalizar el cuestionario, se guardará la nueva configuración en la EEPROM.

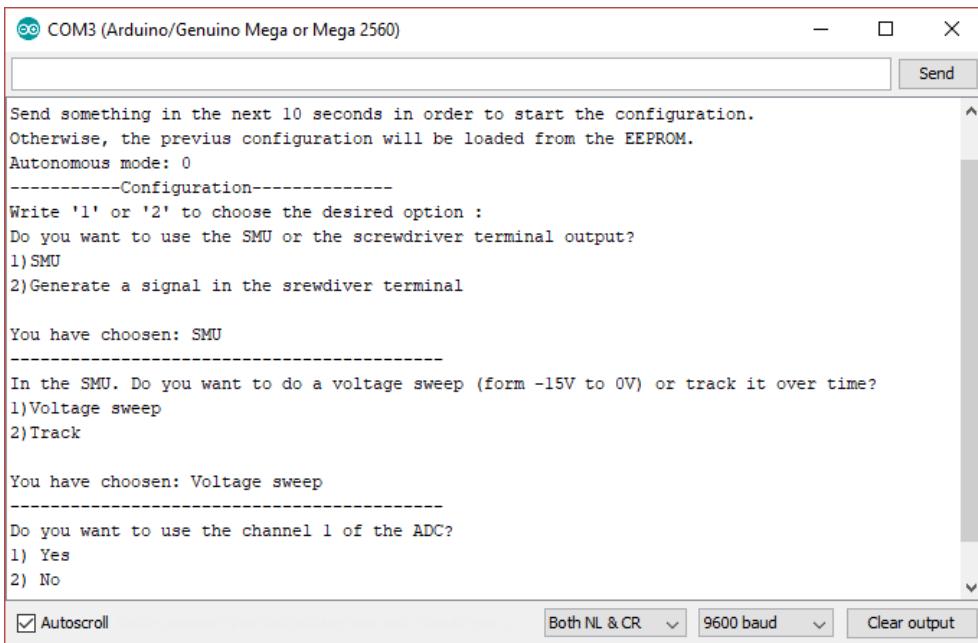


Figura 5.3: Configuración del sistema a través del monitor Serie

Para conseguir realizar tal configuración se han desarrollado cuatro funciones:

- **waitForKey:** Esta función se observará lo que introduce el usuario y esperará hasta que este introduzca un '1' o un '2'. Si el usuario introduce algún otro valor, mostrará un error y seguirá esperando uno de los valores correctos.
- **checkIfConnected:** Define el modo de funcionamiento del sistema (de manera autónoma o conectado por USB a un PC), para realizar tal tarea, le pide al usuario que introduzca cualquier carácter o caracteres a través del monitor Serie, si en 10 segundos no se ha introducido nada, se asumirá que está en modo autónomo.
- **askConfiguration:** Va guiando al usuario a través del monitor Serie para realizar la configuración y va almacenando las respuestas que el usuario ha introducido mediante '1' o '2'.

- **configureSystem:** Esta función llamará a las tres anteriores para ir, mediante condicionales, definiendo la configuración del sistema, dependiendo de si esta en modo autónomo o no, de si ha activado un canal o no ...

5.3.4. Multiplexor

Ya que para el siguiente módulo no se encontró ninguna librería que facilitase su uso, se procedió a la realización de funciones de forma que el código quedase más simplificado y limpio.

Se desarrollaron dos funciones principales para el ADG408 descrito en la Sección.3.4 y son:

- **initMultiplexer:** Función para inicializar el multiplexor, define los pines de dirección y el pin de enable como salidas y pone en alto el pin *enablePin* para habilitar el multiplexor.
- **setChannel:** Esta función tiene como parámetro de entrada una variable tipo *byte* y que corresponderá con el canal a seleccionar . Mediante condicionales fijara el canal del multiplexor siempre que el numero introducido esté entre 1 y 8, en caso de que el número no este en ese rango, mostrará un mensaje por pantalla diciendo que se ha introducido un numero de canal erróneo (siempre que la depuración por pantalla este activada) .

5.3.5. Medición de canales

Primero se debe destacar, que para cada canal activado se crea una variable global de tipo *float* en la que se almacenará el valor de este canal. En este bloque se han desarrollado 3 funciones:

- **readADC:** Tiene como parámetro de entrada un *int* que indica el canal del ADC, y la función crea una variable auxiliar de tipo *float* en la que acumula el valor de la lectura del canal del ADC indicado 100 veces. Posteriormente, el valor de la variable auxiliar se divide entre 100 para obtener la media de las 100 medidas y tener un resultado mas exacto, finalmente devuelve la variable auxiliar.
- **readAnalog:** Su funcionamiento es igual que la función *readADC*, pero en este caso, leerá valores de los canales analógicos del Arduino.
- **measureChannels:** Lee los canales activos usando las 2 funciones anteriores y almacena su valor en la variable global definida para tal uso.
- **generateWaveform:** Tiene como variable de entrada un booleano, este booleano habrá sido definido por el usuario en la configuración y dictará el tipo de señal a generar, ya sea sinusoidal o diente de sierra. Para general la señal sinusoidal se usan unas trasformaciones lineales y la función *sin()* de Arduino, mientras que para generar la función rampa tan solo se incrementará el valor del DAC desde su mínimo hasta su máximo y se repetirá.

5.3.6. Reloj en tiempo real

Para el uso del módulo RTC 3231 se usarán dos librerías de apoyo, la primera será *Wire* la cual permite la comunicación I²C y TWI con los dispositivos, en el caso del RTC usado, se comunicará mediante I²C por lo que la librería sera de gran utilidad. La segunda librería usada es *RTClib* de la cual se comentarán sus principales funciones a continuación:

- **rtc.begin:** Inicializa las comunicaciones I²C con el módulo usando la librería *Wire*.

- **rtc.lostPower:** Devuelve *true* en caso de haber perdido la alimentación externa, la comprobación lo hace leyendo un registro por I²C
- **rct.adjust:** Tiene como parámetros de entrada una fecha y hora y fija el RTC en dichos parámetros. Escribe cada parámetro en su registro correspondiente mediante I²C
- **rtc.now:** Mediante la lectura de los registros asignados a cada dato, devuelve la hora actual del RTC.

Cabe destacar, que la librería *RTCLib* sirve para diferentes módulos RTC por lo que hay que especificar que modelo se usará. Las variables donde se almacena la fecha y hora son declaradas en el código principal de manera global para su fácil acceso tanto por las funciones del RTC como por otro módulos. Para el uso del RTC se han desarrollado las siguientes funciones:

- **initRTC:** Llama a la función comentada anteriormente, *rtc.begin*, y posteriormente fija la hora del RTC a la hora de compilación. Además, en caso de tener activada la configuración de depuración por pantalla, nos informará si el módulo ha sido inicializado correctamente o si hubo algún problema para reconocerlo.
- **getTime:** Usa la función *rtc.now* y almacena la fecha y hora en las variables globales previamente comentadas.

5.3.7. Módulo Micro SD

Para la sección de código referente al módulo micro SD, se han usado dos librerías fundamentales: *SPI* y *SD*, la primera, es una librería integrada por defecto en Arduino y facilita la comunicación por SPI con los dispositivos, mientras que la segunda facilitará la labor con el modulo SD mediante algunas funciones incluidas en ella como:

- **begin:** Usa como parámetros de entrada los pines en los que se encuentra el *Choose Slave (CS)*, *Master Out Slave In (MOSI)*, *Master In Slave Out (MISO)* y *Serial Clock (SCK)*. Inicia la comunicación SPI con el módulo. Devuelve *true* si el módulo ha sido inicializado correctamente y *false* en caso contrario.
- **open:** Abre un archivo en el modo que se le indique (lectura o escritura). Si el archivo no existe, lo creará.
- **print:** Escribe en un archivo abierto previamente en modo escritura, los parámetros introducidos en la función.

Se ha desarrollado un método ágil para la nomenclatura de los archivos en los que se almacenarán los datos. Por lo que cada vez que se inicie el dispositivo, este escribirá en un archivo cuyo nombre sea la fecha del día en el que fue iniciado usando los datos del reloj en tiempo real.

Los datos serán almacenados en formato CSV, para facilitar su posterior análisis y tratamiento. Automáticamente, y según la configuración, se guardará en la primera linea del archivo, la cabecera con el nombre de los canales activos, para que una vez obtenidos los datos, la primera fila de cada columna de datos sea el nombre del canal al que corresponden los datos.

Se desarrollaron las siguientes funciones:

- **initSD:** Inicializa el modulo llamando la función de la librería *SD*, *begin*, y genera o abre el archivo nombrado con la fecha del día en el que se inicializa el modulo usando los datos del RTC. Si se activa la configuración de depuración por pantalla se apreciará un mensaje que indicará si ha sido inicializado correctamente o se dio algún tipo de error.
- **saveToSD:** Abre el archivo generado o abierto, con la función anterior y guarda, en CSV, el *time stamp* y los valores de los canales activos.

