UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO FIN DE MÁSTER

DESARROLLO INTEGRAL Y ENSAYOS DEL ESCUDO DOCENTE UMA_AEB_V2.0.0

Máster en Ingeniería Mecatrónica

Autor: Javier Martínez Lahoz

Tutores: Juan Antonio Fernández Madrigal y Andrés Góngora González

Málaga, Marzo de 2019

Declaración de originalidad

El alumno, Javier Martínez Lahoz, garantiza que este trabajo de investigación y desarrollo es original y ha sido realizado respetando los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

1. Resumen

Un escudo, o *shield*, es una placa electrónica que se acopla a un hardware ya existente y amplia sus capacidades. En este trabajo se desarrolla un escudo de Arduino con carácter docente para la Universidad de Málaga (UMA), este se denomina *UMA_AEB_V2.0.0*. La finalidad principal del proyecto es facilitar y acelerar las prácticas de diversas asignaturas en los laboratorios al suprimir el montaje y los posibles errores derivados del mismo.

El escudo se empleará en asignaturas y grados provenientes de distintas áreas de la ingeniería. Las funcionalidades mínimas necesarias de la placa se han definido en colaboración con el profesorado de la UMA de acuerdo a las prácticas de cada materia. Estas funciones son: estudio del comportamiento de componentes y filtros pasivos, gestión de una interfaz manual, control en bucle cerrado de motores, uso de integrados digitales y comunicación mediante buses digitales.

Hasta este momento se emplea en la UMA un escudo que no cumple con todos los requisitos mínimos ahora establecidos: se trata del *UMA_AEB_V1.1.0*. El producto desarrollado en el presente TFM mantiene una compatibilidad de funcionalidades con dicho modelo, pero amplía sus prestaciones. Se han eliminado los comportamientos indeseados, se han incluido convertidores de precisión y se ha permitido un trabajo simultáneo de todos los circuitos. Sin embargo, la novedad más importante consiste en la posibilidad de operar la placa sin ningún tipo de manipulación física. De forma remota ahora es posible reconfigurar los filtros pasivos, verificar los subsistemas y calibrar los convertidores y circuitos analógicos.

Estas nuevas propiedades posibilitan situaciones que resultan imposibles con el *UMA_AEB_V1.1.0*. Por ejemplo, los alumnos en modalidades no presenciales podrán desarrollar las prácticas sin necesidad de desplazarse hasta el centro. A su vez, el profesorado podrá realizar pruebas y verificar el estado del escudo a distancia.

Todas estas características permiten dedicar más tiempo de las prácticas a los aspectos relevantes de las mismas. Esto facilita tanto las labores docentes del profesorado como el aprendizaje de los alumnos. Y, en definitiva, supone una mejora en los resultados académicos y en la satisfacción general.

A fin de que el nuevo escudo llegue al mayor número de usuarios, se ha puesto especial interés en su coste económico. Se ha decidido investigar tecnologías alternativas para sustituir los componentes de mayor precio, destacando las fuentes de alimentación y el convertidor general de impedancia. Gracias a esto, se ha logrado mantener el coste de cien unidades por debajo de 40€ por placa. Este coste incluye los componentes, la fabricación y el montaje. También se ha minimizado el coste de desarrollo empleando exclusivamente programas con licencia *OpenSource*.

En cuanto al software para la placa, se han desarrollado ejemplos que permiten a los alumnos analizar y comprender los métodos de control y comunicación para cada subsistema. También se ha incluido una clase con métodos que permiten acceder a todas las capacidades de la placa. De cara al profesorado esto permite desarrollar programas complejos a gran velocidad. A la vez, se permite desvincular las prácticas del nivel de programación de los alumnos, ya que se pueden realizar tareas complejas con métodos ya implementados.

Tras verificar el correcto funcionamiento del *UMA_AEB_V2.0.0*, se decide realizar una serie de ensayos de capacidades máximas. En estas pruebas se determina que las características de velocidad y robustez de todos los circuitos del escudo resultan satisfactorias para el correcto funcionamiento de las prácticas. Además, se realizan ensayos de emisiones electromagnéticas (EMIs), donde se comprueba que el escudo tiene un buen comportamiento en radiación, conducción y recepción. Por lo tanto, podría plantearse en el futuro una posible expansión fuera de la UMA.

2. Abstract

A shield is an electronic board that plugs in to an existing hardware and increases its capabilities. In this essay it is developed an Arduino shield with teaching goals for Universidad de Málaga (UMA). This board is named *UMA_AEB_V2.0.0*. The primary goal of this project is to ease and speed up the laboratories practices by removing the assembly and its issues.

This shield will be used in multiples subjects from diverse engineering areas. The minimum functionalities needed were defined in collaboration with the UMA's professors according to each subject's practices. This functionalities are: studying passive filter and component's behavior, managing a manual interface, controlling a motor in closed loop, using digital IC and communicating through digital buses.

To this moment in UMA it is used a shield that does not fulfill all the minimum requirements now established; it is the *UMA_AEB_V1.1.0*. The product developed in this project keeps a compatibility with the old one, but it improves its functionalities and performance.

The unwanted behaviors have been removed, precision converters have been included and simultaneous work of all the circuits has been allowed. However, the most important innovation is the possibility of utilizing the board without any kind of physical manipulation. Remotely now it is possible to reconfigure the passive filters, verify the subsystems and calibrate the converters and analog circuits.

These new properties allow scenarios that used to be impossible while using the *UMA_AEB_V1.1.0*. For example, the students will be able to carry out the practices without the need of being at the university. In the same way, the professors will be able to test and verify the shield's status without being present.

All this characteristics allow to spend more time from the practices in the relevant aspects of themselves. This eases the professor's work as well as the students' learning. In conclusion, it improves the academical results and the general satisfaction.

The economical cost of the shield has been a main focus of the project in order to get to as many users as possible. Alternatives technologies have been investigated to change the most expensive components, highlighting the switch mode power supplies and the gyrator. Thanks to this, it has been managed to keep the shield's price under 40€ per unit. This cost includes the components, the manufacturing and the assembly. The developing cost has also been minimized by using exclusively *Open-Source* software.

Software examples have been coded in order to allow the students to understand and analice the control and communication methods for each subsystem. There has been included a class with methods that allow to exploit all the shield's capabilities. This way the professors can develop complex code at high speed. At the same time, this makes possible to disengage the practices from the programming skills of the students.

After verifying the proper functioning of the *UMA_AEB_V2.0.0*, a series of maximum capabilities test have been accomplished. This test determine that the speed and reliability of all the circuits in the shield are satisfactory for the practices. Also EMI's test are carried out, in those it is proved that the shield has a good performance in EMIs radiation, conduction and reception. Therefore, there could be a future expansion outside the UMA.

3. Objetivo

El objetivo del presente trabajo consiste en desarrollar un escudo para la familia de placas microcontroladoras Arduino. Este escudo recibe la denominación de *UMA_AEB_V2.0.0*. No se realizará un diseño libre de restricciones, puesto que se debe mantener la compatibilidad funcional con un modelo ya existente y en uso por la UMA, el *UMA_AEB_V1.1.0* . Esta herramienta se emplea como apoyo a la docencia en asignaturas de Electrónica y Automática de la Universidad de Málaga con resultados muy satisfactorios. Sin embargo, durante las prácticas con este escudo se han detectado algunos comportamientos incorrectos y carencias que se desean solucionar.

El primer objetivo de este TFM consiste en solucionar las interferencias que se generan entre algunos subsistemas. Estas impiden que determinados componentes se puedan usar al mismo tiempo, reduciendo las capacidades reales de la placa. En el nuevo modelo todos los subsistemas deben permitir un funcionamiento simultáneo sin que exista ninguna alteración en su comportamiento.

Durante el diseño debemos tener en cuenta que cada vez son más comunes los modelos de Arduino con tensión de alimentación de 3V3, además de los clásicos de 5V. Por lo tanto, el segundo objetivo consiste en lograr que el escudo funcione indistintamente en ambos casos.

El profesorado de la UMA también ha fijado como objetivo que la placa se pueda operar de forma remota. La motivación en este caso es permitir que los alumnos en modalidades no presenciales puedan realizar las prácticas sin la necesidad de desplazarse hasta el centro. Por lo tanto, todas las funcionalidades de la placa deben estar disponibles sin realizar ningún tipo de manipulación física sobre la misma (con la excepción de los pulsadores).

Debido a la mejora de resultados académicos experimentada con otros modelos, el centro pretende ampliar el uso de esta herramienta a más asignaturas y grados. Sin embargo, esto vendrá determinado por la inversión económica necesaria. Para lograr que el escudo llegue a un mayor número de alumnos se debe minimizar el coste de fabricación.

Nuestro siguiente paso será el desarrollo del software. De cara al profesorado se debe programar una librería que permita programar la placa y verificar su comportamiento de forma rápida y sencilla. Si el profesor lo considera adecuado, los alumnos también podrían aprovechar esta librería. Esto resulta especialmente interesante en los casos en los que la programación no es un aspecto relevante de la práctica.

Los estudiantes también deben aprender a desarrollar el software necesario para operar con la placa, sin necesidad de recurrir a la librería. Por lo tanto, desarrollaremos una serie de ejemplos para que los alumnos puedan comprender el funcionamiento software de los diferentes subsistemas y ampliarlo por si mismos.

Agrupamos los objetivos de desarrollo de la nueva placa en cuatro frentes:

- Compatibilidad interna (entre subsistemas) y externa (con cualquier Arduino).
- Operación remota con todas las funcionalidades.
- Librería de desarrollo rápido en C/C++ y ejemplos.
- Mínimo coste económico.

4. Introducción

La base, sobre la que se sustenta el nuevo diseño, se divide en tres apartados claramente diferenciados: antecedentes, tecnologías y herramientas. A continuación, se va a realizar un estudio acerca de los aspectos más relevantes de cada uno de ellos.

4.1 Antecedentes

Como ya se ha comentado en el capítulo anterior, la UMA emplea un escudo de Arduino como apoyo en algunas prácticas de laboratorio. Este *shield* fue desarrollado por la propia Universidad de Málaga. Se trata del proyecto de innovación educativa PIE15-93 financiado por dicha universidad durante el bienio 2016-2017, denominado *UMA_AEB_V1.1.0*.

El análisis de los circuitos electrónicos de esta placa sirve como punto de partida para el nuevo diseño. Se comienza analizando el esquemático general, mostrado en la figura 1. En esta se pueden ver los subsistemas en los que se divide la placa y las relaciones entre ellos.

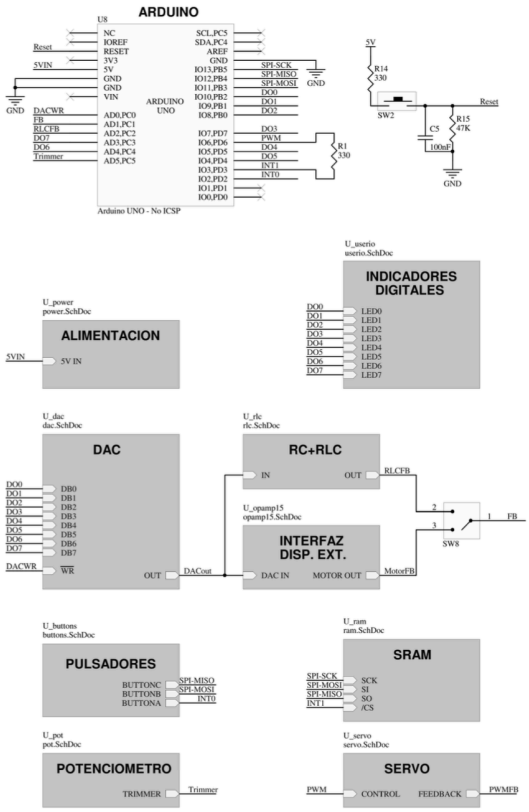


Fig. 1: Esquemático general UMA AEB V2. UMA PIE15-93

Los subsistemas de este modelo y sus funciones son:

- Conversor digital-analógico (DAC) con salida a:
  - Circuito RC/RLC configurables manualmente.
  - Señal de ±15V para control de motor.
- Lectura de ambas señales mediante el ADC de Arduino
- Control de servomotor mediante PWM y realimentación en corriente.
- Memoria RAM SPI.
- Interfaz manual: 8 LEDs, 3 pulsadores y un potenciómetro.

Gracias los comentarios del profesorado acerca de los problemas detectados, se puede identificar el origen algunas incompatibilidades entre circuitos. Gran parte de estas incompatibilidades provienen de la reducida cantidad de entradas o salidas de Arduino disponibles. Esto viene provocado por la necesidad de algunas prácticas de disponer de ocho indicadores LED controlados por sendos pines digitales, y no por un bus digital.

En el diseño del *UMA_AEB_V1.1.0* se decidió compensar esta falta de pines empleándolos en dos circuitos a la vez. Esto provoca que los subsistemas implicados no puedan controlarse de forma simultánea.

Por ejemplo, los pines empleados para la interfaz LEDs también se usan como entrada de datos del conversor D/A. Las líneas de control de la memoria RAM (un bus SPI) también realizan una segunda función, en este caso, la detección de pulsaciones en los botones; esto provoca que si se pulsa algún botón durante una transmisión SPI, esta puede quedar corrupta.

En este esquema también se puede encontrar el circuito del pulsador de *reset*. Se ha comprobado que este pulsador no actúa de forma correcta. Estudiando el funcionamiento del circuito de reinicio de Arduino se encuentra el problema: debería funcionar con lógica negativa (un cero lógico activa el reinicio), pero esta implementado con lógica positiva (reinicio con el uno lógico).

Este diseño empleaba una misma entrada analógica para realizar la lectura de dos subsistemas distintos (RC/RLC y OpAmp15). Para seleccionar cual de las dos señales llegaba al ADC integrado en el Arduino se empleaba un interruptor manual. Pese a no causar problemas, no se trata de una solución aceptable para el nuevo modelo.

Los esquemas electrónicos de los diferentes subsistemas se encuentran en el "Esquemas del *UMA_AEB_V1.1.0*". El comportamiento de estos circuitos se considera adecuado, sin embargo, muchos de ellos deben modificarse de cara al diseño del *UMA_AEB_V2.0.0*. Entre las características que no van a continuarse encontramos: uso de un DAC con entrada en paralelo, reconfiguración de los filtros RC/RLC mediante interruptores manuales, uso de dos amplificadores operacionales para cada etapa (entrada o salida) del OpAmp15 y montaje de una fuente de alimentación conmutada.

Otro aspecto a destacar de este *shield*, actualmente uso, es su altura. Esta se puede ver en el modelo 3D de la placa, mostrado en la figura 2. Para poder mejorar su compatibilidad con otros escudos debemos reducir la altura máxima de los componentes. Los de mayor altura son los condensadores electrolíticos, la bobina, los potenciómetros y la fuente de alimentación conmutada.

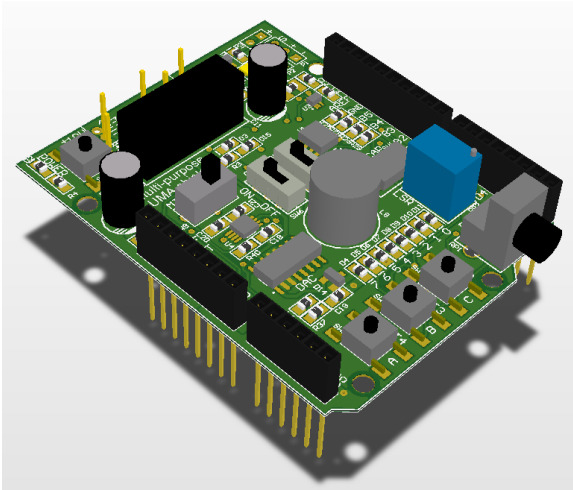


Fig. 2: Modelo 3D del UMA AEB V2. UMA PIE15-93

4.2 Tecnologías

En este apartado se va a realizar una introducción de las tecnologías más importantes para el desarrollo del proyecto. Para cumplir con todos los objetivos ha sido necesario investigar en: fuentes de alimentación, métodos de conmutación, tecnologías de sustitución de componentes pasivos y buses de comunicaciones. La base de estas tecnologías se expone a continuación.

Fuentes de alimentación

Existe multitud de tecnologías para convertir tensiones, y cada una de ellas puede contar con varias topologías. Muchas de ellas están destinadas a transformar la corriente alterna (AC) de la red eléctrica a corriente continua (DC) para el consumo de los aparatos electrónicos. Sin embargo, el *UMA_AEB_V2.0.0* debe generar una corriente continua a partir de otra de menor voltaje, también continua. Por lo tanto, se van a exponer los métodos de conversión DC/DC.

Convertidores DC/DC aislados

El *UMA_AEB_V1.1.0* empleaba un convertidor DC/DC aislado para generar la señal de ±15V a partir de 5V, concretamente el modelo *TMA 0515D* del fabricante *TracoPower*. Este componente es una fuente de alimentación de modo conmutado (SMPS) integrada aislada. Las SMPS integradas aisladas se emplean en multitud de aplicaciones gracias a su amplio abanico de posibilidades y buenas propiedades: tamaño y peso reducidos, aislamiento galvánico, capacidad de emplear entradas en AC y DC (según la etapa de entrada), amplio rango de tensiones de trabajo y gran eficiencia.

Estos componentes suelen incluir internamente todos los elementos necesarios para su funcionamiento, incluyendo el aislamiento y las etapas de filtrado. Aun así pueden aparecer problemas debidos a la conmutación que realizan para mantener la tensión de salida. Esta conmutación puede provocar ruidos electromagnéticos de alta frecuencia en dos rangos distintas. Por lo tanto, suele ser necesario un filtrado adicional a la salida del componente.

Las SMPS pueden emplear distintas topologías (Buck, Boost, Flyback, Half-Forward…). En el caso de las fuentes aisladas, como la serie *TMA*, un método de funcionamiento puede apreciarse en la figura 3. El ciclo de trabajo consiste en: filtrado de entrada, rectificado (en el caso de entrada AC), conmutación de alta frecuencia, transformación, rectificación de salida, filtrado de salida y control realimentado. El control toma medidas de la tensión de salida y, tras compararlo con una referencia, actúa sobre la conmutación de alta frecuencia.

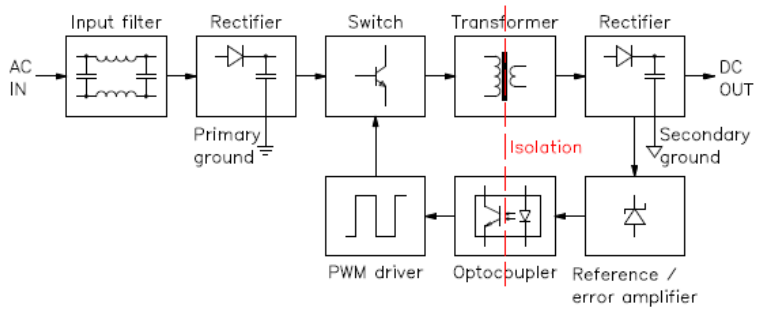


Fig. 3: Estructura interna de una SMPS integrada aislada con entrada CA. giangrandi.ch

En el mercado existen un amplio abanico de SMPS integradas con aislamiento que se pueden adaptar a gran variedad de situaciones. Sin embargo, suelen tener un precio elevado y, aunque son de un tamaño bastante reducido, siguen existiendo alternativas no aisladas de menor tamaño y coste. Otra alternativa, especialmente si se desea minimizar el coste, consiste en emplear soluciones no integradas, aunque el esfuerzo de diseño es mayor.

Convertidor Boost

El convertidor Boost es una topología de convertidor no aislado capaz de generar a su salida una tensión superior a la de su entrada. Para su funcionamiento requiere, en una configuración básica, de un diodo, un transistor, una bobina y un condensador. El funcionamiento del convertidor Boost se basa en alternar entre dos estados distintos: uno de carga y uno de entrega de energía. Se trata, por lo tanto, de una fuente de alimentación en modo conmutado.

En el estado de carga se almacena la energía procedente de la fuente en la bobina. Para ello se cortocircuita el inductor mediante la conmutación del transistor. Esto provoca que la intensidad que circula por el circuito aumente progresivamente, ya que la bobina se opone a los cambios bruscos de corriente. Cuanto mayor sea el valor inductivo, más lentamente aumentará la intensidad, pero también se almacenará una mayor cantidad de energía. Durante esta etapa, la carga se encuentra alimentada por el condensador, que se supone ya cargado.

En el segundo estado el transistor deja de conducir, por lo que la bobina queda conectada en serie con la carga. La corriente de una inductancia no puede variar de forma brusca, por lo que sigue circulando por el resto del circuito. Esta intensidad alimenta la carga y el condensador.

El paso entre los dos estados, mostrados en la figura 4, se suele realizar mediante un control que compara la tensión de salida con una referencia. Si se controlan de forma adecuada los tiempos de corte y conducción se puede lograr una tensión en la salida de casi cualquier valor, siempre y cuando los componentes la soporten.

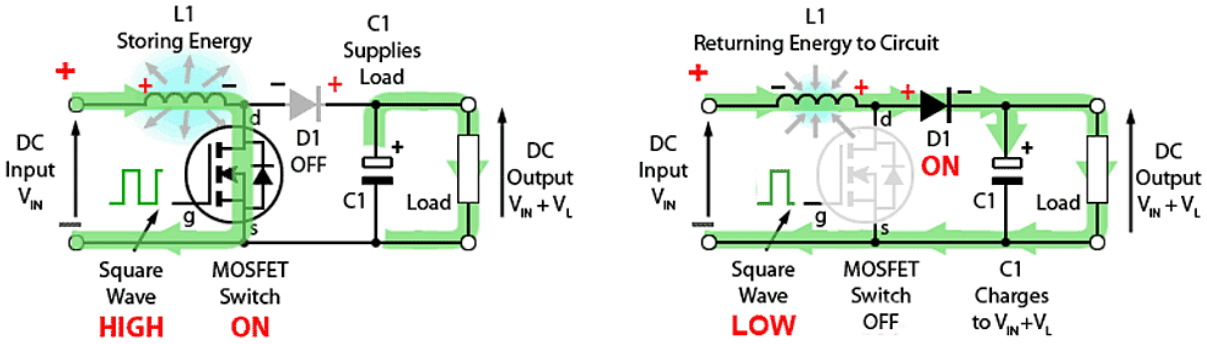


Fig. 4: Etapas de un convertidor Boost. Daniel Graff. LEIFRA

Por otro lado, resulta muy imprescindible contar con una buena etapa de filtrado antes y después del convertidor. Estos filtros se disponen para evitar que el ruido electromagnético, generado por la conmutación de alta frecuencia, afecte al resto del circuito.

Actualmente existen integrados de muy reducidas dimensiones que permiten implementar fácilmente este tipo de convertidores. Internamente cuentan con el transistor de conmutación y toda la electrónica de control; externamente necesitan de la bobina de acumulación de carga, un método de realimentación (normalmente un divisor de tensión) y las etapas de filtrado.

Multiplicador de tensión

Uno de los métodos más sencillos y baratos de aumentar el voltaje consiste en emplear un multiplicador de tensión. La escalera capacitiva es una topología de multiplicador de tensión que emplea diodos y condensadores para aumentar el voltaje de forma escalonada mediante una sucesión de etapas. Existen dos topologías de escalera capacitiva: las de entrada de AC y las de entrada de DC.

Existen topologías con entrada de tensión alterna, como la Cockcroft–Walton, que no requieren de ningún tipo de control para su funcionamiento. Cuando la entrada es de tensión continua, resultan necesarias otras topologías. Por ejemplo, en el caso del multiplicador Dickson, son necesarias dos señales cuadradas que van añadiendo tensión en cada etapa.

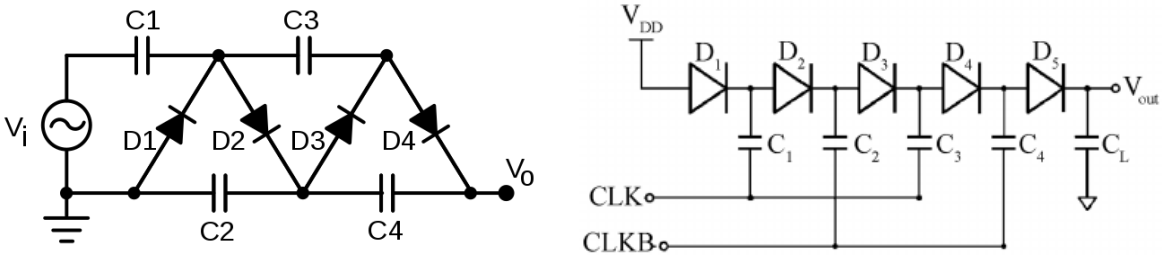


Fig. 5: Multiplicadores de tensión Cockcroft–Walton (izquierda) y Dickson (derecha). Wikiwand

De forma ideal, cada etapa aumenta la tensión final en el valor del pico de la tensión de entrada o en el valor de tensión de la señal cuadrada. Por ejemplo, se logra triplicar el voltaje en tres etapas, es decir, empleando únicamente tres diodos y tres condensadores. La tensión generada también puede ser negativa si se invierte el sentido de los diodos.

La gran ventaja de la topología Cockcroft–Walton es que cada componente solo debe soportar la tensión de su etapa, por lo que se puede lograr, teóricamente, cualquier valor de tensión por elevado que sea. En el caso de la topología Dickson, los condensadores si deben soportar la tensión total. En ambos casos el coste del multiplicador es muy reducido, ya que solo necesita diodos y condensadores.

La gran desventaja de este sistema es que la capacidad de entrega energética del mismo es muy limitada. Cuando aparece un consumo moderado en la tensión de salida, esta cae drásticamente. Para reducir este efecto, se puede aumentar la capacidad de los condensadores o la frecuencia de conmutación.

Interruptores

Para permitir el paso o no de una señal eléctrica se pueden emplear diversas tecnologías, aunque en la actualidad se suelen implementar soluciones basadas en semiconductores. Dentro de las tecnologías de semiconductores, tradicionalmente se empleaban los transistores bipolares. Sin embargo, en la electrónica actual estos han sido sustituidos por otras tecnologías más modernas; MOSFETs, IGBTs, SSR…

A continuación, se va a explicar brevemente el método de funcionamiento de dos tecnologías de semiconductores empleadas en el presente trabajo.

Transistores de efecto de campo metal-óxido-semiconductor

Estos transistores, comúnmente denominados MOSFETs por sus siglas en inglés, son actualmente los más empleados en electrónica. Esto se debe a sus buenas características como: fácil control, baja resistencia en conducción, alta velocidad de conmutación…

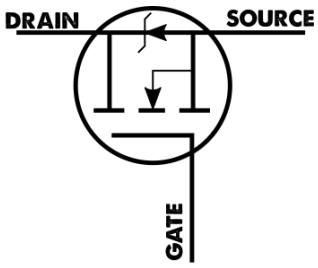


Fig. 6: Símbolo electrónico de un MOSFET. Norwegian Creations

Se suelen emplear referenciados a una tensión conocida, ya que su conmutación se controla mediante la diferencia de tensión entre la base y el colector (tipo P) o el emisor (tipo N). Sin embargo, este método de conmutación puede provocar problemas cuando ninguna de las tensiones de sus terminales sea constante, como se vera más adelante en este trabajo.

Otra característica que puede provocar problemas se debe a su construcción; ya que los MOSFETs conducen en sentido inverso cuando se ven sometidos a una tensión negativa. Esto se representa en su símbolo electrónico, mostrado en la figura 6, como un diodo en paralelo con el propio transistor.

Interruptores analógicos

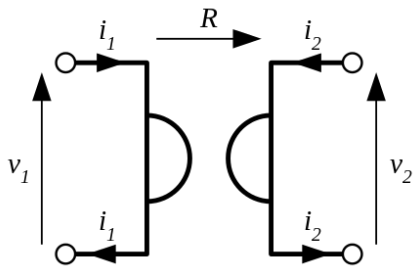
Los interruptores analógicos, *analog switch* (*AS*) en inglés, son componentes electrónicos que permiten una conmutación bidireccional, a diferencia de los transistores. Los *AS* emplean internamente transistores, pero también implementan una etapa de control para que funcionen indistintamente en ambos sentidos, tanto en corte como en conducción.

El principal problema de esta tecnología es que no es capaz de funcionar si las tensiones de sus terminales son superiores a su alimentación positiva o inferiores a su alimentación negativa. Aparte de esto, su comportamiento eléctrico es similar al de un relé, ya sea tradicional o de estado sólido, pero sin aislamiento.

Gyrator

En el *UMA_AEB_V1.1.0* se implementa una bobina como parte de un circuito RLC. Esta bobina es uno de los componentes más caros y voluminosos de la placa. Una de las posibles formas de optimizar de cara a la nueva placa consiste en sustituir esta bobina por otro componente, o circuito, más económico y pequeño. Se ha encontrado una tecnología que puede servir para este propósito: el *gyrator*.

Un *gyrator* es un elemento eléctrico ideal, pasivo, sin pérdidas y de dos puertos propuesto por Bernard D. H. Tellegen en 1948. El *gyrator* relaciona de forma proporcional, mediante un valor característico R, la diferencia de tensión entre los terminales del segundo puerto con la corriente que circula por el primer puerto; de forma similar relaciona la diferencia de tensión entre los terminales del primer puerto con la corriente, en sentido contrario, que circula por el segundo puerto. Todo ello se puede observar en la representación y las ecuaciones de la figura 7.



$$v_2 = R \cdot i_1$$

$$v_1 = -R \cdot i_2$$

Fig. 7: Comportamiento de un gyrator. Wikiwand

Su funcionamiento es parecido al de un transformador, de hecho, la colocación de dos *gyrators* en serie tendría el mismo comportamiento eléctrico que un transformador. También se puede lograr un comportamiento similar a una inductancia en un puerto al colocar un

elemento capacitivo en el otro puerto, o viceversa. Sin embargo, no existen *gyrators* pasivos, y deben ser construidos mediante elementos activos como transistores y amplificadores operacionales.

En la actualidad se emplean algunos *gyrators* activos, dada su gran utilidad. Entre las aplicaciones encontramos la implementación de impedancias de valor negativo o de inductancias de valores muy elevados. Una de las aplicaciones más típicas de los *gyrators* es sustituir bobinas de gran inductancia para miniaturización de circuitos. Mediante esta técnica se consigue un comportamiento más ideal, ahorro de peso, menores pérdidas por calor y menor coste económico frente a una bobina.

Bus de comunicaciones I2C

I2C (*Inter-Integrated Circuit*) es un bus serie de comunicación de datos desarrollado en 1982 por *Phillips Semiconductors*. El uso de este bus de comunicaciones se extendió rápidamente entre otras empresas y actualmente es uno de los buses más comunes de los microcontroladores.

Fundamentalmente se trata de un método de comunicaciones con un maestro y uno o más esclavos; aunque existen formas de trabajo multimaestro. Una de las principales ventajas de este bus es que solo requiere de dos líneas de señal, independientemente del número de nodos. La estructura del bus se muestra en la figura 8.

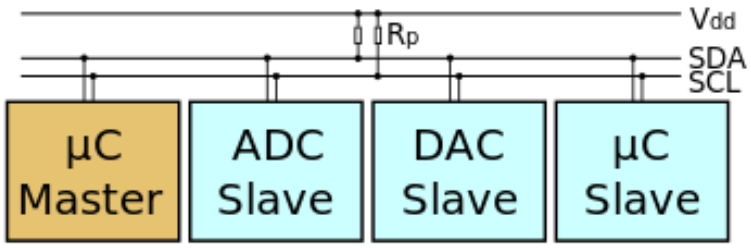


Fig. 8: Estructura de un bus I2C. Wikiwand

Una de las líneas, CLK, está controlada exclusivamente por el maestro y determina la velocidad de transmisión de los datos por el bus. La línea de datos, denominada SDA, puede estar controlada tanto por el maestro como por el esclavo. Esto depende de quién esté transmitiendo la información; sin embargo, el esclavo solo controlará la línea tras una petición de información generada previamente por el maestro. Además, el maestro deberá generar los pulsos de reloj, CLK, incluso cuando sea el esclavo el que transmita la información por el bus.

El primer mensaje de cada comunicación siempre es enviado por el maestro y hace referencia al destinatario del mensaje. Los mensajes siguientes dependen del comportamiento del nodo receptor y del objetivo de la comunicación. Además, existen mensajes estándar (apagado, encendido o reinicio generales) que permiten realizar determinadas acciones sobre todos los nodos esclavos a la vez.

La principal ventaja de este bus es su flexibilidad y su simplicidad en la capa hardware. Su principal problema es la velocidad máxima de transmisión de datos. Esta ha ido aumentando progresivamente con el paso del tiempo desde los 100 Kbits/s hasta los 5 Mbits/s. La velocidad máxima de la comunicación se encuentra relacionada directamente con las resistencias de *Pull-Up* (conectadas entre las líneas y la alimentación positiva) que deben montarse en las dos líneas del bus.

Las resistencias de *Pull-Up* provocan que el estado de reposo del bus sea en el estado eléctrico positivo (*Vdd*). Para realizar las comunicaciones los nodos fuerzan el bus al estado negativo (*Gnd*), representando un cero lógico, o lo dejan libre, representando un uno lógico.

El valor de las resistencias, junto con la capacidad del bus (que depende de los componentes y de las pistas), determina la velocidad a la que el bus retorna al uno lógico después de que el microcontrolador lo deje de forzar al cero lógico. Esta velocidad establece la capacidad máxima de transmisión de información del bus; a mayor resistencia menor velocidad de transmisión. Sin embargo, cuanto menor es el valor de la resistencia, peor es el comportamiento de cara a la compatibilidad electromagnética.

Por lo general se suelen emplear resistencias de 10 KΩ para la velocidad básica de 100 KHz y resistencias de 2 KΩ, o menores, para velocidades de 400 KHz o superiores. Para calcular el valor óptimo de la resistencia en cada caso debemos tener en cuenta la capacidad del bus mediante las ecuaciones de la figura 9.

$$\text{Freq} < 100\text{kHz} \implies R_{\min} = \frac{V_{cc} - 0.4\text{V}}{3\text{mA}}, R_{\max} = \frac{1000\text{ns}}{C_{\text{bus}}}$$

$$\text{Freq} > 100\text{kHz} \implies R_{\min} = \frac{V_{cc} - 0.4\text{V}}{3\text{mA}}, R_{\max} = \frac{300\text{ns}}{C_{\text{bus}}}$$

Fig. 9: Cálculo óptimo de las resistencias del bus I2C. Fuente propia

El I2C es un tipo de bus muy extendido por su flexibilidad y simplicidad hardware. Aunque una distribución en paralelo siempre será más rápida y robusta, las características del I2C lo convierten en una de las opciones más atractivas a la hora de implementar un bus en las placas electrónicas y prototipos.

4.3 Herramientas

El desarrollo del trabajo no se puede desvincular de las herramientas empleadas durante el mismo. En primer lugar se va a exponer brevemente el proyecto Arduino, ya que el *UMA_AEB_V2.0.0* se montará sobre uno de sus productos. Tras esto se mostrará el funcionamiento del software que se ha utilizado para diseñar el escudo (KiCad) y realizar simulaciones (Falstad).

Arduino

El proyecto Arduino está destinado a diseñar y fabricar placas de desarrollo hardware con licencia pública de libre distribución (*GPL* o *LGPL*). El objetivo del proyecto es simplificar el uso de la electrónica y la programación de sistemas embebidos. Existen una gran variedad de placas de Arduino, así como escudos para ampliar sus prestaciones.

Dentro del proyecto Arduino también se engloba el entorno de programación *Arduino IDE*, que proporciona un método fácil de programación de las placas. Éste selecciona automáticamente las librerías deseadas en función de la placa que estemos empleando y simplifica el desarrollo software de las placas Arduino mediante una librería base. La programación de los núcleos de Arduino también se puede realizar mediante el programa *AtmelStudio*, como si se tratara de un microcontrolador discreto.

El Arduino UNO fue la primera placa que se diseñó y también la más extendida del proyecto. Esta monta un microcontrolador AVR de 8bits: el ATmega328P. La tensión de funcionamiento es de 5V, aunque otros modelos más modernos emplean 3V3.

Esta placa cuenta con catorce pines de entrada o salida digitales, seis de los cuales permiten generar una salida PWM, y seis pines de entrada analógica hasta la tensión de funcionamiento, que también se pueden emplear como pines digitales. Sin embargo, dos de los pines digitales (D0 y D1) no se suelen emplear, ya que usan para programación mediante el puerto serie. La distribución, función y denominación de cada uno de los pines se muestra en la figura 10.

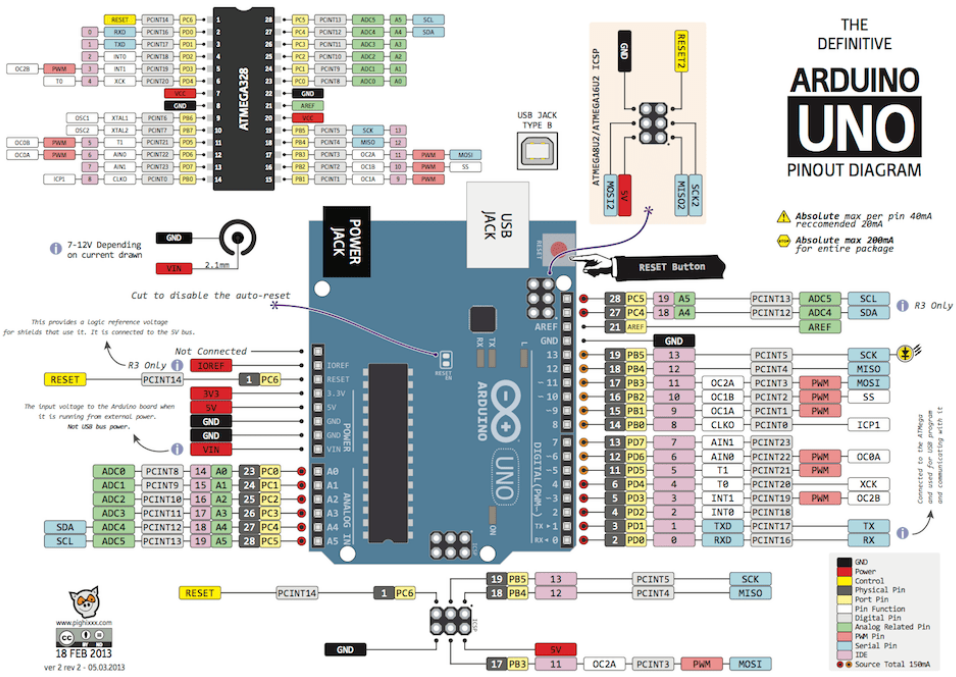


Fig. 10: Mapa de pines del Arduino UNO. pighixxx.com

Una de las principales ventajas de Arduino es la amplia comunidad que lo utiliza. Gracias a la filosofía *OpenSource* cualquier persona puede buscar y reaprovechar trabajo de terceros para sus proyectos, que también podrán ser reutilizados por la comunidad. Esto propicia una gran facilidad y velocidad en el desarrollo. Además, existen una gran cantidad de librerías oficiales en el Arduino IDE que simplifican mucho el desarrollo de software, especialmente de comunicaciones estándar como I2C o SPI.

Para cargar programas en el microcontrolador se suele emplear una comunicación serie y el software gratuito Arduino IDE. Sin embargo, también es posible programar el Arduino como si fuera un ATmega normal en una placa electrónica, para ello se puede utilizar el entorno de programación AtmelStudio.

KiCad

KiCad es un software de código abierto para el diseño electrónico. Se trata de la principal alternativa a los softwares de diseño electrónico propietarios como Altium, Eagle, OrCad… Este programa informático se encuentra dividido en varios módulos interrelacionados entre sí. Todos ellos se gestionan desde un módulo principal, denominado también KiCad, que actúa como gestor del proyecto, dando acceso a los archivos y módulos.

El módulo *Eeschema* es del editor de circuitos electrónicos. En él se diseñan los esquemas electrónicos del proyecto. Para facilitar la claridad y el orden se pueden emplear las herramientas habituales, como etiquetas locales, etiquetas globales y hojas jerarquizadas, en la parte derecha de la pantalla. En esa misma barra de herramientas se encuentran los menús para añadir al esquema componentes, potenciales, líneas, buses, etc.

Existen una amplia diversidad de símbolos de componentes cargados por defecto en KiCad. Además, al tratarse de un software *OpenSource*, en internet se pueden encontrar una gran cantidad de librerías creadas por la comunidad con más símbolos a libre disposición de los usuarios. En caso de que en las librerías no aparezca el componente deseado, y no se encuentre en las librerías de la comunidad, se puede diseñar mediante el otro módulo del programa: el editor de librerías. Desde la barra de herramientas de la parte superior de *Eeschema* se puede acceder directamente a dicho segundo módulo.

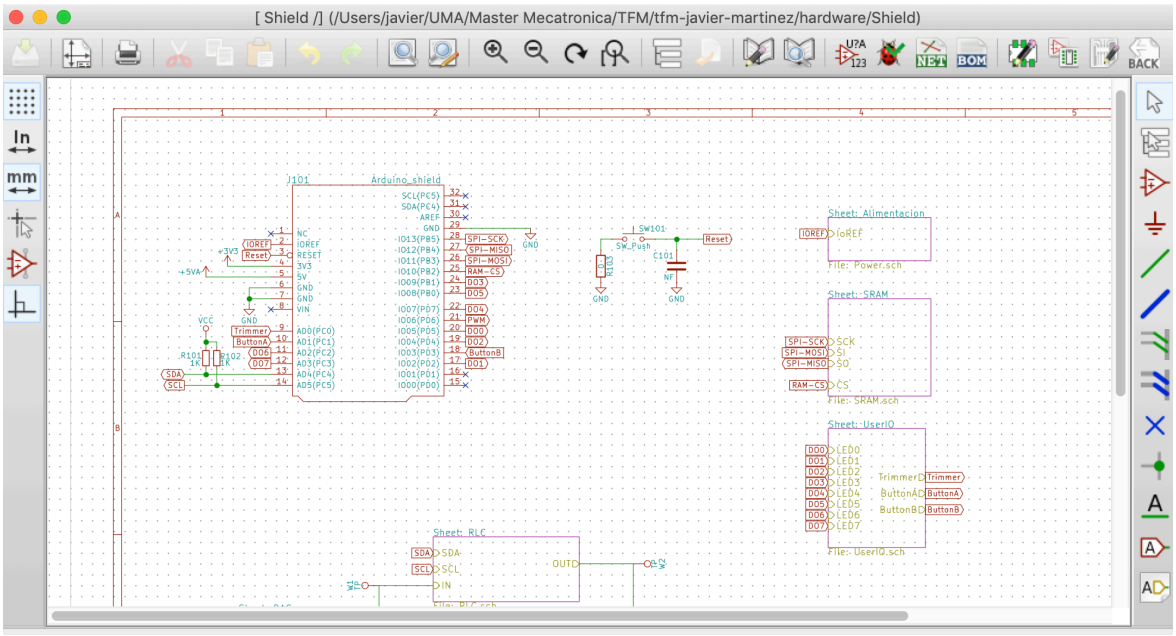


Fig. 11: Módulo *Eeschema* de Kicad. Fuente propia

El editor permite dibujar el componente que se desee desde cero o a partir de un componente ya existente. Una vez hemos se ha terminado la representación del componente y se han dispuesto de la forma deseada los pines, se debe guardar el componente, teniendo la opción de emplear una de las librerías ya existentes en KiCad o pudiendo crear una nueva librería propia con el componente. Estos componentes recién creados pasan a estar disponibles inmediatamente en *Eeschema*, sin necesidad de reiniciar el módulo.

Una vez el diseño electrónico de *Eeschema* está completo se procede a enlazar los componentes de la placa con su huella correspondiente. Por lo tanto, resulta necesario conocer exactamente los encapsulados que montaremos en la placa de circuito impreso (PCB). En caso de que existiera algún problema podrían modificarse posteriormente las huellas, o cambiarlas directamente por otras. Cuando todas las huellas están correctamente enlazadas se puede generar la *Net-list* mediante una de las herramientas de la barra superior. La *Net-list* es el archivo que determina qué huellas irán en la PCB y cómo se realizarán las conexiones entre ellas.

Para realizar el diseño físico de la PCB hay que acceder al tercer módulo de KiCad, *Pcbnew*. Al abrir el módulo la pantalla estará vacía; para comenzar el diseño se debe cargar la *Net-list* que acabamos de generar en *Eeschema*. Esto provoca la aparición de todas las huellas que hemos enlazado previamente a los componentes y de unas líneas que indican las conexiones debemos realizar entre dichos componentes.

A continuación, es necesario disponer los componentes de la forma adecuada para realizar el ruteo. Una de las partes más importantes del diseño consiste en lograr una distribución de componentes que simplifique el trazado de las pistas. La distribución ideal no se logra fácilmente, por lo que resulta necesario ir realizando modificaciones según se vaya considerando oportuno, incluso si ya nos encontramos en la fase de ruteo.

En la fase de ruteo debemos tener en cuenta las necesidades de las pistas que estemos trazando. Las pistas que sirvan de alimentación han de ser de una anchura mayor para permitir el paso de la corriente; las pistas digitales de alta frecuencia deben evitar pasar por sitios delicados (como puede ser por debajo de componentes); las pistas de los buses deben mantener la impedancia entre sí, etc. También hay que tener en cuenta las restricciones del fabricante de PCBs que vayamos a emplear. Esto nos limita la anchura mínima de pistas, la distancia mínima entre pistas, el número de capas, el tamaño y tipo de las vías, etc.

Las herramientas de diseño de *Pcbnew* se encuentran, al igual que en *Eeschema*, en el lateral derecho, mientras que las de visualización se encuentran en el lateral izquierdo. Se puede automatizar la anchura de pistas en función de la red que estemos empleando, aunque no tiene efecto retroactivo. De forma similar si hacemos un plano de cobre debemos modificarlo regularmente para evitar colisiones con otros elementos que se hayan dispuesto más tarde. Una forma de saber si existen problemas consiste en emplear la herramienta de comprobación de reglas de diseño de la parte superior, representado con una mariquita. Esta herramienta actualiza de forma automática los planos de cobre antes de realizar el análisis.