- **readSD:** Esta función mostrará todo el contenido del archivo que se ha generado o abierto. Solo se usará para la depuración por lo que no habrá ninguna llamada a ella en el código final.

5.3.8. Módulo GSM/GPRS

Para el uso del modulo de radio, se ha usado la librería *SoftwareSerial* para crear otro canal Serie mediante software, no se ha usado uno de los otros canales serie que trae el Arduino Mega ya que tenían peor rendimiento. Antes de describir las funciones usadas para este módulo, se van a comentar y explicar los diferentes comandos AT útiles y usados en el código.

- **AT:** Comando simple que realiza un *ping* al módulo para ver si está activo.
- **AT+CIPMODE:** Selecciona el modo de aplicación TCP-IP. Hay dos modos: normal o transparente. Se usará el modo normal.
- **AT+CIPMUX:** Inicia la conexión de una sola IP o multi-IP, para el sistema se usara el modo de IP única.
- **AT+CGATT:** Se conecta o desconecta del servicio GPRS.
- **AT+CREG?:** Comprueba que esta registrado en la red.
- **AT+CSTT:** Se conecta al APN que le introduzcas en los parámetros de entrada del comando.
- **AT+CIICR:** Activa la conexión inalámbrica con GPRS.
- **AT+CIFSR:** Obtiene la dirección IP local.
- **AT+CIPSTART:** Comienza una conexión TCP o UDP, se le deben introducir los siguientes parámetros: Tipo de conexión, dirección y puerto.

- **AT+CIPSEND:** Envía datos a través de la conexión TCP o UDP.
- **AT+CIPSHUT:** Cierra la conexión TCP o UDP.

Conociendo el funcionamiento de los comandos AT y habiendo probado su uso mediante el monitor Serie y softwares como *HyperTerminal* se desarrollaron las siguientes funciones:

- **ShowSerialData:** Muestra por el monitor serie los datos provenientes del módulo GSM/GPRS. Se usa para ver que las respuestas son las esperadas y que no ocurra ningún error en la comunicación.
- **initSIM900:** Inicializa el modulo fijando los baudios a los que se comunicará (19200) y enviando los comandos AT necesarios para que se puedan enviar los datos deseados posteriormente.
- **sendConfig:** Compone la trama a enviar, abre la conexión TCP y envía la trama deseada. Para la el envió de la trama se ha generado un método de agrupar la configuración de manera que se envíen los mínimos campos posibles. La trama contiene que canales están activos y que canales no.
- **sendData:** Compone la trama a enviar, abre la conexión y envía los datos. La trama contiene el valor actual de los canales activos del sistema.

Capítulo 6

Monitorización y visualización de los datos

Para facilitar la visualización de los datos se ha desarrollado un portal: tfg-sergio-gasquez.onlosant.com en un dominio facilitado por Losant, una plataforma *IoT (Interet of Things)* que permite el almacenamiento, tratamiento y visualización de datos.

Sin embargo, aunque los datos sean visualizados a través del portal alojado y que toma los datos de Losant, los datos no son enviados a Losant desde el módulo SIM900, ya que Losant no acepta el envío de datos a través de una conexión TCP-IP [18], que es la usada por el módulo, por lo que se tuvo que usar ThingSpeak [35] como *gateway*. A continuación se mostrará un diagrama simplificado de como procederá el flujo de los datos:

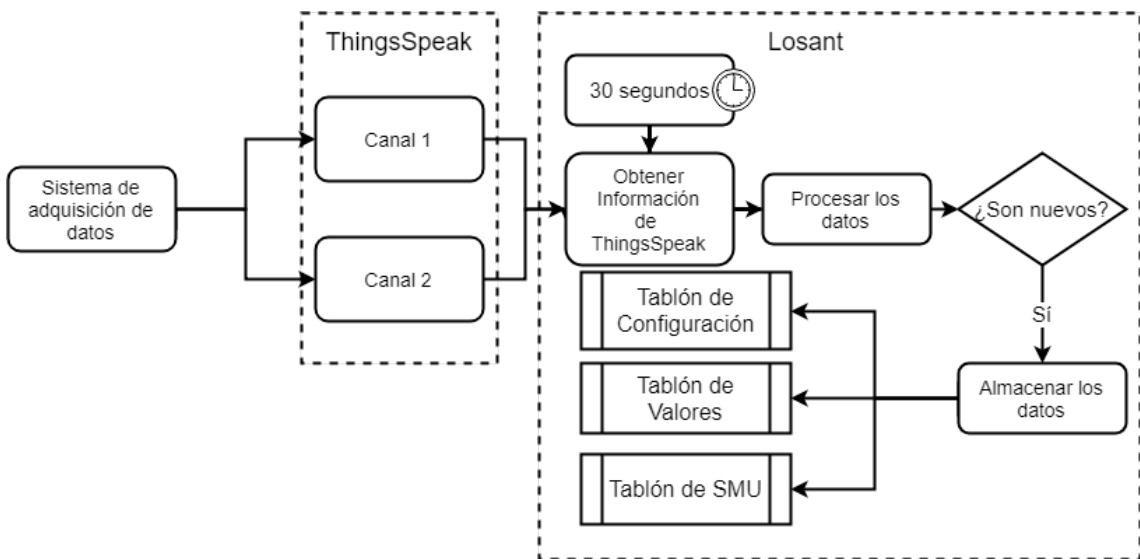


Figura 6.1: Diagrama del flujo de datos entre las plataformas IoT.

ThingSpeak es otra plataforma de *IoT* que pertenece a MathWorks, que también permite recoger, almacenar y mostrar datos, pero al contrario que Losant, si tiene una API, contra la cual se abrirá una conexión TCP por la que se enviarán los datos usando una API Key generada para los dos canales que se usarán. ThingSpeak permite, de forma gratuita, hasta 10 dispositivos, es decir, 10 canales diferentes, en cada canal se pueden guardar hasta 8 variables. Ya que el sistema de adquisición de datos tiene 12 canales y también se guardará la configuración del sistema. Se han usado dos canales de ThingSpeak.

Como se puede observar, para la configuración se usarán dos variables. La configuración será enviada de la siguiente manera:

$Config1 = 9ABCDEFG$ donde A será 1 o 0 en función de si la SMU está activa o no, B será 1 o 0 en función de si el canal 1 del ADC está activo o no y de igual forma para el resto de canales. Por lo que el campo 1, contendrá la configuración de la SMU, el canal 1 y 2 del ADC y los canales 1,2,3,4 del Arduino. La segunda variable relativa a la configuración,

Field 1	Config1	<input checked="" type="checkbox"/>	Field 1	Ardu_11	<input checked="" type="checkbox"/>
Field 2	Config2	<input checked="" type="checkbox"/>	Field 2	Multi_E5	<input checked="" type="checkbox"/>
Field 3	SMU	<input checked="" type="checkbox"/>	Field 3	Multi_E6	<input checked="" type="checkbox"/>
Field 4	ADC_2	<input checked="" type="checkbox"/>	Field 4	Multi_E7	<input checked="" type="checkbox"/>
Field 5	ADC_3	<input checked="" type="checkbox"/>	Field 5	Multi_E8	<input checked="" type="checkbox"/>
Field 6	Ardu_8	<input checked="" type="checkbox"/>	Field 6	Multi_E3	<input checked="" type="checkbox"/>
Field 7	Ardu_9	<input checked="" type="checkbox"/>	Field 7	Multi_E4	<input checked="" type="checkbox"/>
Field 8	Ardu_10	<input checked="" type="checkbox"/>	Field 8		<input type="checkbox"/>

Figura 6.2: Variables almacenadas en el canal 1

Figura 6.3: Variables almacenadas en el canal 2

tiene una forma similar, $Config2 = 9ABCDEF$, pero esta contiene la información de los 6 canales de $\pm 15V$. La razón por la que ambos números comienzan por un '9' es puramente arbitraria, se debía poner un numero conocido ya que si el primer canal estaba desactivado, se enviaría un 0, y ThingSpeak elimina por defecto los 0 a la izquierda. Tampoco se ha podido enviar en una sola variable, ya que, a pesar de que ThingSpeak lo admitiría, Arduino no tiene un tipo de variable que alcance valores tan altos, por lo que se usaron dos variables tipo *unsigned long*. El resto de variables contendrán el valor leído por el ADC, ya sea externo o interno del Arduino.

También se deben destacar la frecuencia con la que ThingSpeak nos permite, de forma gratuita, escribir en su API, una vez cada 15 segundos, por lo que esto se ha tenido en cuenta a la hora de enviar los datos.

ThingSpeak ofrece un tablón donde se pueden visualizar las variables

del canal, pero tiene opciones muy limitadas y, además, se tendría que estar cambiando entre el tablón de los dos canales para consultar todas las variables y no habría forma de representar de forma visual la configuración.