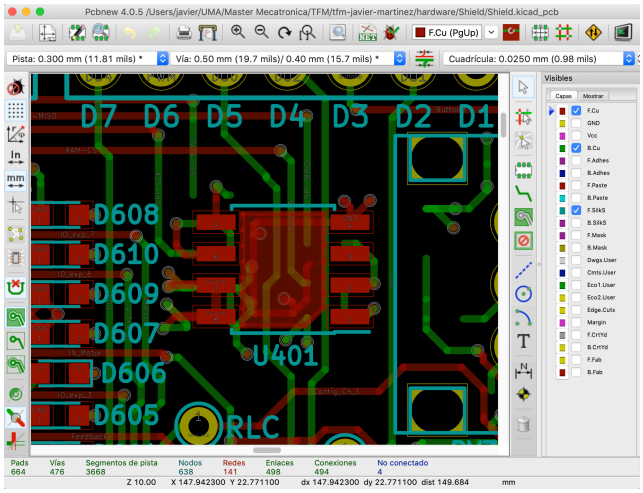


Fig. 12: Pantalla de *Pcbnew*. Fuente propia

En caso de necesitar realizar la modificación de alguna huella se puede acceder directamente al módulo del editor de huellas. Éste funciona de forma similar al editor de librerías.

En cualquier momento del diseño de la placa de circuito impreso se puede acceder al visor 3D mediante las opciones del menú superior. Esto permite ver rápidamente cómo será el resultado final y si van a existir problemas de colisión entre componentes. Por defecto los componentes creados con la librería no tienen un modelo 3D asignado y tan solo aparecerán representados su huella y su serigrafía. Los modelos pueden crearse mediante terceros programas en formato .stl, importarlos a las librerías y enlazarlos manualmente a las huellas en *Pcbnew* para que se muestren en el visor 3D.

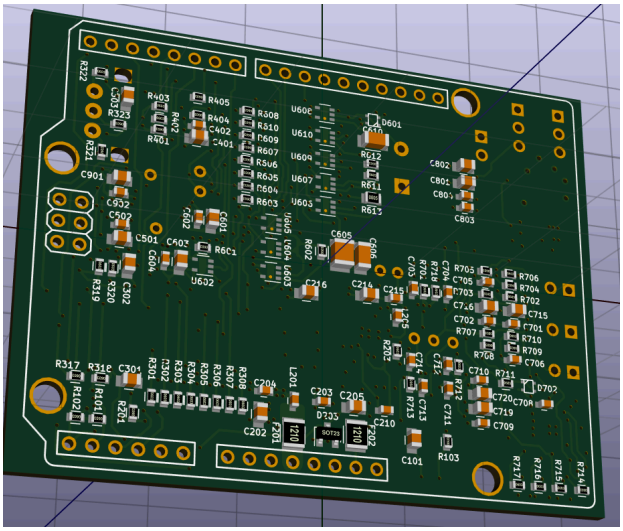


Fig. 13: Visor 3D de KiCad. Fuente propia

Falstad

Falstad es un sitio web que contiene una gran cantidad de aplicaciones con fines educativos en los campos de la matemática, la física y la ingeniería. En concreto la aplicación *Analog Circuit Simulator Applet* permite simular de forma rápida circuitos electrónicos básicos. Su representación es muy visual, modificando el color de los conductores en función de su tensión y empleando puntos amarillos en movimiento para ilustrar el paso de la corriente, cuya velocidad cambia de acuerdo a la intensidad que circula.

Mediante la pestaña "*Draw*", ubicada en la parte superior, se pueden seleccionar componentes para añadir al esquema. El valor de estos componentes se puede modificar haciendo doble click sobre ellos. Aquí también se pueden añadir gráficas, que aparecen en la parte inferior de la pantalla, y que se pueden modificar mediante el doble click. En el último desplegable de la parte superior se localizan una amplia variedad de ejemplos organizados por categorías.

En la parte derecha de la pantalla se encuentran dos deslizables qué sirven para determinar la velocidad de la simulación y la velocidad de representación de las corrientes. Es importante ajustar adecuadamente estos dos valores, así como la escala de las gráficas, para que toda la información representada sea fácil de visualizar y comprender.

Una gran ventaja de *Falstad* es que permite exportar los circuitos creados mediante una URL personalizada. Esto permite guardar o compartir de forma rápida y cómoda las simulaciones llevadas a cabo.

5. Diseño de Esquemáticos

Tras el análisis del *UMA_AEB_V1.1.0*, se puede comenzar a diseñar el nuevo escudo. Al comprender el origen de los problemas de las versión anterior, se puede orientar el trabajo hacia los aspectos más críticos y las características más complejas.

El primer paso del diseño consiste en realizar los esquemas electrónicos de la placa. Durante esta etapa se irán planteando los diversos métodos que se pueden emplear para resolver cada subsistema.

El diseño de los esquemáticos es el momento en el que, además de realizar los esquemas, debemos simular y/o probar las modificaciones que queramos implementar en la placa. En ocasiones se fabricarán pequeñas placas de pruebas para asegurarnos del comportamiento de subsistemas aislados.

Una vez se hayan realizado las pruebas o simulaciones oportunas, se escogerán las soluciones que se consideren mejores dados los objetivos y el comportamiento general de la placa. Las modificaciones y el proceso seguido en cada subsistema se desarrollará en su apartado correspondiente.

5.1 Estudio de necesidades y decisiones principales de diseño

Tras analizar la placa actualmente en uso, y teniendo en cuenta los objetivos que deseamos alcanzar en este nuevo modelo, se han tomado algunas decisiones que marcarán la línea general del diseño.

El escudo debe ser capaz de trabajar con microcontroladores de 5V y de 3V3 indistintamente. Por lo tanto, todas las comunicaciones con este deben ser adaptables. Sin embargo, todos los modelos de Arduino cuentan con un pin de alimentación a 5V. Se puede aprovechar este pin para alimentar los integrados del escudo. Para las comunicaciones emplearemos el bus digital I2C, este protocolo funciona correctamente tanto a 5V como a 3V3 (ambos se consideran 1 digital). Las resistencias de *Pull-Up* deben estar siempre a la tensión de funcionamiento del Arduino; de esta forma se evita que el bus se pueda encontrar a más tensión que la de trabajo del microcontrolador.

Se ha decidido independizar el convertidor de digital a analógico (DAC) del control de los LEDs. El control de este integrado se realizará mediante una comunicación I2C. Aprovechando este bus también se implementa un convertidor de analógico a digital (ADC). Las características del protocolo I2C permiten aumentar el número de nodos sin incrementar el número de pistas del bus, por lo que no supone ningún problema de conexionado para el Arduino. Esta decisión permite generar señales de hasta 5V para los circuitos analógicos aunque el microcontrolador funcione a 3V3.

La implementación de un ADC también libera los pines que se empleaban para la realimentación de los circuitos. Por lo tanto, se pueden usar para separar los botones del interfaz respecto al bus SPI. Este era uno de los principales problemas de la placa anterior, ya que las comunicaciones con la memoria RAM se veían corrompidas si se pulsaban los botones durante una transmisión.

Respecto al *UMA_AEB_V1.1.0* se suprime un botón del interfaz para permitir que cada pin solo tenga una función. Se emplean ocho pines para los LEDs, dos para los botones, uno para el potenciómetro, dos para la comunicación I2C, cuatro para el bus SPI, uno para generar una señal PWM y dos para la comunicación por el puerto serie. De este modo, todos los pines del Arduino se encuentran ocupados, pero sin compartir funciones.

Cualquier nuevo integrado que se añada, y que deba ser controlado por el microcontrolador, deberá permitir una comunicación I2C.

5.2 Esquema principal

Con las decisiones fundamentales establecidas se comienza realizando un diagrama de alto nivel que sirva para fijar la relación general entre los subsistemas.

En comparación con el esquemático de la anterior versión de la placa, se han realizado algunas modificaciones en la agrupación por subsistemas. Se han condensado los subsistemas de “Indicadores Digitales”, “Pulsadores” y “Potenciómetro” en un único bloque: *UserIO*. También se ha añadido un subsistema *ADC*, puesto que se ha añadido este tipo de convertidor. Por lo tanto, el esquemático se divide en los siguientes subsistemas: *Alimentación*, *UserIO*, *SRAM*, *DAC*, *RLC*, *OpAmp15*, *Servo* y *ADC*.

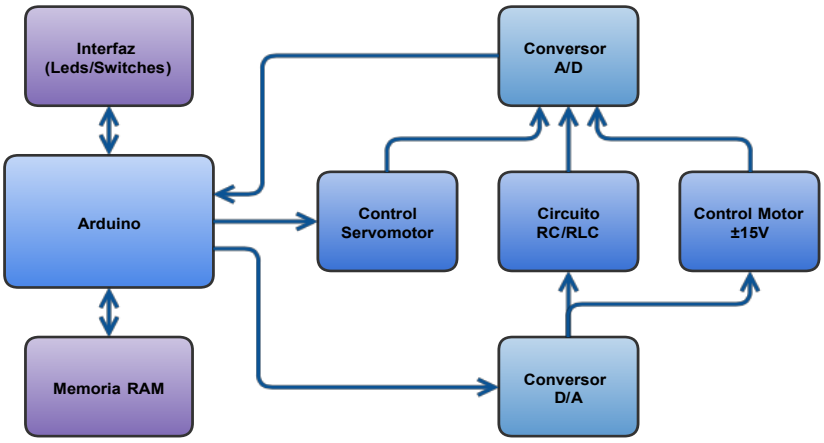


Fig. 14: Diagrama funcional de bloques del *UMA_AEB_V2.0.0*. Fuente propia

Como se puede ver en la figura 14, el Arduino se emplea directamente para controlar la interfaz, la memoria RAM, el servomotor y ambos convertidores. La salida del DAC se conecta a los subsistemas *RLC* y *OpAmp15*. Las señales de salida de estos dos circuitos, junto a la señal de realimentación de *Servo*, se envían al ADC para su digitalización. Las tensiones de alimentación de todos los subsistemas provienen del bloque de *Alimentación*, omitido en el diagrama de alto nivel para lograr una representación más clara.

A continuación, se va a proceder a detallar los componentes que aparecen en el esquema principal, en la figura 16, y el modelo comercial empleado para su implementación. Cada subsistema tiene su propio esquema jerarquizado, representado mediante un rectángulo en la hoja principal. Por lo tanto, los únicos circuitos presentes en la hoja principal son aquellos que no pertenecen a ningún subsistema: El acondicionamiento del bus I2C y el botón de reinicio. Los esquemas completos pueden consultarse en el anexo de esquemas electrónicos.

Para que el bus I2C funcione correctamente se han implementado dos resistencias de *Pull-Up* de 1KΩ en las líneas de *SDA* y *SCL* (*R101* y *R102*). Esta resistencia es de un valor considerablemente bajo, lo que permite una mayor velocidad de comunicación, aunque podrían aparecer problemas debido a interferencias o a la activación por parte de algún integrado. En cualquiera de estos casos, sería necesario aumentar el valor de estas resistencias, sacrificando velocidad del bus.

Se ha decidido que las resistencias y los condensadores montados tengan de huella (0603), en un compromiso entre tamaño y facilidad de soldadura; a partir de ahí, la decisión final se realiza teniendo en cuenta el coste económico y la cantidad de stock disponible. El modelo escogido en este caso es el *CRG0603F1K0* de *TE Connectivity*.

El botón de *Reset* (*SW101*) del escudo se conecta en paralelo al botón original del *Arduino*, ya que este quedará inalcanzable al colocar el escudo. El reinicio se produce cuando

la señal en el pin *Reset* es de 0V, por lo que el botón se dispondrá entre dicho pin y masa. El pin de reinicio cuenta con una resistencia *Pull-Up* de 10KΩ integrada en la placa de Arduino, así que no es necesario añadir otra en el escudo. Por seguridad se disponen una resistencia (*R103*) y un condensador (*C101*) para dar la posibilidad cubrir comportamientos extraños en el futuro. Estos actuarían cómo filtro paso bajo RC y/o como limitación de corriente si se considerase necesario. En un primer momento la resistencia es de 0Ω y el condensador no se monta para que no tengan efecto sobre el circuito.

Empleamos la resistencia (0603) de 0Ω *CR0603-J/-000ELF* de *Bourns* por motivos económicos y de tamaño. En caso de detectarse algún comportamiento extraño se sustituiría por una resistencia de un valor de acuerdo la magnitud de los efectos indeseados detectados.

Respecto a la elección del botón, resulta importante que tenga montaje SMD y que sea del tipo *SPST-NA* (un polo normalmente abierto). También es conveniente que sea de reducidas dimensiones, ya que implementaremos tres en el conjunto del proyecto. Se escoge el modelo *2-1437565-7 de TE Connectivity*. Este modelo presenta un contacto normalmente abierto, sus dimensiones son de 6x6x4'3mm y soporta hasta 50mA y 12V, superando ampliamente nuestras necesidades. Se trata de un tipo de pulsador muy empleado en proyectos de características similares y que ocupa un espacio muy reducido. Sus pines están eléctricamente conectados dos a dos, uniéndose los cuatro al pulsar el botón, lo que puede facilitar algún arreglo en caso de fallo o rotura de un pin.

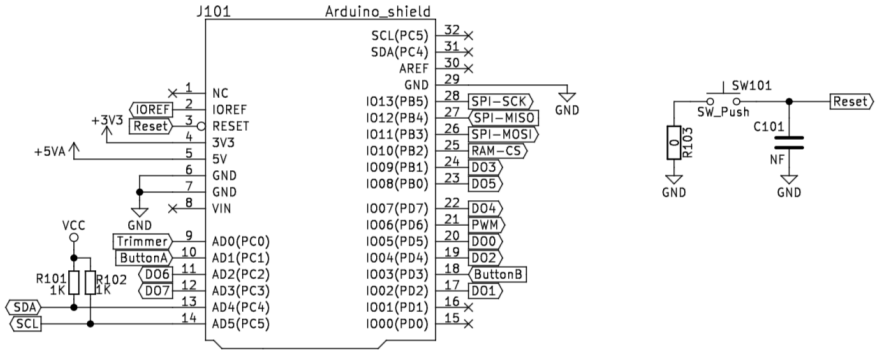


Fig. 15: Esquemático de los circuitos de la hoja principal. Fuente propia

5.3 Alimentación

El subsistema de alimentación es el encargado de generar todas las tensiones necesarias para el adecuado funcionamiento de la placa. Existen cuatro tensiones de alimentación, que se generan mediante tres circuitos.

La primera tensión será la tensión de alimentación de los integrados de la placa. Es una tensión constante de 5V y procede del pin de Arduino del mismo nombre. Emplear una tensión constante para los integrados facilita su elección, al existir menos restricciones en su voltaje de funcionamiento, y regulariza su comportamiento. No resulta adecuado emplear directamente una tensión procedente del exterior. Por lo tanto, vamos a acondicionar y proteger dicha tensión para evitar problemas tanto en nuestro escudo como en el Arduino.

La segunda tensión servirá para las comunicaciones digitales entre el microcontrolador y los integrados del escudo. La obtendremos a partir del pin *IOref,* y será protegida y acondicionada de igual forma que la tensión de alimentación. Esta tensión podrá ser de 5V o de 3V3 según el modelo de Arduino empleado.

Las dos últimas tensiones servirán para permitir las comunicaciones con el motor externo, que requiere de una señal entre -10V y +10V. Para permitir un margen de trabajo mayor, se

decide que estas tensiones sean de, como mínimo ±12V. Finalmente se fija el objetivo de ±15V, aunque cualquier valor entre estos dos se considera adecuado. Estas tensiones tendrán que ser generadas a partir de la tensión de 5V de Arduino, ya que no existe ningún pin que proporcione una tensión mayor. Denominaremos a estos voltajes como tensiones de interfaz, ya que sirven como tal frente a un motor externo.

Tensión de alimentación

La primera parte se encarga de obtener la tensión de 5V del pin de *Arduino* y adecuarla para la placa. Esta tensión se empleará como alimentación de los integrados, por lo que se debe lograr minimizar el ruido existente. Se emplean cuatro etapas diferenciadas: protección contra sobrecorrientes, filtrado inductivo de alta frecuencia, filtrado capacitivo de baja frecuencia y protección contra sobretensiones.

En primer lugar, se realiza una protección contra sobrecorrientes con el fusible autorrearmable *F201*. Dado el consumo esperable, se busca un fusible que permita el paso de al menos 1A antes de cortar. Se ha escogido el modelo *MICROSMD075F-2* de *LittelFuse* para la implementación de este componente. Este fusible tiene una corriente de mantenimiento de 0'75A y una corriente de corte de 1'5A con un tiempo de disparo de 0'1s, todo ello en un encapsulado (1210). Unos valores de corriente más pequeños podrían producir disparos erróneos que no nos interesan, mientras que unos más elevados no protegerían adecuadamente ante sobrecorrientes. Además, el tiempo de disparo tan reducido y el encapsulado SMD lo hacen ideal para nuestra aplicación.

En el segundo paso se elimina el ruido de alta frecuencia empleando un núcleo de ferrita, denominado *L201*. El ruido de alta frecuencia funciona en ambos sentidos, por lo que este componente filtrará tanto el ruido de entrada a la placa como el de salida. Los núcleos de ferrita aumentan su inductancia y su resistencia óhmica en función de la frecuencia de la corriente que los atraviese. Un núcleo ideal es aquel que no presenta impedancia alguna a las frecuencias de trabajo y, sin embargo, esta se vuelve muy elevada a las frecuencias a las que se esperan ruidos, especialmente en el aspecto resistivo. Estos ruidos suelen provenir de señales digitales de alta frecuencia, como las señales de reloj o las comunicaciones de alta velocidad.

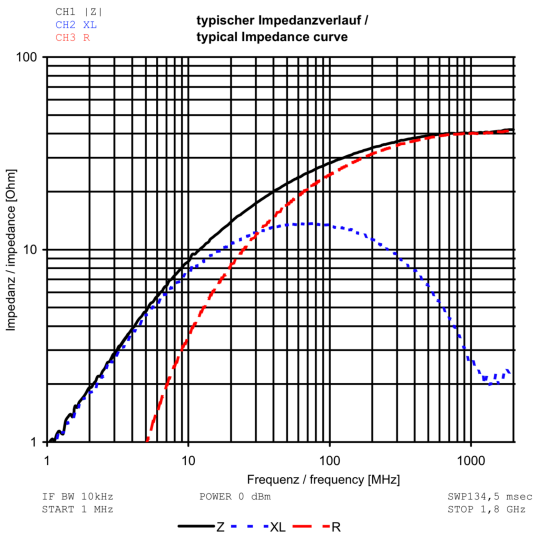


Fig. 16: Comportamiento en frecuencia del núcleo de ferrita. Wurth Elektronik

En nuestro caso se ha escogido implementar el núcleo *742792609* de *Wurth Elektronik*. Como puede verse en la imagen 16, este núcleo no presenta ninguna impedancia antes de 1MHz. A partir de esta frecuencia comienza el comportamiento inductivo. Por su parte el comportamiento resistivo comienza a los 5MHz y aumenta cuanto mayor es la frecuencia.

Tras el núcleo de ferrita, que filtra únicamente la alta frecuencia, disponemos cuatro condensadores en paralelo para filtrar el ruido eléctrico de baja frecuencia. El primero de ellos, *C201*, es electrolítico y se emplea como acumulador principal energía para evitar variaciones bruscas en la tensión de alimentación. Este condensador debe ser de un valor capacitivo bastante elevado, en este caso se ha decidido optar por una capacidad de 220uF. Cuanto mayor sea la capacidad del condensador, más energía acumulará, pero también ocupara un mayor espacio, por lo que es necesario alcanzar un punto medio satisfactorio para la aplicación. El modelo concreto que se ha escogido es el modelo *UWT0J221MCL1GB* del fabricante *Nichicon*, que soporta hasta 6'3V y tiene una tolerancia del 20%.

Los tres siguientes condensadores (*C202*, *C203* y *C204*) son de valores muy diferentes entre sí para que filtren el ruido en un mayor abanico de frecuencias. A estos condensadores los denominaremos, de mayor a menor capacidad, tipo A, tipo B y tipo C. Los modelos concretos para cada tipo se deciden en función de su tensión de trabajo, su tolerancia, su huella, el dieléctrico empleado y su precio. La capacidad concreta de cada tipo se fijará según los modelos encontrados en el mercado, aunque conviene que existan en torno a dos órdenes de magnitud entre las capacidades de cada tipo. El condensador de tipo A será el *GRM21BF51C105ZA01L* de *Murata*, con capacidad de 1uF; el tipo B será el *06031C103KAT2A* de *AVX*, con 10nF; y el tipo C será el *06031A101J4T2A* de *AVX*, con 100pF. Los tres modelos son condensadores cerámicos multicapa ya que ofrecen una buenas prestaciones en un espacio reducido. Los condenadores B y C emplean encapsulado (0603). El tipo A emplea un encapsulado (0805), ya que no encontramos un modelo adecuado con encapsulado (0603).

El último paso para lograr una tensión de alimentación debidamente acondicionada consiste en realizar una protección contra sobretensiones. Para ello se coloca un diodo Zener que denominaremos *D201*. Estos conducen cuando la diferencia de tensión entre sus terminales supera la tensión de ruptura. Es importante la conducción no comience hasta que no se supere claramente la tensión de alimentación establecida, de lo contrario supondría un gasto continuo e innecesario de energía. Se ha escogido el modelo *MMBZ5232BLT1G* fabricado por *ON Semiconductor* puesto que su tensión mínima de conducción es de 5'32V, aunque su tensión nominal de ruptura es de 5'6V. También influyen en esta decisión el bajo coste y el reducido tamaño del componente, siendo un encapsulado SOT-23.

De forma adicional, y sin formar parte del acondicionamiento, se añade un diodo LED de color verde dependiente de esta alimentación, *D202*. De este modo se tiene un indicador luminoso evidente si la tensión de alimentación principal funciona correctamente. Para escoger indicadores LED es importante tener en cuenta la caída de tensión directa del diodo, su corriente nominal y su luminosidad. Se ha decidido emplear el diodo *LP T670-G2J2-1* de *OSRAM*. Este LED tiene una caída de tensión de 2V y una corriente máxima de 30mA.

Se va a trabajar en régimen de trabajo inferior, ya que no necesitamos una alta luminosidad, así también reducimos el consumo energético. Se decide que la corriente del LED sea de 10mA, lo que debería dar 0'014 lúmenes según la hoja de datos. El valor de la resistencia serie necesaria se calcula empleando la caída de tensión del diodo y la corriente nominal mediante la expresión de la figura 17.

$$R = \frac{V_{Supply} - V_{LED}}{I_{LED}} = \frac{(5-2)[V]}{10^{-2}[A]} = 300\Omega$$

Fig. 17: Cálculo del valor de la resistencia. Fuente propia

En este punto también se puede calcular la potencia disipada en la resistencia. Esto es importante por dos motivos; en primer lugar se debe evitar superar el límite de disipación de calor de la resistencia, de lo contrario podría deteriorarse rápidamente o incluso quemarse; y

en segundo lugar, resulta importante tener una idea aproximada del consumo general de la placa y del consumo de cada subsistema. Los diodos LEDs son componentes que consumen una cantidad relevante de energía, tanto por sí mismos como por la disipación de la resistencia. Los cálculos de la potencias consumidas por la resistencia y el LED se muestran en la figura 18.

$$P = R \cdot I^2 = 300[\Omega] \cdot (10^{-2}[A])^2 = 30mW$$

$$P = V \cdot I = 2[V] \cdot 10^{-2}[A] = 20mW$$

Fig. 18: Potencias consumidas por la resistencia (Arriba) y el LED (Abajo). Fuente propia

En este caso la potencia disipada en la resistencia será de 0'03W. Esto permite emplear resistencias SMD (0603), cuya disipación de potencia es de 0'1W. El modelo concreto de resistencia empleada en serie con el LED, a la que se denomina *R201*, será el modelo *MCR03EZPFX3000* de *ROHM*. Se trata de una resistencia económica que cumple con nuestras necesidades de valor resistivo, disipación de potencia y tamaño.

Tensión de comunicaciones

La tensión de comunicaciones se emplea para la interfaz con el Arduino: afecta a los niveles de tensión del bus I2C y a las tensiones del subsistema *UserIO*. Esta tensión será la de funcionamiento del microcontrolador de la placa Arduino a la que se conecte el escudo. Este voltaje puede ser de 5V o 3'3V en función del modelo concreto. El Arduino publica esta tensión por el pin *IOref*. Este pin no se encuentra protegido de ninguna forma, sino que se trata de una derivación más de la tensión de alimentación de la placa. Esto se puede ver en la figura 19, donde el pin 2 (*IOref*) esta unido al pin 5 (*5V*). Por lo tanto, se puede aprovechar para consumir corriente. Esto simplifica enormemente el diseño ya que no resulta necesario disponer ningún tipo de convertidor de tensión (que sería necesario si *IOref* fuera una referencia con limitación en la entrega de corriente).

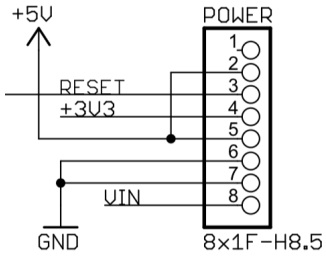


Fig. 19: Conector de alimentación de Arduino. Arduino UNO Schematic

Para acondicionar y proteger esta tensión implementaremos los mismos cuatro pasos empleados en la acometida de alimentación. Las diferencias entre ambos casos residirán en el indicador luminoso, que no se implementará, y en el condensador electrolítico, que en este caso no se montará ya que no se esperan variaciones bruscas de tensión debido a los menores consumos. Por lo demás emplearemos: un fusible *F202*, idéntico a *F201*; un núcleo de ferrita *L202*, idéntico a *L201*; tres condensadores cerámicos (*C205*, *C206* y *C207*) de tipos A, B y C; y un diodo Zener de 5'6V *D203*, similar a *D201*.

Como medida de protección final, colocaremos un diodo Schottky desde esta tensión de comunicaciones hacia la de alimentación. La tensión de alimentación siempre será 5V, mientras que la tensión de comunicaciones puede ser 5V o 3'3V. No hay ningún problema porque la tensión de comunicaciones sea menor que la tensión de alimentación, pero sí puede ha-

ber problemas si es mayor. Esto se debe a que la mayoría de integrados no estén preparados para que la tensión de un pin de comunicaciones supere a la tensión de alimentación. De hecho, algunos integrados incluyen diodos de protección que podrían romperse si esto sucediera.

Para asegurarnos que esta tensión siempre sea inferior vamos a colocar un diodo Schottky *D204A* entre ambas tensiones. Interesa que el diodo tenga la menor tensión directa posible, ya que protegerá mejor, y que ocupe físicamente poco espacio; por lo tanto, hemos escogido el modelo *DB5S308K0R* de *Panasonic* que cuenta con dos diodos independientes en un único encapsulado SSMini5-F4-B, de ahí viene la letra *A* en la denominación. El otro diodo del encapsulado va a resultar útil en el último circuito del subsistema, donde es necesario. Además, tener un encapsulado con dos diodos es especialmente útil en otros subsistemas, donde estos componentes se dispondrán por parejas.

En la figura 20 se muestran las dos etapas de protección de las tensiones de alimentación (arriba) y comunicaciones (abajo).

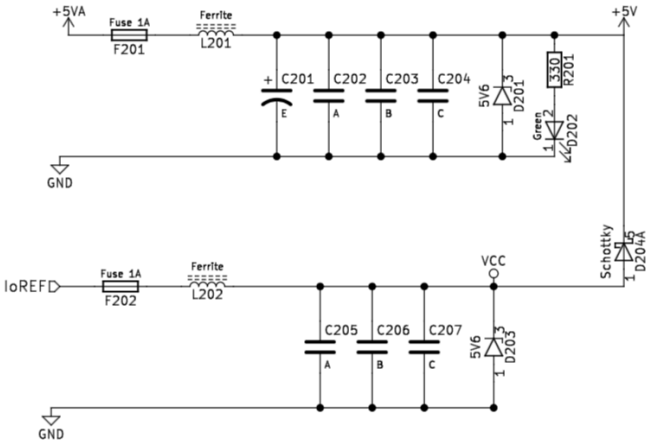


Fig. 20: Protección de las tensiones de alimentación y comunicaciones. Fuente propia

Tensiones de interfaz

Las tensiones de interfaz, *+15V* y *-15V*, tienen dos funciones relacionadas con el control del motor externo: en primer lugar, deben alimentar un integrado de amplificadores operacionales para generar la señal de control; y en segundo, deben servir de alimentación para dos diodos Zener de referencia de 10V para la calibración.

Este circuito debe generar las dos tensiones de interfaz a partir de una única tensión de alimentación de 5V. Además, debe poder suministrar suficiente corriente como para alimentar los amplificadores operacionales, los diodos Zener de referencia y cualquier otro integrado que se añada posteriormente. Pese a no tratarse de una corriente muy elevada es suficientemente relevante como para descartar algunas tecnologías de convertidores. Finalmente, y al igual que el resto del diseño, debe ser lo más compacto y económico posible.

Como ya se ha comentado en el apartado correspondiente, la anterior versión de la placa generaba estas tensiones mediante una fuente de alimentación conmutada integrada aislada. Pese a las buenas prestaciones de estas fuentes, el aislamiento no es necesario para nuestra aplicación y trae consigo dos efectos indeseados. Estos se deben al transformador interno, que supone un aumento de tamaño y coste. Por lo tanto, se decide tratar de buscar otra solución más económica y que ocupe un menor espacio. En caso de no encontrar dicha solución, se mantendrá la fuente de alimentación conmutada.

Multiplicador de tensión

El multiplicador de tensión es un método muy barato y pequeño para elevar voltaje; ya que solo precisa de diodos y condensadores. El problema consiste en que su capacidad de entrega de corriente es muy limitada. Para comprobar si se adapta a las necesidades de este proyecto, se realiza una simulación mediante la herramienta online *Falstad*.

Se llevan a cabo las pruebas de forma simultánea con las dos topologías presentadas previamente: la Cockcroft–Walton y la Dickson. En ambos casos se dispondrá una carga resistiva de 10KΩ para lograr un pequeño consumo. También dispondremos de un condensador para evitar grandes oscilaciones de la tensión de salida. La simulación se muestra en la figura 21.

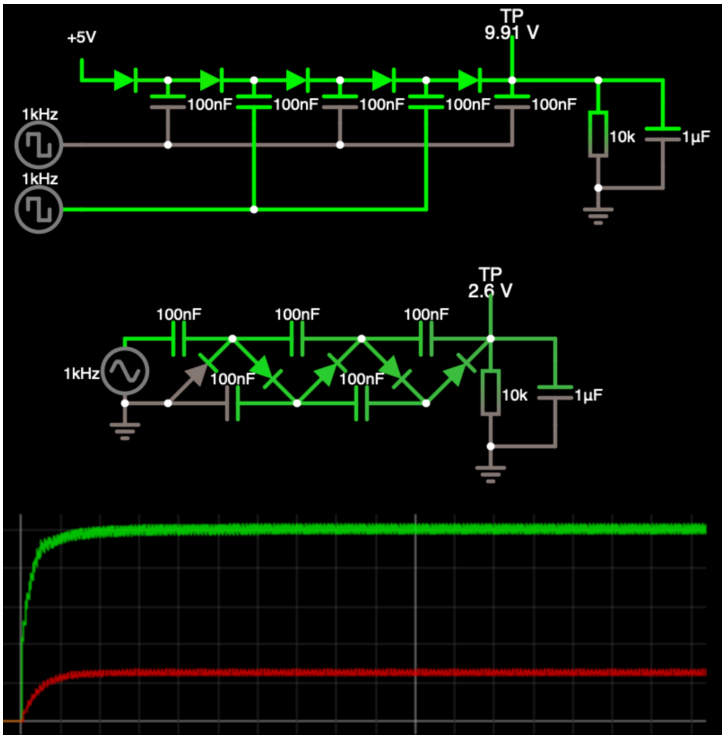


Fig. 21: Simulación de las topologías de multiplicadores de tensión Cockcroft–Walton (abajo y grafica en rojo) y Dickson (arriba y gráfica en verde) ante una carga de 10KΩ. Fuente propia

Como se puede ver en la gráfica de la parte inferior de la simulación, a igualdad de componentes (5 diodos y 5 condensadores), la topología Dickson es capaz de entregar una mayor cantidad de energía frente a la Cockcroft–Walton. La tensión de salida en este circuito se estabiliza en, aproximadamente, 10V; esto equivale a 1mA. Mientras, la topología Cockcroft–Walton apenas es capaz de alcanzar los 0'3mA.

Para aumentar la capacidad de salida de corriente es necesario aumentar la capacidad de los condensadores, el número de etapas o la frecuencia de trabajo, lo que aumenta el coste o la dificultad de implementación. Además, en esta simulación no se ha tenido en cuenta el circuito necesario para generar la tensión alterna o las señales de reloj. Estos circuitos aumentarían la complejidad y el tamaño de la solución, haciéndola aún menos viable. En conclusión, se considera que estas topologías no se adaptan adecuadamente al presente proyecto. Por lo tanto, se procede a buscar otras alternativas para generar las tensiones de interfaz.

Convertidor Boost

Otra posible solución para generar las tensiones de interfaz consiste en emplear un convertidor de tipología Boost. Estos convertidores se emplean para generar tensiones de salida mayores a las tensiones de entrada. La mayoría de los convertidores Boost del mercado son de un único canal de salida; por lo tanto sería necesario emplear dos convertidores in-

dependientes e invertir la salida de uno de los dos. Empleando un modelo con varios canales de salida, se reduciría la cantidad de componentes necesarios. El inconveniente es que se reduce la cantidad de integrados Boost comerciales que encajan con las necesidades del proyecto, y seguiría siendo necesario invertir una salida.

Se realiza una búsqueda de convertidores comerciales de dos o más canales, con tensión de entrada de 5 Voltios y tensiones de salida superiores a 15 Voltios; se encuentra el integrado *LT3463* de *Linear Technology*. Este integrado tiene la gran ventaja de que cuenta con un canal de salida positiva y un canal de salida negativa, resultando ideal para nuestra aplicación. Es capaz de generar tensiones de hasta ±20V en ambos canales y entregar corrientes de hasta 250mA ó 400mA en función del canal.

Dentro del esquema electrónico se denomina *U201* al integrado *LT3463*. El circuito que se va a implementar es el de la aplicación típica mostrada en las hojas de características, mostrado en la figura 22. Para lograr el voltaje de salida deseado de ±15V es necesario adaptar los divisores de tensión.

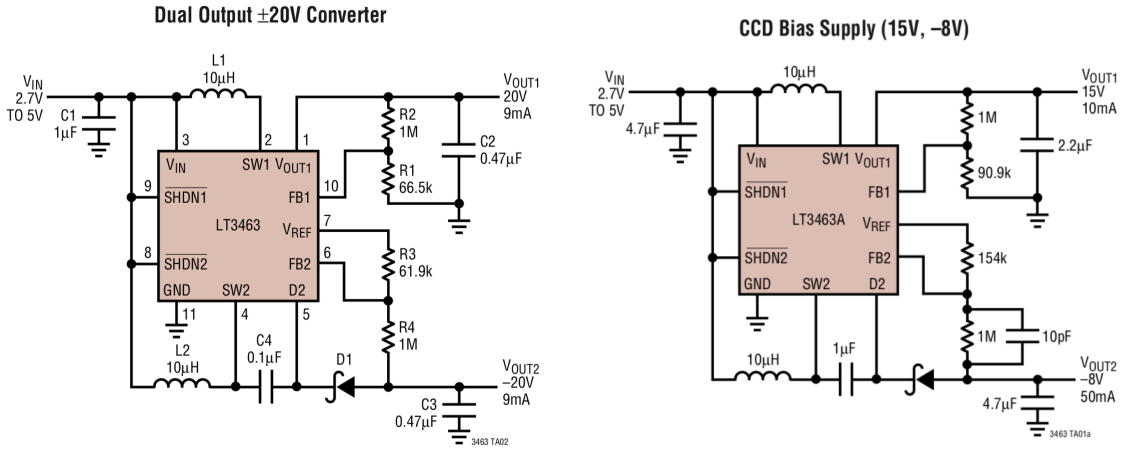


Fig. 22: Aplicaciones típicas del LT3463. LT3463 Datasheet

De acuerdo a la hoja de datos técnicos R1 y R3 deberían ser de un valor entre 50KΩ y 250KΩ, por lo que se fijan arbitrariamente en un valor intermedio: 100KΩ. Los valores de las otras dos resistencias de los divisores de tensión se calculan mediante las expresiones de la figura 23, también obtenidas de la hoja de características.

$$V_{OUT1} = 1'25\left(1+\frac{R_2}{R_1}\right) \Rightarrow R_2 = R_1\left(\frac{V_{OUT1}}{1'25}-1\right)$$

$$R_2 = 100\text{K}\Omega\left(\frac{15\text{V}}{1'25\text{V}}-1\right) = 1\text{M}1\Omega$$

$$V_{OUT2} = -1'25\left(\frac{R_4}{R_3}\right) \Rightarrow R_4 = R_3\left(\frac{V_{OUT2}}{1'25}\right)$$

$$R_4 = 100\text{K}\Omega\left(\frac{15\text{V}}{1'25\text{V}}\right) = 1\text{M}2\Omega$$

Fig. 23: Cálculo de las resistencias de los divisores de tensión. Fuente propia

Para el funcionamiento de este convertidor Boost son necesarios componentes externos, tal y como se observa en los circuitos de aplicación típica de la figura 22. Las bobinas *L1* y

L2, que en nuestro esquema se denominan *L203* y *L204*, son las encargadas de acumular la energía para el convertidor. Se emplea el modelo *LQM18FN100M00D* del fabricante *Murata*. Estas bobinas son de 10uH con una tolerancia del 20% en una huella (0603). Están pensadas para trabajar correctamente hasta los 30MHz. El convertidor trabaja a 125KHz, por lo que este componente se adapta perfectamente a nuestra aplicación.

El condensador *C4* de la aplicación típica de ±20V se va a mantener de ese mismo valor: 100nF. Se implementa el modelo *CGA3E2X7R1H104K080AA* de *TDK*, y se denomina *C210*. Se trata de un condensador cerámico multicapa (0603) que soporta tensiones de hasta 50V con una tolerancia del 10%.

El diodo *D1* debe ser un diodo de conmutación rápida, también denominados diodos Schottky. En este caso se emplea el otro diodo interno del encapsulado del *D204A*; por lo tanto, se identificará como *D204B*. Ambos tienen las mismas características, ya que se trata de dos componentes idénticos fabricados en un mismo encapsulado.

Las resistencias de ambos divisores de tensión *R1*, *R2*, *R3* y *R4* pasan a denominarse *R202*, *R203*, *R204* y *R205*. Los valores de estas resistencias ya se han fijado anteriormente (Figura 24) en 1M1Ω para *R202*, 1M2Ω para *R205* y 100KΩ para *R203* y *R204*. Estas resistencias interesa que sean de una tolerancia pequeña para obtener una tensión lo más precisa posible, sin embargo, una tolerancia demasiado pequeña aumentaría el coste económico de los componentes. Aparte de eso, buscaremos que, al igual que el resto de componentes, sean de pequeña huella y bajo coste. Para *R203* y *R204* se emplea el modelo *CR0603-FX-1003HLF* de *Bourns*; para R202, el *CRCW06031M10FKEA* de *Vishay;* y para R205 se utiliza, también del fabricante *Vishay*, el modelo *CRCW08051M20FKEA*. Todos estos componentes son de un 1% de tolerancia y de huella (0603), excepto *R205* que tiene una huella (0805). Esto se debe a que, debido a su valor no estándar, es más difícil de encontrar justo en el encapsulado deseado.

En el divisor de tensión relacionado con el control de la tensión negativa, la aplicación típica del LT3463 indica que se debe colocar un condensador de filtrado para mejorar su funcionamiento. Este condensador debe ser de un valor muy pequeño para filtrar el ruido, pero sin llegar a interferir en el funcionamiento de la realimentación de control del convertidor. El fabricante lo fija en 10pF, por lo que se decide implementarlo como un condensador de tipo C. Aunque sea de un valor superior (100pF), se considera que no afectará negativamente al circuito.

Un inconveniente de los convertidores Boost es que, al emplear conmutaciones entre dos estados para su funcionamiento, generan elevados ruidos y distorsiones tanto a la entrada como a la salida. El circuito de aplicación típica ya incluye filtros en ambos puntos, pero se decide ampliarlos para minimizar los ruidos emitidos al exterior del circuito.

La hoja de características sitúa en la tensión de entrada un condensador de 1uF o de 4’7uF; en nuestro caso se opta por implementar el condensador de 4’7uF, ya que producirá un mejor filtrado en ambos sentidos. Este condensador, denominado *C208*, será el modelo *1206YG475ZAT1A* de *AVX*. Tiene una tolerancia de -20+80% lo que significa que probablemente su capacidad sea mayor de la nominal. La huella de este componente es (1206) para mejorar su comportamiento.

El problema con los condensadores cerámicos multicapa consiste en que, según aumenta la tensión entre sus terminales, se reduce la capacidad real del componente. Este efecto se incrementa cuanto menor es el tamaño físico del componente y peor es el dieléctrico. Este efecto no es relevante en este subsistema, pero sí lo será más adelante, en los circuitos de los filtros RC yRLC.

En paralelo con *C208* disponemos una huella *C209* para otro condensador. Este segundo condensador no será montado inicialmente. Servirá para aumentar el filtro de entrada del

convertidor Boost en caso de considerarse necesario más adelante, sin la necesidad de desmontar *C208*. También permite montar un condensador de menor capacidad, y por lo tanto mayor velocidad de respuesta, que filtre el ruido de alta frecuencia

Para ambas tensiones de salida del convertidor se va a emplear la misma topología de filtro. Se ha decidido emplear filtros pi ya que reducen en gran medida la cantidad de armónicos y distorsiones frente a otro tipo de filtro pasivos, además de evitar las perdidas por calor típicas del filtro RC. Uno de los principales problemas de los filtros pi es el coste económico cuando la carga consume mucha corriente. Este coste viene dado principalmente por el núcleo de ferrita. Sin embargo, en este caso la corriente consumida es reducida, por lo que este filtro es viable económicamente.

Este circuito se compone de un condensador de entrada, un núcleo de ferrita y un condensador de salida. Su nombre proviene de la similitud del esquema eléctrico con la letra griega π. Además, se emplean dos condensadores de diferentes valores en paralelo a la salida para poder filtrar en un mayor rango de frecuencias.

Los condensadores de entrada *C211* y *C213* serán condensadores de 4'7uF, que es el valor máximo de estos condensadores que puede llegar a observarse en la hoja técnica del convertidor. Con el objetivo de estandarizar componentes se implementa el mismo modelo que el *C208*.

Las necesidades de los núcleos de ferrita aquí son las mismas que en el caso de *L201* y *L202*, por lo que se también se emplea el mismo modelo. Estos núcleos se denominarán *L205* y *L206*.

Como última etapa del filtro se montan condensadores de tipo A (*C214* y *C216*) y de tipo B (*C215* y *C217*). De forma ideal, tras estos filtros no existirá un rizado que llegue a afectar al funcionamiento de la placa. En caso de que durante las pruebas se considere necesario, se modificaran los componentes del filtro para mejorar su comportamiento de acuerdo a los resultados obtenidos.

En la figura 25 podemos ver el resultado final del esquema del subsistema de alimentación con el convertidor *Boost* en la parte inferior. En la parte superior se encuentra la tensión de alimentación *+5V*, que obtiene la energía de la tensión *+5VA*. En la parte central se encuentra la tensión de comunicaciones, denominada *Vcc* y que obtiene su energía de la etiqueta *IoREF*, viniendo del pin de mismo nombre del Arduino. El convertidor *Boost* obtiene su energía de la tensión de alimentación *+5V* generada en la parte superior y proporciona las dos tensiones de interfaz *+15V* y *Vss* (-15V).

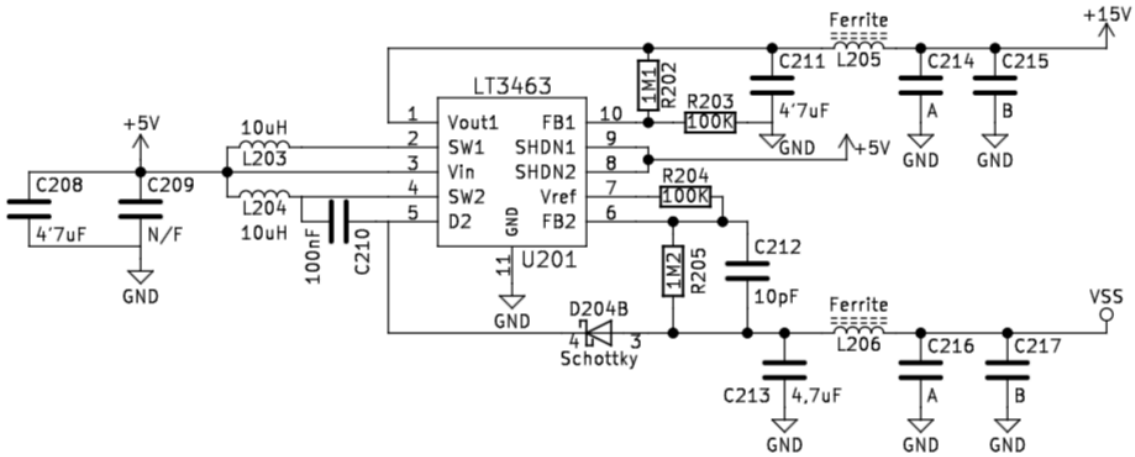


Fig. 24: Implementación del convertidor Boost. Fuente propia

5.4 Interfaz

Este subsistema tiene la función de generar una pequeña interfaz entre el usuario y la placa sin necesidad de recurrir a la comunicación con el ordenador vía puerto serie, u otros métodos. Está formado por cinco bloques: dos de salida de información y tres de entrada. La salida de información se realiza mediante el uso de LEDs. Esta será en forma de componentes discretos ordenados como un vector o registro (también denominado *array*), y en forma de monitor LED de siete segmentos o *display*. La entrada de información se logra mediante una combinación de dos pulsadores y un potenciómetro.