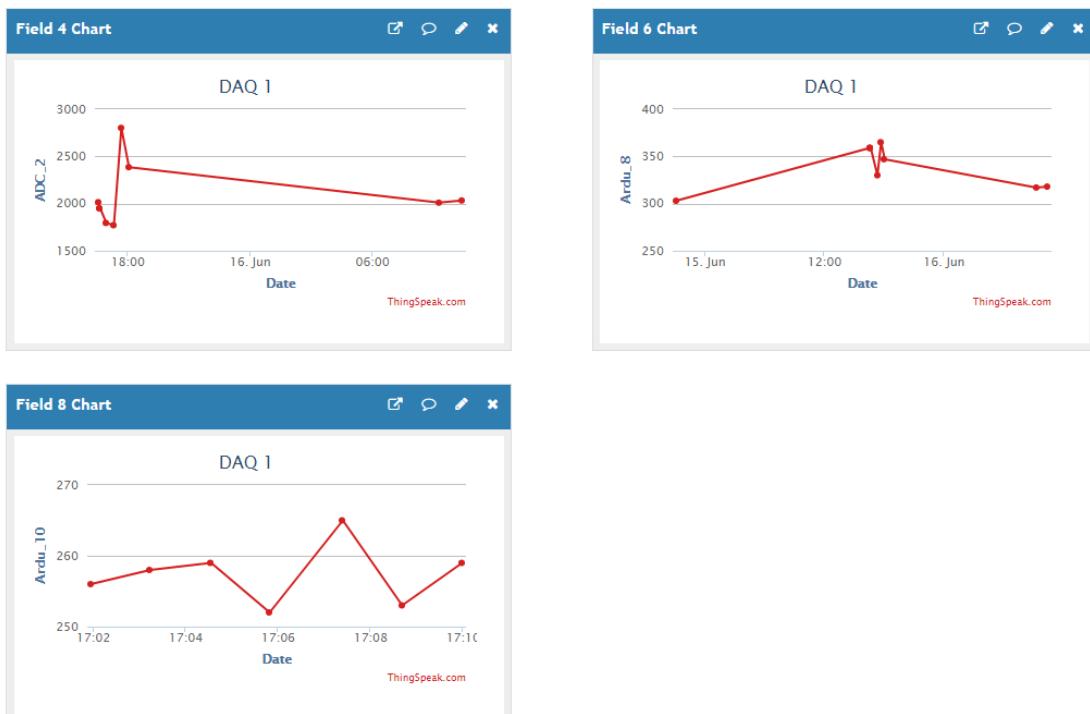


Figura 6.4: Ejemplo de tablón en ThingSpeak.

Para una mejor visualización, mejor procesado de los datos antes de mostrarlos y más variedad a la hora de mostrarlos, se decidió basarse en Losant para mostrar los datos.

Se ha desarrollado una aplicación en Losant, la cual procesará, almacenará y mostrará los datos enviados a ThingSpeak. Para almacenar los datos, se han creado dos dispositivos en Losant, el primero almacenará la configuración y tiene los siguientes atributos:

	Data Type Boolean	Name SMU
	Data Type Boolean	Name ADC_2
	Data Type Boolean	Name ADC_3
	Data Type Boolean	Name ARDU_A9
	Data Type Boolean	Name ARDU_A10
	Data Type Boolean	Name ARDU_A11
	Data Type Boolean	Name multi_E5
	Data Type Boolean	Name multi_E6
	Data Type Boolean	Name multi_E7
	Data Type Boolean	Name multi_E8
	Data Type Boolean	Name multi_E3
	Data Type Boolean	Name multi_E4
	Data Type Boolean	Name ARDU_A8

Figura 6.5: Dispositivo de la configuración en Losant.

El segundo dispositivo de Losant almacenará los valores de los canales y contiene el mismo número de atributos que el primer dispositivo pero en este caso estos serán de tipo *Number*.

Para obtener los datos y preprocesarlos antes de su visualización se ha desarrollado un flujo (*workflow*):

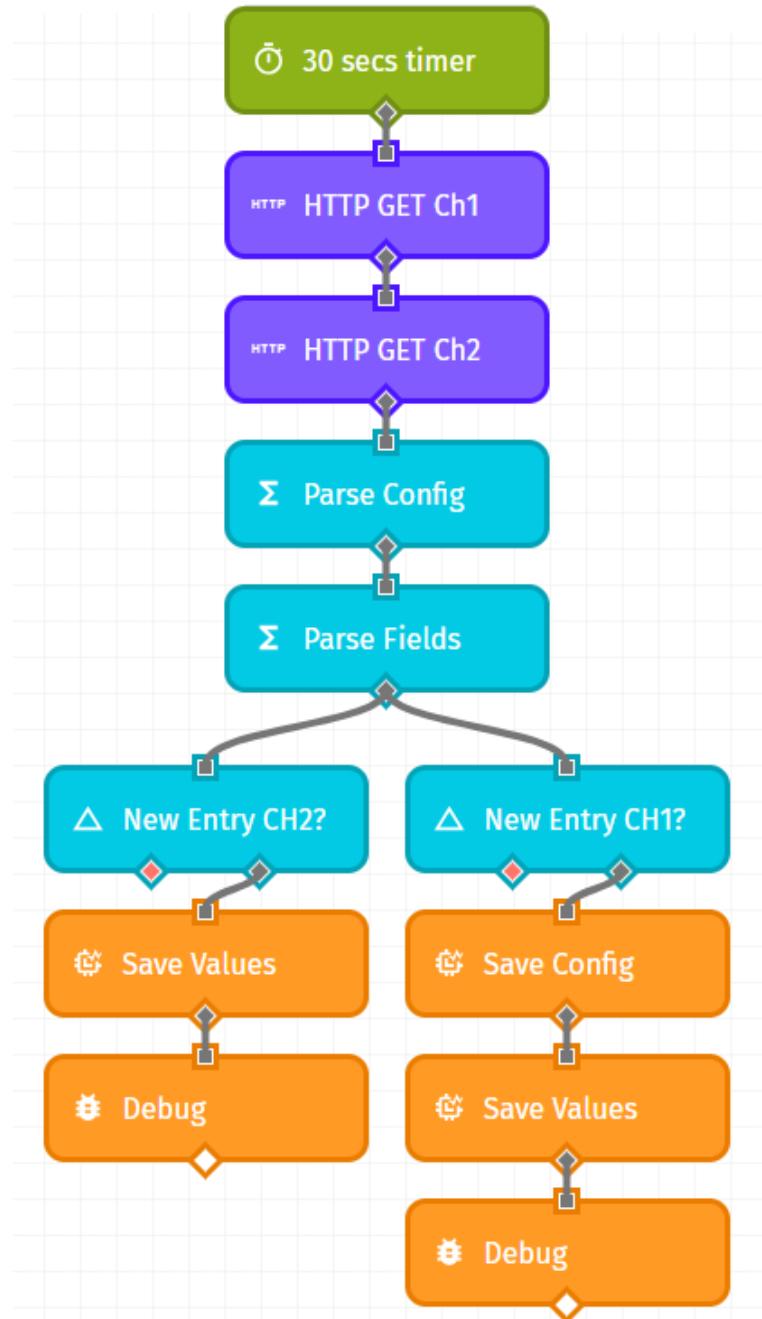


Figura 6.6: Workflow de la aplicación desarrollada en Losant.

A continuación, se van a explicar en detalle cada uno de los bloques por el siguiente orden, de arriba a abajo y de izquierda a derecha:

- **30 secs timer:** La función de este bloque es activar el *workflow* cada 30 segundos, es decir activa el resto del flujo de manera periódica.
- **HTTP GET Ch1:** Este bloque obtiene los valores actuales del canal de ThingSpeak mediante una petición *GET* y usando una *API Key* de lectura generada por ThingSpeak. La respuesta viene en forma de *JSON*.
- **HTTP GET Ch2:** Este bloque es exactamente igual que el anterior, solo que este obtendrá la información del canal 2.
- **Parse Config:** Este bloque es de tipo *function*, los bloques de tipo función permiten, usando JavaScript, usar y transformar los *JSON* obtenidos con los bloques anteriores. La función de este bloque será tomar las dos variables que contienen la configuración, separar los dos numeros dígito a dígito y añadirlos al *JSON* en variables cuyo nombre será *confNombreDelCanal*.
- **Parse Fields:** De nuevo, se ha usado otro bloque *function*, este obtendrá del *JSON* los valores de los diferentes *fields* de los dos canales de ThingSpeak y para cada variable comprobara si este es un valor diferente de *NULL* y en caso de serlo guardará y añadirá al *JSON* dos variables, una llamada *NombreDelCanal*, que contendrá el valor del canal, y otra *V_NombreDelCanal* que tendrá el voltaje equivalente a dicho valor del canal. Este voltaje es calculado sabiendo el número de bits de la resolución de los dos ADCs usados en este mismo bloque.

```

25 payload.ADC_2= payload.data.Channel1.body.feeds[0].field4;
26 payload.V_ADC_2= (payload.data.Channel1.body.feeds[0].field4* 0.1875)/1000;
27 }
28
29 if(payload.data.Channel1.body.feeds[0].field5 != null)
30 {
31 payload.ADC_3= payload.data.Channel1.body.feeds[0].field5;
32 payload.V_ADC_3= (payload.data.Channel1.body.feeds[0].field5* 0.1875)/1000;
33 }
34
35 if(payload.data.Channel1.body.feeds[0].field6 != null)
36 {
37 payload.Ardu_8= payload.data.Channel1.body.feeds[0].field6;
38 payload.V_Ardu_8= (payload.data.Channel1.body.feeds[0].field6*5)/1024;
39 }
40
41 if(payload.data.Channel1.body.feeds[0].field7 != null)
42 {
43 payload.Ardu_9= payload.data.Channel1.body.feeds[0].field7;
44 payload.V_Ardu_9= (payload.data.Channel1.body.feeds[0].field7*5)/1024;
45 }
46
47 if(payload.data.Channel1.body.feeds[0].field8 != null)
48 {
49 payload.Ardu_10= payload.data.Channel1.body.feeds[0].field8;
50 payload.V_Ardu_10= (payload.data.Channel1.body.feeds[0].field8*5)/1024;
51 }
52

```

Figura 6.7: Muestra de parte del código del bloque *Parse Fields*.