Interfaz de salida

En la *UMA_AEB_V1.1.0* se emplea únicamente un array de LEDs como interfaz de salida; en este nuevo escudo se va a emplear también un display de siete segmentos. Esto da la opción de realizar una práctica en la que se programe un *encoder*, o codificador, de binario al formato del display. Además, una vez se tiene el *encoder*, permite representar números o incluso algunas letras en el monitor.

La implementación del *display* se realiza en paralelo con la del *array*. Existe un condicionante destacable: debe existir un correcto funcionamiento de ambos circuitos tanto a 5V como a 3V3. Para ello se deben seleccionar cuidadosamente los componentes y calcular de forma precisa las resistencias de limitación de corriente.

En un primer momento se había propuesto emplear la misma resistencia para ambos componentes, pero no se considera una opción válida. Esto se debe a que la suma de las caídas de tensión de un LED y del display supera los 3V3, por lo que no podrían colocarse en serie. La colocación en paralelo con una única resistencia común tampoco es buena idea debido a la diferencia entre las caídas de tensión de los componentes.

Cuando la tensión de funcionamiento cambie, cambiará la luminosidad de los LEDs. Para minimizar este efecto debemos procurar que la caída de tensión en la resistencia en ambos casos sea lo menor posible, ya que esta determina la corriente, que es proporcional a la luminosidad. La tensión de alimentación no es un aspecto que podamos modificar, por lo que debemos buscar diodos con la menor tensión directa posible. Tampoco debemos olvidar el espacio físico que ocupan los LEDs, debido a su cantidad, y el display, ya que suelen ser de grandes dimensiones.

En la búsqueda de LEDs discretos se encuentra el modelo *KP-1608SRC-PRV* de *Kingbright*. Se trata de un diodo LED de color rojo (640nm), de considerable luminosidad (100mcd nominales), con una caída de tensión menor a dos voltios en un encapsulado (0603). Se trata de un componente que encaja perfectamente con las necesidades del circuito y además es de un color fácil de distinguir de otros LEDs, como el LED verde del encendido. Dentro del esquema estos LEDs recibirán las denominaciones *D301* a *D308*.

Para implementar el *display* de siete segmentos se escoge el modelo *KCSC02-106*, también de *Kingbright*, recibiendo la denominación *S301*. Se trata de un componente de reducidas dimensiones (10x6’9x3mm) y con una caída de tensión inferior a 2’2V. Su luminosidad es considerablemente menor que la de los LEDs discretos, pero los monitores de mayor luminosidad son más voluminosos y quedan descartados.

Para calcular las resistencias serie de los LEDs se deben hallar los valores máximo y mínimo que pueden tomar, y luego decidir un valor intermedio. El valor máximo admisible para la resistencia es aquel que, con una tensión de alimentación 3V3, permita circular la corriente mínima para que el LED luzca. El valor mínimo de la resistencia es aquel que evite la rotura del LED por exceso de corriente cuando este se encuentre alimentado a 5V. Para hacer un cálculo preciso hay que tener en cuenta además la variación de la caída de tensión

en el diodo en función de la corriente que lo atraviesa. En la figura 26 se muestra el cálculo general y el cálculo específico para los LEDs discretos y para el monitor.

$$R_s = \frac{V_{cc} - V_{LED}}{I_{LED}}$$

$$R_{MAX} = \frac{3V3 - V_{LED}}{I_{MIN}}$$

$$R_{MIN} = \frac{5V - V_{LED}}{I_{MAX}}$$

$$R_{L\,MAX} = \frac{3V3 - 1'75V}{5mA} = 310\Omega \qquad R_{D\,MAX} = \frac{3V3 - 1'9V}{5mA} = 280\Omega$$

$$R_{L\,MIN} = \frac{5V - 1'85V}{20mA} = 157'5\Omega \qquad R_{D\,MIN} = \frac{5V - 2'1V}{20mA} = 145\Omega$$

Fig. 25: Cálculo de las resistencias necesarias para LEDs discretos y *Display*. Fuente propia

Debemos emplear resistencias serie de entre 310Ω y 157’5Ω para los LEDs discretos y de entre 145Ω y 280Ω para el *display*. Decidimos estandarizar estas resistencias, ya que cualquier valor entre 158Ω y 280Ω nos sirve para las 16 resistencias serie. Vamos a emplear un valor intermedio de 220Ω, pues este valor debería permitir una luminosidad aceptable con ambas tensiones a la vez que protege los componentes y evita un consumo excesivo.

A la hora de escoger la resistencia de 220Ω es especialmente importante el tamaño y el precio, ya que se van a implementar 16 unidades solo en este subsistema. El modelo *MCR03EZPFX2200* de *ROHM* nos ofrece las características que necesitamos, con un ±1% de tolerancia en un encapsulado (0603) y con un precio inferior a medio céntimo por unidad. Estas resistencias se nombrarán dentro del esquema como *R301* a *R316*. Correspondiendo las ocho primeras para los LEDs discretos y las ocho restantes para el display.

Interfaz de entrada

La interfaz de entrada se compone de dos pulsadores y un potenciómetro. Se ha eliminado un pulsador frente a la anterior versión de la placa; pero, a cambio, el control de los restantes se ha independizado del bus SPI, sobre el que tendía a generar interferencias.

Los dos pulsadores, *SW301* y *SW302*, funcionarán con lógica positiva y un *pull-down* (resistencia a 0V) que asegurará una adecuada detección cuando no se esté produciendo una pulsación. Interesan botones de un polo y un contacto (SPST), de montaje superficial, del menor tamaño posible y de bajo coste. Para estandarizar componentes se va a montar el mismo modelo que en el botón de Reset del esquema general: el *2-1437565-7* de *TE Connectiviy*. Este componente ya se había escogido de acuerdo a estos requisitos.

Como valor resisitivo de los *pull-down* se ha seleccionado arbitrariamente 47KΩ. Si este valor es demasiado elevado, puede no llevar a cabo su cometido correctamente y que aparezcan lecturas incorrectas ante distorsiones externas. Si este valor es demasiado bajo, provoca un consumo continuo de energía y, en casos muy extremos, puede llegar afectar al funcionamiento del botón, de forma que no se detecten correctamente las pulsaciones. Se considera que 47KΩ es un valor que forzará adecuadamente el valor cuando no se active el pulsador, y la corriente de fuga máxima seria de apenas 100uA, por lo que no supondrá un problema energético. Realmente esta corriente será menor ya que las entradas del Arduino se encontrarán configuradas como entrada, por lo que tendrán una gran impedancia.

Se ha escogido el modelo *MCR03EZPFX4702* de *ROHM* debido a sus buenas características, su encapsulado (0603) y su precio reducido. Estas resistencias reciben las denominaciones de *R318* y *R320*, y actúan sobre los pulsadores *SW301* y *SW302* respectivamente.

Es recomendable proteger los pulsadores de forma similar al botón de Reset, mediante una resistencia y un condensador. En un principio se montará una resistencia de 0Ω, pero no el condensador; de esta forma no afectarán al comportamiento de los botones. En caso de ser necesario se cambiarán estos componentes, que pasarían a actuar como un limitador de corriente y un filtro paso bajo. Estas resistencias de 0Ω son, al igual que en el caso de *R103*, el modelo *CR0603-J/-000ELF* de *Bourns*. Las resistencias reciben las denominaciones *R317* y *R319*. Los condensadores, pese a no montarse, se identificarán como *C301* y *C302*.

El potenciómetro (*RV301*) permite a los usuarios generar una información más compleja que los pulsadores, limitados a dos estados. Se ha realizado una búsqueda con el objetivo de encontrar el potenciómetro con la menor altura posible que se pudiera accionar de forma manual sin necesidad de una herramienta. El modelo seleccionado es similar al que montaba la anterior versión de la placa, el *RK09K1110A0J* de *Alps Electric*, de poco más de 10mm de altura. Existen potenciómetros de menor tamaño, pero su manipulación requiere de una herramienta, por lo que se descartan. El valor resistivo del potenciómetro, 10KΩ, no resulta algo importante ya que la lectura se realiza en modo divisor de tensión. Tampoco es un condicionante su capacidad de disipación de potencia, debido a la alta impedancia del pin de lectura analógica.

Para poder proteger el potenciómetro, aunque en un principio no debería ser necesario, se va a implementar una resistencia en serie con cada uno de sus terminales (*R321*, *R322* y *R323*). Para evitar que afecten al comportamiento del potenciómetro estas resistencias van a ser inicialmente de 0Ω, pero pueden modificarse en cualquier momento. También se deja un hueco para un condensador (*C303*) conectado entre la señal de salida del potenciómetro y masa. Este condensador permitiría implementar una estabilización o un filtro paso bajo más adelante.

5.5 SRAM

En el escudo actualmente en uso, la memoria RAM externa tiene problemas al compartir pines con los botones, y por lo tanto sufre interferencias. En esta versión ya se han independizado ambas funciones. Se ha tomado la decisión de no realizar modificaciones significativas en este subsistema, puesto que su comportamiento es satisfactorio. El chip de memoria RAM, denominado dentro del esquema *U401*, seguirá siendo el modelo *23LC512-I/SN* fabricado por *Microchip*. Cuenta con 512 kbits de almacenamiento y se controla mediante el protocolo de comunicaciones SPI. Este protocolo requiere una mayor cantidad de conexiones que el I2C, pero permite una mayor velocidad de traspaso de información. Se trata del único componente de toda la placa que emplea este protocolo de comunicaciones.

En las cuatro líneas existentes del bus SPI se colocan resistencias en serie como protección. Las líneas son *Chip Select*, *Slave Out*, *SPI Clock* y *Slave In*; siendo sus resistencias respectivas *R402*, *R403*, *R404* y *R405*. Estas resistencias en serie cumplen tres funciones: en primer lugar limitan la corriente que circula por el bus, en segundo reducen el subamortiguamiento de las conmutaciones del bus (reduciendo las EMIs radiadas) y en tercer lugar dificultan que el ruido electromagnético externo altere las comunicaciones.

Todas estas resistencias de protección serán de 300Ω. El modelo concreto de la resistencia, similar a otros escogidos anteriormente, es el *MCR03EZPFX3000* de *ROHM*. Toda esta gama de resistencias ofrece unas buenas prestaciones, y destaca especialmente por su muy reducido precio.

La línea de *Chip Select* requiere además de una resistencia de *Pull-Up* (*R401*) que se emplea para evitar posibles ruidos mientras no se utilice. Así se logra que el integrado solo lea la información del bus SPI cuando lo indique el microcontrolador. Para estandarizar componentes se ha decidido que esta resistencia sea del mismo valor que los *Pull-Down* de los pulsadores (*R318* y *R320*), 47KΩ. De este modo se monta el mismo modelo, el *MCR03EZPFX4702* de *ROHM*.

La alimentación de cualquier integrado o componente que tenga un consumo variable debe ser protegida. De no ser así un cambio en el consumo podría provocar una alteración considerable de la tensión de alimentación debido a la inductancia de las pistas de alimentación. Esta variación podría dañar el componente o generar *glitches* en su funcionamiento, en función de si la tensión de alimentación aumenta o disminuye.

La protección de las alimentaciones frente a estas variaciones se realiza añadiendo los denominados condensadores de desacoplo. Al colocarlos en la tensión de alimentación compensan la inductancia de las pistas y suavizan las variaciones de voltaje, evitando tanto las subidas excesivas de tensión como los *glitches*. Estos condensadores también filtran la tensión de alimentación, minimizando que los ruidos electromagnéticos captados por las pistas lleguen hasta el integrado. Para que los condensadores de desacoplo lleven a cabo su función de forma óptima deben estar colocados lo más cerca posible de los pines de alimentación.

Se decide implementar dos condensadores de desacoplo en paralelo. El *C401* será de tipo A y, al ser el de mayor capacidad, tendrá la función principal de compensar la inductancia de las pistas y evitar variaciones. El condensador de menor capacidad será el *C402*, de tipo B, y principalmente se encargará de filtrar el ruido electromagnético en ambos sentidos. Este último se dispondrá, si es posible, más cerca del integrado, ya que su respuesta es más rápida. Todos los integrados del circuito irán acompañados dos condensadores de desacoplo como protección.

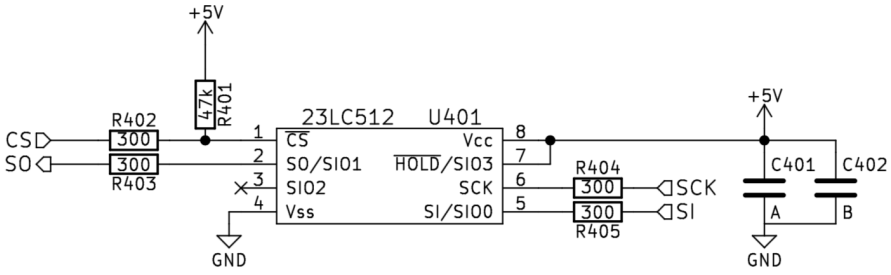


Fig. 26: Subsistema de la memoria RAM. Fuente propia

5.6 DAC

Un convertidor digital-analógico, DAC por sus siglas en inglés, tiene la función de generar tensiones analógicas a partir de datos digitales proporcionados por el microcontrolador. La mayoría de placas de Arduino no cuentan con un DAC integrado. Por lo tanto, para permitir esta característica es necesario implementar un integrado externo.

En la *UMA_AEB_V1.1.0* se emplea un convertidor con entrada paralela, es decir, cada bit de información del dato a generar se recibía por un pin independiente. Esto tiene la ventaja de ser un método muy rápido y sencillo de enviar la información. Sin embargo, es necesario destinar una gran cantidad de salidas GPIO del microcontrolador a esta tarea, uno por cada bit. En el modelo actualmente en uso, los pines encargados de enviar estos datos también se ocupan del control de los LEDs de la interfaz. Otro inconveniente de este método es que la resolución del conversor está limitada por el número de conexiones.

En el presente diseño se ha decidido independizar el control de la interfaz y el DAC; por lo que resulta necesario comunicarse con el conversor mediante un protocolo de comunicaciones. Como ya se ha mencionado anteriormente, hemos optado por usar I2C. Este bus tan solo necesita dos conexiones entre los diferentes nodos conectados, independientemente de su cantidad. Esto simplifica el diseño frente a otros métodos de comunicación como SPI, donde existen tres líneas comunes más una o dos líneas adicionales por cada nodo.

Los DACs, al igual que los ADCs (*Analog to Digital Converter* o convertidor analógico-digital), tienen gran cantidad de parámetros relevantes para su funcionamiento. Entre ellos destacan: la resolución, la tasa de conversión y el *Slew-Rate*.

La resolución se mide en bits y determina la precisión de la conversión. El paso del conversor, es decir la diferencia de tensión entre dos datos que cambian en una unidad, va a ser la tensión máxima de salida (tensión de referencia) dividida entre dos elevado al número de bits de resolución. Por lo tanto, cuanto mayor sea la resolución, mejor precisión tendrá el conversor (un paso menor). El objetivo en este subsistema consiste en lograr un paso de 5mV como máximo y capacidad de trabajar en todo el rango de los 5V; con estos datos se puede calcular el número de bits de resolución necesarios. Para ello se emplea mediante la ecuación de la figura 27.

$$Paso = \frac{V_{REF}}{2^{n\_bits}}$$

$$5mV = \frac{5V}{2^{n\_bits}} \rightarrow n\_bits = \log_2\left(\frac{5V}{5mV}\right) = 9'9657$$

Fig. 27: Cálculo bits de resolución de un convertidor. Fuente propia

Como se acaba de calcular, para tener un paso máximo de 5mV es necesaria una resolución de 9’96 bits. Los convertidores no pueden tener una resolución que no sea de bits enteros, por lo que la primera resolución que cumple con nuestras necesidades es la de 10 bits.

A la hora de generar tensiones que cambian a gran velocidad es muy importante la tasa de conversión. Buscamos tener la posibilidad de generar una onda con una frecuencia de hasta 10KHz. Para poder representarla correctamente, el DAC debe ser capaz de modificar la tensión de salida a una velocidad mucho mayor. Calculamos la frecuencia de conversión necesaria mediante el teorema de Nyquist. Este establece que para poder reconstruir una señal adecuadamente son necesarios, al menos, diez puntos por cada ciclo. Es decir, la frecuencia de conversión debe ser diez veces mayor a la de la onda que se desea generar. De esta forma, el convertidor será capaz de generar una onda con diez tensiones diferentes en cada ciclo, simulando una onda analógica relativamente continua.

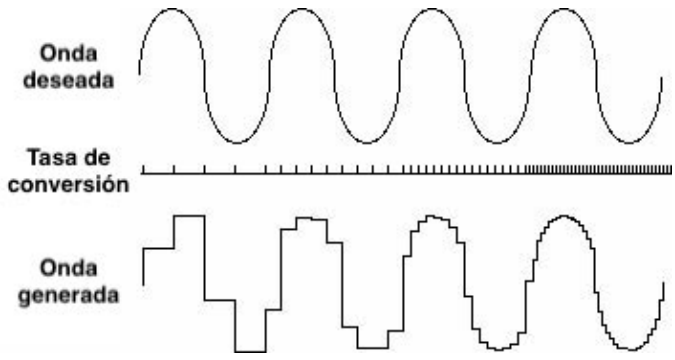


Fig. 28: Representación de una onda generada con diferentes tasas de conversión. Wikiwand

En la figura 28 se muestra el resultado de tratar de generar una señal alterna pura con diferentes tasas de conversión. Cuanto mayor es la tasa de conversión (lado derecho de la gráfica), más se aproxima la onda generada a la onda deseada.

El *Slew-Rate* determina cuál es el cambio máximo que el convertidor es capaz de provocar en la tensión de salida. Esto está limitado por el amplificador de la etapa de salida. Como se ha comentado anteriormente, se desea poder generar señales de hasta 10KHz en todo el rango de funcionamiento. Por lo tanto, el *Slew-Rate* debe permitir que se produzcan variaciones de 5V en 50us o menos (50us es el tiempo de medio periodo en una onda de 10KHz), lo que equivale a un *Slew-Rate* de 0'1V/us.

Con todos estos datos se realiza la búsqueda del integrado. El objetivo es encontrar un DAC comercial con interfaz I2C, de 10 o más bits de resolución, más de 100ksps, con un Slew-Rate de al menos 0'1V/us y de montaje superficial. El convertidor *MCP4725* de *Microchip* cumple perfectamente con estos requisitos. Sus características son:

- Interfaz I2C con 8 posibles direcciones y tres velocidades (100kbps, 400kbps y 3.4Mbps)
- 12 bits de resolución; con una alimentación de 5V equivale a 1'22mV por paso.
- 6us de tiempo de conversión; equivale a una frecuencia de actualización de 166 KHz.
- *Slew-Rate* de 0'55V/us. Esto permite generar una onda de 5V pico a pico a 55 KHz.

El *MCP4725* cuenta además con una memoria EEPROM interna, un modo de bajo consumo, salida Rail-to-Rail y un empaquetado SOT-236. Este encapsulado es de seis pines: dos de alimentación, uno para la salida y tres para el bus I2C. Estos tres últimos son la línea de reloj SCL, la línea de datos SDA y un pin A0 de selección de dirección. Este sirve para poder implementar dos DACs de este mismo modelo en un único bus sin que generen colisiones. Al conectar el pin A0 a masa se fija su dirección I2C en 1100000. En el esquema este convertidor recibe la denominación *U501*.

Este subsistema lo completan dos condensadores, *C501* y *C502* de tipo A y B respectivamente. Estos son los condensadores de desacoplo del convertidor. Se disponen en la alimentación de igual forma que con cualquier otro integrado. Aunque esto es especialmente importante en los convertidores para estabilizar la tensión de alimentación ya que, si no cuentan con ningún pin de referencia, esta es la propia tensión de alimentación. Por lo tanto, el voltaje de salida depende del de alimentación; y cualquier ruido provocará imprecisiones.

5.7 RC/RLC

Este subsistema se encarga de hacer pasar la tensión generada por el DAC a través de unos filtros RC y/o RLC. Estos filtros tienen que poder activarse, desactivarse y ser reconfigurables vía software. Las especificaciones que se han de cumplir son las siguientes:

- Los dos filtros deben poder activarse independientemente.
- En caso de activarse ambos filtros irá primero el RC y posteriormente, en serie, el RLC.
- El circuito RC debe contar con una resistencia de 100Ω, un condensador fijo en torno a 2uF y dos condensadores seleccionables de aproximadamente 5uF y 15uF.
- El filtro RLC debe contar con una resistencia variable entre 0Ω y unos 500Ω, una inductancia fija cercana a 50mH y un condensador final activable de 1uF.
- El filtro RLC debe poder alternar entre un comportamiento subamortiguado, sobreamortiguado y con amortiguamiento crítico.

Para acondicionar los filtros, asegurando su correcto funcionamiento y protegiendo el resto de los componentes de la placa, se decide emplear una etapa de entrada con un seguidor de tensión. También se protege la salida mediante un par de diodos Schottky que evite tensiones fuera del rango de 0-5V.

En este subsistema existen tres apartados en los que debemos prestar una especial atención. En primer lugar, se intentará sustituir la bobina del circuito RLC por un circuito activo equivalente, con el objetivo de ahorrar espacio y coste económico. En segundo lugar, habrá que encontrar la forma de lograr una resistencia variable vía software de los valores necesarios, al tiempo que permita la corriente adecuada. Y, en tercer lugar, se debe tener cuidado con los métodos de activación y desactivación de los elementos de los filtros, ya que pueden aparecer problemas de activaciones indebidas al tratarse de elementos flotantes.

Gyrator - Convertidor general de impedancia

Para generar una inductancia de gran valor es necesario emplear un bobinado de cobre de considerable longitud, lo que resulta en bobinas voluminosas, caras, con perdidas por calor y que, si no están correctamente apantalladas, pueden generar interferencias magnéticas en sus alrededores. Una solución a estos problemas consiste en emplear un convertidor general de impedancia, o *gyrator*, que se comporte como una inductancia sin necesidad de emplear una bobina. Un circuito empleado típicamente para generar inductancias se muestra, junto a su circuito equivalente, en la figura 29.

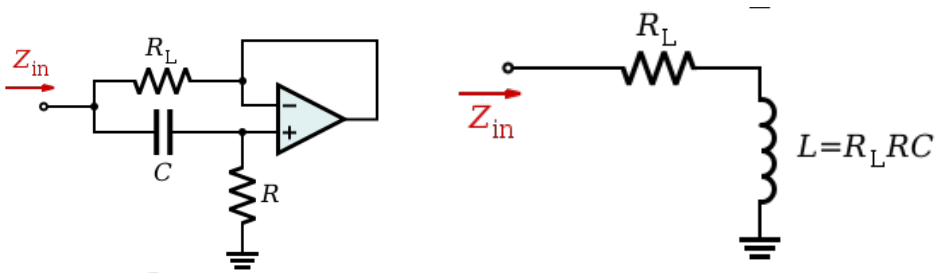


Fig. 29: Gyrator simulando una inductancia y circuito equivalente. Wikipedia

Se decide probar este circuito para comprobar su comportamiento antes de de tomar una decisión de cara al diseño final. Para realizar las pruebas correctamente se decide emplear potenciómetros en lugar de resistencias, esto permitirá probar rápidamente distintas combinaciones de valores. Las medidas se realizarán mediante una resistencia que se dispondrá al principio del circuito. Al medir la caída de tensión en esta resistencia se podrá conocer la corriente que atraviesa el circuito. El esquema final y la placa de pruebas se muestran en la figura 30.

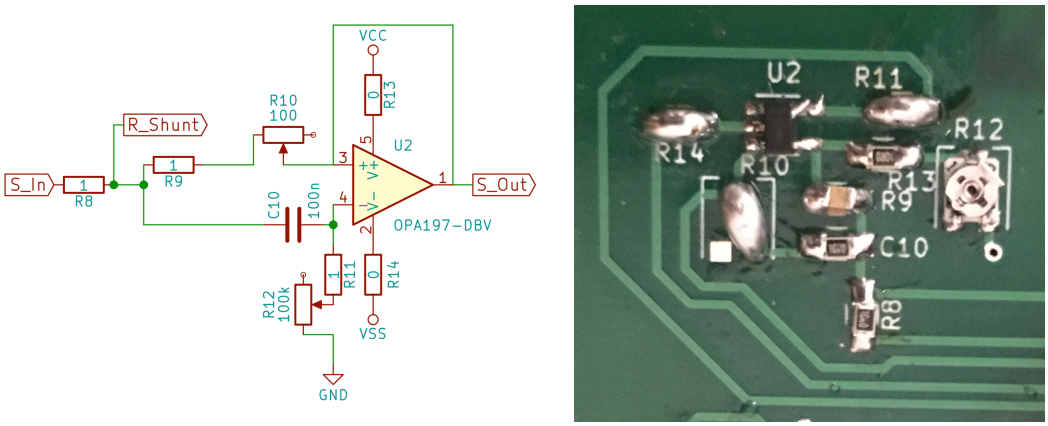


Fig. 30: Pruebas del *gyrator*. Fuente propia

Al probar el circuito, y tras realizar algunos ajustes, se puede ver cómo la corriente que atraviesa el circuito aumenta progresivamente. Este es el comportamiento típico de las inductancias. Además, podemos comprobar cómo este comportamiento es adecuado en un amplio rango de frecuencias y que la inductancia equivalente, obtenida a partir de la constante de tiempo del circuito (R/L), coincide con las formulas presentadas previamente. Se puede observar el resultado de dos pruebas a frecuencias de 100Hz y 1KHz en la figura 31.

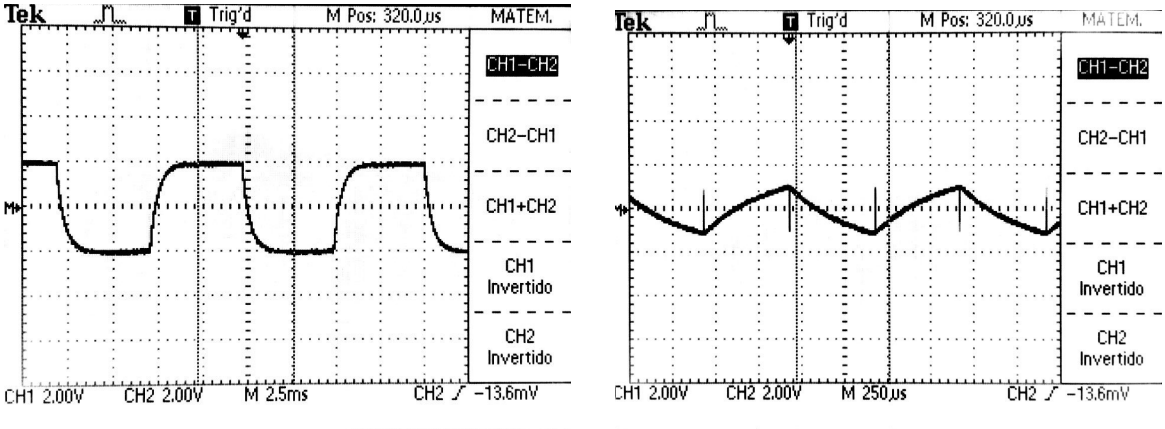


Fig. 31: Pruebas del gyrator a distintas frecuencias. Fuente propia

Este circuito permite generar inductancias de valores elevados en un espacio muy reducido y con componentes muy económicos. Además, apenas requiere de dos resistencias, un condensador y un amplificador operacional, por lo que puede resulta muy rentable a nivel económico.

El mayor inconveniente es que la inductancia generada siempre tiene que estar referenciada a masa. Esto resulta aceptable en la implementación de algunos filtros, como el filtro RL paso alto. Sin embargo, su implementación en el *UMA_AEB_V2.0.0* supondría una modificación importante del comportamiento del filtro RLC. Por lo tanto, es necesario consultar con los profesores que emplearán el escudo antes de tomar una decisión definitiva.

Tras comentar los resultados con los profesores, se decide que no resulta conveniente modificar la estructura original del filtro RLC en serie. En consecuencia, se continúa con la búsqueda de otra alternativa para generar la inductancia.

Para encontrar otra solución a la sustitución de la bobina, decidimos recurrir a la idea base del gyrator, y no a sus implementaciones típicas, ya que no se amoldan a nuestras necesidades. Por definición, un gyrator es un componente que acopla la tensión de cada uno de sus lados con la corriente del otro lado; se trata de una fuente de corriente generada por tensión.

Por lo tanto, se pueden emplear dos fuentes de corrientes controladas por tensión en antiparalelo para crear un gyrator de uso general. Se procede a simular el comportamiento de este circuito mediante el software online *Falstad*.

En primer lugar se dispone el circuito con el comportamiento deseado, incluyendo la bobina. Se emplea un generador de ondas cuadradas de 40Hz y una carga de una resistencia de 100Ω en serie con una bobina de 100mH. Se procede a duplicar este circuito, cambiando la bobina por dos fuentes de corriente controladas por tensión en antiparalelo, tal y como se puede ver en la figura 32.

Al colocar un condensador en el segundo puerto del gyrator, el puerto del primero se comportara como una inductancia de un valor en henrios igual a la capacidad del condensador en faradios.

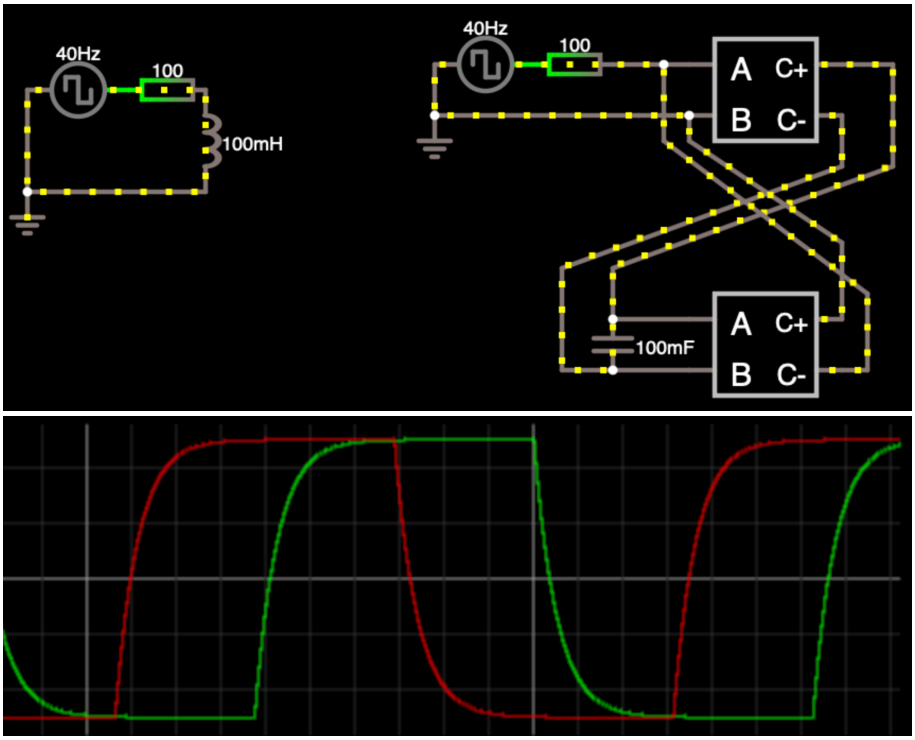


Fig. 32: Simulación del *gyrator* con fuentes controladas en Falstad. Fuente propia

Se puede comprobar que, empleando dos fuentes de corriente, una resistencia y un condensador, se obtiene un efecto idéntico al de un circuito RL en serie. Se han desfasado π/2 las señales de los generadores para poder distinguir las dos gráficas, puesto que son idénticas. Este es el comportamiento deseado; la dificultad ahora reside en encontrar fuentes de corriente controladas por tensión que permitan su implementación.

Las fuentes de corriente controladas por tensión con salida aislada quedan rápidamente descartadas por motivos de coste y de espacio, ya que ocuparían y costarían mucho más que una bobina pasiva. Esta implementación no generaría ninguna ventaja respecto a la bobina tradicional.

Otra posible implementación consiste en emplear amplificadores de transconductancia variable, OTA por sus siglas en inglés. Se trata de un amplificador con salida en corriente, por lo que en la práctica es un fuente de corriente controlada por tensión de pequeña escala. Sin embargo, existen varios inconvenientes en su funcionamiento. Los más importantes consisten en su baja capacidad de entrega de corriente y en que su salida no se encuentra aislada, sino referenciada a masa. Su símbolo electrónico, donde se puede ver que su salida no es aislada, se muestra en la figura 33.

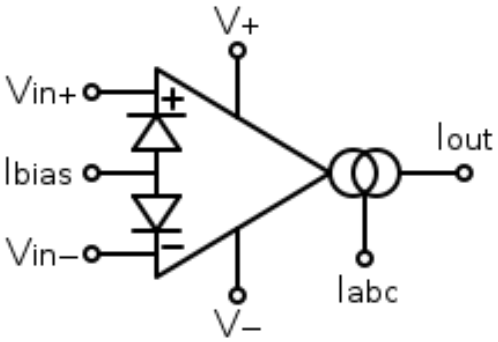


Fig. 33: Símbolo electrónico de un amplificador de transconductancia. Wikipedia

La falta de aislamiento imposibilita una conexión en antiparalelo, como la empleada en la simulación de la figura 32. Además, existen otros inconvenientes que afectarían negativamente aunque pudiéramos realizar la conexión en contraparalelo: pocos modelos en el mercado, elevado precio y la ya mencionada escasa corriente de salida. Queda por lo tanto descartado el amplificador de transconductancia variable como método para implementar el Gyrator.

Finalmente decidimos cesar en el intento de sustituir la bobina por una inductancia activa. Se implementa una bobina pasiva como inductancia del filtro RLC, de igual forma que en la *UMA_AEB_V1.1.0*.

Resistencia variable - Potenciómetro digital

A continuación, se van a exponer los diferentes métodos estudiados para implementar una resistencia para el filtro RLC que sea variable mediante el software. Disponer de una resistencia variable resulta trivial si empleamos un potenciómetro. El inconveniente de este método radica en que las variaciones deben realizarse de forma física. Lograr que un microcontrolador sea el que modifica el valor resistivo no es algo tan fácil. Existen dos métodos fundamentales para modificar el valor de una resistencia de forma digital.

El primer método consiste en emplear un potenciómetro cuyo eje sea solidario al eje de un pequeño motor. De esta forma, controlando la posición angular del motor se controla la del potenciómetro y, por lo tanto, su valor resistivo. Este método puede tener problemas en cuanto a precisión, pero su principal inconveniente reside en el tamaño y el peso. Por lo tanto, no resulta un método viable para el presente escudo.

El segundo método consiste en utilizar una serie de resistencias. Al controlar de forma digital cuántas resistencias actúan entre dos puntos, mediante la activación de interruptores, se obtiene un potenciómetro digital. Esto se puede implementar mediante un multiplexor donde se ha colocado una resistencia entre cada entrada y sus contiguas. Existen integrados comerciales que llevan a cabo esta función. Algunos de ellos emplean protocolos de comunicación I2C, haciéndolos ideales para nuestra aplicación.

Un ejemplo, entre muchos otros, sería la serie *AD5247* de *Analog Devices*. Se trata de integrados que permiten una gran cantidad de pasos (hasta 1024), por lo que no existiría una diferencia relevante de precisión o posibles valores respecto a un potenciómetro manual.

El principal aspecto deficiente que presenta este tipo de integrados es la corriente máxima que permiten. En el caso de la serie mencionada anteriormente, la corriente máxima pulsante es de 20mA, reduciéndose este valor hasta únicamente 5mA cuando la corriente es continua. Si el valor deseado fuera ligeramente superior, sería posible implementar dos potenciómetros en paralelo y controlar ambos de forma simultánea. Sin embargo, para el correcto funcionamiento de este circuito resulta necesaria una corriente hasta de 50mA. Por lo tanto, serían precisos demasiados integrados, así que se opta por descartar el uso de integrados comerciales. Aun así, la idea de emplear una serie de resistencias nos parece la más adecuada.

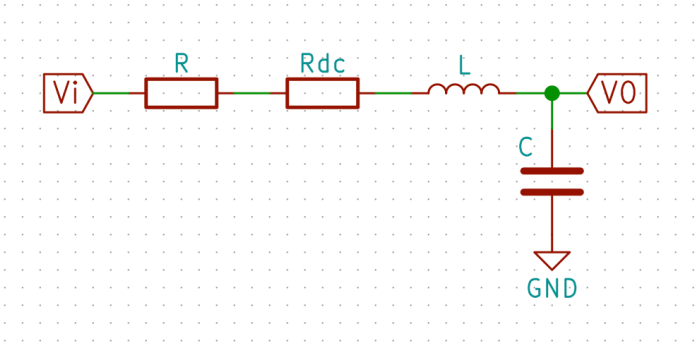
Tras no encontrar ninguna solución comercial que satisfaga las necesidades del proyecto, se decide que la única solución válida consiste en implementar un potenciómetro digital mediante resistencias discretas e interruptores. El problema de esta solución es que el tamaño y el coste aumentan en relación con la cantidad de valores resistivos distintos que se puedan seleccionar.

Junto al profesorado se decide que se van a emplear dos resistencias de valores diferentes. Estas se deben poder disponer en paralelo para generar un tercer valor. Además, también se desea poder eliminar completamente la resistencia en serie con la bobina.

Los valores resistivos se escogerán de forma que cada posible valor resistivo genere un tipo distinto de amortiguamiento en el filtro RLC. Para ello es necesario conocer qué valor resistivo produce el amortiguamiento crítico. Los valores superiores producirán sobreamortiguamiento y los valores inferiores producirán subamortiguamiento (también denominado sobreoscilación).

Para calcular este valor se debe analizar el comportamiento ideal del filtro. La base del análisis con las ecuaciones que relacionan la tensión de entrada (Vi) y la tensión de salida (Vo) con la corriente que atraviesa el filtro. Para deducir estas ecuaciones tan solo es necesario conocer el comportamiento individual de cada componente: mostradas en la figura 34.

El circuito que se va a analizar estará formado por dos resistencias (la resistencia variable y la resistencia serie de la bobina), una inductancia y un condensador. En la figura 34 se muestra el circuito que vamos a emplear.



$$V_R = R \cdot i$$

$$V_L = L\frac{di}{dt}$$

$$V_C = \frac{1}{C}\int i\,dt$$

Fig. 34: Esquema del filtro RLC y ecuaciones de comportamiento individuales. Fuente propia

Al combinar estas ecuaciones individuales se puede obtener la relación que aparece entre la corriente que atraviesa el circuito y diferentes caídas de tensión. Las relaciones de la intensidad con los voltajes de entrada y de salida se muestran en la figura 35.

$$V_i = i(R + R_{DC}) + L\frac{di}{dt} + \frac{1}{C}\int i\,dt$$

$$V_o = \frac{1}{C}\int i\,dt$$

Fig. 35: Tensiones de entrada y salida en función de la corriente. Fuente propia

Al derivar ambos términos de la ecuación de la tensión de salida, podemos despejar la corriente. Luego se puede sustituir esta expresión en la ecuación de la tensión de entrada; de esta forma se obtiene una única ecuación general que relaciona la tensión de entrada con la tensión de salida, tal y como se muestra en la figura 36.

$$i = C\frac{dV_o}{dt}$$

$$C\frac{dV_o}{dt}(R + R_{DC}) + LC\frac{d^2V_o}{dt} + V_o = V_i$$

Fig. 36: Ecuación de comportamiento del sistema. Fuente propia

Para obtener la función de transferencia entre la tensión de entrada y la tensión de salida es necesario realizar la transformada de Laplace. Esta permite trabajar más fácilmente con los términos de las derivadas de la corriente. Tras realizar la transformación se puede despejar la relación entre ambos voltajes, mostrada en la figura 37.

$$CV_o(R+R_{DC})s+LCV_os^2+V_o=V_i$$

$$\frac{V_o}{V_i}=\frac{1}{C(R+R_{DC})s+LCs^2+1}$$

Fig. 37: Relación de la tensión de entrada con la de salida. Fuente propia

El comportamiento de amortiguamiento crítico se dará cuando exista un único polo doble en el denominador de la función de transferencia. Por lo tanto, en la fórmula general empleada para hallar los polos de esta función, mostrada en la figura 38, debemos encontrar el valor de R que anula la raíz cuadrada del numerador.

$$\frac{-C(R+R_{DC})\pm\sqrt{C^2(R+R_{DC})^2-4LC}}{2LC}$$

Fig. 38: Cálculo de los polos de la función de transferencia. Fuente propia

Como lo único que importante para hallar el polo doble es el contenido de la raíz, es posible omitir el resto de la fórmula de la figura 38. Después es necesario despejar *R* en función del resto de términos; y, tras ello, únicamente será necesario sustituir los valores empleados en los otros componentes para conocer la resistencia necesaria en cada caso.

$$C^2(R+R_{DC})^2-4LC=0$$

$$R=2\sqrt{\frac{L}{C}}-R_{DC}$$

Fig. 39: Cálculo de la resistencia necesaria para el amortiguamiento crítico. Fuente propia

Si aplicamos la fórmula de la figura 39 con los valores nominales de los componentes del filtro (bobina de 47mH y condensador de 1uF), obtenemos un resultado de 381Ω. Sin embargo, las tolerancias de los componentes, incluyendo la resistencia serie de la bobina, provocan que este valor pueda variar en varias decenas de Ohmios.

Para estudiar todas las posibles combinaciones y calcular la resistencia necesaria en cada caso empleamos la tabla de la figura 40. También hemos representado gráficamente en la figura 41 la variación del valor resistivo que provoca amortiguamiento crítico en función del valor real del resto de componentes.

El amortiguamiento crítico ocurre con una resistencia de entre 348Ω y 418Ω en función del valor exacto del resto de componentes. De acuerdo a esto, y a la decisión previa de disponer dos resistencias que puedan colocarse en paralelo, los valores que hemos decidido emplear son 520Ω y 390Ω. Al colocarlos en paralelo se genera una resistencia equivalente de 220Ω. La resistencia de 520Ω siempre generará sobreamortiguamiento, la resistencia de 390Ω se encontrará alrededor del amortiguamiento crítico y la combinación de ambas siempre provocará subamortiguamiento.

Componente	Rdc	L	C	R
Tolerancia	5 %	5 %	10 %	-
Nominal	52	4,70E-02	1,00E-06	**381,59**
	49,4	4,47E-02	9,00E-07	396,07
	49,4	4,47E-02	1,10E-06	353,54
R Máxima	49,4	4,94E-02	9,00E-07	**418,93**
	49,4	4,94E-02	1,10E-06	374,22
	54,6	4,47E-02	9,00E-07	390,87
R Mínima	54,6	4,47E-02	1,10E-06	**348,34**
	54,6	4,94E-02	9,00E-07	413,73
	54,6	4,94E-02	1,10E-06	369,02

Fig. 40: Tabla de posibles valores para la resistencia. Fuente propia

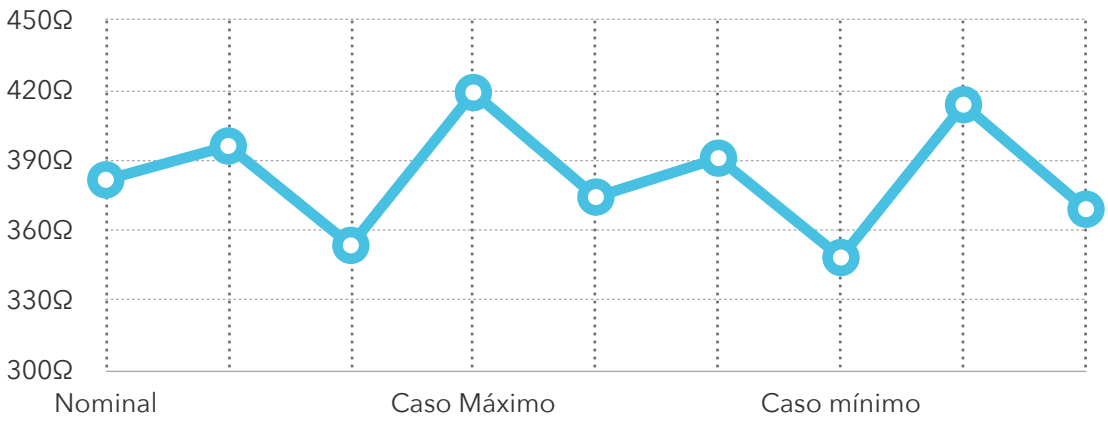


Fig. 41: Representación de los diferentes valores de la resistencia. Fuente propia

Interruptores digitales

Controlar las activaciones y desactivaciones de los elementos flotantes de los filtros no resulta una tarea trivial. La primera solución que nos planteamos, siendo satisfactoria en una gran cantidad de aplicaciones, consiste en emplear transistores MOSFET. La activación de los MOSFET se controla mediante diferencia de tensión entre la base y el colector (tipo P) o emisor (tipo N). Además, debido a su construcción, los MOSFET siempre permiten conducir en uno de los dos sentidos. Los MOSFET se suelen emplear referenciados siempre a una tensión conocida, normalmente la tensión de alimentación (tipo P) o masa (tipo N). Esto consigue que siempre exista una tensión conocida en el terminal que afecta al control de la activación.

Sin embargo, estas características pueden generar una serie de comportamientos indeseados si el transistor no se encuentra referenciado a una tensión conocida, como ocurre en nuestro circuito. A continuación, se muestra el comportamiento en un circuito RC donde se emplea un MOSFET de tipo P en flotante para activar o desactivar el condensador. Vamos a realizar las pruebas mediante el simulador online *Falstad*. El circuito empleado consta de una resistencia, un condensador, un generador de ondas y transistor MOSFET de tipo P; la disposición de los componentes se muestra en la figura 42.

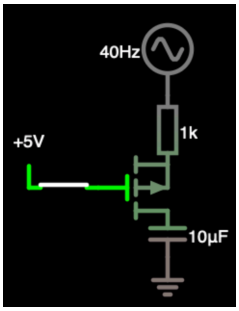


Fig. 42: Circuito empleado en la primera simulación: MOSFET P. Fuente propia

Al comenzar la simulación se mantiene el interruptor cerrado, por lo que la tensión en la base del MOSFET es de 5V. La fuente superior genera tensión alterna senoidal de 0V a 5V. Como la tensión de base siempre es igual o mayor que la tensión de colector, el interruptor permanece cerrado y no circula ninguna corriente.

Cuando se abre el interruptor la tensión de base cae a 0V, en un circuito real se implementaría una resistencia de *Pull-Down* para asegurar un correcto funcionamiento. Debido a la diferencia de tensión entre la base y el colector, el MOSFET comienza a conducir. En este escenario aparecen dos problemas. La tensión generada es de 0V a 5V, por lo que en algunos momentos la tensión de colector es menor a la tensión que necesita el MOSFET para activarse. Esto se puede observar en la figura 43 como unas pequeñas discontinuidades en el gráfico de corriente en el paso por cero. El condensador reduce este efecto al mantener la tensión por encima de ese valor. Sin embargo, si el transistor controlara la activación o desactivación de una resistencia, el efecto sería mucho más acusado.

El otro problema surge cuando se cierra el interruptor con el circuito cargado. El MOSFET vuelve a cortar el paso de la corriente; sin embargo, el condensador mantiene una diferencia de tensión entre sus terminales. Cuando esta supera la tensión de la fuente, el diodo interno de libre circulación permite el paso de corriente en sentido inverso. Esto se puede ver en la última parte de la figura 43.

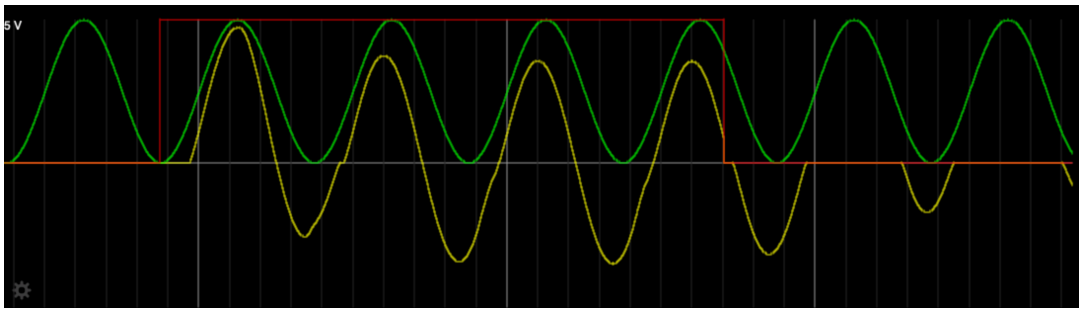


Fig. 43: Gráfico de la simulación empleando un MOSFET tipo P: tensión de base del MOSFET (rojo), tensión generada (verde) y corriente (amarillo). Fuente propia

Es necesario tener en cuenta que este transistor no puede colocarse encima de la resistencia, como sería típico de un MOSFET tipo P. Esto viene determinado por las restricciones de diseño, ya que se tiene que poder controlar la activación de los condensadores independientemente de la resistencia.

Si empleásemos transistores de tipo N, que se activan mediante una tensión de base superior a la tensión de colector, también aparecerían problemas. En este caso colocaríamos el transistor referenciado a masa. Así se evitaría el problema existente con la tensión de activación. Sin embargo, la carga acumulada en el condensador sigue provocando inconvenientes. Cuando el condensador se carga genera una diferencia de tensión entre sus terminales. Si el condensador se encuentra cargado y el transistor deja de conducir, el terminal

negativo del condensador pasa a estar en circuito abierto. La tensión del terminal positivo iguala su tensión con la de la fuente, y la tensión del terminal negativo mantiene la diferencia.

Por ejemplo, si el condensador acumula 3V de carga y la tensión de alimentación baja a 0V, la tensión del terminal positivo será de 0V. El terminal negativo tiene que mantener la diferencia de tensión, ya que al no haber corriente no puede existir un cambio en la carga acumulada; por lo tanto, el voltaje en el segundo terminal del condensador será de -3V. Este terminal está conectado con el MOSFET tipo N. Se da el caso de que el transistor tiene menor tensión en el emisor que en el colector, así que comienza a conducir en sentido inverso debido al diodo parásito. Esto puede verse en la figura 44.

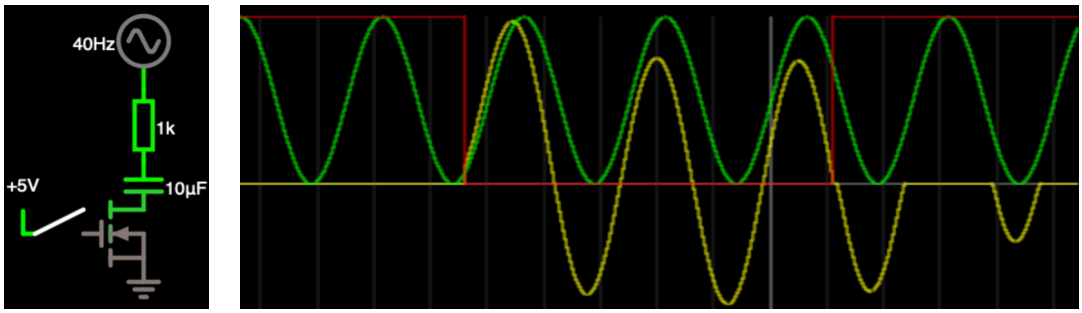


Fig. 44: Simulación empleando un MOSFET tipo N: Switch (rojo), tensión generada (verde) y corriente (amarillo). Fuente propia

La combinación de los comportamientos de la tensión de puerta de los MOSFET, el diodo parásito y la acumulación de carga del condensador provoca que ninguna combinación de menos de cuatro MOSFET nos dé el resultado adecuado. Además, la combinación de cuatro MOSFET que nos sirve no es la de un puente en H completo, por lo que no podemos emplear integrados comerciales. Esta solución requeriría de una gran cantidad de integrados distintos, incluyendo los de control.

Al no poder emplear MOSFETs de la forma habitual, procedemos a realizar un listado de todas las tecnologías o circuitos que nos podrían servir. Las opciones que nos planteamos para resolver la activación son las siguientes:

1. Emplear cuatro MOSFETs (dos P y dos N) controlados de 0V a +5V.
2. Emplear dos MOSFETs controlados de -15V a +15V .
3. Emplear interruptores analógicos (*Analog Switch* en inglés).
4. Emplear relés de estado sólido (SSR).
5. Emplear relés convencionales (electromecánicos).

La primera opción, emplear cuatro MOSFETs, tiene el problema de que se necesitan una gran cantidad de integrados, con sus respectivos métodos de control. El circuito requiere de la implementación de 8 interruptores. Esto implica entre 8 y 16 integrados de transistores (pueden ir 2 o 4 por empaquetado), integrados de inversión lógica para el control (dependiendo del número de puertas) y un expansor de E/S. Todo esto exige una gran cantidad de componentes, implicando mucho espacio, una gran cantidad de pistas, un elevado coste económico y una reducción de fiabilidad.

Si se emplean MOSFETs controlados a ±15V, nuestra segunda opción, se eliminan algunos de los problemas, fundamentalmente los asociados a las activaciones indeseadas. Seguirían siendo necesarios entre 8 y 16 integrados, dependiendo de si encontramos empaquetados dobles. Además, este método complica el control, ya que se debería realizar a

±15V y no podemos emplear integrados de lógica de 0V a +5V. Por otro lado, esta solución presenta el problema de que la alimentación de ±15V no está pensada para suministrar la corriente necesaria para alimentar los integrados necesarios.

La tercera opción consiste en emplear interruptores analógicos. Es una solución menos generalizada que las otras. Se trata de integrados cuyo funcionamiento es similar al de los relés de estado sólido, pero que no mantienen aislamiento entre la entrada de control y los pines de señal. Internamente suelen funcionar mediante dos IGBTs controlados mediante un circuito de activación que asegura un adecuado funcionamiento. Se trata de componentes completamente bidireccionales, por lo que solo sería necesario un integrado por interruptor, además del expansor E/S.

Las dos últimas opciones tienen un comportamiento óptimo pero presentan problemas de tamaño y precio, originados por el aislamiento óptico (que no necesitamos) en el caso de los SSR y por el uso del electroimán en los relés convencionales. Ambas soluciones quedan rápidamente descartadas.

En conclusión, se descartan las opciones 4 y 5 por los motivos aludidos en el párrafo anterior y se desechan la primera y la segunda por problemas de funcionamiento. Tan solo queda una opción, aunque no se conoce si existe algún problema con esta tecnología. Se procede a la compra de interruptores analógicos, concretamente el modelo *TS5A3166* de *Texas Instruments*, para comprobar su funcionamiento. Se ha escogido este integrado porque sus características de tensión y corriente encajan con las necesidades de los ocho interruptores de este subsistema. También destaca por su baja resistencia serie cuando está activado, siendo menor a un Ohmio.

Realizamos pruebas de activación y desactivación con el mismo circuito de las simulaciones anteriores (Fig. 42 a 44). Durante las pruebas comprobamos que el comportamiento del interruptor analógico es muy similar a un SSR. Aparte de la falta de aislamiento, la única diferencia es que, si se supera la tensión de alimentación en los pines de contacto, el interruptor comienza a conducir. Esto se debe a dos diodos internos de protección que aseguran que las tensiones en los pines de señal nunca superen la alimentación.

El resultado de las pruebas es satisfactorio, por lo que se decide que el funcionamiento del *TS5A3166* es adecuado para este subsistema. Tan solo hay que tener en cuenta que si la tensión se vuelve negativa o supera los 5V se activan los diodos internos y el sistema pasa a conducir. Esto provoca que no sea adecuado colocarlos entre los condensadores y masa, debido a la posible aparición de tensiones negativas por la carga acumulada de condensadores volantes. Pero sí se pueden colocar sin ningún problema por encima de estos, ya que son integrados completamente bidireccionales, al contrario que los MOSFET.

Implementación final - Etapa de entrada

Una vez que hemos resuelto los principales problemas del subsistema, podemos pasar a su implementación definitiva. La parte fundamental del subsistema está formada por los filtros y sus protecciones. Existirán protecciones tanto a la entrada como a la salida.

Al comienzo del subsistema se monta un amplificador operacional, el *U602*, con una configuración en modo seguidor para que actúe como buffer. De esta forma se separa y protege al DAC frente al resto del circuito. Se ha calculado que este amplificador debe ser un amplificador *Rail-to-Rail* (funcionamiento correcto en todo el rango de alimentación) capaz de entregar al menos ±50mA y permitir ondas de 10KHz o más.

Se ha escogido el modelo *NCV20081* de *ON Semiconductor* ya que es un amplificador operacional *Rail-to-Rail*, capaz de entregar hasta ±100mA y seguir frecuencias de hasta 83KHz, puesto que su Slew-Rate es de 0'4V/uS. Todo ello viene en un encapsulado *SC-70* y a un precio muy asequible.

Como protección adicional, se coloca una resistencia que limita la corriente entre el DAC y el amplificador. Esta resistencia también limita el ruido recibido por el buffer debido a las radiaciones electromagnéticas externas. El valor de esta resistencia no es algo relevante, puesto que la entrada al amplificador operacional ya es de alta impedancia. Se decide mantener el valor de la versión anterior de la placa: 1KΩ. Con anterioridad se ha empleado algunas resistencias de este valor, por lo que se estandariza al mismo modelo, el *CRG0603F1K0* de *TE Connectivity*. Esta resistencia se denomina *R601*.

Al igual que en el resto de integrados se filtra la alimentación del amplificador operacional mediante dos condensadores de desacoplo. Uno de ellos de tipo A, el *C603*, y otro de tipo B, el *C604*.

Implementación final - Filtro RC

La resistencia del filtro RC, *R602*, mantendrá el mismo valor que en la versión anterior de la placa, 100Ω. Se monta el modelo *MCR03EZPFX1000* de *ROHM*. Se trata de la misma serie que se ha usado en otras resistencias ya que tiene una tolerancia muy buena (1%) con un precio muy asequible (menos de medio céntimo por unidad). Para poder desactivar el filtro RC debe existir la posibilidad de cortocircuitar esta resistencia. Con este objetivo se implementa un interruptor analógico en paralelo con ella. El circuito de implementación concreto de los interruptores será tratado más adelante.

El primer condensador de este filtro será de 2'2uF, nombrado como *C605* en el esquema. Los valores de todos los elementos de los filtros vienen fijados como requisitos del profesorado, por lo que no es necesario realizar cálculos, pero tampoco cabe ninguna modificación. Este condensador no debe ser físicamente muy pequeño, ya que afecta a la linealidad de su comportamiento. Para lograr un compromiso entre linealidad y espacio, se ha decidido emplear huellas (1206) o (1210). Tras ver las opciones del mercado se decide emplear el modelo *12103C225KAT2A* de *AVX*. Este tiene una huella (1210) y una tolerancia de ±10% gracias a su dieléctrico X7R. Este condensador estará siempre activo debido a especificaciones de diseño, por lo que no le colocaremos ningún *Analog Switch* en las inmediaciones.

El segundo condensador del filtro debe ser de un valor de aproximadamente 5uF. Los condensadores *C208*, *C211* y *C213* del subsistema de alimentación son de 4'7uF y tolerancia de -20%+80%, por lo que su valor real oscila en torno a 5uF. Estos condensadores se habían elegido anteriormente con una huella (1206) pensando en la implementación de los mismos en este subsistema, donde su tamaño sí resulta relevante. En consecuencia, el condensador *C606*, segundo condensador del filtro RC, será el modelo *1206YG475ZAT1A* de *AVX*.

El tercer condensador ha de ser de un valor en torno a 15uF. Con el objetivo de estandarizar, y ante la escasez de condensadores de 15uF del mercado que encajen de forma satisfactoria con las necesidades de este circuito, hemos decido colocar en paralelo tres condensadores de 4'7uF: *C607*, *C608* y *C609*. Esto eleva su capacidad equivalente a 14'1uF (-20% +80%) a la vez que se reduce la resistencia serie a un tercio. Además, ello nos permite emplear nuevamente el modelo *1206YG475ZAT1A* de *AVX*, ahorrando costes.

Implementación final - Filtro RLC

Como se ha calculado anteriormente, se implementarán dos resistencias en el filtro RLC: una de 390Ω y otra de 520Ω. La resistencia de 390Ω, *R613*, será el modelo *CRG0805-F390R* de *TE Connectivity*. Se ha escogido este modelo por su tolerancia y su bajo precio, aunque tiene el inconveniente de estar fabricado con una huella (0805) ligeramente mayor a la de la mayoría de componentes pasivos (0603); sin embargo, no se ha encontrado ningún otro modelo que resulte más adecuado.

La resistencia de 520Ω se va a lograr al colocar en serie dos resistencias que ya se encuentran en la placa, esto permite estandarizar componentes, reduciendo el coste final del

escudo. Además, esto aumenta las posibilidades de ajustar el valor de forma precisa en caso de desear un cambio en el valor resistivo más adelante. Se emplean las resistencias de 220Ω, *R611*, y de 300Ω, *R612*. Los modelos, ya implementados en ocasiones anteriores, son el *MCR03EZPFX2200* y el *MCR03EZPFX3000* de *ROHM*.

Como ya se ha decidido, se va a montar la misma bobina que en la *UMA_AEB_V1.1.0*, ya que nuestros intentos de generar una inductancia activa no han dado los resultados esperados. El modelo concreto es el *RL181S-473J-RC* de *Bourns*. Se trata de una bobina de 47mH y 50Ω de resistencia serie que recibe la denomincación *L601*.

El condensador *C610* de *1uF* completa el filtro RLC. No montaremos, pese a que sea del mismo valor, un condensador de tipo A. En su lugar emplearemos el modelo *222278115663* de *Phycomp*. Debido a su mayor tamaño (1206) y su dieléctrico X7R, debería proporcionarnos un comportamiento más predecible y estable en el filtro.

Implementación final - Etapa de salida

Para evitar tensiones negativas o superiores a 5V que puedan dañar algún componente, se incluyen dos diodos Schottky como etapa de salida. El comportamiento del filtro en configuración LC puede provocar estas tensiones, que son peligrosas para los interruptores o el ADC. El *TS5A3166* dispone internamente de diodos con este mismo propósito, pero se decide montar el diodo doble *D601* como protección adicional de la línea. Este diodo será el mismo modelo empleado anteriormente en el subsistema de alimentación: el *DB5S308K0R* de *Panasonic*.

La configuración final de los filtros, junto a las etapas de entrada y salida, se muestra en la figura 45.

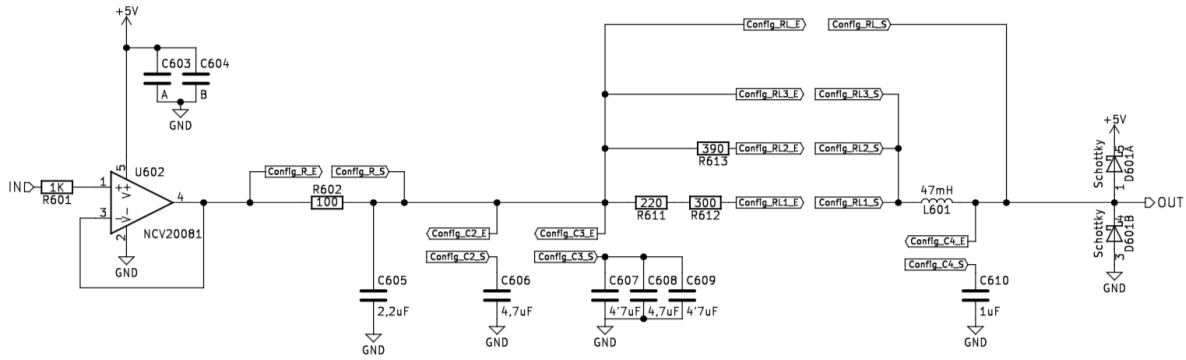


Fig. 45: Filtros del subsistema RC/RLC junto con las etapas de entrada y salida. Fuente propia

Implementación final - Interruptores

Como se puede ver en la figura 45, se ha decidido emplear ocho interruptores analógicos para controlar la configuración de los filtros (las conexiones están representadas por las etiquetas globales). Tres de ellos serán para el filtro RC y cinco para el RLC. No se deben emplear pines del Arduino como control, ya que entonces compartirían dos funciones. Debemos evitarlo, ya que un objetivo de esta versión de la placa consiste en acabar con todas las colisiones de funciones en los pines de Arduino. En su lugar vamos a implementar un integrado expansor de E/S con interfaz I2C, al que denominaremos *U601*.

Los expansores de E/S son integrados que permiten controlar pines digitales a partir de un bus digital. Se emplean para ampliar de forma efectiva la cantidad de GPIOs de un microcontrolador. La mayoría permiten configurar los pines individualmente como entrada o como salida. Los expansores permiten ajustar el nivel lógico de los pines de salida y enviar de vuelta lecturas realizadas en las entradas. En el presente diseño no resulta importante la

velocidad de comunicación del bus, ya que no se va a modificar ni a leer el estado de los pines a gran velocidad.

El expansor que se implemente debe contar con ocho puertos de salida digital de 0V a 5V. Su comunicación debe ser mediante el bus I2C, la velocidad no es un aspecto relevante. Lo que sí resulta fundamental es que la dirección que tiene dentro del bus pueda configurarse, porque más adelante se empleará otro expansor, y es necesario poder diferenciarlos dentro del bus I2C. Tampoco se puede obviar en la búsqueda el tamaño físico del integrado, ya que la mayoría son bastante voluminosos y puede resultar un problema para la distribución de componentes en la placa.

El integrado que hemos encontrado que mejor cumple con estos requisitos es el modelo *PCA9554BS3* fabricado por *NXP*. Cuenta con ocho pines bidireccionales que permiten corrientes hasta 50mA, se comunica mediante I2C a una velocidad de hasta 400KHz, emplea 3 pines para configurar la dirección I2C y emplea un encapsulado de apenas 3x3mm (HVQFN 16pin 3x3). Al conectar los tres pines de dirección del I2C a masa se provoca que, de acuerdo a la hoja de datos técnicos, el integrado responda en el bus a la dirección *0x20*.

En la alimentación del *U601* se disponen dos condensadores de desacoplo: uno de tipo A (*C601*) y uno de tipo B (*C602*).

Como interruptor analógico se emplea el mismo con el que realizamos las pruebas anteriormente: el *TS5A3166* de *Texas Instruments*. Se trata de un interruptor completamente bidireccional de tipo SPST con 0'9Ω de resistencia serie. Su funcionamiento puede verse alterado cuando alguna de las tensiones no está dentro del rango de las tensiones de alimentación. Por lo tanto, y como ya se ha comentado anteriormente, no se van a colocar entre los condensadores y masa, debido a las tensiones negativas que pueden aparecer. Se montan ocho de estos interruptores, desde el *U603* hasta el *U610*; cada uno de ellos contará con un condensador de desacoplo de tipo A, de *C619* a *C626*, y uno de tipo B, de *C611* a *C618*.

Llegados a este punto, el subsistema ya resulta completamente funcional; sin embargo, se decide añadir un indicador LED en cada interruptor para poder saber en todo momento cuál es la configuración de los filtros, sin necesidad de realizar medidas ni recurrir al programa cargado. Todos los LEDs (*D603* a *D610*) se conectarán entre el pin de control del *Switch* de su misma numeración y masa, excepto dos: el *D603* y el *D606*. Los interruptores asociados a estos LEDs son los encargados de puentear los filtros. Por lo tanto, cuando el interruptor conduce, el filtro se encuentra desactivado. En ese caso queremos que el LED se encuentre apagado, y al viceversa. Por ello estos dos LEDs se montan entre el pin de control y la alimentación de 5V.

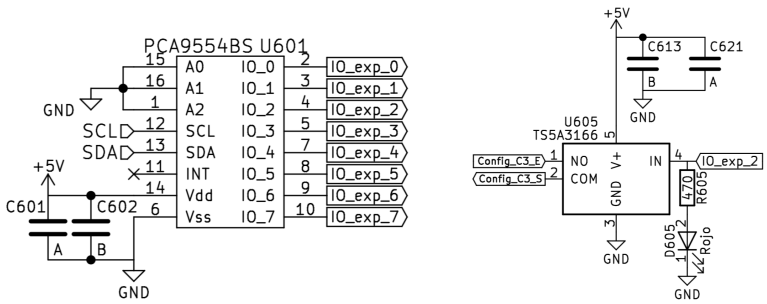


Fig. 46: Esquemas del expansor (izquierda) y Analog-Switch (derecha). Fuente propia

Montaremos el mismo modelo de LED que en el array del subsistema de la interfaz: el *KP-1608SRC-PRV* de *Kingbright*. Sin embargo, no emplearemos las mismas resistencias limitadoras, ya que en este caso la circuito será siempre de 5V. La resistencia óptima para 5V es de 470Ω, así que implementaremos el *MCR03EZPFX4700* de *ROHM*. Se trata de la misma gama de resistencias que hemos usado ya en varias ocasiones por su buena tole-

rancia y bajo precio. Denominaremos estas resistencias *R603* a *R610* manteniendo constante la numeración del LED, del *Analog Switch* y de la resistencia correspondientes.

5.8 Amp. Op. 15V

La función principal de este subsistema es controlar un motor externo en lazo cerrado. Este motor se controla mediante una tensión de consigna que puede ir desde -10V hasta +10V. Este subsistema debe ser capaz de generar una señal de, al menos, esos valores. Al mismo tiempo, también tiene que acondicionar la señal de realimentación del motor. Esta realimentación es del mismo rango de tensión que la salida, y debe tratarse para permitir que el microcontrolador la procese.

Por lo tanto, este subsistema contará con dos etapas diferenciadas: la etapa de salida, que generará una señal de ±15V a partir de una de 0-5V; y la etapa de entrada, que acondicionará la señal de ±15V a niveles que pueda procesar el ADC (0-5V).

En la actualidad es necesario calibrar este subsistema antes de trabajar con él. En este nuevo escudo se ha decidido que, con el objetivo de aprovechar mejor el tiempo disponible en las prácticas de laboratorio, esta calibración se tiene que poder realizar de forma automática vía software. Se considera una calibración correcta aquella que se ha realizado en, al menos, tres puntos alejados entre sí. Uno de ellos debe ser cero (para fijar el desfase), otro debe ser positivo y el útlimo debe ser negativo.

Etapa de salida

La etapa de salida es aquella encargada de convertir la señal del DAC, de 0V a 5V, en la señal de control del motor, de -15V a +15V. En el escudo *UMA_AEB_V1.1.0*, se realiza esta conversión mediante dos etapas consecutivas. En primer lugar, se añade un offset a la señal y posteriormente se amplifica. Todo ello se logra mediante dos amplificadores operacionales independientes. Para reducir la cantidad de componentes necesarios, se va a plantear un circuito que permita usar un único amplificador operacional que realice ambas dos funciones a la vez. Se va a emplear un circuito de sumador-restador ponderado, mostrado en la figura 47.

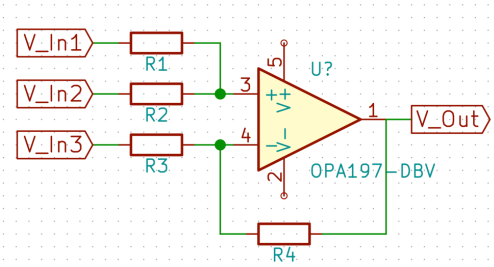


Fig. 47: Esquema del sumador-restador ponderado. Fuente propia

A continuación, se va a analizar matemáticamente el comportamiento de este circuito. Para comenzar se cuantifican las caídas de tensión en los dos lazos abiertos que existen: uno de *V_In1* a *V_In2* y otro de *V_In3* a *V_Out*; de esta forma se pueden despejar *V+* y *V-*. Tras esto se pueden igualar *V+* y *V-* para despejar la tensión de salida. Este proceso se muestra en las figuras 48 y 49.

$$\frac{V_{In1}-V_{+}}{R_1}=\frac{V_{+}-V_{In2}}{R_2}\Rightarrow V_{+}=\frac{V_{In1}\cdot R_2+V_{In2}\cdot R_1}{R_1+R_2}$$

$$\frac{V_{In3}-V_{-}}{R_3}=\frac{V_{-}-V_{Out}}{R_4}\Rightarrow V_{-}=\frac{V_{In3}\cdot R_4+V_{Out}\cdot R_3}{R_3+R_4}$$

Fig. 48: Cálculo de V+ y V- mediante la ley de Ohm en el circuito. Fuente propia

$$\frac{V_{In1}\cdot R_2 + V_{In2}\cdot R_1}{R_1+R_2} = \frac{V_{In3}\cdot R_4 + V_{Out}\cdot R_3}{R_3+R_4}$$

$$V_{In1}\cdot\frac{R_2(R_3+R_4)}{R_3(R_1+R_2)} + V_{In2}\cdot\frac{R_1(R_3+R_4)}{R_3(R_1+R_2)} - V_{In3}\cdot\frac{R_4}{R_3} = V_{Out}$$

Fig. 49: Cálculo de Vout mediante la igualdad de V+ y V-. Fuente propia

Se conoce el rango de la tensión de salida: 30V (-15V +15V). La entrada tiene un rango de 5V, así que entre ambos puntos debe existir una amplificación de 6 veces. Además, la salida debe estar centrada, por lo que es necesario un offset negativo. Se puede ver en la figura 49 que *V_In3* resta tensión, por lo que se emplea como offset negativo. También se puede ver que todas las tensiones son independientes entre sí, por lo que este offset tendrá que ser de -15V. Por último, cabe destacar que únicamente hemos de amplificar una tensión, la del DAC, por lo que uno de los dos primeros términos no es relevante. Se conecta una de las dos entradas a masa, por lo que su término será nulo de cara al cálculo del comportamiento del circuito.

Se implementa un regulador de tensión de 2'5V para asegurar la estabilidad y precisión de *V_In3*. Su término en la ecuación debe alcanzar los -15V, por lo que la amplificación debe ser de 6 veces. Así, se sabe que *R4* debe ser 6 veces superior a *R3*. De acuerdo a esto, y a que todo el término que multiplica a *V_In2* debe ser 6, se puede calcular la relación entre *R1* y *R2*; tal y como se muestra en la figura 50.

$$\frac{R_1(R_3+6R_3)}{R_3(R_1+R_2)} = 6 \Rightarrow \frac{7\cdot R_1}{(R_1+R_2)} = 6$$

$$7\cdot R_1 - 6\cdot R_1 = 6\cdot R_2 \Rightarrow R_2 = \frac{R_1}{6}$$

Fig. 50: Cálculo de la relación entre R2 y R1. Fuente propia

Sabiendo *R4* debe ser 6 veces superior a *R3*, y que *R1* debe ser 6 veces superior a R2, procedemos a buscar los valores valores exactos. Con el objetivo de estandarizarlos se decide que *R4* y *R1* sean del mismo valor, por lo que *R2* y *R3* resultan también del mismo valor. El valor concreto no es algo muy relevante, siempre que se respeten las relaciones entre ellos. Sin embargo, no resulta recomendable emplear valores que demasiado bajos (aumentarían el consumo) o demasiado altos (aumentarían el ruido del circuito). Se decide arbitrariamente emplear valores de 20KΩ (*R2* y *R3*) y de 120KΩ (*R1* y *R4*).

Etapa de entrada

El motor genera una señal de realimentación, también de -15V a +15V, que hay que procesar. Para acondicionarla y poder digitalizarla con el ADC se va a emplear el mismo circuito que en la etapa de salida. Pero se modifican las entradas de las tensiones y los valores de las resistencias a fin de lograr el efecto inverso. Lo que se busca es reducir la amplitud de la onda a una sexta parte y añadir un offset que la deje siempre positiva.

Se vuelve a emplear la fórmula general de la tensión de salida de la figura 49. En este acondicionamiento no es necesario ningún offset negativo, por lo que *V_In3* va a ser 0V. *V_In1* y *V_In2* serán la tensión de offset y la tensión de entrada respectivamente. Se deben encontrar los valores de resistencias que hagan que el término que acompaña a *V_In1* sea 1

(para añadir un offset de 2'5V) y que el término que acompaña a *V_In2* sea 1/6 (para alcanzar los niveles adecuados de tensión).

$$V_{In1}\cdot\frac{R_2(R_3+R_4)}{R_3(R_1+R_2)}+V_{In2}\cdot\frac{R_1(R_3+R_4)}{R_3(R_1+R_2)}=V_{Out}$$

Fig. 51: Fórmula del comportamiento de la etapa de entrada. Fuente propia

En la figura 51 se puede observar que la única diferencia que existe entre los términos que acompañan a las tensiones es que en un caso encontramos *R1,* y en el otro *R2*. Por lo tanto, para que el término que acompaña a *V_In1* sea 6 veces mayor al de *V_In2*, *R2* debe ser 6 veces mayor a *R1*. Además, que el término que acompaña a *V_In1* debe ser 1, ya que el offset debe ser de 2'5V y va a recibir una tensión de 2'5V. Así pues, se puede obtener la relación entre *R3* y *R4*, al igualar este término a 1.

$$\frac{R_2(R_3+R_4)}{R_3(R_1+R_2)}=1\Rightarrow\frac{6R_1(R_3+R_4)}{R_3\cdot 7R_1}=1$$

$$\frac{6(R_3+R_4)}{R_3\cdot 7}=1\Rightarrow 6R_3+6R_4=7R_3$$

$$6R_4=R_3$$

Fig. 52: Relación entre R3 y R4 para la etapa de entrada. Fuente propia

En conclusión, *R2* debe ser 6 veces mayor a *R1*; y *R3* debe ser 6 veces mayor a *R4*. Se opta por estandarizar los valores entre sí y con la etapa de entrada, por lo que *R2* y *R3* valdrán 120KΩ, mientras que *R1* y *R4* valdrán 20KΩ.

Calibración

Uno de los objetivos principales de este escudo consiste en lograr que este subsistema sea capaz de autocalibrarse sin intervención externa. Para conseguirlo deben calibrarse ambas etapas por separado y con al menos, tres puntos de calibración, uno de ellos en 0V (para minimizar el *offset*). Emplearemos una aproximación lineal por tres puntos, por lo que cuanto más separados estén los puntos de la calibración, menor error relativo se cometerá.

Una primera propuesta se basa en emplear dos interruptores para la calibración. Uno de ellos conectaría el DAC con la etapa de entrada. El otro haría lo propio con la etapa de entrada y de salida. En primer lugar se calibraría la etapa de entrada con los valores proporcionados por el convertidor y, posteriormente, se calibraría la etapa de salida al conectarla con la etapa de entrada, que ya se encontraría calibrada. El esquema quedaría como se muestra en la figura 53.

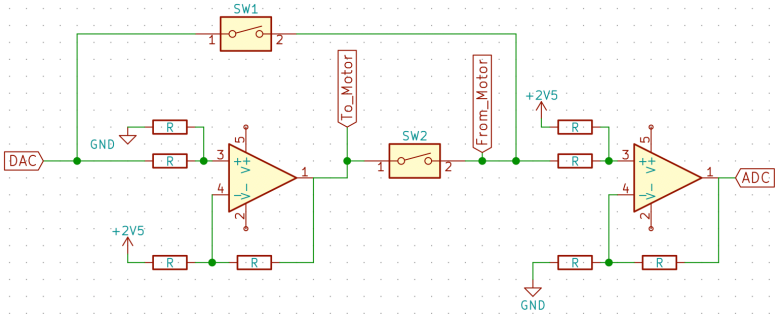


Fig. 53: Propuesta inicial para la autocalibración. Fuente propia

Esta calibración sería bastante sencilla de implementar con dos interruptores analógicos de ±15V, no se podrían emplear MOSFETs ya que se trata de interruptores flotantes. Tan solo habría que tener cuidado para lograr una protección adecuada del DAC, aunque no debería suponer un problema.

Sin embargo, existe un inconveniente en la calibración con este circuito: cuando se calibrase la etapa de entrada, solo podríamos emplear tensiones de 0V a 5V, esto es, el rango de tensiones del DAC. Esto provocaría que la calibración solo se realizara en un rango de la sexta parte del rango de trabajo, y pudiendo generar únicamente tensiones positivas, mientras que el rango de funcionamiento cuenta también con negativas. Debido a esto, se descarta este circuito y se opta por buscar otro método de calibración.

En vez de emplear una calibración en un rango continuo empleando el DAC, se decide calibrar la etapa de entrada mediante tres puntos. Estos tres puntos serán -10V, 0V y +10V. De esta forma se asegura una calibración de la etapa en un rango similar al de funcionamiento. Las tensiones se obtendrán a partir de diodos de referencia de tensión, y se llevarán al circuito mediante interruptores analógicos. Este método requiere de más componentes, pero se consigue una mejor calibración. Además, se mantiene el DAC aislado del circuito de ±15V, evitando que pueda sufrir daños por las tensiones elevadas de este subsistema. Una vez la etapa de entrada se encuentre correctamente calibrada, se procederá con la etapa de salida de la misma forma que en la primera propuesta: conectando ambas en serie.

Para este método de calibración se necesitan cuatro interruptores analógicos: tres para las tensiones de referencia y uno para conectar ambas etapas. Los problemas a los que nos enfrentamos son similares a los que ocurrían en el subsistema de los filtro RC/RLC. Debido a las tensiones que aparecerán en el circuito, que pueden ser superiores o inferiores a las tensiones de referencia, no podemos emplear MOSFETs. El uso de relés convencionales o SSR resulta excesivo tanto en coste económico como en espacio ocupado, por lo que decidimos implementar nuevamente interruptores analógicos. En esta ocasión deben permitir tensiones en el rango de ±15V, así que es necesario buscar un nuevo modelo.

Implementación - Circuito principal

El integrado central de este subsistema es el amplificador operacional. Se busca un integrado con dos amplificadores operacionales internos que permita tensiones de ±15V. Entre sus características interesa especialmente que tenga un elevado s*lew-rate*, para permitir rápidos cambios en la tensión de salida; y que no sea necesario realizar acondicionamiento. También resulta importante que el propio integrado sea pequeño y barato.

El modelo escogido es el *LM324D* de *Texas Instruments*; cuenta con cuatro operacionales internos de 0'5V/us de *Slew-Rate* por un precio muy económico. Dentro del circuito recibe la denominación de *U701*. Debido a la existencia de varios componentes dentro del mismo encapsulado, el amplificador de la etapa de salida será el *U701A* y el de la etapa de entrada *U701B*. Este integrado se alimenta de forma simétrica a ±15V mediante las tensiones de interfaz. Por lo tanto, necesita 4 condensadores de desacoplo: un tipo A a +15V (*C715*), un tipo B a +15V(*C701*), un tipo A a -15V (*C716*) y un tipo B a -15V (*C702*).

Previamente se ha calculado que cada amplificador requiere de dos resistencias de 20KΩ y dos de 120KΩ. Las resistencias del circuito de salida (*R1-R4*) reciben las denominaciones de *R702* a *R705*, manteniendo el orden; mientras que en el circuito de entrada reciben las denominaciones de *R707* a *R710*.

Las cuatro resistencias de 20KΩ, dos para cada amplificador, se implementarán con el modelo *MCR03EZPFX2002* de *ROHM*. Se trata de una serie que muy empleada en este proyecto. Sin embargo, no se encuentran resistencias de esta misma serie de 120KΩ, por lo que se va a montar el modelo *ERJ3RED1203V* de *Panasonic*. La tolerancia de esta es mejor; sin embargo, su precio también es superior.

Adicionalmente se coloca un condensador de filtrado de tipo *C* entre la entrada negativa de cada operacional y su salida. De esta forma se reduce el ruido de alta frecuencia y mejora el comportamiento del circuito.

Como protecciones, se disponen resistencias para evitar excesos de corriente y diodos Schottky para evitar sobretensiones. Las dos resistencias se colocan a la salida de los amplificadores operacionales. Ambas serán de 300Ω, de esta forma se podrá emplear el mismo modelo que *R201* o *R402-R405*. En el circuito de salida se denomina *R706*, y en el de entrada *R711*.

Los diodos Schottky se montan al inicio y al final del circuito. Su función es evitar que salgan tensiones negativas o superiores a 5V de este subsistema. Estas tensiones podrían provocarse tanto por un mal funcionamiento o por un contacto accidental durante la manipulación de la placa. Se emplea un diodo Schottky doble en cada una de las localizaciones, recibiendo las denominaciones de *D703* (inicio) y *D702* (final). El modelo es el mismo que se ha empleado con anterioridad en otros subsistemas, el *DB5S308K0R* de *Panasonic*.

Las dos etapas del subsistema, junto a sus protecciones, se muestran en la figura 54.

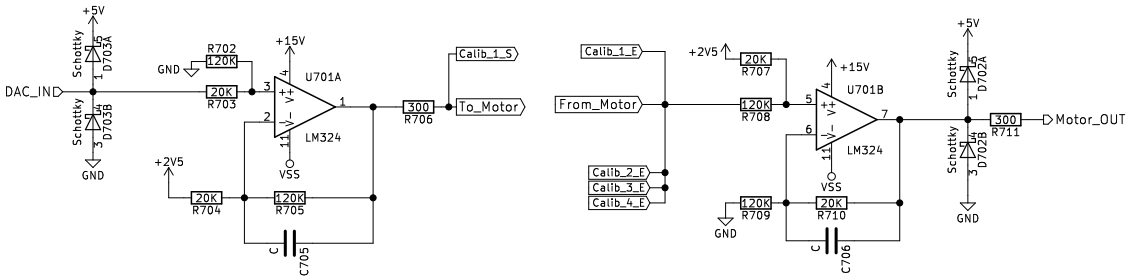


Fig. 54: Circuito principal de las etapas de salida (izq.) y entrada (der.). Fuente propia

Durante las prácticas, no se emplean conectores para realizar las conexiones físicas con el motor. Por lo tanto, no se van a montar conectores, en su lugar se realizarán agujeros plateados en la placa donde puedan anclarse pinzas de tipo cocodrilo. Aun así, en el circuito se trata como si fuera un conector normal, ya que necesitará una huella. Además, da más claridad al circuito al ver los hipotéticos conectores, aunque no se monte ningún componente. Disponemos un conector para la etapa de salida (*J701*), otro para la etapa de entrada (*J703*) y otro de masa, que actúa como referencia para ambas tensiones (*J702*).

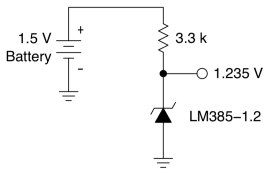
Implementación - Tensiones de referencia

Para el adecuado funcionamiento del circuito es necesario generar tres tensiones de referencia: una de 2'5V, una de +10V y una de -10V. Es importante emplear métodos que tengan una buena precisión, que operen correctamente en ese rango de tensiones y que no requieran de demasiados componentes de acondicionamiento; de esta forma se reduce el coste total y el espacio del conjunto.

Los tres voltajes van a ser generados mediante diodos de referencia de tensión. Dentro de esta tecnología, se emplean componentes que se encuentran ya calibrados a la tensión de salida que deseamos, de esta forma se reduce la cantidad de componentes externos necesarios y se mejora la precisión.

Para lograr la referencia de 2'5V se monta el *LM285D-2.5G* de *ON Semiconductor*, recibiendo la denominación *D701*. Ya está prefijado para trabajar a 2'5V y permite operar con corrientes entre 10uA y 20mA manteniendo una precisión del 1%. El circuito de aplicación típica de este componente se muestra en la hoja de datos técnicos. Al igual que en todos los diodos de referencia, resulta necesaria una resistencia de limitación de corriente, que se calcula de acuerdo a la corriente máxima que pueda requerir el circuito posterior. Se estima

un consumo máximo de un miliamperio, por lo tanto la resistencia serie, calculada en la figura 55, debe ser de aproximadamente de 2K5Ω.



$$\frac{V - V_{REF}}{I_{MÁX}} = \frac{5V - 2'5V}{1mA} = \frac{2V5}{1mA} = 2K5\Omega$$

Fig. 55: Aplicación típica y cálculo de la resistencia. Datos técnicos LM285D y Fuente propia

Se montan dos resistencias de 5K1Ω en paralelo para lograr una resistencia equivalente de 2K55Ω. Esto permite estandarizar valores con otras resistencias que se utilizan más adelante. Emplearemos el modelo *MCR03EZPFX5101* de *ROHM* para las dos resistencias: *R701* y *R718*.

Además, se emplean dos condensadores de 100nF para mejorar el comportamiento del sistema. Uno de ellos se coloca a la entrada del circuito (*C703*) y el otro a la salida (*C704*). Se monta el mismo modelo que el condensador *C210* del subsistema de alimentación.

Con el objetivo de proteger la tensión de referencia y asegurar un consumo inferior a 1mA, se va a implementar una etapa de salida mediante un amplificador operacional en modo seguidor no inversor. Esta decisión se toma debido el U701 cuenta con cuatro operacionales, de los cuales dos se encuentran desocupados. Además, no hace falta ningún componente de acondicionamiento para realizar este montaje. El circuito completo encargado de generar la tensión de 2'5V se muestra en la figura 56.

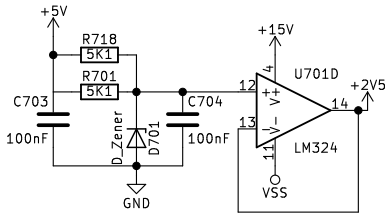


Fig. 56: Circuito generador de la referencia de 2'5V. Fuente propia

Para las tensiones de calibrado de +10V y -10V se van a a usar circuitos similares entre sí. La única diferencia entre ellos será el sentido del diodo, que se invierte para trabajar con tensiones negativas (en el caso de -10V). En ambos casos se emplea el integrado *LM4040-DIM3-10.0/NOPB* de *Texas Instruments*, y recibe las denominaciones de *D704* (+10V) y *D705* (-10V). Se trata de un integrado de referencia de tensión de precisión, fijo a 10V, con una tolerancia del 1%, y en un encapsulado muy reducido: SOT-23.

El circuito de aplicación es el mismo al utilizado en la referencia de 2'5V. También se calcula de forma idéntica la resistencia necesaria. Sin embargo, la caída de tensión en la resistencia en este caso es de 5V por lo que la resistencia debe duplicar su valor hasta los 5KΩ. Se emplea el mismo modelo que en el caso anterior, aunque ahora solo es necesaria una resistencia por cada integrado. Dentro del esquema las resistencias se denominan *R712* (+10V) y *R713* (-10V).

También son necesarios los condensadores de filtrado en las tensiones de referencia. Se mantiene su valor y, por lo tanto, el modelo escogido. Se colocan a la entrada y a la salida de ambos circuitos recibiendo denominaciones desde *C711* hasta *C714*.

En esta ocasión no dispondremos de un amplificador operacional en modo seguidor. Esto se debe a que tan solo existe un amplificador libre en el encapsulado del LM324. Como es-

tas tensiones solo se emplearán para realizar la calibración, y no continuamente como ocurría en el caso de los 2'5V, no se considera un problema el hecho de que no tengan este circuito de protección adicional.

Implementación - Calibración

Para activar y desactivar las diferentes etapas de la calibración se van a implementar interruptores analógicos, la misma tecnología que en el subsistema RC/RLC. La decisión se basa en los mismos criterios que en el caso anterior, ya que no es posible emplear MOSFETs debido a las variaciones de las tensiones del circuito.

En este circuito son necesarios interruptores analógicos que permitan controlar tensiones desde -15V hasta +15V. La resistencia serie no es especialmente relevante, ya que apenas afecta a los ya elevados valores de las resistencias de los amplificadores operacionales. Una característica que sí resulta decisiva de estos interruptores analógicos es la tensión de control, que debe ser a 5V; de ser mayor, sería necesario buscar un método de control distinto al expansor de E/S empleado en el circuito RC/RLC.

Con estas características se opta por el componente *DG411DY-T1-E3* fabricado por *Vishay*. Este integrado cuenta con cuatro interruptores analógicos internos y un pin para seleccionar el nivel de tensión de la entrada de control. Su resistencia serie es de 25Ω, se trata de un valor bastante elevado, pero no afectará al comportamiento del circuito debido a las mucho mayores resistencias de los circuitos de los amplificadores operacionales. El bloqueo funciona correctamente hasta los niveles de alimentación del integrado. Estos niveles son los mismos que alimentan los amplificadores operacionales, por lo que no debería generar ningún problema. Un aspecto, que será necesario tener en cuenta durante el desarrollo software, es que este integrado funciona con lógica negativa: una señal positiva en la pata de control deja el interruptor abierto, mientras que una señal negativa conecta la entrada con la salida.

Este integrado, al que se denomina *U703*, requiere de cuatro tensiones de alimentación distintas: alimentación positiva (+15V), alimentación negativa (-15V), alimentación lógica (+5V) y masa (0V). Por lo tanto, necesitará 6 condensadores de desacoplo; tres de ellos de tipo A (*C718*, *C719* y *C720*) y tres de tipo B (*C708*, *C709* y *C710*).