Las variables añadidas a la trama tras haber transcurrido por los bloques explicados tendría la siguiente forma:

```

    "V_Multi_E4": 0.576
    "Multi_E4": "3072"
    "V_Multi_E3": 0.5938125
    "Multi_E3": "3167"
    "V_Multi_E8": 0.621
    "Multi_E8": "3312"
    "V_Multi_E7": 0.63975
    "Multi_E7": "3412"
    "V_Multi_E6": 0.610875
    "Multi_E6": "3258"
    "V_Multi_E5": 0.6375
    "Multi_E5": "3400"
    "V_Ardu_11": 1.34765625
    "Ardu_11": "276"
    "Conf_multi_E4": "1"
    "Conf_multi_E3": "0"
    "Conf_multi_E8": "1"
    "Conf_multi_E7": "0"
    "Conf_multi_E6": "1"
    "Conf_multi_E5": "0"
    "Conf_ARDU_A11": "1"
    "Conf_ARDU_A10": "0"
    "Conf_ARDU_A9": "1"
    "Conf_ARDU_A8": "0"
    "Conf_ADC_3": "1"
    "Conf_ADC_2": "0"
    "Conf_SMI": "1"

```

Figura 6.8: Muestra de parte de una trama tras los bloques *funtion*

Tras estas variables continuaría el resto de la trama JSON [11] con los *time stamp*, valores de todos los *fields* de los dos canales y demás información.

Primero se detallará la rama de la izquierda.

- **New Entry CH2?**: Este bloque comprueba si una variable ha cambiado con respecto a la anterior trama recibida, en este caso, comprueba

si la *entry_id* de la información recibida de la trama dos es igual a la anterior, es decir, si los datos que se han recogido de ThingSpeak son los mismos que se habían recogido en una trama anterior o son nuevos. La variable *entry_id* es autogenerada por ThingSpeak e incrementa en uno cada vez que se actualiza un dato del canal.

El bloque tiene dos salidas, una para el caso en el que la variable haya cambiado, y otra en caso de que la variable no haya variado. Solo se usará la salida que implica que la variable haya cambiado.

- **Save Values:** La función de este bloque es guardar en el dispositivo creado para almacenar el valor de los canales la información preprocesada por los bloques *function* para posteriormente poder mostrarla.
- **Debug:** Este bloque muestra la trama JSON, su utilidad es meramente depurativa, ya que no realiza ningún cambio, solo muestra la información que ha llegado hasta este punto.

Por ultimo, se va a comentar la rama de la derecha, la cual es muy similar a la anterior, salvo que incluye un bloque extra:

- **New Entry CH1?:** Al igual que el bloque *New Entry CH2?* comprueba si hay una nueva entrada, pero esta vez del canal 1 de ThingSpeak.
- **Save Config:** En el canal 1 se encuentran las dos variables que guardan la configuración del sistema, por lo que este bloque realiza la función de guardar dicha información, una vez separados los dos números que se envían a variables booleanas, en el dispositivo creado con el fin de almacenar tales variables
- **Save Values:** Guarda el valor de los canales SMU, canal 1 y 2 del ADC, y de tres de los cuatro canales del Arduino. Los guarda en el mismo dispositivo que guarda la información el otro bloque *Save Values* de la otra rama.

- **Debug:** Al igual que el otro bloque de *debug*, se utiliza para la depuración de tramas.

Por ultimo, para la visualización de datos, se han usado *Dashboards* de Losant, los cuales permiten mostrar de forma fácil y variada, las variables almacenadas en los diferentes dispositivos, se han creado 4 dashboards.

Antes de comentar cada uno de los dashboards, se explicarán los tipos de bloques usados para mostrar los datos:

- **Time Series Graph:** Este bloque permite mostrar el valor de una variable a lo largo de un tiempo personalizable.
- **Gauge:** Permite mostrar el valor actual de una variable de forma numérica o visual.
- **Indicator:** Comprueba el valor de una variable y muestra un mensaje en función del valor de la variable.

El primer *dashboard* se ha usado para mostrar la configuración del sistema, de manera que el usuario pueda conocer de forma rápida y visual que canales están activos.

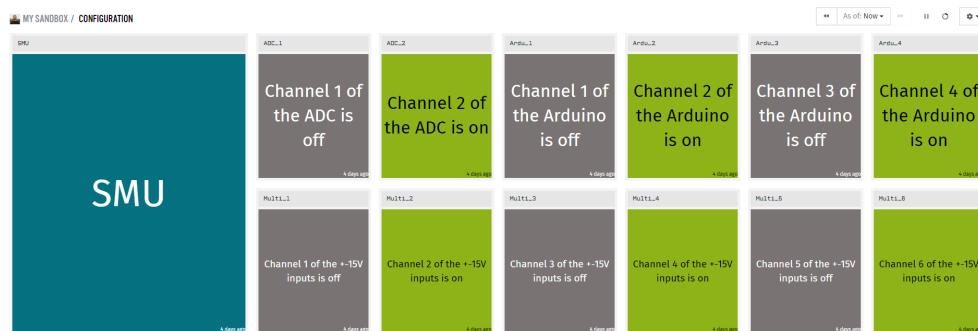


Figura 6.9: Resultado del *dashboard Configuration*

Para el tablero en el que se mostrará la configuración se han usado bloques de tipo *indicator* y en función de la variable almacenada en el

dispositivo *configuration* el bloque se pondrá del color oportuno y mostrará el estado del canal.

A continuación se crearon dos dashboards para ver tanto los valores como los voltajes a los que estos equivalen en su valor numérico y gráficas en los que se observa su progresión a lo largo del tiempo. Se han usado bloques de tipo *Time Series Graph* y *Gauge*.

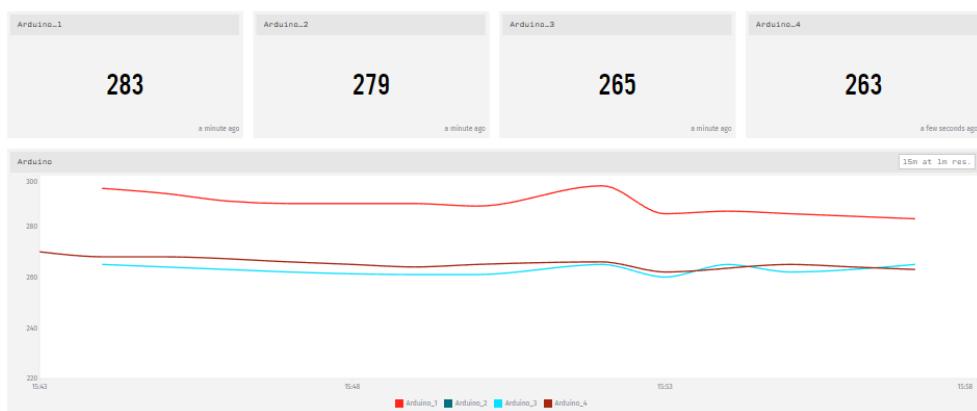


Figura 6.10: Resultado de parte del *dashboard* donde se muestran los valores de los canales

Por ultimo, también se desarrolló un *dashboard* en el que apreciar las variables relacionadas con la *SMU*, de nuevo se usaron los bloques *Time Series Graph* y *Gauge*.

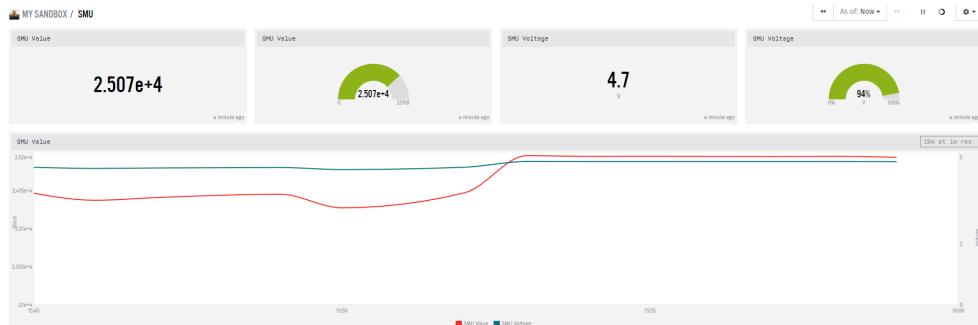


Figura 6.11: Resultado del *dashboard* donde se muestran los valores la SMU

Para que los dashboards pudiesen ser observados por cualquier persona y tener una mayor facilidad para cambiar entre dashboards y demás opciones, se ha usado un domino ofrecido por Losant para realizar un portal de visualización para el sistema. Dicho dominio será publico y no se necesitarán credenciales para visitarlo, aunque dicha característica se podría integrar en caso de ser necesario. Cada una de las pestañas renderizará y mostrará el *dashboard* con cuyo nombre coincide.