Al emplear una tensión lógica de 5V podemos recurrir al mismo integrado expansor que en el subsistema RC/RLC, el *PCA9554BS3,118*, para controlar los interruptores analógicos. Esta vez el expansor recibirá la denominación de *U702*. Se llevan sus pines de dirección a 5V para configurar su dirección dentro el bus I2C distinta a la del *U601*, por lo que su dirección será 0x27. Y lógicamente, lo acompañaremos de sus correspondientes condensadores de desacoplo de tipos A (*C717*) y B (*C707*). Ambos integrados se muestran, junto con sus conexiones y sus condensadores de desacoplo, en la figura 57.

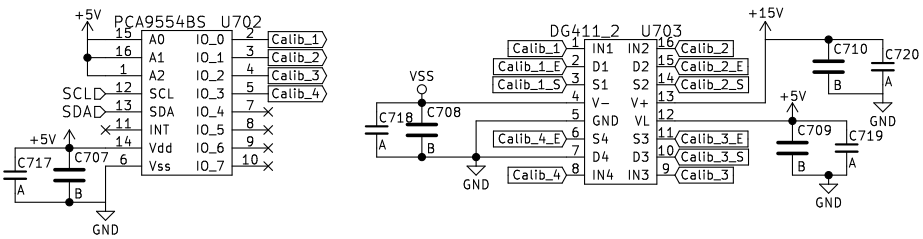


Fig. 57: Circuito de control de la calibración. Fuente propia

También se montan de indicadores luminosos para conocer cuándo se encuentra en funcionamiento cada etapa de la calibración. Para la implementación se meplan los diodos LED *KPH-1608SECK* de *Kingbright*. Son unos diodos LED de color naranja que ofrecen hasta 250mcd en un encapsulado (0603). Reciben las denominaciones *D706* a *D709*. Para contro-

lar la corriente que circula por ellos se montan en serie resistencias limitadoras de 470Ω con las denominaciones *R714* a *R717*. Para estandarizar se usa el mismo modelo que las resistencias ya montadas *R603* a *R610*. Debido a la lógica negativa del DG411, dispondremos los LEDs desde la señal hacia la tensión de alimentación.

5.9 Servo

El subsistema de control del servomotor y cargas externas no genera ningún problema en el escudo actualmente en uso. Por lo tanto, se ha decidido no implementar ninguna nueva función, así que solo vamos a realizar una actualización de componentes, sin alterar el comportamiento de los circuitos.

El objetivo principal de este subsistema es controlar un servomotor externo mediante una onda PWM de consigna y una etapa de salida. De forma adicional, permite actuar sobre cualquier carga externa mediante una onda PWM de potencia. En ambos casos también se realiza una medición del consumo en corriente de estas cargas para llevar a cabo un control en lazo cerrado.

Al emplear una onda PWM como alimentación de una carga externa, aparecen ruidos electromagnéticos que se pueden extender a otros circuitos o afecta a la precisión de las medidas de corriente. Por lo tanto, es fundamental asegurar un filtrado adecuado de este ruido.

Para realizar un filtrado adecuado en todas las frecuencias y evitar variaciones en la tensión, se ha decidido montar cuatro condensadores en paralelo en la alimentación de 5V: dos de tipo A (*C801* y *C802*), uno de tipo B (*C803*) y uno C (*C804*).

Salidas Servomotor y Potencia

Las dos salidas de este subsistema funcionan en paralelo. Esto se debe a que ambas están controladas por el mismo pin de Arduino, el pin 6, que es el único que permite generar una onda PWM en el modelo “UNO”.

Salida Servomotor

La salida pensada para el control del servomotor se podría llevar directamente al exterior, pero es necesario tener en cuenta que existen dos conexiones distintas de servomotor. Las dos necesitan de tres pines de conexión en línea: masa, alimentación y control. La diferencia es que, en ocasiones el pin central del conector es el de masa, y en ocasiones es el de alimentación. Para permitir que el escudo trabaje correctamente con todos los servomotores del mercado se opta por implementar los dos tipos de conexionado mediante dos conectores independientes: *J801* publica la alimentación en el pin central, y *J802* publica la masa. Ambos conectores emplean tres pines de conexionado *251-8092* de *RS Pro*. A fin de distinguirlos correctamente entre si, se realizan marcas en la serigrafía acerca del método de conexionado de cada uno.

Esta salida, al igual que todas las demás salidas de la placa, debe estar protegida frente a sobrecorrientes. Esto dificultará que una conexión incorrecta dañe los componentes del circuito. Se monta una resistencia de 300Ω, denominada *R802*, con este fin. El valor concreto no es un factor determinante, pero es necesario que evite corrientes excesivas, por lo que se ha seleccionado según el criterio de estandarizar componentes. Esto nos permite utilizar el mismo modelo de resistencia que habíamos montado anteriormente en las protecciones serie del bus SPI: la *MCR03EZPFX3000*.

Salida Potencia

La salida de potencia PWM requiere de un transistor (*Q801*) como etapa de salida, ya que el pin del Arduino UNO no es capaz de entregar más de 20mA. Se usa un MOSFET de

tipo N referenciado masa para realizar el control. En esta ocasión se puede emplear un transistor MOSFET, ya que va a trabajar referenciado a una tensión conocida y constante (0V), por lo que no generará ningún problema.

Este transistor debe permitir corrientes de más de 1A, tensiones de 5V y controlarse a menos de 3V3, por si se monta el escudo sobre un Arduino con esa tensión de funcionamiento. El modelo *DMG3420U-7* de *DiodesZetex* ofrece estas características a un precio muy razonable, en un encapsulado SOT-23 y con una resistencia serie típica de apenas 25mΩ, lo que minimiza las pérdidas y el calor generado.

Para proteger el transistor se coloca un diodo en antiparalelo, denominado *D801*. Este diodo se encuentra, por lo tanto, en paralelo con el diodo interno del MOSFET. Su función se evitar que el transistor pueda dañarse debido a tensiones inversas. Se decide implementar el mismo modelo de diodo Schottky que hemos usado en el resto de la placa (*D204*, *D601*…). En este caso desaprovechamos uno de los dos diodos internos, pero resulta más eficiente que buscar un diodo distinto solo para este circuito.

También se monta una resistencia de 0Ω en la conexión de base del transistor *Q801*, a la que denominaremos *R803*. Esto no modificará el comportamiento del circuito, pero permitirá desconectarlo o protegerlo más adelante si se considera necesario. Para esto solo sería necesario quitar la resistencia o cambiarla por otra.

El conector de esta salida, *J803*, será el mismo modelo que el del servomotor. La diferencia es que en este caso solo son necesarios dos pines. Con el objetivo de estandarizar, utilizaremos el mismo componente, aunque ahora se eliminará de forma manual el tercer pin durante el montaje. Para minimizar la posibilidad de conectar las cargas externas de forma incorrecta, se indica la polaridad del conector en la serigrafía.

Indicador LED

En paralelo con ambas salidas, se emplea un diodo LED a modo de indicador del estado de la onda PWM de control. Este diodo brillará con una intensidad proporcional al ciclo de trabajo de la onda. Por lo tanto, proporcionará información acerca de si se está generando alguna onda, y de cuál es aproximadamente su ciclo de trabajo.

Para la implementación de este LED se emplea el mismo modelo de diodo que en los indicadores del subsistema *OpAmp15*: el *KPH-1608SECK* de color naranja. Recibe la denominación *D801* y, al igual que en el caso anterior, se protege con una resistencia de 470Ω (*MCR03EZPFX4700*), esta vez denominada *R801*.

Realimentación de la corriente

Un requisito necesario para realizar un control en lazo cerrado, es conocer en todo momento la corriente que circula por el circuito. La corriente eléctrica se mide de forma indirecta, existiendo dos métodos principales para ello. Uno de ellos aprovecha el campo magnético generado por la corriente (utilizando un sensor Hall); y el otro, la caída de tensión que aparece en una resistencia de valor conocido. Emplear un sensor Hall es menos intrusivo y no genera pérdidas en el circuito, pero su precisión es limitada, su coste más elevado y ocupa un mayor espacio. En este circuito se considera más adecuado medir la caída de tensión que aparece en una resistencia de tipo s*hunt*, se trata de resistencias de muy buena precisión y un valor óhmico muy reducido.

Cuanto menor es el valor de la resistencia menos interfiere en el circuito, ya que la caída de tensión que provoca es menor. Sin embargo, cuanto menor este voltaje, mayor tendrá que ser la amplificación, siendo más susceptible a aumentar también los ruidos. Así pues se debe encontrar un compromiso entre la interferencia provocada en el circuito y la precisión. Si se diseña adecuadamente el circuito, con un buen ruteo, se pueden minimizar las interfe-

rencias externas captadas. Esto permite trabajar con caídas de tensión menores, y por lo tanto se minimiza la intrusión en el circuito.

La tensión de salida del amplificador será la caída de tensión de la resistencia (su valor resistivo multiplicado por la corriente que la atraviesa) multiplicada por la ganancia del amplificador. Para comenzar es necesario fijar el rango de lectura, es decir, cuál va ser la corriente máxima que se podrá leer. Esta corriente será aquella con la que el amplificador genere la tensión máxima admisible, en este caso 5V o 3V3. Decidimos arbitrariamente que nuestro rango de lectura sea de 0A a 1A. Este rango se puede modificar después fácilmente cambiando la resistencia o el amplificador.

$$V_{OUT} = I \cdot R \cdot G$$

$$5\text{V} = 1\text{A} \cdot R \cdot G$$

Fig. 58: Relación entre la corriente de la resistencia y la tensión de salida. Fuente propia

De acuerdo a la figura 58, el producto entre la resistencia s*hunt* y la ganancia del amplificador debe ser 5. En este punto es necesario fijar uno de estos dos parámetros; para ello se busca un amplificador pensado para esta función. Se va a emplear la serie NCS199A, que tiene una precisión del 1'5% y una salida Rail-to-Rail; esto es muy importante ya que se va a emplear una alimentación de 5V y se espera una salida de hasta 5V. Esta familia de integrados cuenta con tres ganancias fijas: 50V/V, 100V/V y 200V/V. La ventaja de emplear ganancias fijas es que se eliminan las imprecisiones que introducen las tolerancias de resistencias externas. Se opta por emplear el valor intermedio de ganancia (100V/V); en el caso de detectar demasiados ruidos en la salida se reducirá esta ganancia. Por lo tanto, se emplea el componente *NCS199A2SQT2G* de *ON Semiconductor* con la denominación *U801*. Una vez conocida la ganancia, se puede calcular el valor de la resistencia s*hunt* necesaria. Este cálculo se muestra en la figura 59.

$$R = \frac{5}{G \cdot 1\text{A}} = \frac{5}{100} = 50\text{m}\Omega$$

Fig. 59: Cálculo de la resistencia s*hunt*. Fuente propia

Los 50mΩ no son un valor estándar, esto es un problema, por lo que es necesario encontrar otro valor cercano adecuado y que se encuentre en el mercado. Si se emplea un valor superior, la caída de tensión será mayor, y por lo tanto también la salida del amplificador. Para no reducir el rango de lectura se debe usar un valor resistivo ligeramente más reducido. Se ha decidido implementar la resistencia s*hunt ERJL06KF47MV* de *Panasonic*; se trata de una resistencia de 47mΩ con una tolerancia del 1% en un encapsulado (0805). Esta resistencia, denominada *R805*, no irá conectada directamente a las entradas del amplificador. Para proteger el amplificador es recomendable disponer de protecciones entre ambos componentes.

Las primeras protecciones son dos resistencias (*R804* y *R806*) de 0Ω, una en cada conexión. Estas actúan a modo de fusible; en caso de detectar problemas se podría modificar su valor. También se monta un condensador (*C805*) de tipo C entre ambos pines para filtrar las componentes de alta frecuencia de la señal que llega al amplificador. Todos estos componentes (s*hunt*, protecciones y Amp.) deben montarse muy próximos entre sí y de forma simétrica para minimizar las posibles captaciones de ruidos por las pistas del circuito.

Los amplificadores de precisión son muy sensibles a las variaciones de la tensión de alimentación. Al igual que en el resto de integrados, se emplean dos condensadores de desacoplo de tipos A (*C806*) y B (*C807*); pero además, se implementa un núcleo de ferrita que

ayuda a reducir los ruidos de alta frecuencia de la alimentación (*L801*). Se trata del núcleo de ferrita empleado ya en otros subsistemas: el *742792609* de *Wurth Elektronik*.

El amplificador cuenta también con un pin de referencia que sirve para modificar el nivel de la salida. Se conecta a masa para que la salida no tenga ningún offset. Como último componente del subsistema se coloca una resistencia *pull-down* de 47KΩ en la salida del amplificador. Esto se hereda de la anterior versión de la *UMA_AEB_V1.1.0*, y sirve para reducir los ruidos en la línea que conecta la salida del amplificador con el ADC.

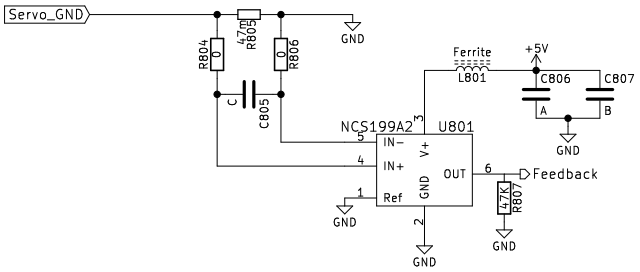


Fig. 60: Circuito de lectura de corriente. Fuente propia

5.10 ADC

Un conversor analógico-digital es un integrado que captura el voltaje de una señal analógica y codifica ese valor de forma digital. En este escudo, y como ya se ha nombrado en apartados anteriores, el ADC debe capturar y procesar tres señales analógicas en el rango de 0V a 5V. La primera señal proviene del subsistema RC/RLC, la segunda del OpAmp15 y la tercera de la realimentación del servomotor. Dentro del esquema este integrado recibe la denominación *U901*.

Las señales que se pretenden procesar serán de una frecuencia máxima de 10KHz, tal y como se vio en el subsistema del DAC. Según el teorema de muestreo de Nyquist, si se desea digitalizar una señal para poder reconstruirla de forma fiable posteriormente, la velocidad de muestreo debe ser, al menos, diez veces superior a la frecuencia de la señal original. Por lo tanto, se debe emplear un convertidor que permita muestrear a una velocidad de 100KHz o más.

Además, queremos mantener una precisión mínima en las lecturas de 5mV. Esto, para una señal de 5V, requiere de 1000 pasos. Esta cantidad de pasos implica una resolución de 9'96 bits (se calcula igual que en el DAC). Por lo tanto, buscamos un conversor de 10bits, que tendrá 1024 pasos, y en consecuencia una resolución real de 4'88mV.

Se realiza una búsqueda con todos estos parámetros, teniendo en cuenta que el integrado debe permitir una comunicación I2C y la importancia de minimizar el coste económico y el espacio ocupado. El modelo que mejor se amolda a estas necesidades es el *AD7995YRJZ-1500RL7* de *Analog Devices*. Cuenta con cuatro entradas independientes, 10 bits de resolución y permite muestrear a velocidades de hasta 140ksps.

No cuenta con buffers internos para guardar los datos, por lo que la velocidad del bus I2C es muy importante para evitar la pérdida de datos. Su comunicación I2C permite las tres velocidades estándares (100KHz, 400KHz y 3'4MHz), es decir, no supondrá ningún problema. No dispone de pines de configuración de la dirección, pero esta no coincide con ninguna empleada en otros componentes de la placa.

Se decide utilizar una entrada para cada una de las señales que se van a muestrear. Esto supone una ventaja respecto a la versión anterior de la placa, ya que aísla los subsistemas entre sí, evitando la propagación de problemas.

La cuarta entrada de lectura del convertidor también puede actuar como señal de referencia. Se decide llevar a dicho pin una señal conocida estable para poder realizar una calibración del convertidor, ya que su salida depende de la tensión de alimentación. En este caso se conecta ese cuarto pin a la tensión de 3V3 del microcontrolador. Al igual que en el resto de integrados, se va a filtrar la tensión de alimentación con dos condensadores de desacoplo, uno de tipo A (*C901*) y uno de tipo B (*C902*).

Con esto queda finalizado el diseño de los esquemáticos de la placa. Los esquemas completos se muestran en el anexo de esquemas electrónicos.

6. Diseño de la PCB

Una vez los esquemas electrónicos están definidos, se procede a realizar el diseño de la PCB. El objetivo principal del diseño debe ser minimizar las posibles fuentes de interferencias y asegurar el comportamiento óptimo de los subsistemas. También conviene realizar un diseño que simplifique las posteriores tareas de montaje y verificación, aunque debe primar el correcto comportamiento electrónico del sistema.

Debido a la gran cantidad de componentes y el reducido espacio disponible, se ha tomado la decisión de realizar un diseño de cuatro capas con componentes en ambas caras. Esto aumentará el coste de fabricación, pero reducirá enormemente la complejidad del trazado de las pistas y mejorará la compatibilidad electromagnética de los circuitos.

En primer lugar se van a presentan las restricciones de diseño, estas se deben cumplir obligatoriamente durante el diseño. Posteriormente, y de acuerdo a las restricciones, se toman las decisiones de diseño, que es recomendable. aunque no obligatorio, cumplir. Finalmente se muestra la disposición de componentes definitiva y su ruteo.

6.1 Restricciones de diseño

Existe una restricción importante de diseño debido a la construcción física de las placas Arduino. Los conectores, el Jack de carga y el USB, tienen una altura considerable y pueden colisionar con los componentes del escudo. El conector Jack está fabricado en plástico, por lo que el único problema es que choque con otro componente de la placa y no permita una conexión adecuada con el Arduino. En el caso del conector USB, el problema puede ser mayor, ya que su parte exterior es metálica. Este conector puede provocar cortocircuitos accidentales entre componentes en el escudo si se disponen en la misma zona que ocupará el conector. También se debe evitar disponer componentes en la zona que ocupa el ATmega en el Arduino UNO, ya que en algunas versiones emplea un encapsulado DIP de altura notable.

Otra restricción de diseño esta relacionada con la operación de la placa; el alumno, o cualquier otro usuario, debe poder operar cómodamente los conectores, los botones y el potenciómetro de la placa. Para ello, estos deben estar ubicados en los bordes de la placa y siempre en la parte superior. Los elementos situados en la parte central de la placa son más incomodos de manejar, y además quedan ocultos si se dispone otro escudo encima.

También resulta conveniente, aunque no es una restricción inviolable del diseño, que la placa tenga una orientación normal de funcionamiento. En esta orientación el alumno debe de ser capaz de trabajar con la placa de forma correcta y cómoda, pudiendo operar todos los subsistemas sin necesidad de cambiar su orientación. En este proyecto se ha decidido que la placa se empleará de forma horizontal, con el conector USB del Arduino hacia la izquierda y los pines de alimentación hacia el alumno.

6.2 Decisiones de diseño

Para asegurar el comportamiento óptimo del sistema se toman una serie de decisiones de diseño. Estas decisiones no son restricciones y podrán alterarse en los casos en los que se considere oportuno.

Al emplear cuatro capas, se tiene la posibilidad de realizar un plano de masa completo en una de ellas. Al hacer que toda una capa, a excepción de las vías, esté conectada a la referencia de masa, se reducen enormemente las impedancias y los lazos de todas las pistas de la placa. Esto se potencia aún más si se llevan las líneas de mayor frecuencia por la cara superior y se emplea la segunda capa como plano de masa. Ambas capas se encuentran muy próximas entre sí, por lo que los lazos que captan (o provocan) interferencias tienen un área mínima, siendo esta proporcional a la captación y emisión de EMIs.

De forma similar al plano de masa general, también vamos a se va a emplear de un plano de masa local en cada uno de los integrados de la placa. Este estará en la misma capa de cobre donde se monte el integrado y cumplirá dos funciones. Al igual que el plano de masa general, este plano reducirá el área de los lazos por los que circule la corriente, por lo que se minimizarán las interferencias. Además, supone que no se pueden trazar pistas por debajo de los integrados. Una línea por debajo de un integrado puede afectarle a causa del campo electromagnético que genera la corriente que la atraviesa, o debido al ruido que emite, especialmente las líneas de alta frecuencia.

Las líneas sensibles que pueden generar más problemas, las de alta frecuencia y las analógicas, se intentarán disponer por la capa superior. Esto facilita la detección de problemas, a la vez que minimiza el lazo con el plano de masa.

Otro aspecto a tener en cuenta relativo a las pistas es la aparición de condensadores parásitos generados por los cruces entre estas a distintas alturas. Estos van a aparecer cada vez que dos pistas se crucen a distintos niveles, por lo que no se pueden evitar. Esto puede afectar negativamente al comportamiento de los circuitos, así que se debe minimizar su aparición. Para ello hay que evitar que dos pistas discurran una encima de la otra durante una longitud considerable, ya que el aumento de superficie incrementa la capacidad del condensador.

También resulta muy recomendable que los subsistemas estén agrupados entre sí de forma compacta. De esta forma, se reduce la longitud de muchas pistas, evitando que actúen como antena ante interferencias externas. Esto también ayuda a la identificación de problemas. Al colocar componentes en ambas caras de la placa, se pueden disponer los componentes de acondicionamiento justo debajo (o encima) del integrado, minimizando la distancia entre ambos. Esto se puede observar en la figura 61, que muestra el integrado *U701*, en la capa superior (en rojo y azul), y todos los componentes pasivos que se relacionan con él, en la cara opuesta (en verde y morado).

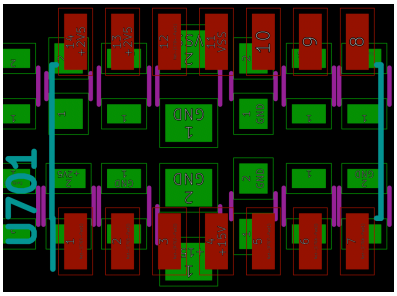


Fig. 61: *U701*, en rojo y azul, y componentes pasivos, en verde y morado. Fuente propia

La proximidad entre componentes de un mismo subsistema resulta especialmente importante en el caso de los condensadores de desacoplo. La función de estos condensadores es compensar la inductancia de las pistas y filtrar la alimentación. Por lo tanto, deben estar al final de las pistas de alimentación de los componentes. Las pistas estos condensadores deben ser lo más cortas posible, minimizando la inductancia serie que aparece con el condensador.

Se procura que los componentes se dispongan, en la medida de lo posible, en la misma orientación y centrados. Esto ayuda a las labores de montaje y comprobación. A fin de facilitar las labores de verificación también se colocan puntos de comprobación en distintas señales de la placa. Para poder realizar verificaciones de los subsistemas con osciloscopio se añaden puntos de medida en la señal de salida del DAC y en las tres señales de entrada del ADC. También se emplean puntos de verificación de las tensiones de calibración de los amplificadores (-10V, 2’5V y 10V). Estos puntos no deben suponer una derivación de la pista, sino que deben encontrarse en el curso de la misma. De lo contrario podría actuar como an-

tena receptora o emisora de EMIs. En caso necesario se modificará el recorrido de la pista en la medida necesaria. Cuanto mayor sea la derivación de la pista, mayores serán las interferencias que generará o recibirá. La colocación correcta de los puntos de verificación se muestra en la figura 62.

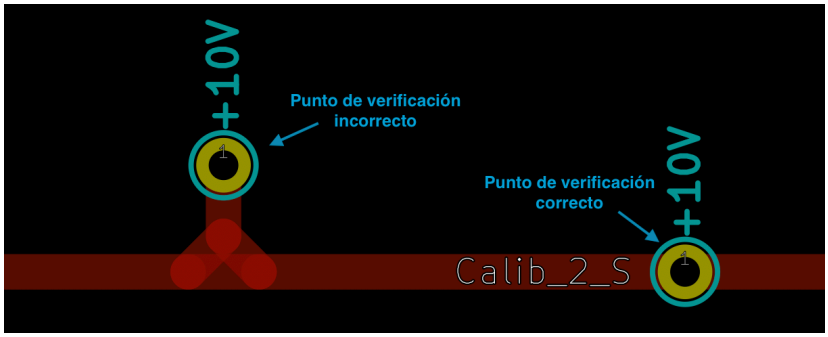


Fig. 62: Representación de como colocar adecuadamente un *Test-Point*. Fuente propia

Finalmente, es necesario que prestar especial atención a la serigrafía. La función principal de esta consiste en identificar correctamente todos los componentes. Pero además, la serigrafía debe servir para distinguir cada indicador, pulsador, conector o punto de comprobación. En el caso de los indicadores LED que indican la configuración del filtro RC/RLC y la calibración del subsistema Amp.Op.15, la serigrafía debe servir para conocer qué componentes se encuentran activos en el filtro, o con qué tensión se está calibrando el amplificador operacional.

6.3 Disposición de subsistemas

Ya se ha decidido anteriormente que la orientación de uso de la placa sea en horizontal. De acuerdo a esta decisión se van a disponer los componentes de tal forma que permitan trabajar de forma cómoda, ya sea en la verificación de la placa o en el transcurso de las prácticas. Para distribuir los subsistemas en la placa hay que diferenciar entre los que requieren de una conexión al exterior, los que dependen de algún pin concreto de Arduino y los que apenas tienen restricciones.

A continuación, se decide la disposición de cada uno de los subsistemas de acuerdo a sus restricciones y sus dependencias. Es importante que los subsistemas se encuentren organizados de la forma más compacta posible y minimizando la longitud de las pistas.

El subsistema de alimentación se dispone en la mitad inferior de la parte central de la placa. Esto se realiza por dos motivos: en primer lugar es ahí donde se encuentran los pines de alimentación de Arduino de donde se obtendrá la energía; y en segundo, se reducen las distancias máximas de las pistas de alimentación. Las pistas más cortas tienen una inductancia menor, lo que favorece el correcto filtrado de los condensadores de desacoplo.

Los subsistemas con conectores, botones o potenciómetros, deben estar situados en los bordes de la placa. Concretamente, los os subsistemas tienen estas restricciones son Servo, OpAmp15 y UserIO. Como se puede ver en la figura 63, los dos subsistemas con conectores al exterior (Servo y OpAmp15) se disponen en el lateral izquierdo. Los elementos del subsistema *UserIO* que permiten la entrada de información (botones y potenciómetro) deben ser accesibles por el alumno. Por lo tanto, se colocan en el borde derecho, lo que permite una manipulación cómoda.

Los elementos de salida de información del modulo UserIO, los LEDs, se disponen adyacentes a los botones, aunque hacia el interior de la placa. Estos componentes no tienen restricciones de posición, pero resulta interesante que se mantenga la cercanía entre los componentes del mismo subsistema.

En el subsistema OpAmp15 se pueden diferenciar dos partes. La parte principal, compuesta por los amplificadores operacionales y los conectores, se encuentra muy próxima al borde de la placa. La parte encargada de generar las tensiones de referencia se encuentra entre la parte principal y el subsistema de alimentación. Estos componentes no tienen salida directa al exterior y, además, actúan eléctricamente entre la alimentación de ±15V y los amplificadores operacionales. Por lo tanto, se sitúan físicamente entre ambos. Muy próximos a estos componentes se colocan los puntos de verificación con el objetivo de minimizar la longitud de las pistas.

El resto de subsistemas se acomodan en los huecos restantes de la placa. Para su colocación se deben tener en cuenta las dependencias entre subsistemas, los pines de Arduino que afectan a cada uno y los buses de comunicación.

El subsistema de los filtros RC/RLC ocupa una gran parte de la placa, ya que requiere de un gran número de componentes para poder realizar las reconfiguraciones. Se decide colocar este subsistema en la parte central superior de la placa. Conviene dejar espacios entre los componentes ya que la cantidad de pistas que habrá que trazar es muy elevada.

El ADC y el DAC se colocan próximos entre sí, puesto que se trata de dos integrados de tipo similar y que emplean el mismo bus de comunicaciones. De esta forma los puntos de comprobación que corresponden a sus entradas o salidas se encuentran también próximos entre sí, facilitando las labores de verificación.

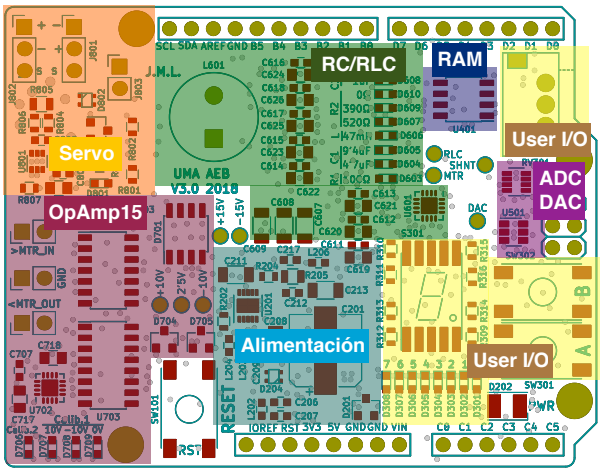


Fig. 63: Distribución de subsistemas en la placa. Fuente propia

Con esta distribución final de componentes, que aparece en la figura 63, se logra que todos los subsistemas se encuentren compactados entre sí, a excepción del UserIO. El potenciómetro se dispone separado del resto de componentes de la interfaz, pero es la única forma de mantenerlo en el borde de la placa. Se consigue que los conectores, los botones y el potenciómetro se sitúen en los bordes de la placa, facilitando su uso; además, ninguno de estos componentes se solapa con los conectores de la placa del Arduino Uno (USB, Jack e ICSP).

Todos los laterales de la placa se encuentran ocupados por este tipo de componentes excepto el extremo izquierdo del lado inferior. En este caso, se trata de un tramo bastante corto que no se podía usar para ningún conector debido a la proximidad con el conector Jack de la placa Arduino. Se decide aprovechar este espacio se aprovecha para colocar los indicadores LED de la calibración del subsistema OpAmp15*;* mientras que los conectores se disponen entre los conectores Jack y USB del Arduino.

En la parte superior del lateral izquierdo se dispondrán los dos conectores del servomotor y el conector para la salida PWM. Estos tres conectores deben colocarse de forma que sea

imposible conectar una carga externa entre dos de ellos por accidente. El conector PWM esta desplazado hacia el interior de la placa para separarlo físicamente de los agujeros del Arduino, que están pensados para anclar los escudos mediante tornillería.

Los indicadores LED discretos de la interfaz se localizan justo debajo del display. Estos diodos se ordenan con el correspondiente al bit menos significativo al lado derecho y el más significativo al lado izquierdo, de forma similar a la ordenación de los bits en un registro de un byte.

Los elementos que se emplearán de forma habitual durante el uso de la placa se resaltan en la figura 64. Se ha procurado que ningún componente de altura considerable, como el condensador *C201* o la bobina *L601*, esté colocado delante de otro al que pueda molestar, como un botón o un LED.

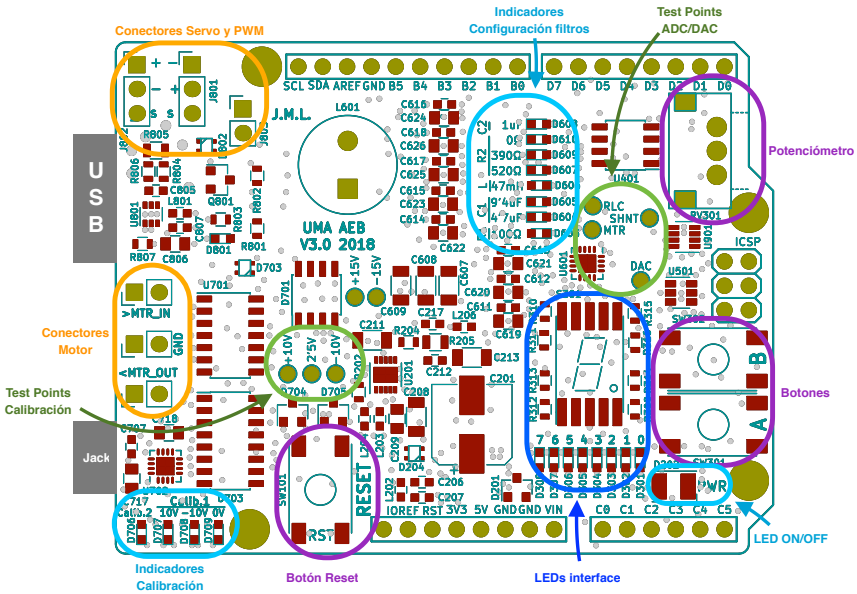


Fig. 64: Elementos importantes en el uso de la placa. Fuente propia

6.4 Trazado de pistas

Como se ha comentado anteriormente, las pistas de la capa superior se procuran trazar en horizontal y las pistas de la capa inferior en vertical. Esto minimiza la capacidad de los condensadores parásitos generados en los cruces de pistas a distintas alturas.

La anchura mínima permitida para el trazado de pistas es de 0’3mm. Las pistas más de dimensiones menores generan mayor inductancia serie y, además, aumenta el coste de fabricación. El tamaño normal de las pistas será de 0’4mm para las pistas de señales y de 0’6mm como mínimo para las pistas de alimentación. Dos ejemplos del trazado general de la placa se muestran en la figura 65.

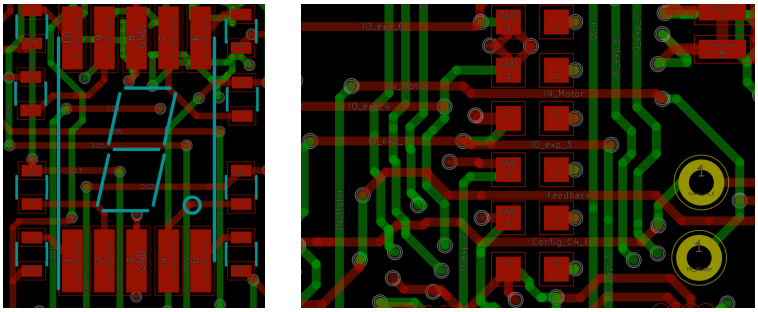


Fig. 65: Ejemplos de trazado de pistas en la placa. Fuente propia

En el caso de los buses de comunicaciones como I2C y SPI resulta importante que todas las pistas tengan la misma impedancia. Para lograr esto es recomendable emplear el mismo numero de vías en las distintas pistas. Además, si las pistas son de gran longitud, conviene que discurran de forma paralela, esto será importante en el bus I2C que se emplea en varios integrados de la placa.

En las pistas de alimentación hay que prestar especial atención a aquellas que entregan la energía a la carga PWM y al servomotor. Estas cargas externas pueden consumir una gran cantidad de corriente, pudiendo superar el amperio. Se decide que las pistas sean capaces de soportar consumos de hasta dos amperios.

Se puede emplear el módulo de KiCad *pcbCalculator*, que ayuda a realizar cálculos importantes de cara al diseño de las PCBs, en esta ocasión se accede la pestaña de "Ancho de pista". Introduciendo los datos de corriente, resistividad del conducto, aumento de temperatura permitido y grosor del cobre, se obtiene la anchura que debe tener la pista de cobre, ya sea por una capa externa o interna. Todo ello se muestra en la figura 66.

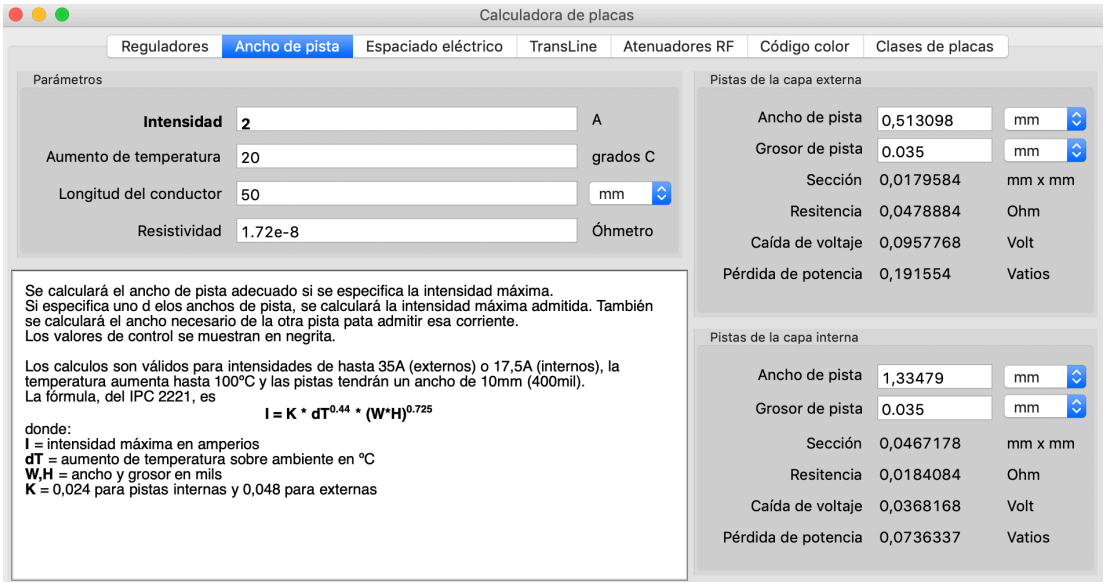


Fig. 66: Calculadora de anchura de pistas de KiCad. Fuente propia

De acuerdo a los resultados anteriores, las pistas internas de alimentación que conectan estos componentes deben ser de más de 1'4mm. Se procura realizar pistas de 2mm o más para contar con un margen de seguridad y reducir el aumento de temperatura producido. Realizar pistas de esta anchura es complicado, debido a la gran cantidad de componentes y vías ya existentes. Por lo tanto, se emplea una red de alimentación donde no existe un único camino de alcanzar cada punto. Esto aumenta la sección equivalente de las pistas a al vez que facilita las tareas de diseño de pistas.

Todas las pistas de alimentación se realizan por su propia capa. Cabe destacar que esta capa también se emplea para el trazado puntual de pistas de control, siempre que esto simplifique el diseño y no afecte al trazado de las pistas de alimentación.

7. Fabricación, montaje y verificación

7.1 Fabricación

Una vez el diseño electrónico se considera en un estado finalizado, se fabrica un primer prototipo. Gracias a las empresas actuales de prototipado rápido, se pueden obtener productos con una calidad profesional a un reducido coste y en un corto espacio de tiempo.

Durante todo el proceso de diseño de la placa resulta de vital importancia no perder de vista las capacidades de fabricación de la empresa de prototipado. El tamaño de pistas, su espaciado y el tamaño de vías debe mantenerse por encima de unos valores determinados o el coste de fabricación aumentará. En el caso de la fabricación de placas de cuatro capas puede resultar útil el uso de vías ciegas o enterradas; sin embargo, esto aumenta el coste en un orden de magnitud, por lo que no es rentable a no ser que resulte necesario.

Para la fabricación es necesario enviar a la empresa los archivos GERBER. Se trata de un formato estandarizado, y KiCad los genera de fácilmente mediante el menú "*Plot Gerbers*". Aquí se puede escoger de que capas generar los archivos y también algunas opciones adicionales, tales como si se desean cubrir las vías o la precisión de la exportación.

Se ha escogido para la fabricación a la empresa de prototipado rápido *ALLPCB*. En su página web hay que seleccionar la cantidad de placas que se desean fabricar, así como algunos parámetros de la fabricación. Para conocer el coste del prototipado hay que indicar el numero de capas, el tamaño de la pcb, el espaciado mínimo entre pistas y el tamaño mínimo de taladro. Además, se puede escoger el grosor de la pcb, la profundidad de cobre en las pistas, el tipo de acabado y los colores de la máscara de soldadura y la serigrafía. Todas estas opciones se muestran en la figura 67.

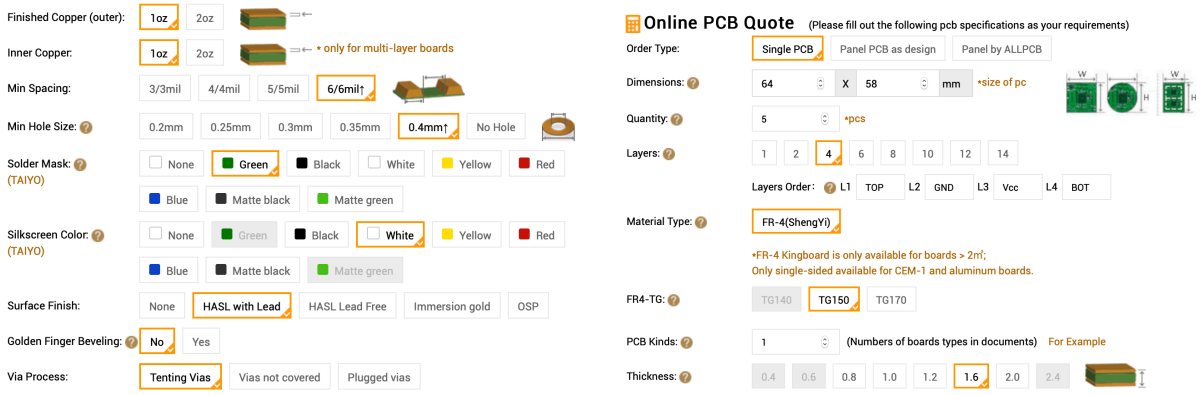


Fig. 67: Configuración del prototipado. AllPCB.com

Tras introducir las características del proyecto hay que incluir los archivos GERBER para su comprobación antes de la fabricación. Si no hay ningún problema se fabrica la placa y se envía por mensajería en un plazo normalmente inferior a una semana, aunque puede variar en función de las características del prototipo, las fechas concretas y la empresa.

Al recibir el producto final se observa su buena calidad y se comienzan las pruebas para saber si el prototipo es adecuado. Estos ensayos consisten en montar el escudo y realizar pruebas simples para cada subsistema, comprobando su correcto funcionamiento.

7.2 Montaje

El método más cómodo de montaje consiste en comenzar por los componentes más pequeños, los integrados de montaje superficial (SMD) de paso fino o sin patillas (QFN), seguir por el resto de componentes SMD y acabar por los componentes de montaje pasante, de-

jando para el final los de mayor altura. Este método evita que sea necesario soldar componentes pequeños al lado de componentes grandes que dificulten el acceso del soldador. Una vez se encuentran soldados todos los componentes, se procede a realizar las pruebas.

En el prototipo se encuentra un problema en la primera prueba de alimentación: el LED *D202*, que indica la presencia de los 5V de la tensión de alimentación, no se ilumina. Esto solo puede apuntar a dos errores: que el LED se encuentra mal montado (aspecto fácilmente comprobable) o que existe algún problema en la etapa de alimentación. Se comprueba que existe un error en la etapa de alimentación al no detectar tensión con el voltímetro. Finalmente se acota el fallo: existe un cortocircuito entre la red de alimentación +5V y el plano de masa. Se comprueba que no se trata de un caso aislado y que el error persiste en todas las placas que se han fabricado.

La fabricación de placas de cuatro capas dificulta encontrar errores de este tipo cuando se producen en las capas internas. Para encontrar la fuente del problema resulta necesario intercambiar un mensajes con los fabricantes de la placa y realizar varias revisiones y modificaciones sobre los escudos y los diseños de KiCad. Según el servicio técnico del fabricante, los archivos GERBER enviados contienen dicho cortocircuito, pero este no aparece en los análisis de KiCad.

Finalmente se encuentra el origen del problema al realizar la revisión de los archivos GERBER generados por KiCad. Se observa que algunas vías se encuentran ligeramente desviadas respecto a su pista correspondiente. En algunos casos estas desviaciones llegan a generar cortocircuitos entre diversas redes, no solo entre alimentación y masa. Este problema proviene de la precisión en la exportación, que es de 0'1mm. En algunos puntos se trabajó con una cuadricula inferior a esta precisión para facilitar el trazado de todas las pistas, por lo que al realizar la exportación se modificaron las posiciones originales, ajustando a una cuadricula de 0'1mm.

Se decide modificar la precisión de la exportación en las opciones de la misma, fijándola en 0'02mm. Tras esto se generan y revisan los nuevos archivos GERBER en busca de desviaciones. Se comprueba que en esta ocasión todas las vías se encuentran correctamente centradas en sus pistas. Al enviar estos nuevos archivos a los fabricantes seo tiene la confirmación de que los cortocircuitos han desaparecido. Por lo tanto, se realiza un nuevo pedido con las placas arregladas.

Con las nuevas placas se comienza otra vez la soldadura de los componentes y se comprueba el funcionamiento de todos los subsistemas. En esta ocasión se verifica que la alimentación se comporta de forma correcta y no existe ningún cortocircuito.

La siguiente comprobación que se realiza consiste en asegurar que las tensiones de *+15V*, *-15V*, *+10V*, *-10V* y *+2'5V* son correctas. Estas medidas se toman de forma rápida gracias a los puntos de verificación distribuidos por el escudo.

Durante las pruebas de verificación del resto de subsistemas se localizan algunos errores en la placa. Estos ya se encuentran corregidos en la revisión del diseño mostrada anteriormente. Sin embargo, en el prototipo empleado para la verificación y el desarrollo software, son necesarias algunas modificaciones manuales para que todos los subsistemas funcionen de forma adecuada.

Entre estos errores destaca un problema con ambos expansores de puertos (*U601* y *U702*). La huella diseñada originalmente era más grande que el componente, por lo que no se podía realizar una soldadura correcta. Se ofrecen diferentes encapsulados para este componente y se realizó la compra del modelo de mayor tamaño, mientras que la huella era la del modelo de menor tamaño. Existían dos posibles soluciones: modificar la huella (requiriendo una alteración del diseño y una nueva fabricación de las placas) o comprar el integrado con el encapsulado adecuado para el montaje. Se opta por esta última.

Otro problema similar ocurre con el buffer del subsistema RC/RLC: *U602*. En esta ocasión existía un error en el conexionado de los pines. La conexión de los pines de alimentación era correcta, de forma que el integrado no sufrió daños. El pin positivo de entrada y el de salida se encontraban intercambiados, por lo que fue necesaria una modificación física de la placa para solucionarlo a fin de que el circuito funcionara correctamente.

También se detecta un problema en el circuito de calibración del subsistema *OpAmp15*. Originalmente se había aprovechado el último amplificador operacional para proteger las señales de calibración de *+10V* y *-10V*; sin embargo, esto generaba problemas (al tratar de proteger dos señales con un único operacional), por lo que se ha eliminado en los esquemas finales. También se ha modificado físicamente la placa para evitar este problema.

El último problema de la placa esta relacionado con el display de siete segmentos. Debido a una falta de stock del modelo deseado se optó por la compra de otro modelo similar de la misma serie, donde aparentemente la única diferencia era el color de los LEDs del display. Sin embargo, este modelo suministrado es de cátodo común, mientras que el modelo del diseño es de ánodo común. Por lo tanto, en el primer prototipo el display permanece inoperable, pero tan solo es necesario montar el modelo adecuado, o cualquier modelo de ánodo común.

El resultado final de las placas montadas con todos los componentes se muestra en las imágenes de la figura 68.

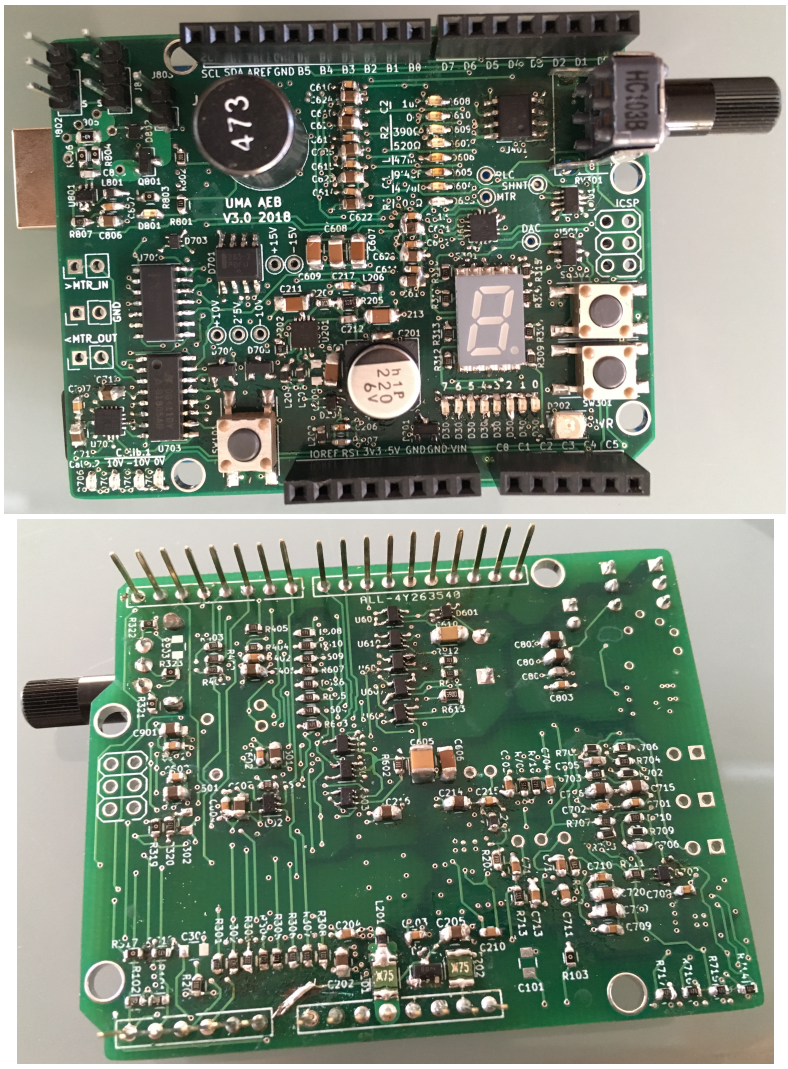


Fig. 68: Prototipo sobre el que se realizarán las pruebas. Fuente propia

8. Diseño Software

Una vez el escudo esta fabricado y es funcional, hay que desarrollar el software. En primer lugar, resultan necesarios programas de pruebas y verificación para comprobar el funcionamiento de los todos los circuitos. En segundo lugar, se desarrollan programas de ejemplo para que los alumnos tengan una referencia a la hora de conocer el comportamiento del escudo.

8.1 Software de verificación y pruebas

Las funciones de prueba sirven para asegurar el correcto funcionamiento de los diferentes subsistemas. Estas pruebas no están pensadas para ser empleadas por los alumnos, tan solo como verificación. Por lo tanto, se decide implementarlas como una clase y no como programas de ejemplo. Dentro de esta clase se encontrarán todos los métodos y atributos necesarios para las diferentes pruebas. También se incluyen métodos más complejos que permiten determinar el alcance las posibilidades del escudo.

Todas las funciones de verificación se incluyen en un único archivo de verificación general denominado *Full_Test.ino*. Para realizar las pruebas tan solo será necesario ir descomentando los trozos de código correspondientes a cada una de las pruebas.

Interfaz

El primer subsistema que se verifica es el subsistema de la interfaz con el usuario. Este consta de dos botones, un potenciómetro, siete LEDs y un display de siete segmentos. Son necesarios dos métodos: uno de configuración (necesario para el correcto funcionamiento de los circuitos) y otro de verificaciones.

Se comienza con la implementación del método que configura los botones y el potenciómetro como entrada y los LEDs como salida. Este método, denominado *configIO*, es independiente del propio método de verificación, ya que se llamará siempre que se vaya a emplear algún elemento de la interfaz.

Se etiquetan los pines que controlan la interfaz en la librería mediante *#define,* ya que son constantes invariables de la placa. Esto, además, permite emplearlos con tan solo cargar la librería, sin necesidad de realizar una instancia de la clase.

Se ha decidido que la forma más rápida de probar este subsistema consiste en relacionar todos los componentes mediante un único algoritmo, al que se accede con el método *testIO*. El funcionamiento del mismo es el siguiente: se lee el valor del potenciómetro; se acota entre 0 y 7, representando un LED; si se pulsa el botón ‘A’, se enciende el LED correspondiente y si se pulsa el botón ‘B’, se apaga dicho LED.

Al realizar un barrido, primero con un botón pulsado, y después con el otro, se puede comprobar rápidamente si algún elemento no funciona correctamente. De forma adicional, se añade la opción de enviar por el bus serie todos los datos manejados por el microcontrolador. Esto permite acotar rápidamente el componente que estaría provocando un hipotético error. Esta opción se activa al realizar el paso de un parámetro distinto de cero al método. En caso de no indicarse nada en la llamada, no se volcará ninguna información al bus serie.

DAC

El conversor *MCP4725* tiene varios modos distintos de funcionamiento, así como varias funciones adicionales. El objetivo es desarrollar métodos que permitan probar todo de forma rápida y eficiente. Hay que tener en cuenta que, para permitir la comunicación con el conversor, es necesario activar el bus I2C en el Arduino mediante la orden *Wire.begin()*. Si se desea modificar la velocidad del bus, se emplea *Wire.setClock(x)*; siendo *x* la velocidad en bits/s deseada. Arduino solo admite las velocidades de 100KHZ y 400KHz.

Para establecer la comunicación con el DAC hay que conocer su dirección dentro del bus I2C. Como ya se indicó anteriormente, se ha conectado su pin de selección (*A0*) de dirección a masa, lo que genera la dirección 1100000 en binario, 0x62 en hexadecimal. Este valor se incluye como atributo constante dentro de la clase.

El modo de operación lo definen dos conjuntos de bits: los PDx y los Cx. Los bits PD0 y PD1 sirven para colocar el integrado en modo bajo consumo y seleccionar una resistencia de *Pull-Down* para la salida. Solo si ambos son '0' se realizarán las conversiones. Esto se muestra en la tabla 5-2 de la hoja de datos técnicos.

La implementación del modo de bajo consumo se hace mediante el método *dacPower-Down*. Este posibilita el paso de dos parámetros distintos, si bien ambos tienen valores por defecto. El primero permite elegir la resistencia de *Pull-Down*; se han definido tres constantes para el paso de este parámetro. En caso de no emplear ninguna de las tres, se asignará automaticamente el valor intermedio (100KΩ). El segundo parámetro sirve para indicar si se guarda el nuevo estado del registro en la memoria EEPROM del convertidor. Por defecto se encuentra desactivado, pero el paso de cualquier dato no nulo lo activa.

Los bits C0, C1 y C2 se emplean para definir el régimen de funcionamiento del convertidor, siempre que este no se encuentre en modo de bajo consumo. El *MCP4725* permite operar en modo normal (C1=1) o en modo rápido (C1=0). En modo normal aparece la posibilidad de guardar los datos del registro en la EEPROM interna (C0=1). En modo rápido tan solo se modificará el registro volátil. El modo normal sin guardado y el modo rápido tienen el mismo resultado, aunque el modo rápido emplea un byte menos en la comunicación. La estructura de los comandos que hay que utilizar en cada caso se muestran en las figuras 6-1 y 6-2 de la hoja de características.

Se decide implementar cinco métodos distintos de escritura. El primero de estos métodos empleará la llamada *dacTest* y servirá para comprobar rápidamente si el convertidor es capaz de generar tensiones en su salida. Para ello se genera una tensión en forma de escalera o diente de sierra. Se puede modificar el comportamiento al realizar un paso de parámetros con la altura y la duración del escalón. Mediante este método se puede comprobar el estado del DAC empleando únicamente un voltímetro, aunque de esta forma solo se puede comprobar que la tensión varía entre ciertos límites y a una velocidad determinada. Se puede realizar una mejor verificación empleando un osciloscopio, tal y como se muestra en la figura 69.

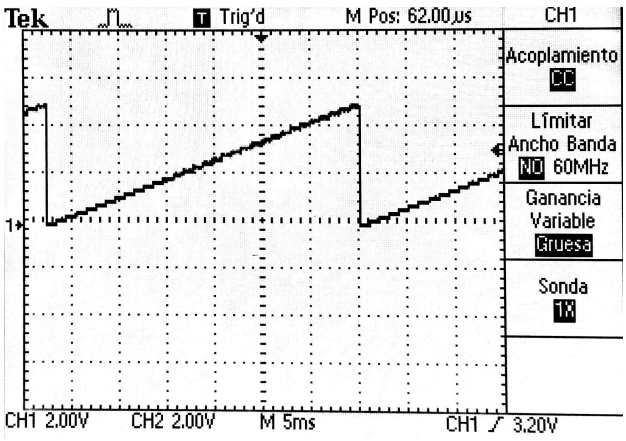


Fig. 69: Prueba del DAC con el método *dacTest*. Fuente propia

Los otros cuatro métodos de escritura sirven para generar tensiones concretas. Tener cuatro métodos permite cubrir todas las diferentes opciones que generan los dos modos de funcionamiento (normal y rápido). Ya que se permite el paso por parámetro, tanto del dato numérico que se va a generar, como directamente el valor de la tensión de salida deseada,

que deberemos escalar previamente. La llamada a estos métodos sirve para distinguirlos: *dacFastData*, *dacFastVoltage*, *dacNormalData*, *dacNormalVoltage*.

Para reducir el peso del archivo, se ha decidido que las funciones que reciben por parámetro la tensión deseada realicen el cálculo necesario y, posteriormente, llamen al método que genera la salida a partir del dato numérico. Se plantea la posibilidad emplear un paso de parámetro de tipo *auto* para emplear un único método que permitiera ambas posibilidades; sin embargo, Arduino no admite esta práctica.

Adicionalmente le podemos pedir al DAC que nos proporcione la información contenida en el registro y en la memoria. Esto se implementa mediante el método dacReadStatus, que guarda toda la información en la estructura *dacReadStruct* de tipo *dacReadStructType*, siendo esta un atributo de la clase. Si realizamos un paso de parámetro no nulo a la función, esta, además, volcará los datos al bus serie, si este se encuentre activo.

ADC

El convertidor analógico-digital tiene cuatro canales distintos, y permite activar varios a la vez. Además, todas las conversiones pueden realizarse empleando como referencia la tensión de alimentación o la tensión de la tercera entrada. Por lo tanto, será necesario un métodos de configuración, además de los métodos de lectura.

La dirección del ADC, necesaria para la comunicación I2C, se puede ver en la tabla 8 de la hoja de datos técnicos. Este dato depende únicamente del modelo concreto, ya que no dispone de pines de selección de dirección. En este caso la dirección es 0x29, y se incluye como atributo constante de la clase. Se va a controlar el convertidor mediante el uso de cuatro métodos con los siguientes objetivos: configuración, lectura discreta, lectura continua y lectura rápida.

El método de configuración, *adcConfig*, tendrá dos implementaciones distintas, aunque una única llamada. En la primera de ellas la configuración se realizará mediante el paso de un parámetro por cada canal disponible más otro parámetro para la selección de la referencia, aunque este último se encuentra por defecto desactivado. Si un parámetro es positivo, el canal correspondiente se activará y viceversa.

La segunda implementación del método emplea un único paso de parámetro para realizar toda la configuración, además permite controlar adicionalmente el filtro del I2C, el *Bit Trial Delay* y el *Sample Delay*. Este parámetro se puede definir fácilmente usando operaciones con las constantes públicas de la clase: *Channel0*, *Channel1*, *Channel2*, *Channel3*, *RefSel, ADCFilter, BitTrialD* y *SampleD*.

El *Bit Trial Delay* y el *Sample Delay* se utilizan para evitar que se realicen operaciones críticas mientras el bus I2C está activo. Se ha comprobado experimentalmente que las medidas no son adecuadas cuando se desactivan estas opciones.

Para realizar una lectura individual se emplea el método *adcRead*. En las tablas 12 y 13 de la hoja técnica se puede ver el formato del paso de información. Este método decodifica el valor de la lectura y lo da como resultado de su llamada. Si se realiza el paso de un puntero a entero en la llamada al método, este se actualizará con el número del canal empleado. Esta opción puede desactivarse al no pasar parámetro o cuando este es el valor 0. Finalmente, y cómo ayuda a las verificaciones, se puede realizar el paso de un segundo parámetro no nulo para que el método envíe al bus Serial toda la información de la lectura y del canal, así como la tensión estimada de acuerdo al atributo de escala, *adcScale*.

La lectura continua, *adcContRead*, permite automatizar la toma de una cantidad determinada de datos y su guardado en un vector. Además, permite obtener un segundo vector con los tiempos de toma de cada dato, facilitando la reconstrucción de la onda capturada. Es necesario pasar por parámetro el puntero del vector donde se deseen guardar los datos y el

número de lecturas que se quieren hacer. No existe un límite predefinido para el número de lecturas. El vector de tiempos no es obligatorio para el correcto funcionamiento del método, y se puede omitir al no realizar el paso por parámetro del puntero correspondiente.

El método de lectura rápida, *adcFastRead*, realiza una petición de 16 lecturas consecutivas y guarda toda la información en dos vectores, uno de bits altos y otro de bits bajos. Al final se decodifican los datos recibidos en un array, cuyo puntero debe pasarse por parámetro en la llamada al método. Se han realizado diversas pruebas y no se ha conseguido que el sistema permita lecturas de más de 32 bytes seguidos del bus I2C mediante una sola orden. Si se desean encadenar más de 16 lecturas, existirá un intervalo temporal mayor entre las lecturas *16n* y *16n+1* frente al resto.

Esto se produce porque, al realizar la petición de 16 datos, el ADC realiza las 16 conversiones y, posteriormente, comienza la transmisión de los datos. Por lo tanto, las medidas se realizan en un espacio de tiempo muy corto, aunque no se pueden encadenar más de este límite. Este método se recomienda para realizar medidas rápidas de un intervalo temporal reducido (hasta 16 muestras); para realizar medidas continuadas es más recomendable emplear el método *adcContRead*, aunque su intervalo temporal entre muestras es mayor.

RC/RLC

En el subsistema RC/RLC hay que considerar un aspecto fundamental durante el desarrollo de los métodos de la clase. Todos los interruptores analógicos funcionan con lógica positiva; sin embargo, dos de ellos están dispuestos en paralelo con los componentes, mientras que los otros seis están en serie. Esto hace que eso dos integrados funcionen de cara al circuito con lógica negativa. Los componentes del filtro que se ven afectados son la resistencia del filtro RC y la bobina del filtro RLC. Otro aspecto importante del circuito es que, cuando se active la bobina, también es necesario poner una resistencia en serie. De lo contrario el circuito seguirá estando abierto.

Se implementan dos métodos, uno de inicialización y otro de configuración. En ambos casos hay que modificar los registros del expansor de puertos I2C *PCA9554*. La dirección en el bus de este integrado es 0x20, ya que sus pines de direccionamiento se han llevado a 0V. Toda la comunicación con el integrado se realiza mediante el envío dos bytes. El primer byte indica qué registro se va a modificar y el segundo byte, la información que se transmite a dicho registro. Esto no se cumple al usar los pines del expansor como entradas digitales, pero en este proyecto no se van a emplear como tales.

En el método de inicialización, declarado como *filterInit*, se accede al registro de configuración, que es el número 3, y se escribe un cero lógico en todos los bits del mismo. De esta forma, se configuran todos los pines como salida digital. Este paso resulta vital ya que por defecto los pines están establecidos como entrada, tal y como se explica en el apartado 6.1.5 de la hoja de datos técnicos.

En un segundo paso de la inicialización se accede al registro del puerto de salida, número 1, y se configura con el dato 0x09. Esto implica que se activan la primera y la cuarta salida. Estas salidas son las que actúan contra los interruptores analógicos que controlan la primera resistencia y la bobina. Por lo tanto, estas se puentean, quitando todos los elementos de los filtros. Es importante llevar desde el comienzo los filtros a un estado conocido para evitar posible confusiones.

Para configurar de cualquier otra forma los filtros debemos emplear el método *filterConfig*. Este cuenta con dos implementaciones distintas. En la primera, es necesario realizar el paso por parámetros del estado deseado de cada componente de forma individual. Esto permite la configuración de los filtros de cualquier forma deseada, pero es fácil cometer errores frente al segundo método.

La segunda implementación emplea únicamente el paso de un parámetro. Este se escribe directamente en el registro de salida del expansor de entradas o salidas. Dicho parámetro es un número puede ser calculado a mano, al conocer la relación de cada bit con el comportamiento de los componentes del circuito. Sin embargo, para agilizar el uso de este método, se han predefinido atributos constantes de la clase para una configuración rápida y cómoda. No se han definido atributos para todas las combinaciones, tan solo para algunas representativas: *Filter0*, *FilterRC1*, *FilterRC2*, *FilterRL*, *FilterLC*, *FilterRLC* y *FilterRCRLC*.

Op. Amp. 15

En este caso, al igual que en el anterior, todo el control se realiza a través del expansor de E/S *PCA9554*. Pese a tratarse del mismo integrado, la dirección en el bus I2C es distinta debido a los pines de direccionamiento, en este caso es 0x27.

Los métodos utilizados para el control del integrado son similares a los anteriores, aunque con algunos ajustes. Se utilizan dos métodos, uno de inicialización y otro de configuración. Sin embargo, debemos tener en cuenta que el *DG411* funciona con lógica negativa.

El método de inicialización, *opAmpInit*, funciona igual que el método *filterInit*; tan solo es necesario modificar la dirección a la que se envían los paquetes, y modificar el registro de salidas. En este caso, debido a la lógica negativa de los interruptores analógicos, se fijan todos los bits del registro a 1. Por defecto, el método de inicialización deja abiertos todos los interruptores, si se desea acceder a las calibraciones tendrá que emplearse el método correspondiente.

La calibración en este subsistema no cuenta con una gran cantidad de combinaciones, como en el caso anterior, tan solo existen cinco posibles configuraciones del circuito. En cada una de las etapas de calibración solo se encontrará activo un interruptor cada vez. Por lo tanto, se pueden emplear parámetros predefinidos para configurar cualquier caso, esto además simplifica la configuración al ser necesario emplear lógica negativa. La llamada al método será *opAmpCalib* y emplearemos los atributos públicos de la clase *OpAmp_No_Calib*, *OpAmp_Calib_1_0V*, *OpAmp_Calib_1_pos*, *OpAmp_Calib_1_neg* y *OpAmp_Calib_2*.

Servo

El control del servomotor es directo, tan solo es necesario utilizar una señal PWM generada por el pin correspondiente, el pin 6. Por lo tanto, no implementaremos ningún método de control. Lo que sí vamos a implementar es un método de verificación, donde emplearemos la señal leída del potenciómetro para actuar sobre el servomotor. Este método emplea la llamada *testServo*.

La lectura de la realimentación del consumo del servomotor se realizará leyendo el canal 2 del ADC, por lo que se emplean los métodos implementados para el convertidor.

SRAM

La memoria SRAM cuenta con tres modos de funcionamiento en relación al autoincremento realizado por el integrado en operaciones consecutivas. El modo más simple es el modo byte, aquí no existe ningún autoincremento y se harán las operaciones sucesivas sobre un mismo registro. En el modo de funcionamiento por páginas, se realizará un autoincremento tras cada operación, pero al llegar al final de una página (32 bytes) se volverá al comienzo de la misma. El modo secuencial lleva a cabo incrementos tras cada operación ignorando los límites de página, por lo que se escribirá en toda la memoria hasta alcanzar el último registro, tras el cual se volverá al primero.

Para el control de este integrado, permitiendo múltiples modos de funcionamiento, vamos a emplear siete métodos: uno de configuración, dos de escritura, dos de lectura y dos auxiliares.

El método de configuración se usa para indicarle a la memoria si se van a realizar operaciones de lectura o escritura, y en que modo. Al llamarlo, con la orden *sramConfig*, se realiza el paso de dos parámetros. El primero de ellos indica sobre qué modo actuar (lectura o escritura) empleando los atributos *SRAMreadRegister* o *SRAMwriteRegister*. Mientras que en el segundo parámetro se configura el modo de funcionamiento a byte, página o secuencial mediante los atributos *SRAMbyteMode*, *SRAMpageMode* o *SRAMseqMode*.

Los métodos auxiliares ayudan a simplificar el código en los métodos de lectura y escritura, ya que tanto la apertura y configuración del bus como el cierre del mismo son similares en todos los casos. El método *sramBegin* sirve para comenzar la comunicación con la memoria y *sramEnd*, para finalizarla.

Los métodos *sramRead* y *sramWrite* sirven para realizar operaciones individuales de lectura o escritura respectivamente. La lectura solo requiere un paso por parámetro del registro al que se quiere acceder, y devuelve el dato contenido en dicho registro. El método de lectura requiere el paso de dos parámetros, el registro y el dato a escribir, y no devuelve nada.

Los métodos *sramContRead* y *sramContWrite* sirven para realizar lecturas o escritura continuas empleando un vector para almacenar o tomar los datos. En ambos casos es necesario proporcionar, mediante el paso de parámetros, el registro donde se comienzan las operaciones, el número de operaciones a realizar y el puntero al vector de datos.

Para verificar rápidamente este integrado podemos emplear el método *sramTest*. Este método realiza una escritura en un registro e, inmediatamente, solicita ese mismo dato. De esta forma se comprueba si los métodos de lectura y escritura funcionan correctamente.

Métodos de calibración

Los métodos de calibración son aquellos en los que se busca compensar el comportamiento no ideal de algunos circuitos. Los subsistemas que deben ser calibrados son ambos convertidores y el subsistema OpAmp15, tanto en la etapa de entrada con en la de salida.

En primer lugar, es necesario realizar la calibración de los convertidores, comenzando por el ADC y posteriormente el DAC. En estos dos procesos se utiliza un único método, denominado *conversorsCalib*.

Las señales de ambos integrados emplean como referencia la tensión de alimentación de 5V. Como esta no está regulada y puede variar, es recomendable calibrar los conversores tras cada encendido.

Para calibrar el ADC se va a emplear la señal conocida que existe en su tercera entrada. Esta señal será siempre de 3V3, por lo que se puede comparar esa información con la lectura obtenida del convertidor y calcular la relación real entre el dato recibido y la tensión existente. Este dato se guarda en el atributo *adcScale*.

Tras la calibración del ADC se conecta en serie con el DAC para ajustar este último. Para ello se genera una tensión determinada con el DAC y se compara el valor teórico generado con la señal real recibida en el ADC, que ya ha sido previamente calibrado. Conociendo la relación entre ambos datos se puede calcular la escala real del DAC, que se guarda en el atributo *dacScale*.

Con ambos conversores calibrados se pueden ajustar las etapas de entrada y salida del subsistema *OpAmp15*. En primer lugar, se calibra la etapa de entrada mediante las tensiones de referencia de *+10V*, *-10V* y *0V*. Con la referencia de 0V se obtiene directamente el offset real de la amplificación, que idealmente sería 2’5V. La ganancia del sistema se calcula como la pendiente de la recta que se forma al unir las lecturas de *+10V* y *-10V*.

Con estos dos valores (ganancia y offset), que se guardan como atributos de la clase, se puede calcuñar la tensión real del motor en función de la tensión leída por el conversor mediante el método *opAmpIn*. En la figura 70 se muestra cómo obtener los parámetros y cómo ajustar las lecturas de acuerdo a dichos parámetros.

$$Offset = f(0V)$$

$$G = \frac{f(-10V) - f(10V)}{20V}$$

$$V_{\text{Real}} = G \cdot V_{\text{Med}} + Offset$$

Fig. 70: Fórmulas para la calibración de la etapa de entrada. Fuente propia

Para calibrar la etapa de salida es necesario emplear la recién calibrada etapa de entrada. Ahora se dispone de todo el rango de funcionamiento para realizar la calibración, aunque se simplifica empleando los tres mismos puntos que en el caso anterior, así también se evitan posibles discordancias que podrían aparecer entre las dos etapas si se calibraran de formas completamente diferentes.

Se comienza generando la tensión que idealmente debería producir la señal deseada. Mediante el ADC se puede comprobar en cuánto difieren el resultado real y el ideal. Se realizan ajustes sucesivos en el dato enviado al DAC hasta que esta diferencia es mínima. Al realizar este proceso con los tres puntos se obtiene, de igual forma al caso anterior, la ecuación de una recta que sirve como función de transferencia entre la tensión final deseada y la tensión real inicial que debemos generar. Esta función se aplica con el método *opAmpOut*.

8.2 Ejemplos

Los ejemplos son programas sencillos donde se enseña a los alumnos a controlar los diferentes subsistemas y a realizar interacciones entre ellos. Están pensados para que los alumnos puedan aprender sin necesidad de recurrir al interior de la librería. Se han desarrollado los siguientes ejemplos:

- Control de la interfaz (*IO_Example.ino*).
- Uso de los conversores (*Conv_Example.ino*).
- Uso de los filtros RC/RLC (*RC-RLC_Example.ino*).
- Uso del subsistema OpAmp15 (*OpAmp_Example.ino*).
- Uso en bucle cerrado del Servomotor (*Servo_Example.ino*).
- Uso de la memoria RAM (*SRAM_Example.ino*).

El ejemplo *IO_Example* funciona de forma similar a los métodos *IOconfig* e *IOtest*. Se trata de un código que emplea todos los componentes de la interfaz con interacciones entre sí. De esta forma se espera que los alumnos aprendan a controlar botones, potenciómetros y LEDs a la vez que comienzan a acostumbrarse al uso de algoritmos de control.

Conv_Example.ino es un código que muestra cómo comunicarse con ambos convertidores. Lo más importante de este ejemplo es comprender el protocolo de comunicación I2C y la codificación de la información en los mensajes. Este ejemplo se centra solo en la comunicación.

El archivo *RC-RLC_Example.ino* sirve para comprender cómo reconfigurar los filtros. En él se muestra cómo se configura el expansor mediante los parámetros preestablecidos. En este archivo también se encuentra implementado, aunque no en funcionamiento, una función de configuración por parámetros discretos que permite conocer exactamente cómo se trabaja con el expansor, además de permitir el acceso a todas las posibles combinaciones de los filtros. En el cuerpo del programa se emplea el botón A para generar un escalón de tensión de 500ms de duración. Tras esto el ADC registra en un vector bidimensional la lectura recibida y el momento de su captura. De esta forma se puede reconstruir fácilmente la señal mediante MatLab o cualquier otro software similar.

Para lograr un buen funcionamiento en el subsistema *OpAmp15* resulta necesario realizar una calibración. La primera parte del ejemplo *OpAmp_Example.ino* se centra en lograr la calibración de ambas etapas, aunque para ello se deben calibrar previamente los conversores. La calibración del motor, cuyo algoritmo ya se ha explicado previamente, se encapsula en una función, de forma que puedan llevarse a cabo las prácticas sin necesidad de saber cómo realizar dicho ajuste.

La parte principal del código para el control del motor comienza con la obtención de un dato a partir de la señal del potenciómetro. Posteriormente este dato se ajusta según las funciones obtenidas de la calibración y se envía a la etapa de salida. Finalmente se captura y se ajusta la señal realimentación proveniente de la etapa de entrada.

Una ventaja de este ejemplo es que enseña cómo controlar el motor empleando funciones para los ajustes. Por lo tanto, el control del motor puede realizarse sin necesidad conocer los métodos de la calibración, tan solo es necesario emplear estas mismas funciones.

El ejemplo de control del servomotor, *Servo_Example.ino*, es uno de los más sencillos. Para lograr una salida tan solo es necesario generar una señal PWM por el pin 6. La realimentación es directa a través del canal 2 del conversor A/D (preferiblemente calibrado). Tras esto solo debemos escalar las lecturas de acuerdo a la ganancia del circuito.

SRAM_Example.ino es el último ejemplo que se va a incluir, se trata de un código que permite guardar una palabra en la memoria y posteriormente recuperarla. Pese a que el código no tiene una gran complejidad, sí que añade una nueva característica interesante con la que el alumno se debe familiarizar: el protocolo de comunicaciones SPI.

9. Resultados

En este apartado final se van a analizar aspectos del comportamiento final de la placa.

Velocidad de captura de datos del ADC

Una gran parte de las funciones de la placa requieren de leer información mediante el conversor A/D. Así, una de las características más relevantes de esta nueva placa es la velocidad de captura de los datos. Esto viene dado por la propia frecuencia de captura del conversor, pero también por la programación realizada en el microcontrolador.

Como se ha comprobado anteriormente, existe un límite en la cantidad de datos que se pueden pedir de forma simultánea al conversor. Por lo tanto, vamos a comprobar cuál es la velocidad máxima de toma de muestras en los dos casos posibles de lecturas continuas: tomando muestras discretas, o realizando una serie de 16 muestras rápidas.

En ambos casos se muestrea la misma señal subamortiguada generada en el circuito LC tras generar un escalón de 1V a 4V con el DAC. Sin embargo, cabe destacar que en el caso de las muestras rápidas no se puede obtener el valor del instante de tiempo de captura. Esto se debe a que primero se toman todas las lecturas, y posteriormente se realiza la comunicación. Otro problema es que durante la representación gráfica aparecen zonas sin datos, correspondientes al momento de traspaso de información, y durante el cual no se realizan medidas (marcadas con las líneas rojas). La gran ventaja de este método es la elevada frecuencia de muestreo que se puede alcanzar. Ambas características se pueden observar en la figura 71.

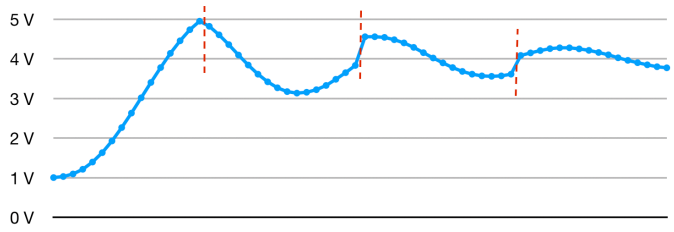


Fig. 71: Reconstrucción de la señal empleando muestras rápidas. Fuente propia

En el caso de las lecturas discretas se puede reconstruir completamente la señal, puesto que se dispone de los valores de tiempo en los instantes de captura. Al representar gráficamente los datos capturados (figura 72) se observa que se cumple con el teorema de Nyquist, ya que existen más de diez muestras en cada ciclo de la onda.

Esto también es comprobable matemáticamente: el tiempo medio entre dos muestras es de 112us, equivalente a 8'9KHz, mientras que el tiempo medio entre dos picos de la onda es de 1348uS o 741Hz. Por lo tanto, el muestreo se realiza a una frecuencia doce veces superior a la de la onda original.

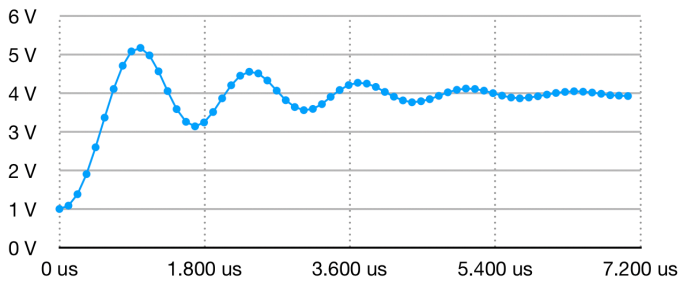


Fig. 72: Reconstrucción de la señal empleando muestras consecutivas. Fuente propia

Onda de mayor frecuencia generada

La capacidad de generar señales se estudia separándola en dos casos: solo el conversor y el conversor con la etapa de salida OpAmp15. En ambos casos se van a realizar dos pruebas fundamentales. En la primera se comprueba el s*lew-rate*, es decir, el tiempo que tarda el conversor en modificar su tensión desde el valor mínimo al máximo sin pasos intermedios. La segunda prueba servirá para comprobar la capacidad de generar ondas senoidales construidas con al menos 10 puntos por ciclo. Para asegurar la precisión en los resultados se emplea un osciloscopio externo para las mediciones. Todos los resultados se muestran en la tabla de la figura 73.

	Slew-Rate		Onda senoidal	
	V/us	Freq.	Tiempo	Freq.
DAC (0-5V)	5V/9'2uS	543 KHz	1,09ms	917 Hz
OpAmp15 (-10V→10V)	20V/90uS	222 KHz	1,09ms	917 Hz

Fig. 73: Tabla de frecuencias máximas del DAC y de la salida del OpAmp15. Fuente propia

Analizando la tabla se puede comprobar que el *slew-rate* del amplificador operacional de salida es bastante menor al del DAC, por lo que la frecuencia máxima que es capaz de seguir es inferior. Esto se puede deber al condensador de filtrado de tipo C que se encuentra en dicho circuito.

Sin embargo, en ambos casos, con o sin amplificador, el factor más determinante es la velocidad a la que se puede transmitir la información al convertidor. Al emplear una comunicación I2C de 400KHz (ya que el Arduino no soporta más velocidad) y ser necesarios cuatro bytes por cada dato que queramos escribir, no se puede lograr una tasa de variación muy elevada. Este retardo implica que el periodo mínimo de una onda de diez puntos sea de 1'09ms, por lo que la frecuencia máxima es de 917Hz.

Fiabilidad de las comunicaciones

Otro de los aspectos que podemos analizar es el comportamiento de las líneas de los buses de comunicaciones. Al trabajar con velocidades de comunicación muy elevadas, puede ocurrir que integrados no sean capaces de modificar los niveles lógicos con la rapidez suficiente. Esto está muy relacionado con los transistores que controlan el bus y la capacidad parásita de éste.

Para estudiar este efecto vamos a analizar las líneas de reloj de los buses SPI e I2C. La pendiente de subida de esta señal va a determinar la fiabilidad de estas comunicaciones. Cuanto más abrupta sea, más cuadrada será la onda, y por lo tanto más fiable será la comunicación. Pero una pendiente abrupta también genera armónicos de mayor energía, que generan EMIs. Una subida suave no generará interferencias, pero será más sensible, pudiendo llegar a provocar lecturas incorrectas. Cuanto mayor sea la velocidad, más abrupta deberá ser la subida para evitar lecturas incorrectas. En la figura 74 se muestran las líneas de reloj de ambos buses.

Como puede observarse, la línea de reloj del bus I2C mantiene una forma aproximadamente cuadrada y sin sobreoscilaciones, por lo se considera que las resistencias de *Pull-Up* se encuentra bien escogidas. Se puede calcular la constante de tiempo (tau) del circuito, que sirve también para conocer la capacidad del bus. Esta constante se puede calcular como el tiempo que tarda la señal en subir un 63% o el un tercio del tiempo que tarda en alcanzar el 95% de su valor.

De acuerdo a la figura 74, el 95% del valor final en la subida se alcanza en aproximadamente 1uS (tau de 300ns). En bajada la tau es de aproximadamente 100ns. Podemos calcular la capacidad del bus dividiendo la tau de subida entre la resistencia de *Pull-Up*. Esto da una capacidad de aproximadamente 300pF.

El bus SPI trabaja a una velocidad mucho mayor, como se ve en las escalas temporales de ambas gráficas, por lo que tiene una forma más similar a una onda triangular. Esto podría llegar a provocar problemas en las comunicaciones debido a la poca definición que existe en los niveles de tensión del bus. Una posible solución consistiría en bajar la velocidad del reloj del bus, permitiendo más tiempo para que las señales lleguen a unos niveles adecuados y generen ondas cuadradas. En este caso la constante de tiempo de subida y de bajada es similar, de aproximadamente 50ns. Pese a tener una constante de tiempo menor, que implica una subida más abrupta, esta onda es menos adecuada debido a que la velocidad de comunicación es mucho mayor frente al bus I2C. La capacidad de este bus es de aproximadamente 166pF.

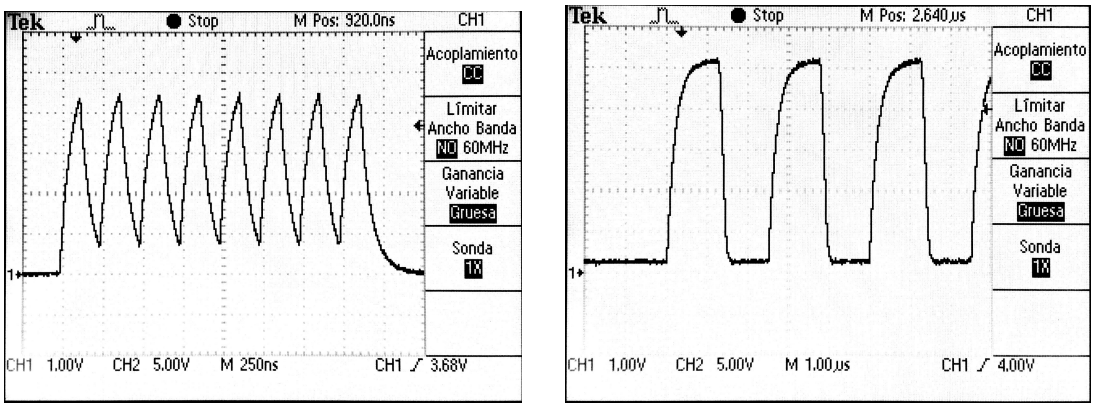


Fig. 74: Líneas de reloj del SPI (izq.) e I2C (der.). Fuente propia

En caso de empezar a detectar problemas en el bus SPI a una velocidad a la que se desee trabajar, se podrían cambiar las resistencias serie de las líneas. Esto permitiría un mayor paso de corriente, y por lo tanto, un cambio más rápido en los niveles lógicos del bus. Aun así, estas resistencias también limitan la emisión de ruidos (al reducir la energía de los armónicos), por lo que si no existen problemas no es recomendable realizar modificaciones.

Ensayos de EMIs

Los ensayos de EMIs, o interferencias electromagnéticas por sus siglas en inglés, son pruebas destinadas a comprobar los niveles de emisión y sensibilidad de la placa a los ruidos electromagnéticos. Los ensayos de EMIs homologados son necesarios para vender productos electrónicos con el sello CE en la Unión Europea. En este caso no se van a realizar ensayos homologados ya que no se pretende exportar la placa; sin embargo, resulta muy interesante conocer el comportamiento del escudo en este aspecto, ya que puede ayudar a detectar problemas y a planificar futuras mejoras.

Para comprobar el comportamiento de placa se realizan dos ensayos de EMIs emitidas por la placa (radiadas y conducidas) y uno de inmunidad frente a radiaciones externas.

EMIs Radiadas

El ensayo de EMIs radiadas se puede hacer mediante una antena o mediante una sonda de campo cercano. En este caso se utiliza una sonda de campo cercano conectada a un analizador de espectros para poder comprobar las radiaciones electromagnéticas generadas por cada circuito de la placa; el montaje se muestra en la figura 75. En la placa se carga un programa que emplea simultáneamente comunicaciones I2C, SPI y ondas PWM.

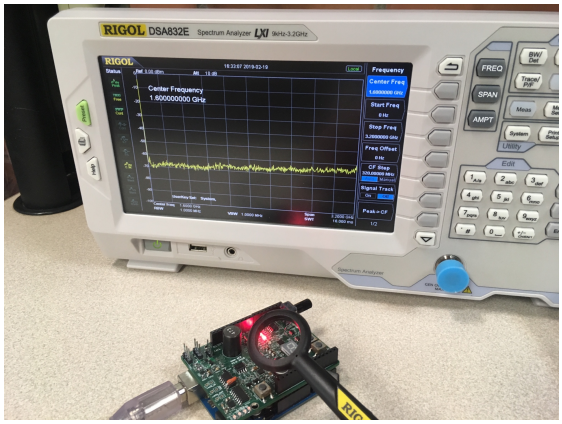


Fig. 75: Montaje para detección de EMIs radiadas. Fuente propia

Durante los ensayos no se aprecian ruidos electromagnéticos provenientes de la placa. Lo que sí se puede detectar es el ruido generado por el cristal de cuarzo del Arduino. Esto se puede observar en la figura 76, el ruido existente esta situado en las frecuencias de los múltiplos de 16MHz, por lo que se trata del oscilador y sus armónicos.

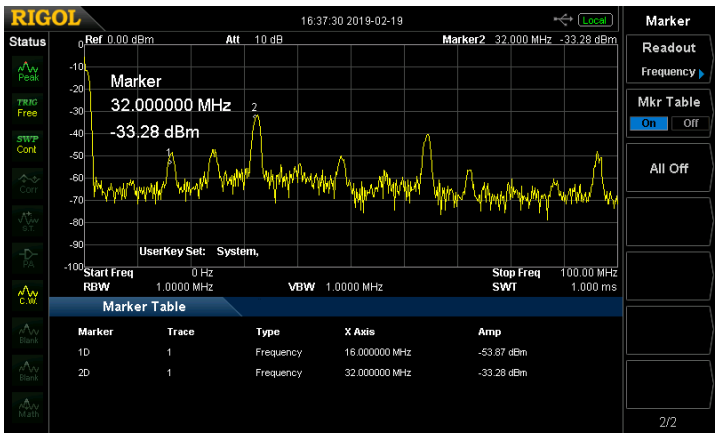


Fig. 76: EMIs radiadas desde el oscilador del Arduino UNO. Fuente propia

Se puede afirmar que el escudo produce menores interferencias electromagnéticas que el propio Arduino. Esta ausencia de ruido se achaca a dos factores. En primer lugar, cabe destacar el plano de masa que minimiza los lazos de corriente que pueden actuar como antena; y, en segundo lugar, las resistencia serie que se han implementado en el bus SPI, limitando su corriente máxima y, por lo tanto, su emisión electromagnética.

EMIs Conducidas

Para estudiar las EMIs que genera la placa por los conductores de alimentación hay que emplear un LISN (Line Impedance Stabilization Network). Se trata de un aparato que se conecta en las líneas de alimentación de un aparato electrónico y permite analizar los ruidos presentes en ellas. Mediante este ensayo se puede comprobar cuál es la efectividad de los filtros en la alimentación de la placa. Idealmente se debe utilizar una batería que no genere ruido para no afectar a los resultados. En este caso no se dispone de una batería de 5V, por lo que hay que en cuenta que puede aparecer ruido generado por la propia fuente de tensión.

Se considera que la mayor fuente de ruido conducido en la placa será la fuente de alimentación Boost, debido a la conmutación de corrientes de alimentación. Por lo tanto, para comenzar, se van a estudiar con el osciloscopio las frecuencias a las que aparecen ruidos en las tensiones de *+15V* y *-15V*, mostradas en la figura 77.

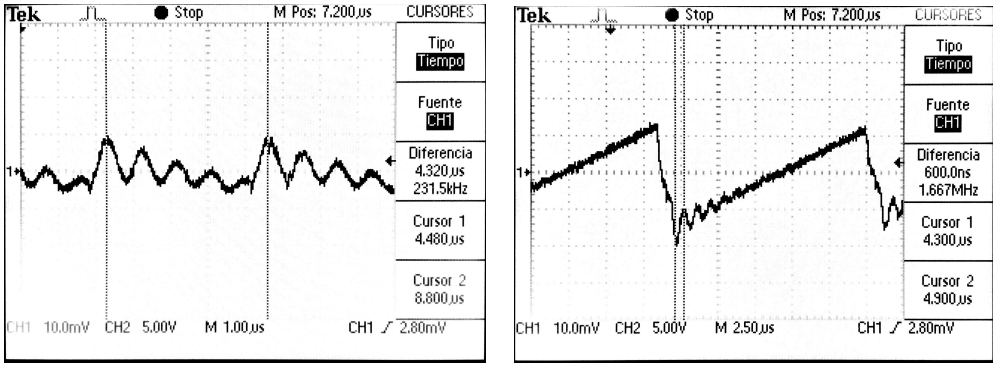


Fig. 77: Señales de alimentación *+15V* (izq.) y *-15V* (der.) usando acoplo de AC. Fuente propia

Las frecuencias importantes para ambas señales son la de conmutación del convertidor (231 KHz y 71KHz) y la del ruido que genera en cada conmutación (1'04 MHz y 1'6 MHz).

Al analizar la señal generada por el LISN, en la figura 78, se observa que estas frecuencias no coinciden con las frecuencias más importantes del LISN. Al igual que en el caso anterior, las frecuencias donde se encuentran los ruidos son armónicos de 16MHz, por lo que se trata de ruidos del oscilador de cuarzo del propio Arduino.

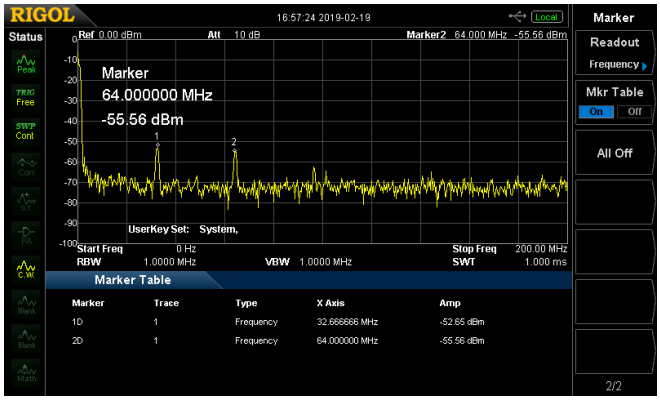


Fig. 78: EMIs conducidas detectadas mediante la LISN. Fuente propia

Inmunidad

El ensayo de inmunidad es aquel que se encarga de comprobar que la placa es capaz de funcionar correctamente pese a la existencia de interferencias externas. Para realizar este ensayo se va a introducir la placa dentro de una antena TEM, con la que se generan interferencias en todas las frecuencias hasta 1GHz. El montaje se muestra en la imagen 79.