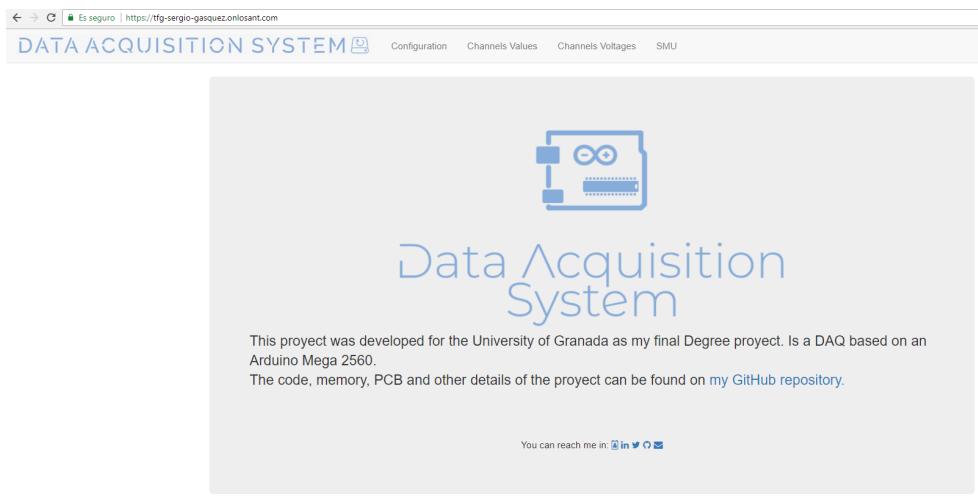


Figura 6.12: Web desarrollada para la visualización de datos.

Capítulo 7

Especificaciones del sistema

En este apartado se resumirán las características más relevantes del sistema desarrollado:

		Mín	Tip	Máx	Unidad
Alimentación ^[1]		5	5	12	V
Consumo ^[2]		98	130	250	mA
Resolución ^[3]		0.1875	0.1875	4.88	mV
Frecuencia de muestreo ^[4]	Modo rápido	112	112	8333	Hz
	Modo resolución	1	1	83	Hz

Tabla 7.1: Especificaciones del sistema de adquisición de datos

Notas:

^[1] Se puede alimentar de diferentes formas: por USB, mediante baterías o a través del conector Jack, el cual acepta tensiones de entre 7 y 12 V.

^[2] El consumo mínimo se consigue cuando el sistema se encuentra dormido, el consumo del microprocesador despierto es en torno a 130 mA, mientras que cuando la SMU requiere de la máxima potencia el sistema llega a consumir 250 mA.

^[3] La resolución dependerá de con que ADC se este midiendo el canal, el

ADC externo, el ADS1115, tiene una resolución de 0.1875 mV mientras que el ADC interno del Arduino Mega tiene una resolución de 5mV.

[4] En el modo rápido de lectura solo se realiza una medición al canal, mientras que en el modo resolución se realizan 100 medidas del canal y se calcula la media.

A continuación se detallarán el resto de características importantes del sistema:

- 12 entradas analógicas:
 - 6 entradas del rango [0,5] V.
 - 6 entradas del rango [-15,15] V.
- SMU que permite medir corrientes.
- Salida por terminal de tornillo de una tensión configurable, con formas de onda preconfiguradas como sinusoidal o rampa.
- Dispositivo portátil con modo de trabajo autónomo.
- Pantalla LCD informativa.
- Almacenamiento de datos en un fichero de texto.
- Envío de datos a un portal web donde se pueden visualizar las medidas en tiempo real.

Capítulo 8

Resultados y conclusiones

8.1. Resultados

8.1.1. Canales

Para ver el correcto funcionamiento de la lectura de los canales se ha monitorizado un sensor de humedad en tierra y un sensor de temperatura, el LM35Z. El sensor de humedad era leído un canal asociado al ADC interno del Arduino, mientras que el valor del LM35Z era leído por el ADS1115. El tiempo entre mediciones fue fijado a 20 minutos.

```
Date,Time,ADC_1,Ardui_1,  
15/6/18,16:31:4,2015,359,  
15/6/18,16:53:13,1795,336,  
15/6/18,17:15:40,1771,330,  
15/6/18,17:38:7,2802,365,  
15/6/18,18:0:34,2386,347,  
15/6/18,18:23:0,1689,317,  
15/6/18,18:45:27,3097,367,  
15/6/18,19:7:54,2476,342,  
15/6/18,19:30:21,2548,343,  
15/6/18,19:52:48,2584,345,  
15/6/18,20:15:15,2650,345,  
15/6/18,20:37:42,2654,345,  
15/6/18,21:0:10,2707,346,  
15/6/18,21:22:37,3054,358,  
15/6/18,21:45:4,2404,334,  
15/6/18,22:7:31,2470,334,  
15/6/18,22:29:58,2428,332,  
15/6/18,22:52:26,2824,347,  
15/6/18,23:14:53,2716,343,  
15/6/18,23:37:20,2615,339,  
15/6/18,23:59:46,2634,340,  
16/6/18,0:22:13,2664,341,  
16/6/18,0:44:39,3062,356,  
16/6/18,1:7:6,2062,317,  
16/6/18,1:29:32,2059,317,  
16/6/18,1:51:59,2058,317,  
16/6/18,2:14:25,2054,316,  
16/6/18,2:36:51,2054,316,  
16/6/18,2:59:18,2051,316,
```

Figura 8.1: Archivo generado en la SD para almacenar los resultados

En el portal desarrollado, se han procesado los datos y se han convertido a sus respectivas unidades, a continuación se mostrará un tablón en el que se aprecia su estado actual así como su progresión a lo largo del tiempo:

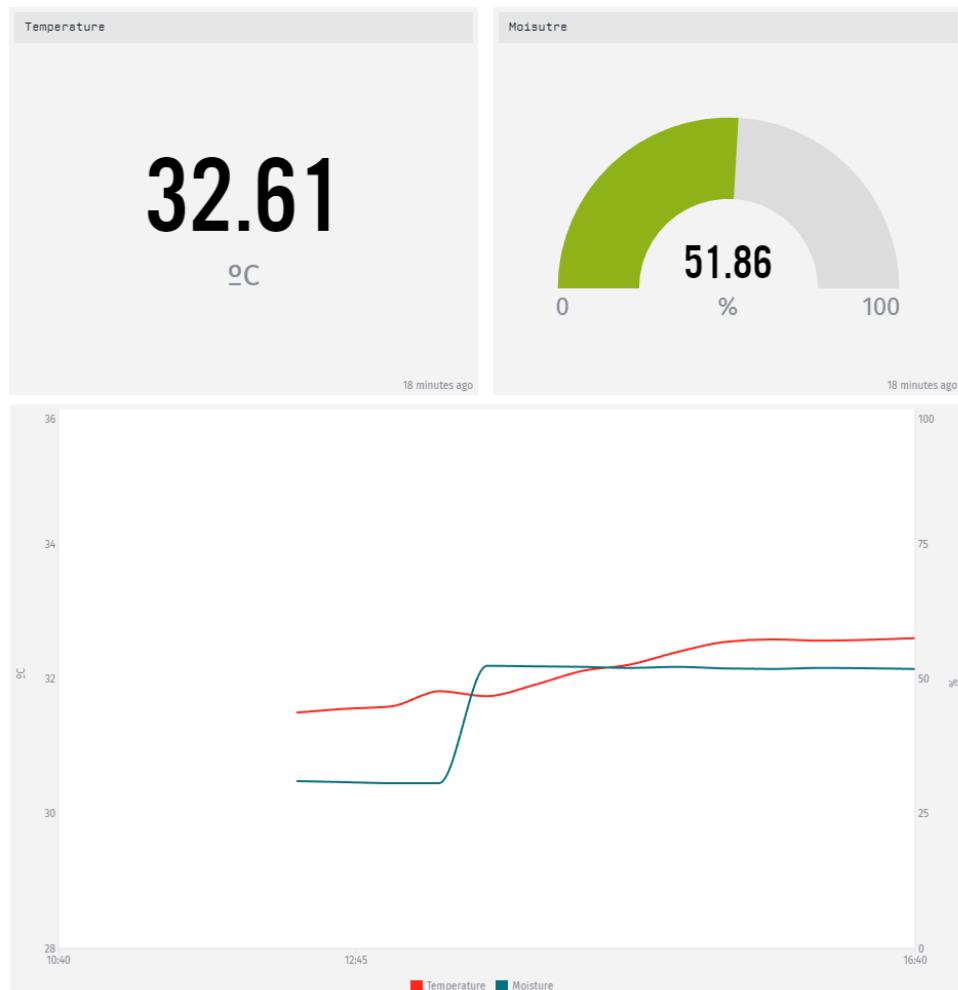


Figura 8.2: Resultados visualizados en el portal web.