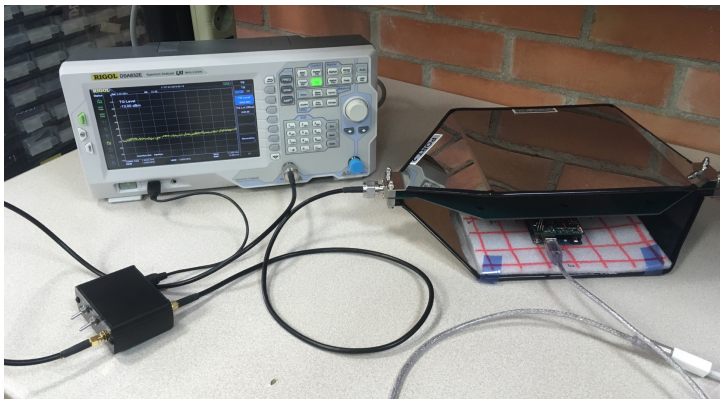


Fig. 79: Montaje para el ensayo de inmunidad. Fuente propia

Para analizar la inmunidad del escudo se prueban diferentes programas de verificación y se estudia si existe algún comportamiento anómalo en el sistema. Se comprueba que no existe ninguna alteración en el funcionamiento de los subsistemas pese a la radiación externa. Esto se debe, al igual que en el caso de las EMIs radiadas, a la reducida área de los lazos de corriente y a las resistencias de limitación de corriente que existen en diversas líneas. También cabe destacar que resulta crítico que los condensadores de desacoplo se hayan dispuesto lo más cerca posible de los pines de alimentación; de lo contrario, podrían aparecer *glitches*.

Coste económico

El coste económico de la placa se ha intentado mantener al mínimo. Durante todo el proyecto se ha empleado software gratuito: *KiCad* para el desarrollo electrónico, *Atom* para el desarrollo de los archivos de programación y *Arduino IDE* para la compilación y la subida de los códigos al microcontrolador.

Pese a ello, los componentes y la fabricación de las placas siguen teniendo un coste económico importante. Para la fabricación de cada placa el coste de los componentes es de 37’903€. Sin embargo, el coste del envío de los componentes es mucho mayor debido a las cantidades mínimas de pedido, alcanzándose los 165’897€. Esta diferencia se va reduciendo conforme se aumenta la cantidad de placas a fabricar y se aprovechan todos los componentes de los envíos.

El coste de los componentes, teniendo en cuenta las cantidades mínimas de pedido pero no los descuentos por grandes pedidos, para la fabricación de diez placas asciende a 426’997€. Esto equivale a 42’6997€ por placa, un valor mucho más cercano al coste individual de los componentes. El coste individual por placa queda por debajo de los 40€ realizando pedidos de material para 20 o más placas; y el coste real sería aun menor gracias a los ya mencionados descuentos por grandes cantidades.

A este precio hay que añadirle el coste de fabricación de la placa. Al tratarse de una placa de cuatro capas el coste económico es ligeramente superior a la fabricación de placas de dos capas. Empleando el mismo fabricante que ha realizado el prototipo el coste de 10 placas sería de 47$; mientras que el coste de 20 placas sería 67$. Por lo tanto, el coste material de cada placa es aproximadamente 47’4€ para diez unidades y 42’8€ para veinte unidades.

En estos precios no se ha incluido el coste de montaje industrial, ya que es un proceso bastante caro, debido a la gran cantidad de componentes y a que estos se sitúan por ambas caras. El montaje se puede realizar en la propia universidad de forma manual para ahorrar este coste. En caso de desear realizar un ensamblaje industrial el mismo fabricante de los prototipos oferta el servicio. El coste aproximado para nuestra placa sería de unos 370$ para diez placas o de 410$ para veinte. Esto situaría el coste total entre 75€, para diez unidades, y 63€, para veinte unidades.

Además se puede tener en cuenta el coste del proyecto completo. Este se estima en algo más de 5.000€, tal y como se puede ver en su anexo correspondiente. Añadiendo esto, y suponiendo una fabricación de 100 unidades, el coste final de cada placa ascendería a 92€.

10. Conclusiones

A continuación se va a realizar una valoración general de los resultados del proyecto, y se va a concluir con una lista de posibles mejoras de cara a una futura revisión.

10.1 Conclusiones

En primer lugar, conviene destacar que se ha cumplido uno de los objetivos fundamentales del proyecto; ya que se ha comprobado que no existen interferencias entre los diferentes sistemas y que se pueden emplear todos de forma simultánea.

Técnicamente se ha conseguido fabricar una placa que mejora ampliamente las características de su predecesora, evitando los problemas de colisiones y permitiendo trabajar con más precisión en las señales analógica gracias a los conversores A/D y D/A.

También se han cumplido los objetivos de que la reconfiguración de los filtros y la calibración de la etapas de amplificadores sean autónomas y no requieran de manipulación física de la placa. Esto permite la operación remota de las mismas y acelera el desarrollo de las prácticas.

Sigue quedando pendiente la sustitución del inductor fijo por un circuito equivalente que permita ahorrar en coste y espacio. Sin embargo, en las condiciones actuales, y teniendo en cuenta la investigación realizada, se considera poco probable encontrar una alternativa satisfactoria al inductor fijo.

Se ha logrado realizar un diseño electrónicamente robusto con un buen comportamiento electromagnético. Esto facilitará trabajar con varios escudos diferentes de forma simultánea en el mismo Arduino sin que existan problemas entre ellos.

Otro de los objetivos de diseño más importantes era mantener la placa en un coste económico razonable. Al lograr un coste inferior a 50€ por placa, suponiendo un montaje manual en la universidad, se puede plantear el uso extendido de la misma en practicas de diversas asignaturas. Aún realizando un montaje automatizado, el coste total podría mantenerse por debajo de los 65€.

Se espera que esta versión de la placa ayude a continuar la mejora del rendimiento académico de los alumnos que la emplean. Esto debería ir apoyado por la capacidad de la placa de realizar las reconfiguraciones y las calibraciones de forma rápida y autónoma; permitiendo un mejor aprovechamiento del tiempo de prácticas para profundizar sobre los aspectos más relevantes de la misma.

En líneas generales se considera que el desarrollo del proyecto ha sido un éxito, alcanzando la mayoría de los objetivos propuestos; logrando un producto final robusto y fiable.

10.2 Mejoras

Existen tres apartados de la placa en los que se podrían realizar algunas ligeras mejoras en futuros trabajos.

En primer lugar, y más importante, resultaría interesante añadir referencias de tensión para los conversores A/D y D/A. De esta forma se eliminaría la necesidad de realizar calibraciones vía software de los mismos y se mejoraría su precisión.

En segundo lugar se podría retomar la investigación acerca del gyrator de impedancia. Se podría investigar el uso de amplificadores de transimpedancia o integrados de funcionamiento similar para lograr un gyrator de uso general. Aunque el coste de estos integrados puede ser elevado debido a su escasez, podrían permitir generar inductancias variables o de valores muy elevados.

En último lugar considero que podría resultar interesante aumentar las posibles combinaciones de reconfiguración del subsistema RC/RLC, especialmente si se emplea un gyrator. Con las tecnologías ya implementadas podría investigarse la posibilidad de emplear un sistema de componentes dispuestos en flotante, y que ambos extremos pudieran conectarse en varios puntos. Se trataría un método similar al funcionamiento interno de las FPGAs.

Para llevar a cabo este último concepto podría resultar necesario emplear componentes de menor tamaño respecto a los actuales. Esto dificultaría el montaje manual, pero podría resultar rentable un montaje automatizado si se aplica a una serie relativamente grande de placas (veinte o más).

11. Bibliografia

B. D. H. Tellegen (April 1948). *"The gyrator, a new electric network element"*

Carl Nelson & Jim Williams (June 1986). *"Boost Converter Operation". LT1070 Design Manual*

Lin, Yu-Shiang (2008). *Low Power Circuits for Miniature Sensor Systems,*

NXP Semiconductors (April 2014). *UM10204 I2C-bus specification and user manual*

Motorola Inc. (Feb. 2003). *SPI Block Guide v3.06*

Kushner, David. (Oct. 2011). *The Making of Arduino*

KiCad (Jan. 2019) *Getting Started in KiCad*

James Bryant, Walt Kester (2004), *Data Converter Architectures*

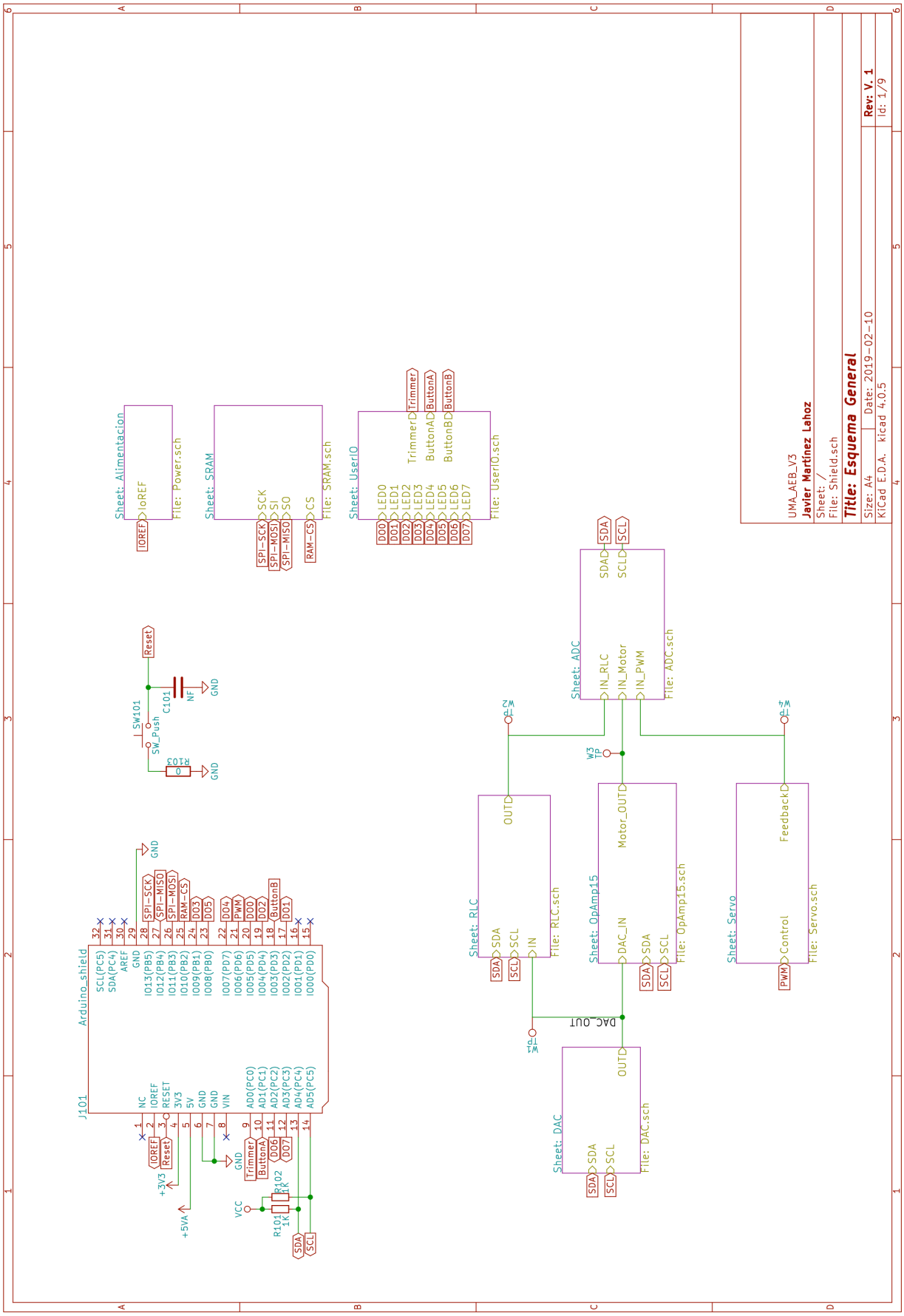
Horn, Elton T. *Analog Switches Applications and Projects*

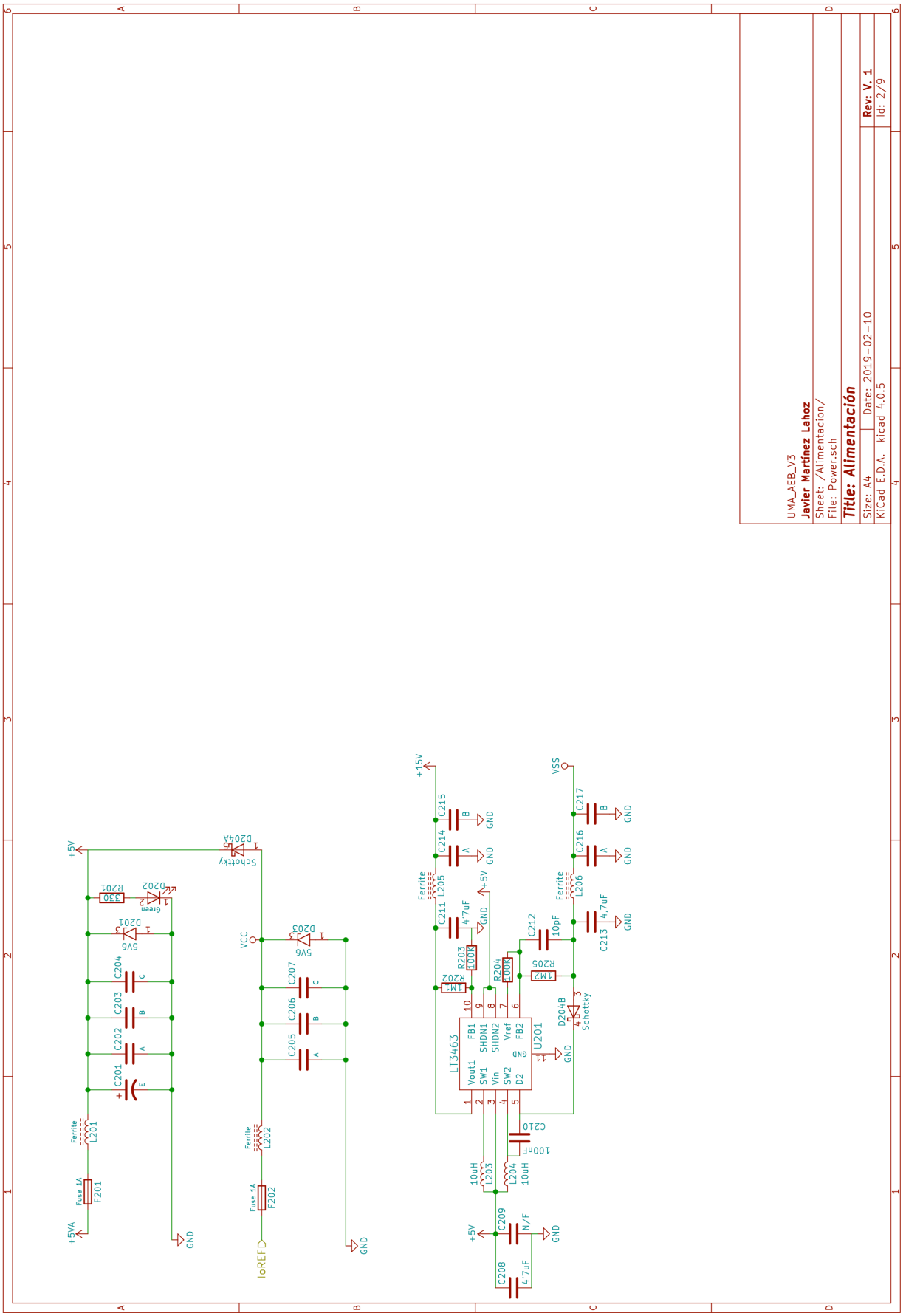
Analog Devices (2009), *MT-101 Decoupling Techniques*

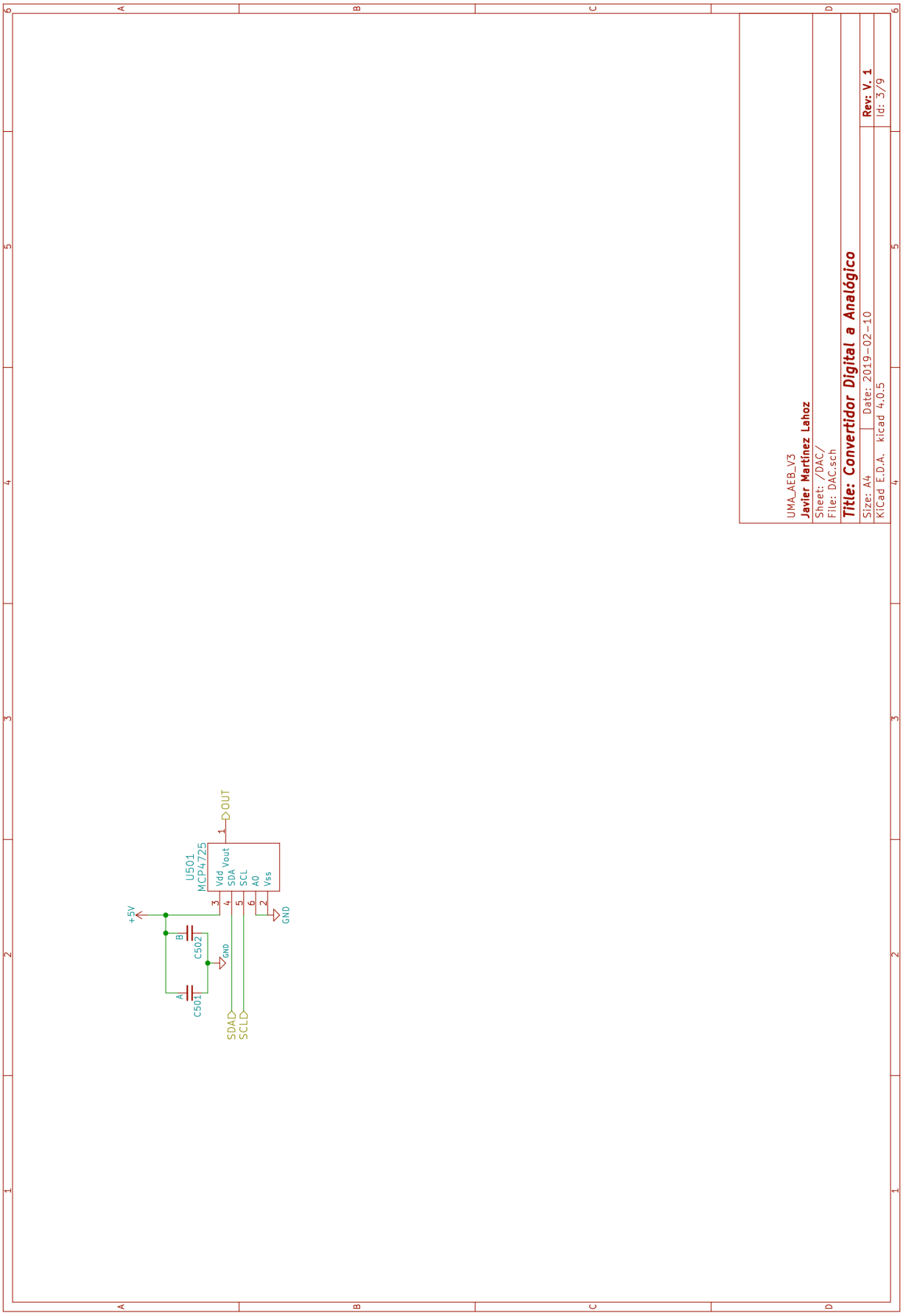
TABLA DE FIGURAS

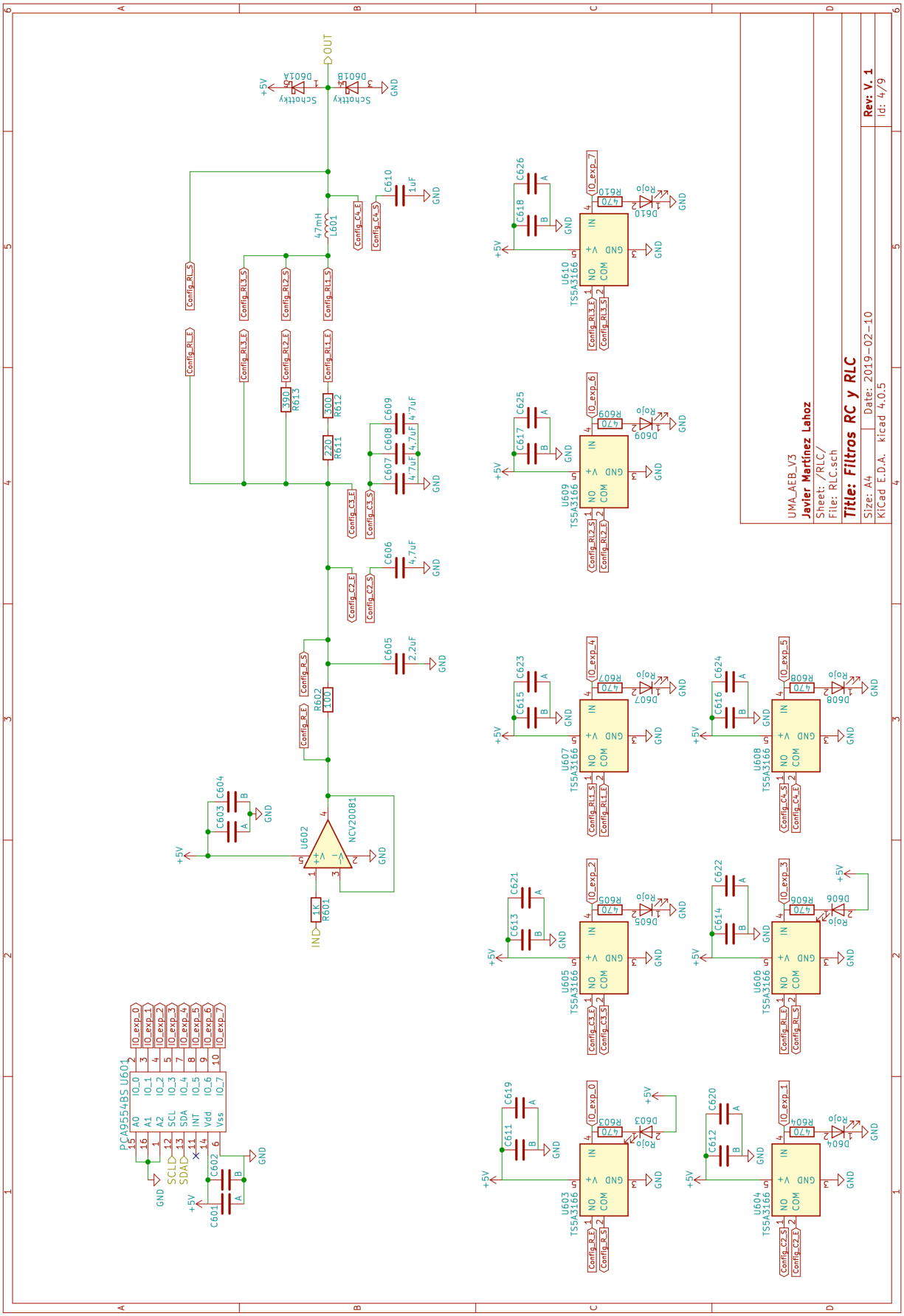
12. Anexos

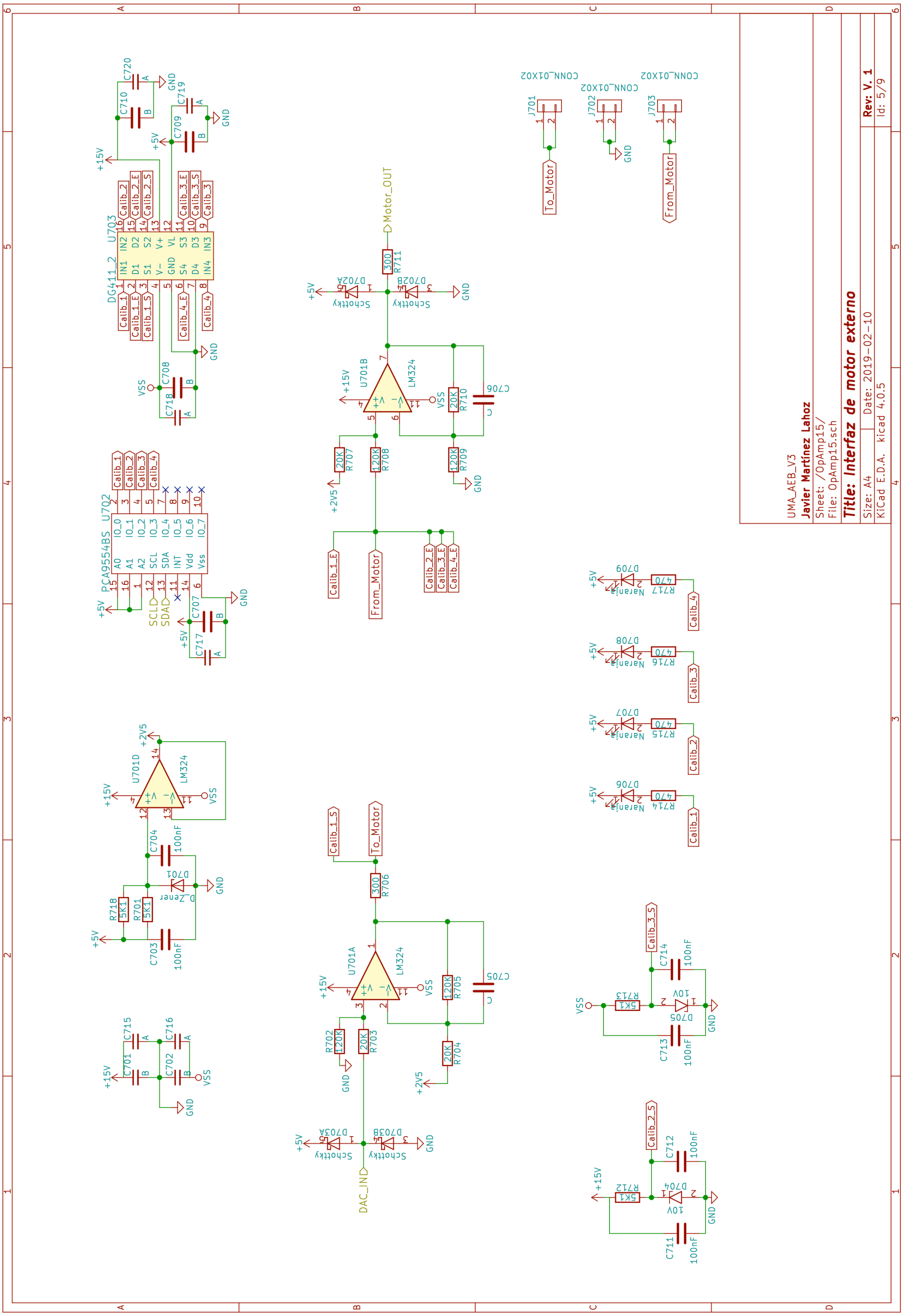
12.1 Esquemas electrónicos

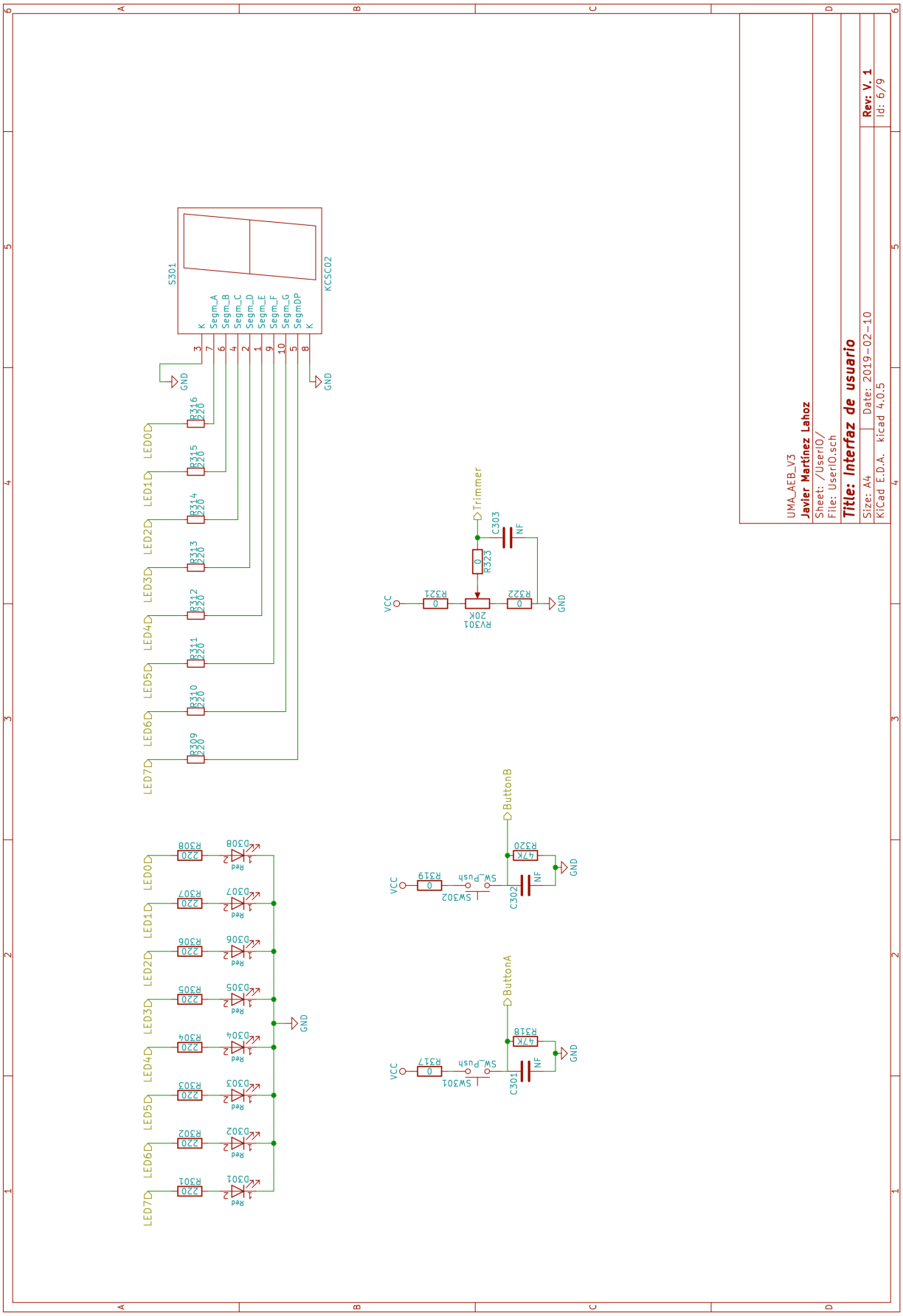


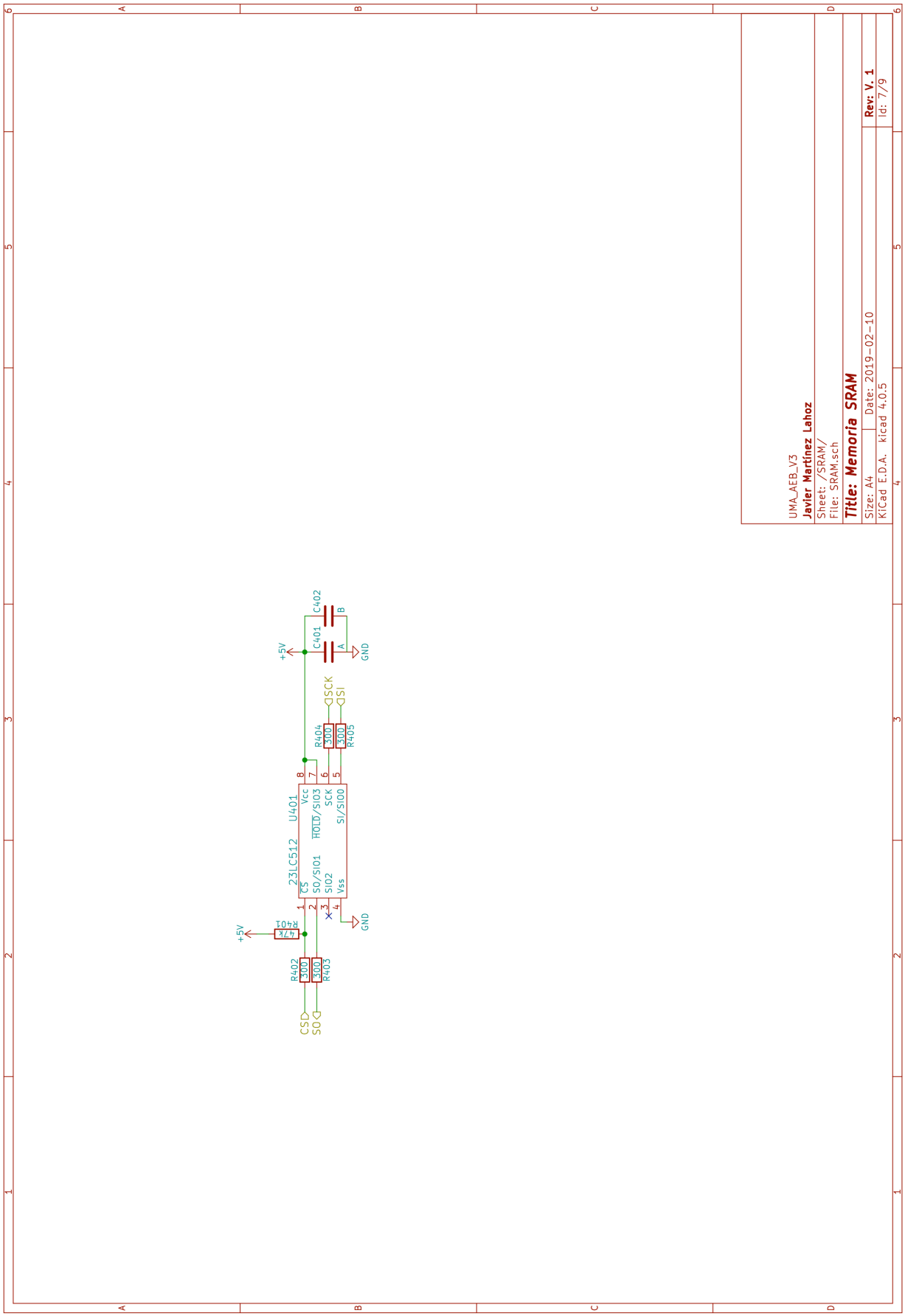

UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /Alimentacion/
File: Power.sch
Title: Alimentación
Size: A4
Date: 2019-02-10
Rev: V. 1
KiCad E.D.A. kicad 4.0.5
Id: 2/9
+5V
+5VA
Fuse 1A
F201
Ferrite
L201
C201
C202
C203
C204
D201
D202
R201
330
Green
5V6
GND
Schottky
D204A
VCC
D203
C205
C206
C207
F202
L202
IoREFD
+15V
C215
C214
C211
4.7uF
Ferrite
L205
R203
100K
R202
1M1
R204
100K
R205
1M2
C212
10pF
LT3463
Vout1
SW1
Vin
SW2
D2
GND
FB1
SHDN1
SHDN2
Vref
FB2
U201
D204B
Schottky
C210
100nF
10uH
L203
L204
10uH
C209
N/F
C208
4.7uF
VSS
C217
C216
C213
4.7uF
L206

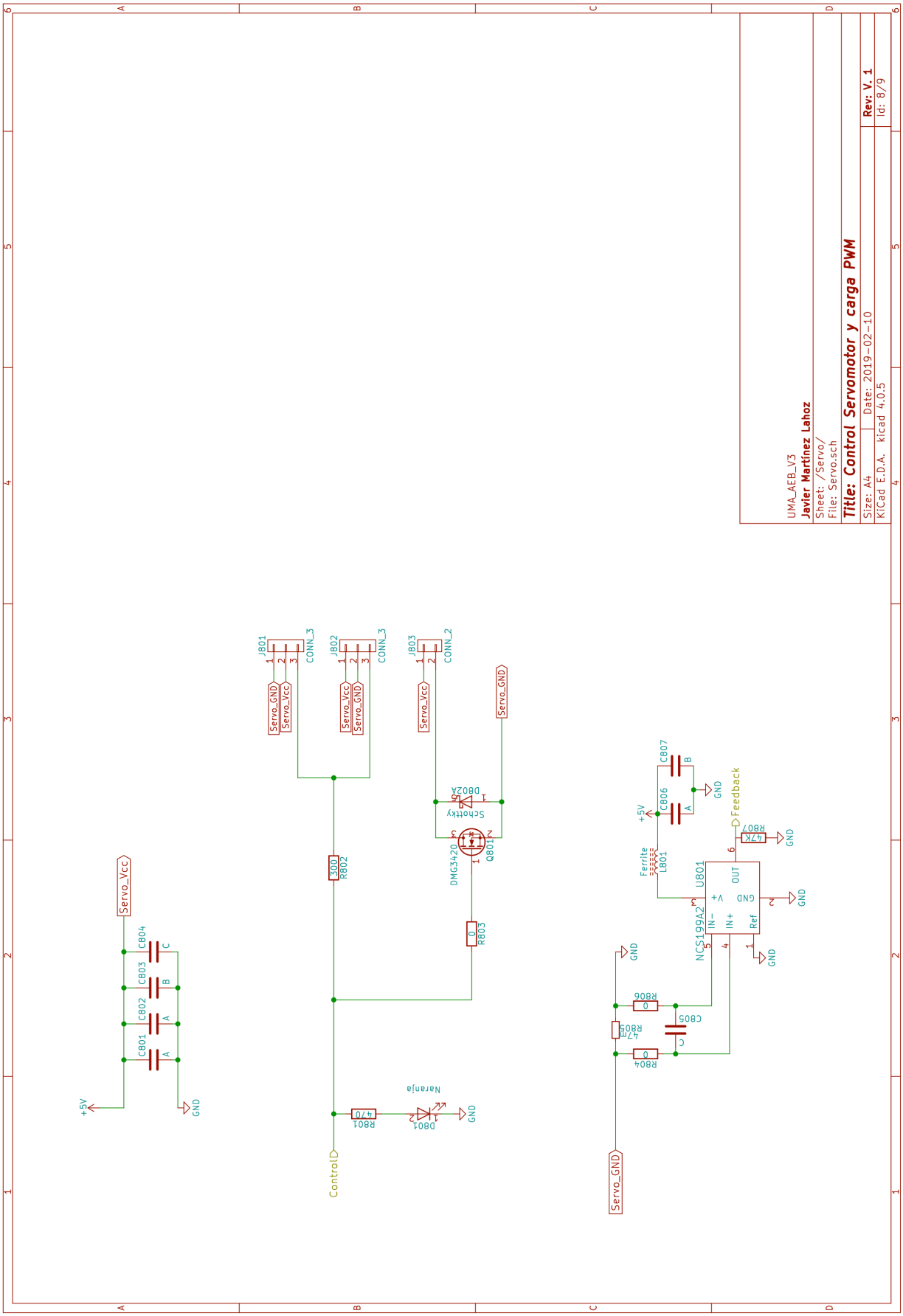

UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /DAC/
File: DAC.sch
Title: Convertidor Digital a Analógico
Size: A4
Date: 2019-02-10
KiCad E.D.A. kicad 4.0.5
Rev: V. 1
Id: 3/9
U501
MCP4725
Vdd
Vout
SDA
SCL
A0
Vss
OUT
GND
+5V
C501
C502
SDA
SCL

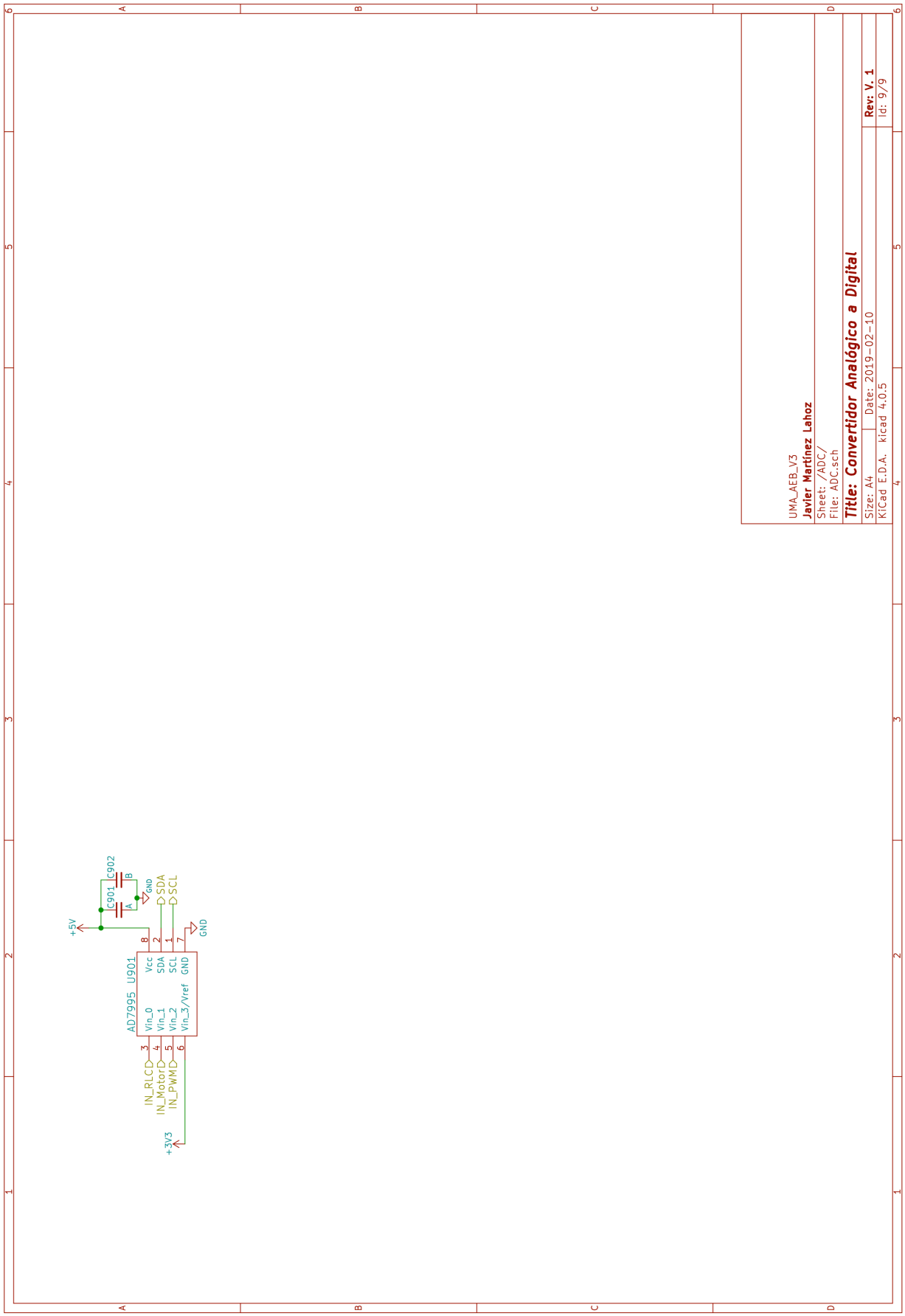

UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /RLC/
File: RLC.sch
Title: Filtros RC y RLC
Size: A4
Date: 2019-02-10
Rev: V. 1
KiCad E.D.A. kicad 4.0.5
Id: 4/9
OUT
IN
U602
NCV20081
U601
PCA9554BS
U603
U604
U605
U606
U607
U608
U609
U610
TS5A3166


UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /OpAmp15/
File: OpAmp15.sch
Title: Interfaz de motor externo
Size: A4
Date: 2019-02-10
Rev: V. 1
KiCad E.D.A. kicad 4.0.5
Id: 5/9


UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /UserIO/
File: UserIO.sch
Title: Interfaz de usuario
Size: A4
Date: 2019-02-10
KiCad E.D.A. kicad 4.0.5
Rev: V. 1
Id: 6/9
S301
KCSC02
K
Segm_A
Segm_B
Segm_C
Segm_D
Segm_E
Segm_F
Segm_G
SegmDP
K
GND
R316 R315 R314 R313 R312 R311 R310 R309
220
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
Trimmer
C303
NF
R323
0
R321
R322
RV301
20K
VCC
ButtonB
R320
4.7K
R319
SW_Push
SW302
C302
ButtonA
R318
R317
SW301
C301
D308 D307 D306 D305 D304 D303 D302 D301
Red
R308 R307 R306 R305 R304 R303 R302 R301