Se puede apreciar como incrementó el porcentaje de humedad cuando se realizó un riego en torno a las 13:30. La temperatura se mantiene más o menos constante debido a que el sistema estaba monitorizando una planta en interior.

8.1.2. Generador de señales

Para ver el resultado de las funciones generadas, se ha usado el osciloscopio para tener mayor número de muestras por segundo. Se ha de tener en cuenta que las señales son generadas de la siguiente forma: El DAC fija el voltaje entre [0,5] V y posteriormente se hacen pasar por la etapa analógica de amplificación para obtenerlas del rango ± 15 V. No se han usado amplificadores operacionales *rail-to-rail* en la etapa de amplificación por lo que se puede observar la señal recortada en ambos extremos:

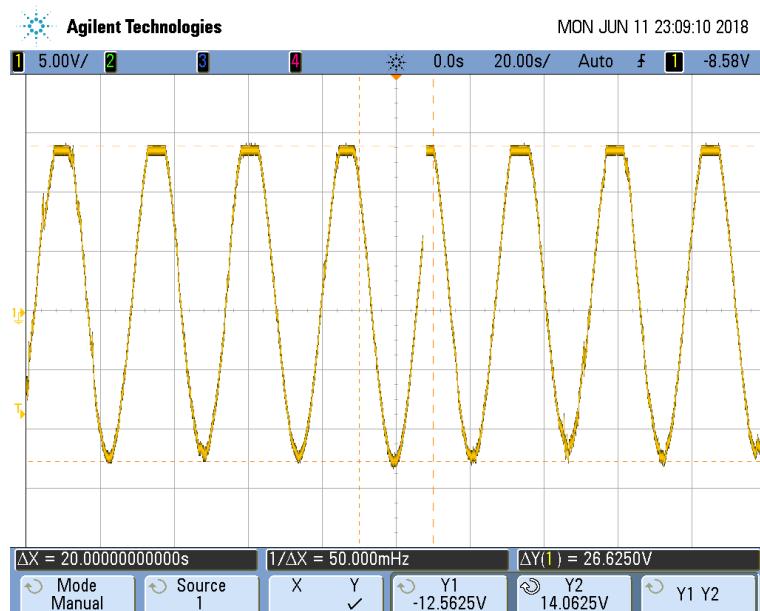


Figura 8.3: Resultado de la señal seno generada por el DAC

Si se reduce el rango de voltajes, se obtendría sin problema la señal completa:

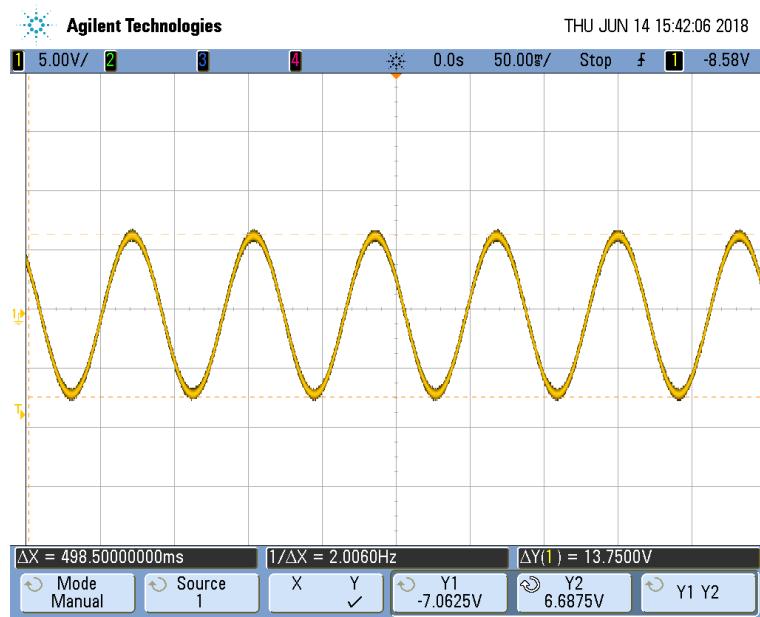


Figura 8.4: Resultado de la señal seno generada por el DAC

También se han generado señales diente de sierra y triangulares:



Figura 8.5: Resultado de la señal rampa generada por el DAC

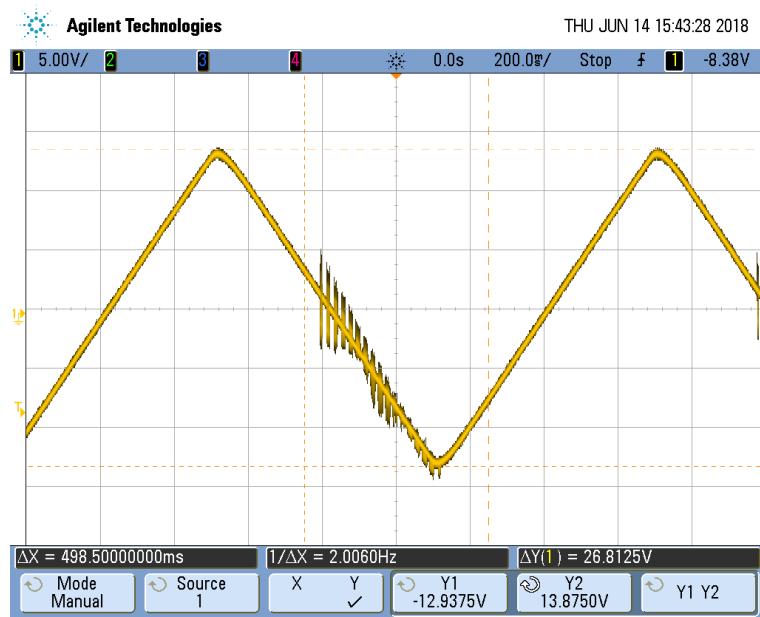


Figura 8.6: Resultado de la señal triangular generada por el DAC

8.1.3. SMU

A continuación se mostrarán las medidas realizadas con la SMU. La primera medición realizada fue comprobar la curva I-V de una resistencia de 100 kΩ:

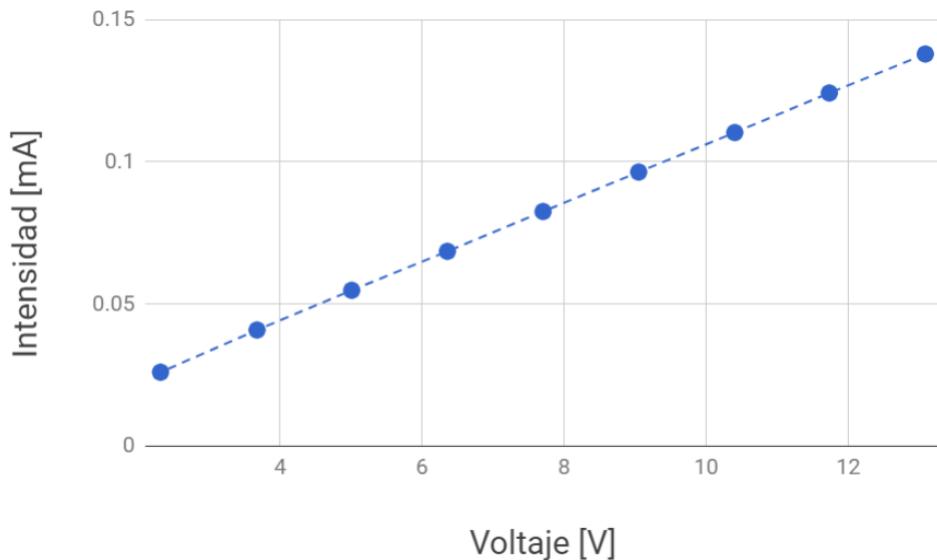


Figura 8.7: Gráfica I-V de una resistencia de $100\text{ k}\Omega$

Se ha tomado como corriente positiva aquella que va desde la salida inversora del amplificador operacional hasta la tensión generada por el DAC y posteriormente amplificada. Como era de esperar, se obtiene un comportamiento lineal. La recta obtenida es:

$$y = 0,025 + 0,010x \quad (8.1)$$

Si se calcula el valor de la resistencia aplicando la ley de Ohm en alguno de los puntos, se realizará el ultimo punto como ejemplo, se obtendrá el valor de la resistencia:

$$R = \frac{V}{I} = \frac{13,2\text{ V}}{0,135\text{ mA}} = 97,77\text{ k}\Omega \quad (8.2)$$

Teniendo en cuenta que la resistencia usada tiene una tolerancia de $\pm 5\%$ se considera un resultado certero.

Posteriormente se caracterizó, mediante su curva I-V, de nuevo, un

transistor de canal-P, el ZVP4424A, pero el barrido en voltaje que se pudo realizar fue pequeño debido a su gran resistencia interna $R_{DS(ON)}$. El transistor se estudió en la siguiente configuración:

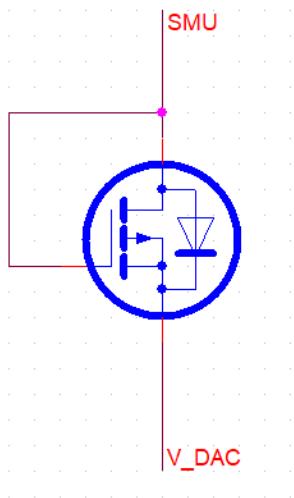


Figura 8.8: Gráfica I-V de un ZVP4424

Tras evaluarlo, estos fueron los resultados obtenidos.

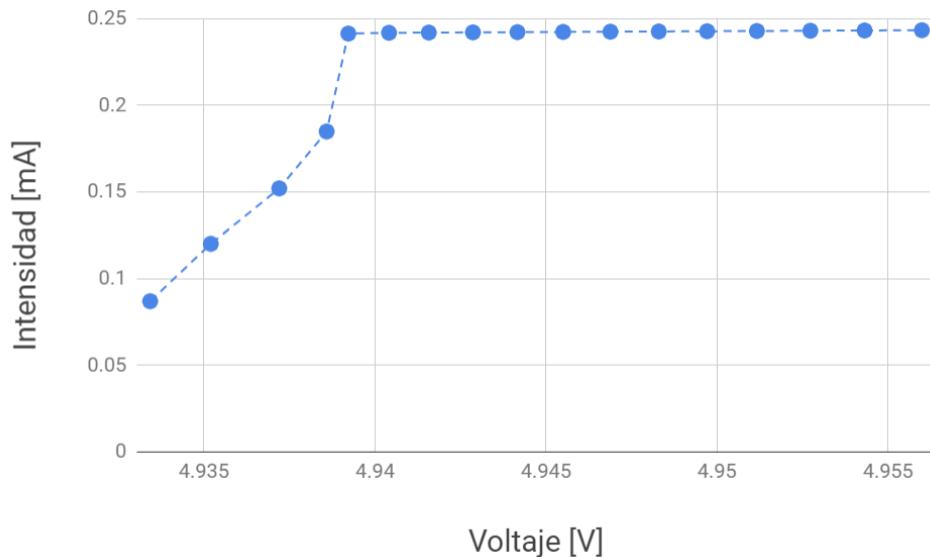


Figura 8.9: Gráfica I-V de un ZVP4424

Se puede ver que se satura en corriente rápidamente debido a que las características propias del transistor (voltaje de saturación, resistencia interna, ...) no son las más adecuadas para evaluarlo con la SMU realizada ya que el valor en tensión que hay en las dos entradas del amplificador es de 1.26 V y viene fijado por el divisor de tensión usado. Para un estudio mejor del transistor, se deberían cambiar el valor de las resistencias que forman el divisor de tensión.

8.2. Conclusiones

La realización del sistema ha supuesto el diseño de circuitos electrónicos, selección de componentes, la programación del software, el diseño de la PCB y muchas más tareas han tenido que ser realizadas y superadas con el fin de conseguir el sistema de adquisición de datos realizado.

Cabe destacar que los objetivos propuestos para el proyecto en la Sección 1.3 han sido cumplidos:

- Sistema portátil y con capacidad de autonomía.
- Incorporación de una SMU para la medida de corrientes.
- Conseguir la medición de entradas de dos rangos diferentes: [0,5] V y [-15,15] V.
- Lograr incorporar al sistema un módulo para el envío de los datos.
- Capacidad de almacenar los datos en una tarjeta de memoria.

Cabe destacar que el hecho de que el sistema este capacitado para enviar datos a través de un modulo GSM/GPRS lo hace un sistema muy versátil y Vanguardista.

Incluso, se ha realizado un portal para visualizar los datos en tiempo real, lo cual, no era uno de los objetivos iniciales pero fue desarrollado para darle un añadido al sistema.

8.3. Trabajo futuro

Se ha de tener en cuenta que el principal objetivo de este trabajo fin de grado era desarrollar el hardware con la idea de que otro compañero realizase otro proyecto fin de grado en el que realizaría una aplicación para el sistema, por lo que estas son algunas de las líneas de trabajo futuro:

- **Optimización de los amplificadores operacionales:** Como se ha podido comprobar, el uso de amplificadores operacionales los cuales no eran *rail-to-rail* ha supuesto algunos problemas en el sistema.

El uso de un amplificador *rail-to-rail* como el LT1366 mejoraría las señales que el sistema genera y también ampliaría el rango en el que el sistema de adquisición de datos es capaz de medir. Dicho cambio no se introdujo en el trabajo debido a que el pedido de LT1366 no llegó a tiempo.

- **Capacidad de fijar la configuración a través del portal:** Se podría desarrollar una tablón de configuración mediante botones o deslizantes de forma que el usuario pudiese configurar el sistema de forma remota y *online*.
- **Mostrar la localización del dispositivo:** Otro elemento útil a incluir en el portal del sistema es la localización del sistema de adquisición de datos en todo momento, ya que saber la posición en la que se ha realizado cada medida puede ser de gran utilidad en algunos casos.
- **Mejora del consumo del sistema:** El consumo de los sistemas siempre es uno de los grandes objetivos a minimizar y que tiene muchas formas de realizarse, se podría intentar reducir aun mas el tiempo que el dispositivo permanece dormido para reducir su consumo.

Capítulo 9

Apéndice

9.1. Presupuesto

Articulo	Cantidad	Precio/unidad [€]
Arduino Mega 2560	1	32.92
Caja - Hammondmfg 1598C	1	12.54
Convertidor Analógico-Digital - ADS1115	2	4.35
Convertidor Digital-Analógico - MCP4725	2	2.25
Pantalla LCD - Nokia 5110	1	2.79
Reloj en tiempo real - DS3231	1	2.57
Multiplexor 8 a 1 - ADG408	2	12.81
Comutador Analogico - DG419	1	1.22
Módulo Micro SD	1	2.29
Módulo SIM900	1	11.82
Referencia de tensión - LM4040	3	0.64
Amplificador operacional - LM358	5	0.73
MAX743	1	7.17
Condensadores	19	0.08
Bobinas	1	0.30
Resistencias	23	0.05
Placa Circuito Impreso	1	7.00
Diodo Schottky	2	0.13
Pines	91	0.02
Terminales de tornillo	5	0.36
Mano de obra	400 horas	15€/hora
Total		6128.53€

Tabla 9.1: Presupuesto del proyecto

9.2. Bill of Materials

Designator	Description	Quantity	Supplier	Reference
J1	Arduino Mega	1	Arduino	A000067
J2	Pin Header Straight 1x10	1	METZ CONNECT	31017124
J3,J4,J7,J8,J6	Arduino Pin Socket 8	4	HARWIN	D01-9922046
J5	Pin Header Straight 2x18	2	HARWIN	D01-9922046
J14	Pin Header Straight 1x05	1	HARWIN	D01-9922046
U1	Nokia 5110	1	ADAFRUIT	338
U2	DS3231	1	Adafruit	3013
U3	microSD	1	SparkFun	12761
U4	ADG408	1	ANALOG DEVICES	ADG408BNZ
U5	ADS1115	1	Adafruit	1085
U6,U7,U8,U9,U10	LM358	5	TEXAS INSTRUMENTS	LM358AN
U11	MCP4725	1	SparFun	12918
U12	DG419	1	VISHAY	DG419DJ-E3
IC1	MAX743	1	MAXIM INTEGRATED	MAX743CPE+
Q3,Q4,Q5	LM4040	3	TEXAS INSTRUMENTS	LM4040AIZ-5.0/NOPB
R1	Resistor 68kΩ	1	TE CONNECTIVITY	CFR16J68K
R2	Resistor 18kΩ	1	WELWYN	MFR3-18KFC
R5,R14	Resistor 200kΩ	2	VISHAY	MRS25000C2003FRP00
R6	Resistor 51kΩ	1	MULTICOMP	MF25 51K
R7,R15	Resistor 47kΩ	2	WELWYN	MFR4-47KFI
R8,R12,R16,R4	Resistor 100kΩ	4	MULTICOMP	MCRC100G104KB-RH
R9,R20,R3	Resistor 22kΩ	3	MULTICOMP	MCRC100G223KB-RH
R10,R11,R18,R23	Resistor 10kΩ	4	WELWYN	MFR3-10KFC
R13,R19	Resistor 50kΩ	2	VISHAY	CMF7050K000FKEB
R17	Resistor 270kΩ	1	MULTICOMP	MCRC100G274KB-RH
R21	Resistor 1kΩ	1	WELWYN	MFR3-1K0FC
R22	Resistor 330Ω	1	TE CONNECTIVITY	CFR16J330R
C1,C5	Capacitor 0.01μF	2	VISHAY	BFC237041103
C2	Capacitor 0.1μF	1	VISHAY	BFC237021104
C3	Capacitor 10μF	1	VISHAY	BFC237021106
C4	Capacitor 1μF	1	VISHAY	BFC237021105
C6,C7,C11,C12,C13,C14, C15,C16,C17,C18,C19	Capacitor 100nF	11	KEMET	C320C104K5R5TA
C8,C9,C10	Capacitor 100μF	3	VISHAY	BFC237021107
D1,D2	Schottky Diode	2	STMICROELECTRONICS	1N5817
L1,L2	Inductor 100μH	2	PANASONIC ELECTRONIC	ELLCTP101MB

Tabla 9.2: Bill of Materials

Bibliografía

- [1] Rasha Abbasi y col. “The IceCube data acquisition system: Signal capture, digitization, and timestamping”. En: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 601.3 (2009), págs. 294-316.
- [2] Francesco Adamo y col. “FFT test of A/D converters to determine the integral nonlinearity”. En: *IEEE Transactions on Instrumentation and Measurement* 51.5 (2002), págs. 1050-1054.
- [3] SA Arduino. “Arduino”. En: *Arduino LLC* (2015).
- [4] Yusuf Abdullahi Badamasi. “The working principle of an Arduino”. En: *Electronics, computer and computation (icecco), 2014 11th international conference on*. IEEE. 2014, págs. 1-4.
- [5] David San Segundo Bello y col. “An interface board for the control and data acquisition of the Medipix2 chip”. En: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 509.1-3 (2003), págs. 164-170.
- [6] Jean-Pierre Bono. *Multi-processor system having a watchdog for interrupting the multiple processors and deferring preemption until release of spinlocks*. US Patent 7,162,666. Ene. de 2007.
- [7] Jonathan C Borg, Xiu-Tian Yan y Neal P Juster. “A KICAD tool for proactive exploration support to ‘Design Synthesis for Multi-X’”. En: *Knowledge Intensive Computer Aided Design*. Springer, 2000, págs. 295-322.

- [8] Kuor-Hsin Chang. "Bluetooth: a viable solution for IoT?[Industry Perspectives]". En: *IEEE Wireless Communications* 21.6 (2014), págs. 6-7.
- [9] JP Charras. "KiCAD EDA Software Suite". En: URL: <http://www.kicad-pcb.org> ().
- [10] Chris CP Cheung y col. "Multi-channel pre-beamformed data acquisition system for research on advanced ultrasound imaging methods". En: *IEEE transactions on ultrasonics, ferroelectrics, and frequency control* 59.2 (2012), págs. 243-253.
- [11] Douglas Crockford. "The application/json media type for javascript object notation (json)". En: (2006).
- [12] Laura Dabbish y col. "Social coding in GitHub: transparency and collaboration in an open software repository". En: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM. 2012, págs. 1277-1286.
- [13] ZHAO Da_cheng y JIA Hai_yan. "Short Message Transmission and Receiving with AT Commands [J]". En: *Journal of Information Engineering University* 2 (2004), pág. 025.
- [14] Paul G Davis, Quoc B Huynh y John C Peck Jr. *Serial peripheral interface*. US Patent 8,589,717. Nov. de 2013.
- [15] Jerry Doland y Jon Valett. "C style guide". En: (1994).
- [16] Maurizio Di Paolo Emilio. *Embedded systems design for high-speed data acquisition and control*. Springer, 2016.
- [17] Brian Evans. *Beginning Arduino Programming*. Apress, 2011.
- [18] Behrouz A Forouzan y Sophia Chung Fegan. *TCP/IP protocol suite*. McGraw-Hill Higher Education, 2002.
- [19] Mei-zhen GAO, Yu-min MAO y Jin-jun LIU. "Interface design of AVR microcomputer with serial clock DS3231". En: *International Electronic Elements, China* 5 (2007), págs. 15-18.

- [20] C Gaudin y col. “A wireless high-speed data acquisition system for geotechnical centrifuge model testing”. En: *Measurement Science and Technology* 20.9 (2009), pág. 095709.
- [21] Roy Gelbard, Nava Pliskin e Israel Spiegler. “Integrating system analysis and project management tools”. En: *International Journal of Project Management* 20.6 (2002), págs. 461-468.
- [22] Ravender Goyal. “Managing signal integrity [PCB design]”. En: *IEEE spectrum* 31.3 (1994), págs. 54-58.
- [23] Paul R Gray y Robert G Meyer. “MOS operational amplifier design-a tutorial overview”. En: *ieee journal of solid-state circuits* 17.6 (1982), págs. 969-982.
- [24] Timo Halonen, Javier Romero y Juan Melero. *GSM, GPRS and EDGE performance: evolution towards 3G/UMTS*. John Wiley & Sons, 2004.
- [25] Roland Holcer y Linus Michaeli. “DNL ADC testing by the exponential shaped voltage”. En: *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*. Vol. 1. IEEE. 2001, págs. 693-697.
- [26] Johan H Huijsing y Daniel Linebarger. “Low-voltage operational amplifier with rail-to-rail input and output ranges”. En: *IEEE Journal of Solid-State Circuits* 20.6 (1985), págs. 1144-1150.
- [27] Texas Instruments. *LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers*. 2005.
- [28] Texas Instruments. “LM4040-N/LM4040Q-N Precision Micropower Shunt Voltage Reference”. En: *Technical Document, Literature* 11323 (2000).
- [29] John Keown. *OrCAD PSpice and circuit analysis*. Prentice Hall Upper Saddle River, 2001.
- [30] Brian Kernighan y Dennis M Ritchie. *The C programming language*. Prentice hall, 2017.

- [31] Hermann Kopetz. "Internet of things". En: *Real-time systems*. Springer, 2011, págs. 307-323.
- [32] Lukas Kull y Thomas H Toifl. *Analog-digital converter*. US Patent 9,191,018. Nov. de 2015.
- [33] M Lamontagne, DC Bradley y ED Lemaire. "Data acquisition and analysis system on micro computer for biomechanical studies". En: *Journal of Biomechanics* 22.10 (1989), pág. 1044.
- [34] Claudiu Lung y col. "Wireless data acquisition system for IoT applications". En: *Carpathian Journal of Electronic and Computer Engineering* 6.1 (2013), pág. 64.
- [35] Marcello A Gómez Maureira, Daan Oldenhof y Livia Teernstra. "ThingSpeak—an API and Web Service for the Internet of Things". En: *Retrieved7/11/15World WideWeb, http://www. Mediatechnology. leiden. edu/images/uploads/docs/wt2014_ thing speak. pdf* (2011).
- [36] Raphael Mukaro y Xavier Francis Carelse. "A microcontroller-based data acquisition system for solar radiation and environmental monitoring". En: *IEEE transactions on instrumentation and measurement* 48.6 (1999), págs. 1232-1238.
- [37] R Palit y col. "A high speed digital data acquisition system for the Indian National Gamma Array at Tata Institute of Fundamental Research". En: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 680 (2012), págs. 90-96.
- [38] John P Porter. *Data acquisition system*. US Patent 3,999,046. Dic. de 1976.
- [39] Ahmed G Radwan, Ahmed M Soliman y Ahmed S Elwakil. "First-order filters generalized to the fractional domain". En: *Journal of Circuits, Systems, and Computers* 17.01 (2008), págs. 55-66.
- [40] Christopher G Regier. *Source-measure unit based on digital control loop*. US Patent 7,903,008. Mar. de 2011.

- [41] S Rosiek y FJ Batlles. “A microcontroller-based data-acquisition system for meteorological station monitoring”. En: *Energy Conversion and Management* 49.12 (2008), págs. 3746-3754.
- [42] Apostolis K Salkintzis, Chad Fors y Rajesh Pazhyannur. “WLAN-GPRS integration for next-generation mobile data networks”. En: *IEEE Wireless communications* 9.5 (2002), págs. 112-124.
- [43] Philips Semiconductors. “The I2C-bus specification”. En: *Philips Semiconductors* 9397.750 (2000), pág. 00954.
- [44] Jesper Steensgaard. “Bootstrapped low-voltage analog switches”. En: *Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on*. Vol. 2. IEEE. 1999, págs. 29-32.
- [45] Christopher Swannack y col. *Data acquisition system*. US Patent App. 09/800,888. Jul. de 2002.
- [46] Suet Fong Tin y col. *Dual-barrel, connector jack and plug assemblies*. US Patent 7,914,344. Mar. de 2011.
- [47] Jeffrey Travis y Jim Kring. *LabVIEW for everyone: graphical programming made easy and fun*. Prentice-Hall, 2007.
- [48] Sanroku Tsukamoto. *Digital analog converter*. US Patent 7,928,880. Abr. de 2011.
- [49] SIM900 SpyTracer User. “Guide V1. 00”. En: (2003).
- [50] Feng Xia y col. “Internet of things”. En: *International Journal of Communication Systems* 25.9 (2012), págs. 1101-1102.
- [51] Fu-I Yang. *Universal USB power supply unit*. US Patent 6,362,610. Mar. de 2002.
- [52] Ichiro Yokomizo, Sachito Horiuchi y Mayuka Matsumae. *DC/DC converter*. US Patent 6,400,211. Jun. de 2002.