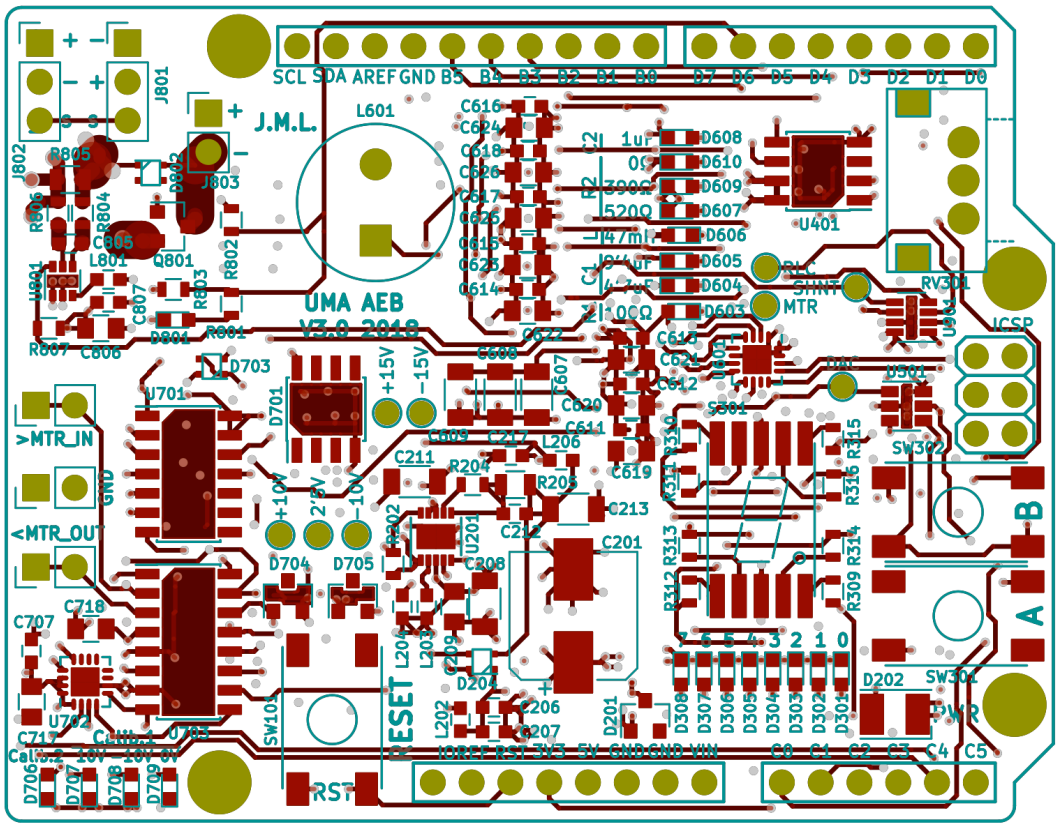

UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /SRAM/
File: SRAM.sch
Title: Memoria SRAM
Size: A4
Date: 2019-02-10
KiCad E.D.A. kicad 4.0.5
Rev: V. 1
Id: 7/9
C402
C401
B
A
+5V
GND
SCK
SI
R404
R405
300
U401
23LC512
CS
SO/SIO1
SIO2
Vss
Vcc
HOLD/SIO3
SCK
SI/SIO0
R401
4.7K
R402
R403
CS
SO


UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /Servo/
File: Servo.sch
Title: Control Servomotor y carga PWM
Size: A4
Date: 2019-02-10
KiCad E.D.A. kicad 4.0.5
Rev: V. 1
Id: 8/9
J801
J802
J803
CONN_3
CONN_2
Servo_GND
Servo_Vcc
ControlD
R802
300
R803
0
R801
470
Naranja
D801
GND
+5V
C801
C802
C803
C804
C806
C807
Feedback
R807
47K
L801
Ferrite
U801
NCS199A2
OUT
GND
Ref
IN+
IN-
R805
4.7
C805
R804
R806
DMG3420
Q801
Schottky
D802A

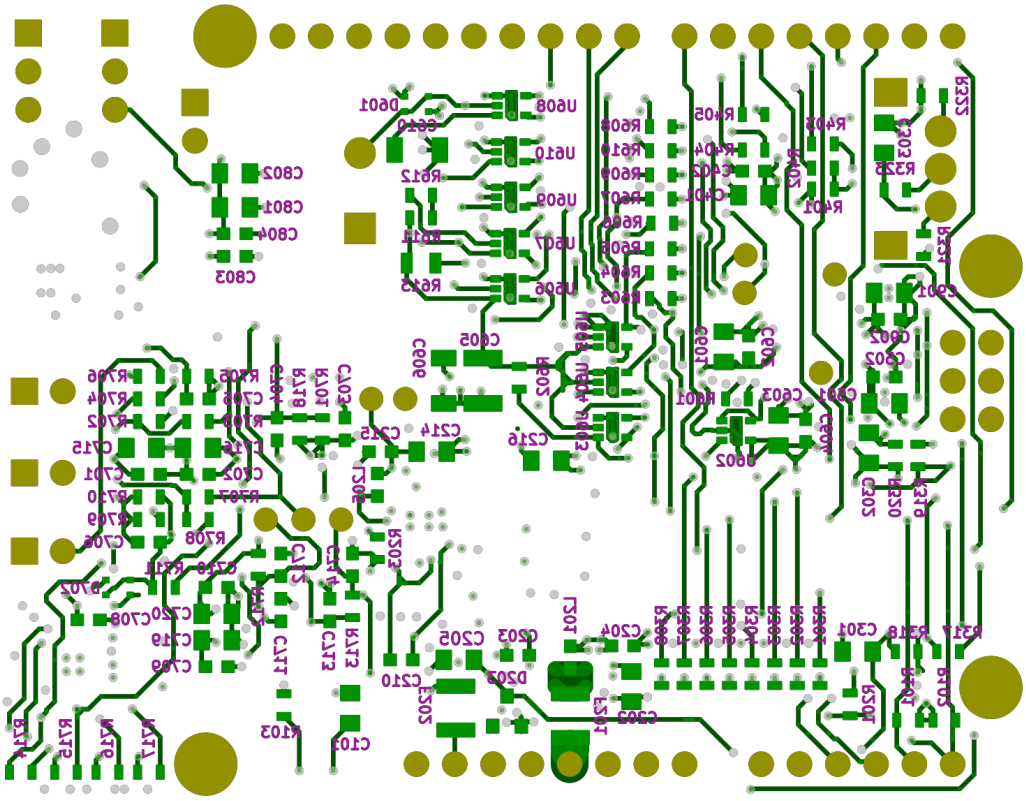

UMA_AEB_V3
Javier Martínez Lahoz
Sheet: /ADC/
File: ADC.sch
Title: Convertidor Analógico a Digital
Size: A4
Date: 2019-02-10
KiCad E.D.A. kicad 4.0.5
Rev: V. 1
Id: 9/9
+5V
C901
C902
A
B
GND
SDA
SCL
GND
AD7995 U901
Vcc
SDA
SCL
GND
Vin_0
Vin_1
Vin_2
Vin_3/Vref
IN_RLC
IN_Motor
IN_PWM
+3V3

12.2 Placa de circuito impreso (PCB)

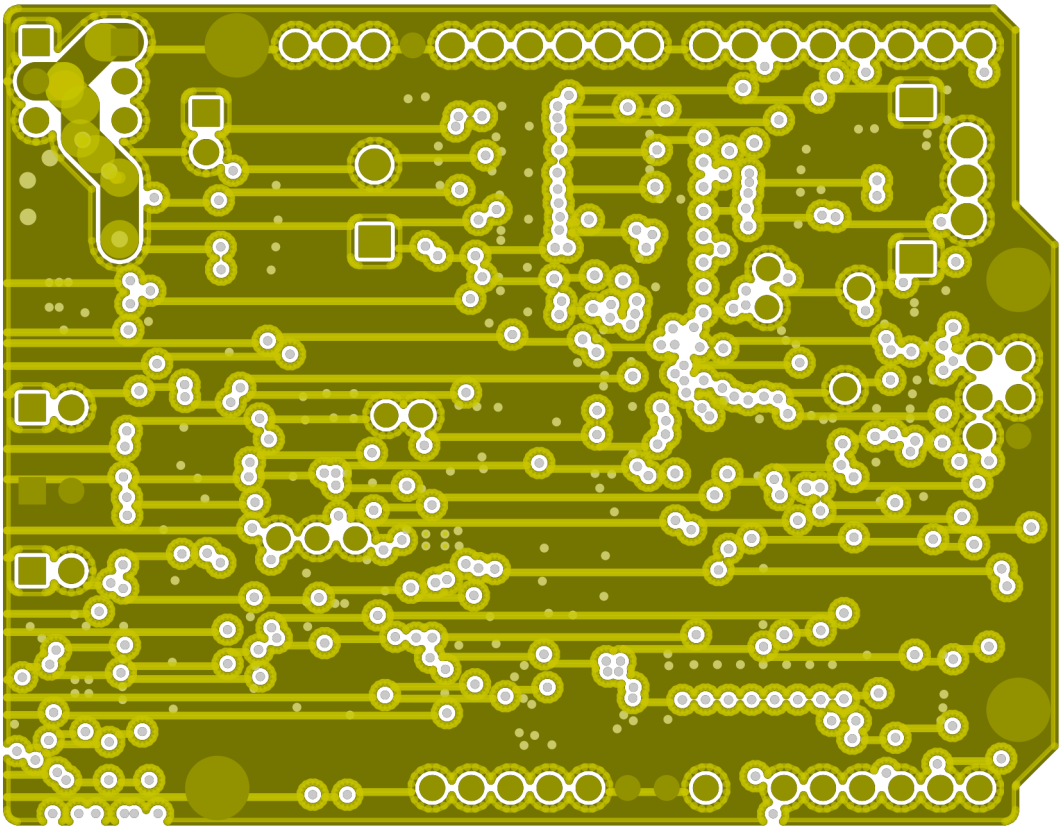
Capa superior (Cobre y serigrafía)

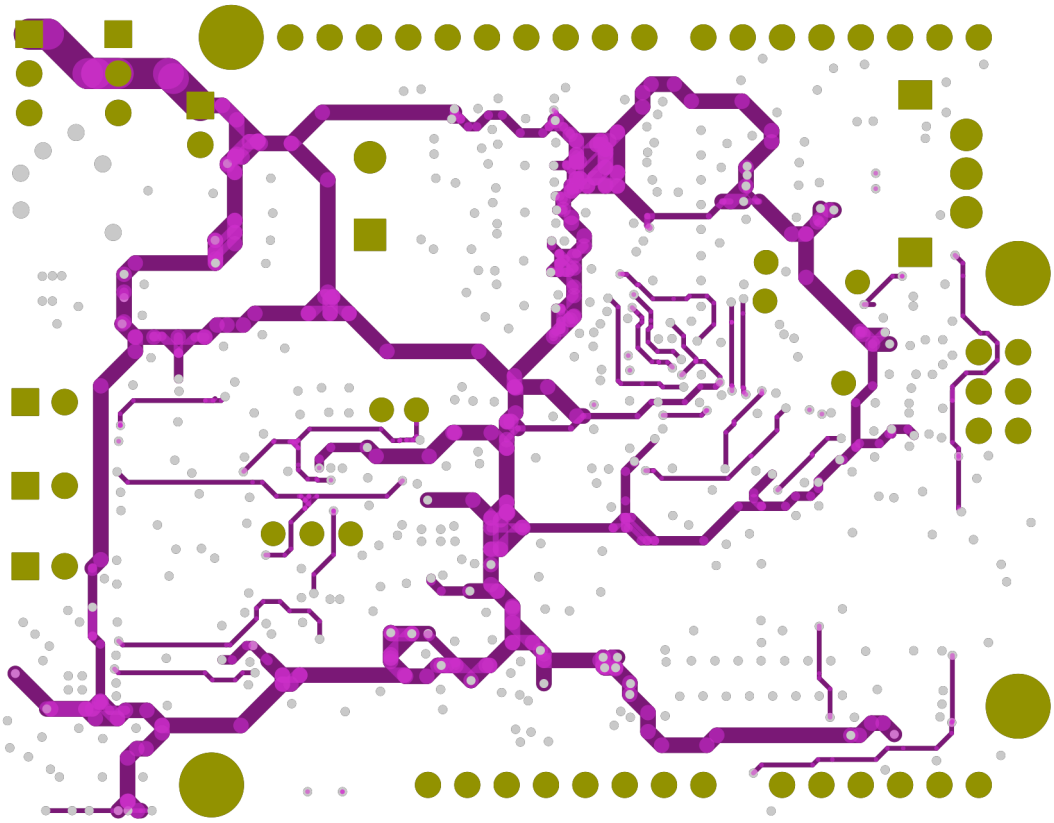


Capa inferior (Cobre y serigrafía)

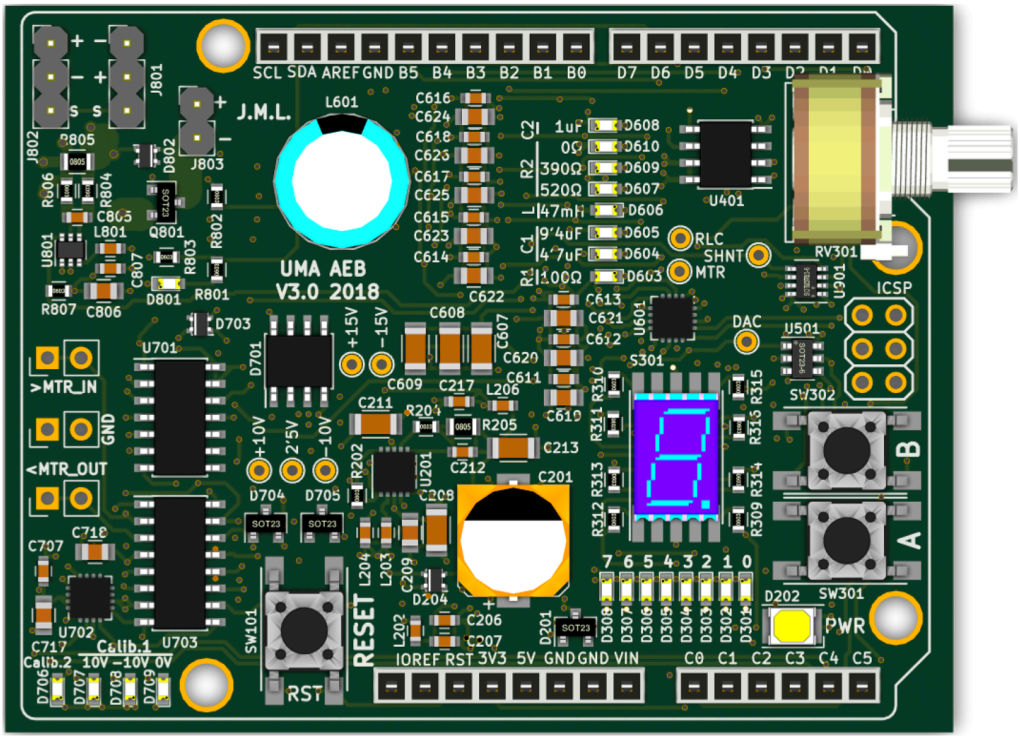


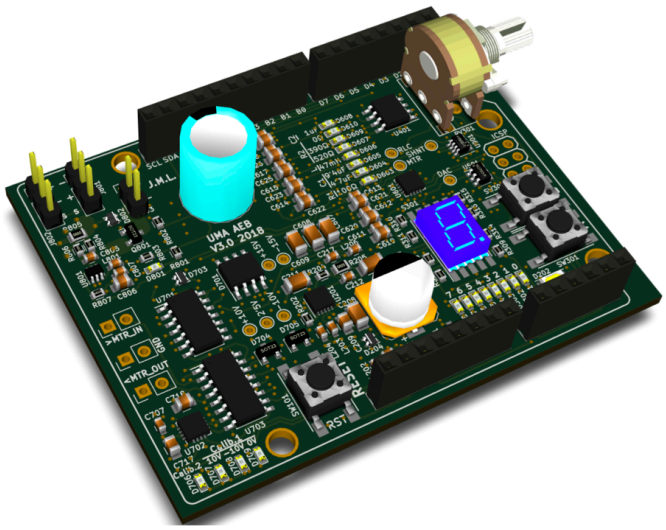
Capas internas (GND y Vcc)

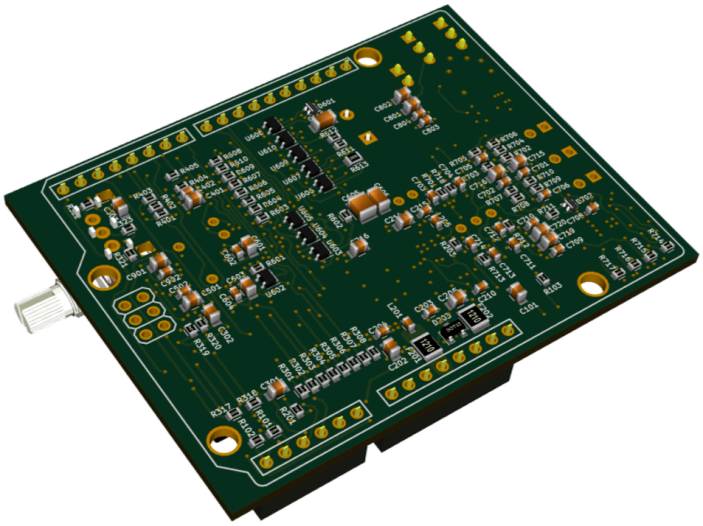




Modelo 3D

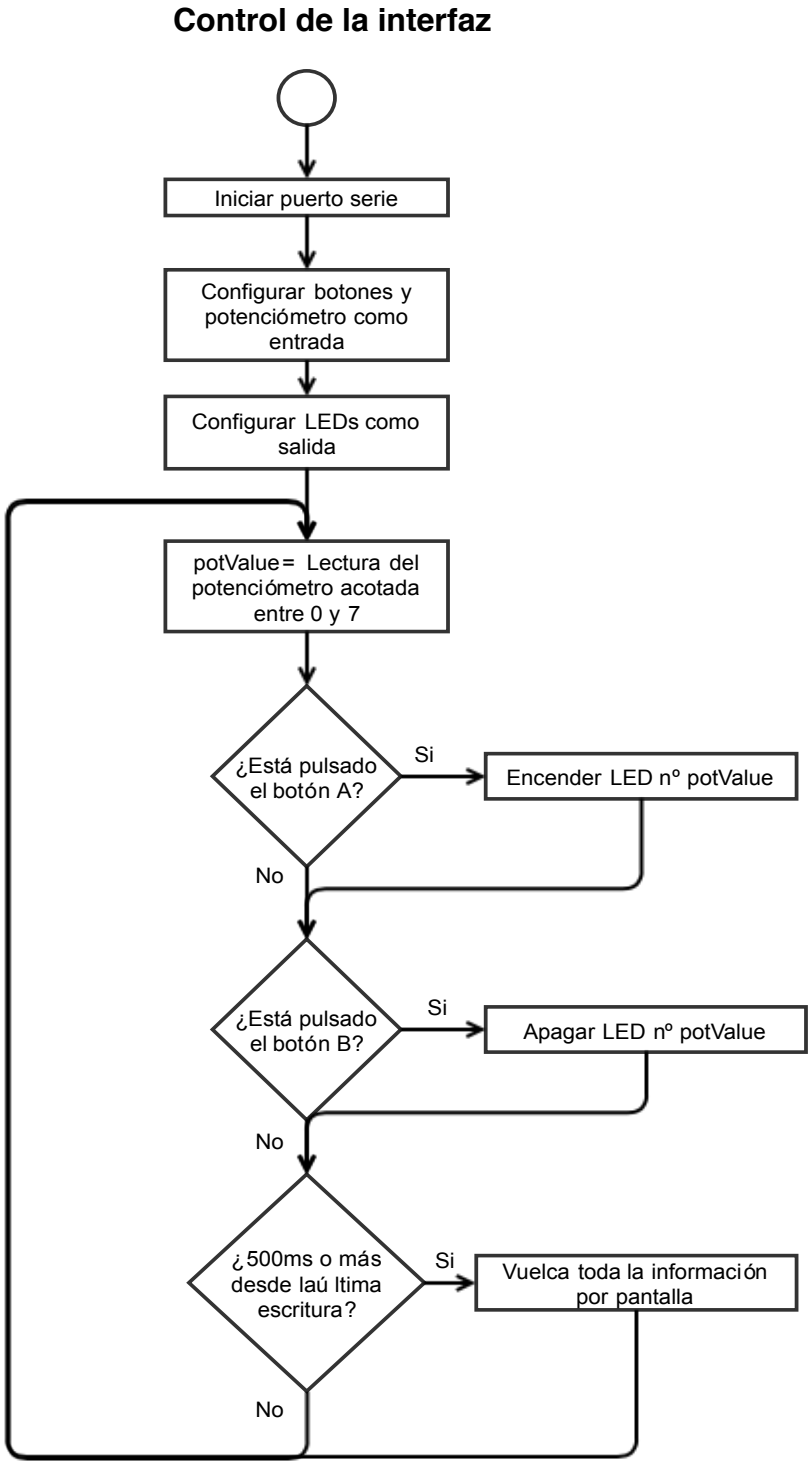

UMA AEB
V3.0 2018
J.M.L.
RESET
RST
PWR
ICSP
DAC
RLC
SHNT
MTR
>MTR_IN
<MTR_OUT
GND
SCL SDA AREF GND B5 B4 B3 B2 B1 B0 D7 D6 D5 D4 D3
IOREF RST 3V3 5V GND GND VIN C0 C1 C2 C3 C4 C5



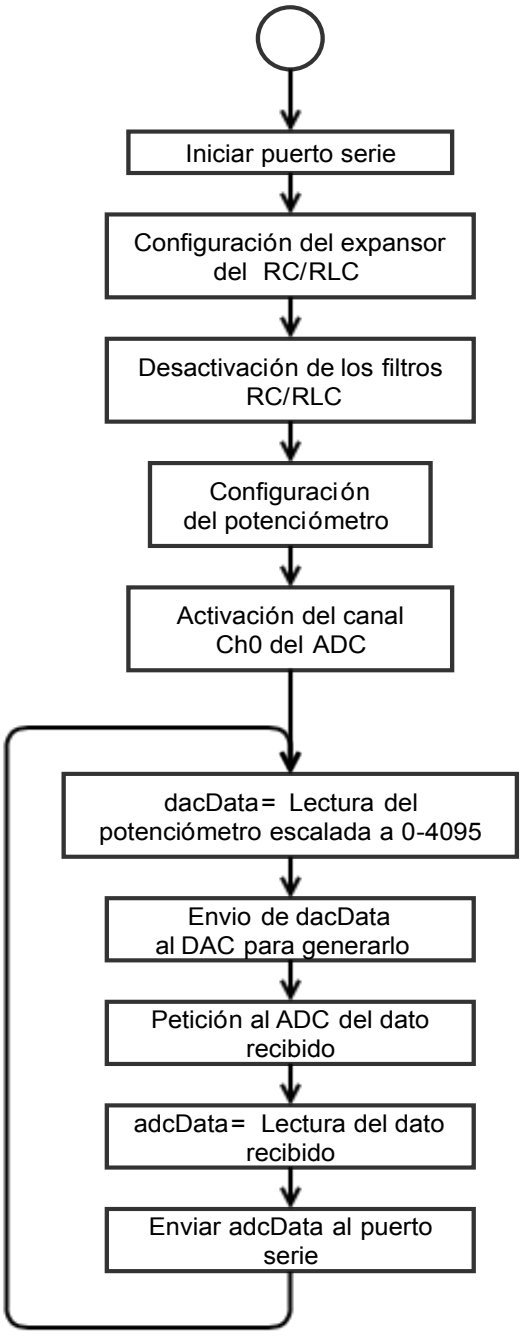


12.3 Software

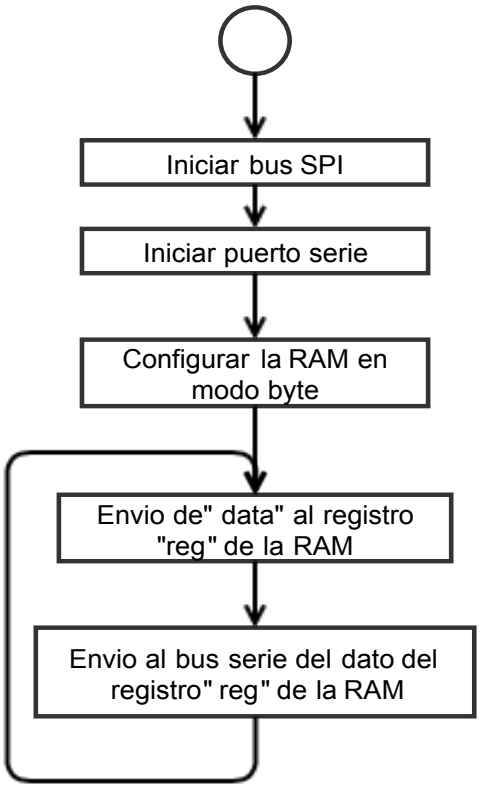
Ejemplos



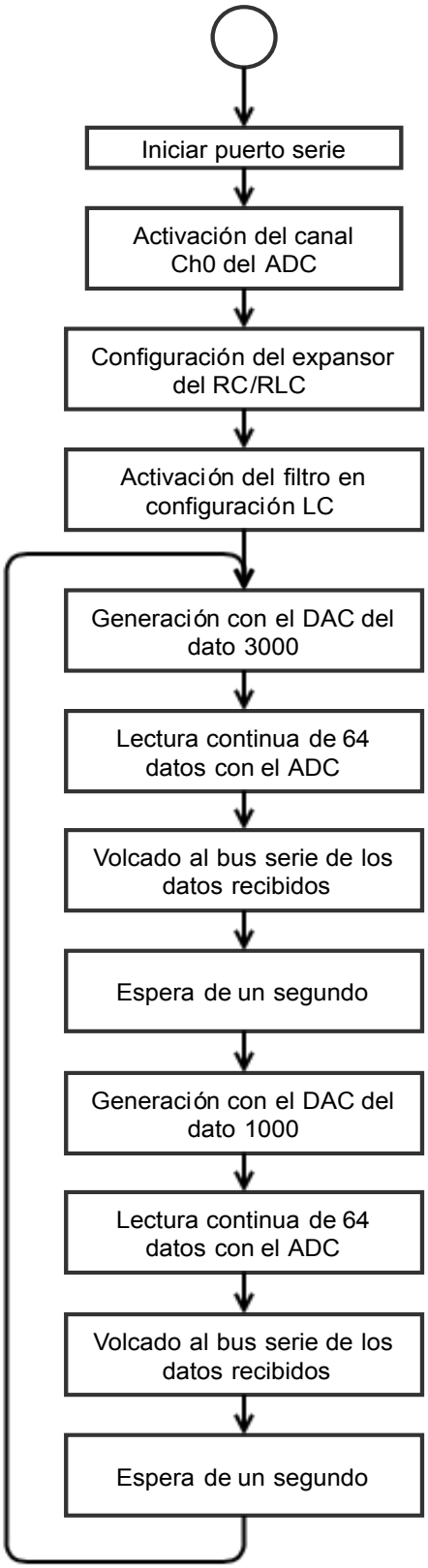
Uso de los convertidores ADC y DAC



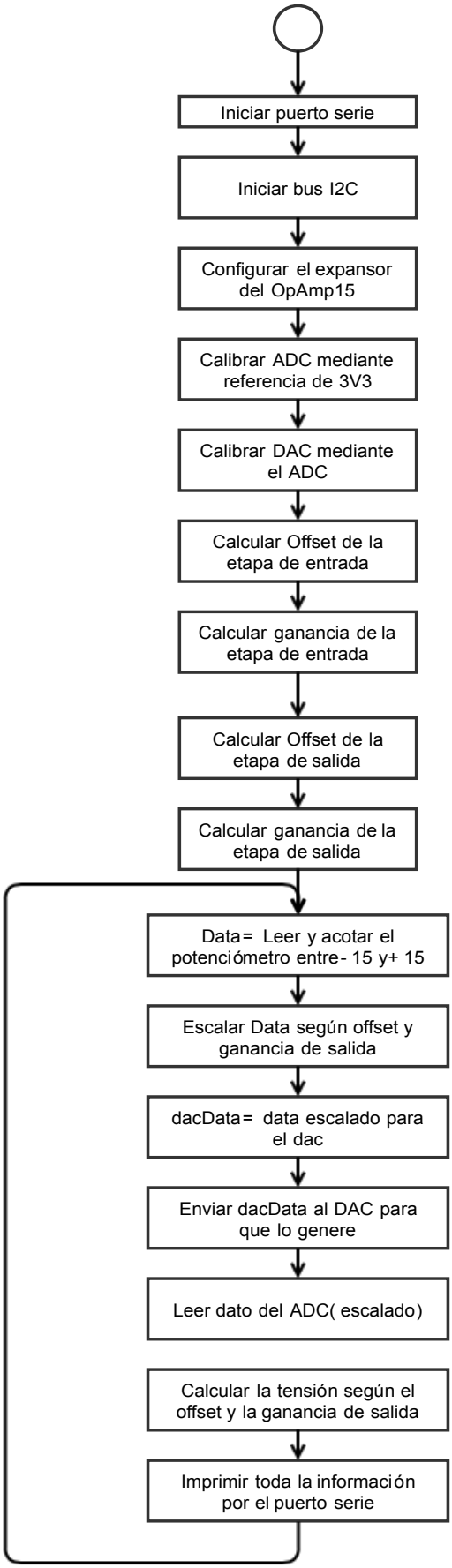
Uso de la memoria RAM



Configuración y estudio de los filtros pasivos



Control del motor externo



Código de la librería - Archivo de definiciones (.h)

```
#ifndef UMA_AEB_V3_h
#define UMA_AEB_V3_h
#include "Arduino.h"
#include <Wire.h>
#include <SPI.h>

/* Estructura que permite almacenar la información del registro del dac */
   struct dacReadStructType{
     bool ready;  //Indica si se esta escribiendo en la EEPROM
     bool por; //Power-ON-Reset
     byte Register_PowerDown;  //Bits de PD del registro
     int Register_Data;    //Bits de dato de salida del registro
     byte EEPROM_PowerDown;  //Bits de PD de la EEPROM
     int EEPROM_Data;    //Bits de dato de salida de la EEPROM
   };

/* A continuacion se definen los pines que controlan el interface:
   Se definen fuera de la clase para poder acceder a ellos en todo momento  */
    #define LED0 5
    #define LED1 2
    #define LED2 4
    #define LED3 9
    #define LED4 7
    #define LED5 8
    #define LED6 16
    #define LED7 17
    #define BUTT_A 15
    #define BUTT_B 3
    #define POT 14
    #define SERV 6

    #define SRAM_CS 10

class UMA_AEB_Class
{
  public:
    Shield_Class(void);  //Constructor de la clase, no tiene codigo propio

    void IOconfig(void);  //Configuracion de los pines necesarios para el control de interface
    void IOtest(bool _printed = false);    //Funcion probar el interface en su totalidad

    void dacTest(int step = 128, int time = 1000);   //  Prueba del DAC
    void dacFastData(unsigned int dacData);   //  Modo Fast del dac, se escribe unicamente el dato deseado
    void dacFastVoltage(float dacVoltage);   // Modo Fast del dac, se escribe unicamente el voltaje deseado
    void dacNormalData(unsigned int dacData, bool writeEEPROM = false);  // Modo Normal del dac (Dato)
    void dacNormalVoltage(float dacVoltage, bool writeEEPROM = false);  // Modo Normal del dac (Voltaje)
    void dacPowerDown(byte resistor = 0x4, bool writeEEPROM = false); // Modo ahorro de energía del dac
    void dacReadStatus(bool dataPrint = false); //Esta funcion lee toda la info del dac y la guarda en la estructura dacReadStruct

    void adcConfig(bool _CH0, bool _CH1, bool _CH2, bool _CH3, bool _ref = false);  // Configurar el ADC con datos individuales
    void adcConfig(int configData); // Configurar el ADC con los datos ya en conjunto
    float adcRead(bool format = 0, int* ch=0, bool _print = false);  //Esto sirve para leer del adc
    void adcContRead(int* DataArray, int readsNumber, float* timeArray=0); //Esta función realiza medidas consecutivas
    void adcFastRead(int *_outputArray, float *_timeArray, int _dataNumber = 1); //esto sirve para realizar medidas consecutivas

    void filterInit(void); //Esto hace la configuración inicial necesaria para emplear el expansor de los circuitos RC y RLC
```

```
        void filterConfig(bool _R1, int _C1, bool _L, int _R2, bool _C2); //Esto reconfigura los filtros mediante parámetros discretos
        void filterConfig(short int _config);//Esto reconfigura los filtros mediante un único parámetro

        void opAmpInit(void); //Inicialización del AO15
        void opAmpConfig(short int calib_mode);  //Configuración del OA15, usar parámetros predefinidos
        void opAmpCalib(void);
        float opAmpIn(void);
        void opAmpOut(float _data);

        void sramConfig(byte sramModeRegister, byte mode);  //Este método sirve para configurar la SRAM
        void sramBegin(unsigned int sramRegister, short int opMode); //Este método sirve para abrir el bus y configurar la comunicacion
        void sramEnd(void); //Este método sirve para cerrar el bus
        byte sramRead(unsigned int sramRegister);  // Esto se usa para leer un único byte de la SRAM
        void sramContRead(unsigned int sramRegister, byte* dataArray, short int datanumbers); // Lecturas continuas a un array
        void sramWrite(unsigned int sramRegister, byte data);  // Esto se usa para escribir un único byte en la SRAM
        void sramContWrite(unsigned int sramRegister, byte* dataArray, short int datanumbers); // Escrituras continuas desde un array

        void testServo(void);

        void conversorsTests(void); // Método de prueba de DAC y ADC en serie
        void conversorsCalib(void); //Método de calibración de los conversores empleando la referencia 3V3

        void generalReset(void);   // Reiniciar todos los elementos del I2C
        void generalWakeUp(void);  // Despertar todos los elementos del I2C

        dacReadStructType dacReadStruct;

        const short int LedArray[8]={LED0,LED1,LED2,LED3,LED4,LED5,LED6,LED7};

      // constantes para la configuración del dac
        const short int DacAddr = 0x62; //Dirección I2C del DAC
        const short int DAC_PullDown_1K = 0x2;   // Se miden 4K
        const short int DAC_PullDown_100K = 0x4; // Se miden entre 190K y 1M1
        const short int DAC_PullDown_500K = 0x6; // Se miden 2M2
        float dacScale = 819.2;   // Escala base del DAC para el paso de tensión a dato

        // constantes para la configuración del adc
        const short int AdcAddr = 0x29;   //Dirección I2C del ADC
        const short int Channel0 = 0x10;  //Activación de los canales
        const short int Channel1 = 0x20;
        const short int Channel2 = 0x40;
        const short int Channel3 = 0x80;
        const short int RefSel = 0x08;   // Selección de la referencia para la lectura
        const short int ADCFilter = 0x04; // Filtro del bus I2C en el ADC
        const short int BitTrialD = 0x02; // Esto desactiva el Bit Trial Delay
        const short int SampleD = 0x01;   // Esto desactiva el Sample Delay
        float adcScale = 0.00488281;   // Escala base del ADC para el paso de dato a tensión

        // constantes para la configuración ambos expansores E/S
        const short int InputPort = 0;
        const short int OutputPort = 1;
        const short int Polarity = 2;
        const short int Config = 3;

        // constantes para la configuración de los filtros RC/RLC
        const short int FilterAddr = 0x20;
        const short int Filter0 = 0x09;
        const short int FilterRC1 = 0x08;
        const short int FilterRC2 = 0x0E;
        const short int FilterRL = 0x41;
```

```
    const short int FilterLC = 0xA1;
    const short int FilterRLC = 0x61;
    const short int FilterRCRLC = 0x60;

    // constantes para la configuración de los AO15,
    //hay que tener en cuenta que estos AS tienen lógica negativa
    const short int OpAmpAddr = 0x27;
    const short int OpAmp_No_Calib = 0xFF;
    const short int OpAmp_Calib_2 = 0xFE;
    const short int OpAmp_Calib_1_pos = 0xFD;
    const short int OpAmp_Calib_1_neg = 0xFB;
    const short int OpAmp_Calib_1_0V = 0xF7;
    float gainOut=1.0/6.0;
    float offsetOut=2.5;
    float gainIn=6;
    float offsetIn=-2.5;

     //Constantes para el control de la SRAM
    const short int SRAMreadMode = 0x03;
    const short int SRAMwriteMode = 0x02;
    const short int SRAMreadRegister = 0x05;
    const short int SRAMwriteRegister = 0x01;
    const short int SRAMbyteMode = 0x00;
    const short int SRAMpageMode = 0x80;
    const short int SRAMseqMode = 0x40;

};
#endif
```

Código de la librería - Archivo de implementación (.cpp)

```
#include "Arduino.h"
#include "UMA_AEB_V3.h"

UMA_AEB_Class::Shield_Class(void){   //Constructor de la clase, no tiene codigo propio
}

void UMA_AEB_Class::IOconfig(void){    //Configuracion de los pines necesarios para el control de interface
  for(int i=0;i<8;i++) pinMode(LedArray[i], OUTPUT);  //Definir todos los LEDs como salidas.
  pinMode(BUTT_B, INPUT);                             //Definir los botones y el pot como entradas.
  pinMode(BUTT_A, INPUT);
  pinMode(POT, INPUT);
}

void UMA_AEB_Class::IOtest(bool _printed){    //Funcion probar el interface en su totalidad
  static int _timer;
  int potValue=analogRead(POT)/128;       //Leer el valor del pot y acotarlo en un entero entre 0 y 7.
  if(digitalRead(BUTT_A)==HIGH)               // Si el boton A esta pulsado
     digitalWrite(LedArray[potValue], HIGH);    // encender el LED indicado por el pot.
  if(digitalRead(BUTT_B)==HIGH)               // Si el boton B esta pulsado
     digitalWrite(LedArray[potValue], LOW);     // apagar el LED indicado por el pot.

      if(_printed == true && Serial == true && (millis()-_timer)>500){  // Esto muestra por pantalla los datos cada
500ms
        Serial.println();
        Serial.print("Botón A:  "); Serial.println(digitalRead(BUTT_A));
        Serial.print("Botón B:  "); Serial.println(digitalRead(BUTT_B));
        Serial.print("POT:  "); Serial.println(analogRead(POT));
        Serial.print("Valor:  "); Serial.println(potValue);
        Serial.print("LEDs:  "); for(int i=0;i<8;i++) Serial.print(digitalRead(LedArray[i]));
        Serial.println();
        _timer=millis();    // Toma de tiempo
       }
```

```
}

void UMA_AEB_Class::dacTest(int step, int time){    //  Prueba del DAC donde va aumentando automática-
mente su valor en incrementos de 128 cada medio segundo
  static int dacData = 0;
  static int _timer;
  if((millis()-_timer)>time){  // Comprobación de tiempo
    dacData=(dacData+step)&0xFFF;   // Aumento del dato de salida
    dacFastData(dacData);       // Actualización del dato de salida
    _timer=millis();   // Toma de tiempo
  }
}

void UMA_AEB_Class::dacFastData(unsigned int dacData){   //  Modo Fast del dac, se escribe unicamente el
dato deseado
  if(dacData>4095)dacData=4095;        //Comprobación de que el dato esta en los valores admisibles
  if(dacData<0)dacData=0;
  Wire.beginTransmission(DacAddr);   // Inicio de comunicación I2C
  Wire.write(dacData >> 8);          //Envio de los datos
  Wire.write(dacData & 0xFF);
  Wire.endTransmission();            // Fin de comunicación I2C
}

void UMA_AEB_Class::dacFastVoltage(float dacVoltage){    // Modo Fast del dac, se escribe unicamente el
voltaje deseado
  dacVoltage*=dacScale;   //Convertir la tensión a dato
  dacFastData((int)dacVoltage/1);  //Eliminar los decimales y paso a int para usar la función dacFastData
}

void UMA_AEB_Class::dacNormalData(unsigned int dacData, bool writeEEPROM){   // Modo Normal del dac
(Dato), se puede escribir en la EEPROM
  if(dacData>4095)dacData=4095;     //Comprobación de que el dato esta en los valores admisibles
  if(dacData<0)dacData=0;
  Wire.beginTransmission(DacAddr);     // Inicio de comunicación I2C
  if (writeEEPROM)  Wire.write(0x60);   // Escritura modo (modo normal + bit de guardado EEPROM)
  else              Wire.write(0x40);    //  Escritura modo (modo normal)
  Wire.write(dacData >> 4);                //Envio de los datos
  Wire.write((dacData & 0x0F) << 4);
  Wire.endTransmission();
}

void UMA_AEB_Class::dacNormalVoltage(float dacVoltage, bool writeEEPROM){    // Modo Normal del dac
(Voltaje), se puede escribir en la EEPROM
  dacVoltage*=dacScale;      //Convertir la tensión a dato
   dacNormalData(dacVoltage,writeEEPROM);         //Eliminar los decimales y paso a int para usar la función
dacNormalData
}

void UMA_AEB_Class::dacPowerDown(byte resistor, bool writeEEPROM){      // Modo ahorro de energía del
dac, se puede escribir en la EEPROM
   if (resistor != DAC_PullDown_1K && resistor != DAC_PullDown_100K && resistor != DAC_PullDown_500K)
resistor = DAC_PullDown_100K;
    Wire.beginTransmission(DacAddr);
    if (writeEEPROM)  Wire.write(0x60 + resistor);  //Aqui se añade al mensaje el valor de la resitencia de Pull-
Down
   else              Wire.write(0x40 + resistor);  // Su puede guardar en la EEPROM (if) o no (else)
    Wire.write(0);    //Estos dos datos son el dato que se guarda en el registro
    Wire.write(0);    //Se pone a 0 por defecto, no debería afectar
   Wire.endTransmission();
}

void UMA_AEB_Class::dacReadStatus(bool dataPrint){   //Esta funcion lee toda la info del dac y la guarda en
la estructura dacReadStruct
  byte readBytes[5];
  Wire.requestFrom(DacAddr, 5);  //Petición de los datos
```

```
    for(int i=0;i<5;i++) readBytes[i]=Wire.read();   //Lectura de los dats
    dacReadStruct.ready=readBytes[0] & 0x80;    //Esta a 0 si esta escribiendo en la EEPROM
    dacReadStruct.por=readBytes[0] & 0x40;      //Power On Reset
    dacReadStruct.Register_PowerDown=(readBytes[0]>>1) & 0b11;  //Bits de Power Down en el registro
    dacReadStruct.Register_Data=((int)readBytes[1]<<4)+((int)readBytes[2]>>4);  //Bits de datos en el registro
    dacReadStruct.EEPROM_PowerDown=(readBytes[3]>>5) & 0b11;   //Bits de Power Down en la EEPROM
     dacReadStruct.EEPROM_Data=(((int)(readBytes[3]& 0x0F)<<8)+readBytes[4]);   //Bits de datos en la EE-
PROM
    if(dataPrint == true && Serial == true){
     Serial.println();
     Serial.print("Ready: ");Serial.println(dacReadStruct.ready);
     Serial.print("POR: ");Serial.println(dacReadStruct.por);
     Serial.print("Register PD: ");Serial.println(dacReadStruct.Register_PowerDown);
     Serial.print("Register Data: ");Serial.println(dacReadStruct.Register_Data);
     Serial.print("EEPROM PD: ");Serial.println(dacReadStruct.EEPROM_PowerDown);
     Serial.print("EEPROM Data: ");Serial.println(dacReadStruct.EEPROM_Data);
    }
   }

   void UMA_AEB_Class::adcConfig(bool _CH0, bool _CH1, bool _CH2, bool _CH3, bool _ref){  // Configurar el
ADC con datos individuales de cada canal
    Wire.beginTransmission(AdcAddr);
     Wire.write((_CH3<<7)+(_CH2<<6)+(_CH1<<5)+(_CH0<<4)+(_ref<<3)); //Cada canal se manda a su bit co-
rrespondiente
    Wire.endTransmission();
   }

   void UMA_AEB_Class::adcConfig(int configData){  // Configurar el ADC con los datos ya en conjunto
    Wire.beginTransmission(AdcAddr);
    Wire.write(configData);   //Hay que emplear los datos publicos Channel0-Channel3
    Wire.endTransmission();
   }

   float UMA_AEB_Class::adcRead(bool format, int* ch, bool _print){  //Esto sirve para leer del adc
    Wire.requestFrom(AdcAddr, 2);  // petición de dos bytes de información por I2C
    byte _x1 = Wire.read(); // Byte alto
    byte _x2 = Wire.read(); // Byte bajo
    unsigned int _output = ((_x1&0x0F)<<6)+(_x2>>2);  // Composición del dato a partir de los dos bytes
    if(ch!=0) (*ch)=_x1>>4; //En esta parte se guarda el canal de lectura mediante el puntero
    if(_print == true && Serial == true){      //Esto sirve para mostrar por pantalla toda la información en casos de
verificación
    Serial.print("Canal: ");
    Serial.print(_x1>>4);
    Serial.print("\t Dato: ");
    Serial.print(_output);
    Serial.print("\t Tensión: ");
    Serial.println(_output*adcScale,3);
    }
    if (format == 0)return(_output);
    else return (_output*adcScale);
   }

   void UMA_AEB_Class::adcContRead(int* DataArray, int readsNumber, float* timeArray){  //Esto sirve para leer
del adc
    byte _x1,_x2;
    for(int i=0 ; i < readsNumber; i++){
    Wire.requestFrom(AdcAddr, 2);  // petición de dos bytes de información por I2C
    _x1 = Wire.read(); // Byte alto
    _x2 = Wire.read(); // Byte bajo
    DataArray[i] = (((_x1&0x0F)<<6)+(_x2>>2));  // Composición del dato a partir de los dos bytes
    if (timeArray) timeArray[i]=micros();
    }
   }
```

```
void UMA_AEB_Class::adcFastRead(int *_outputArray, float *_timeArray, int _dataNumber){  //esto sirve para realizar medidas consecutivas rapidas
  byte _i=0;
  byte _loop;
  byte _highData[_dataNumber*16];   //Arrays para almacenar rapidamente todos los bytes recibidos
  byte _lowData[_dataNumber*16];
  for(_loop=1; _loop<(_dataNumber+1); _loop++){
    Wire.requestFrom(AdcAddr, 32);  // Petición de 16 lecturas (32 bytes)
    for(; _i < (_loop*16); _i++){
      _highData[_i] = Wire.read(); // guardado de los bytes altos
      _lowData[_i]  = Wire.read(); // guardado de los bytes bajos
      _timeArray[_i] = micros();
    }
  }
  Wire.endTransmission();
    for(int _i=0; _i<(_dataNumber*16); _i++) _outputArray[_i]=((_highData[_i]&0x0F)<<6)+(_lowData[_i]>>2); // Composición de los datos a partir de los arrays
}

void UMA_AEB_Class::filterInit(void){   //Esto hace la configuración inicial necesaria para emplear el expansor de los circuitos RC y RLC
  Wire.beginTransmission(FilterAddr); // Llamada I2C al expansor del filtro
  Wire.write(Config);   // Acceso al registro de configuración
  Wire.write(0x00);     // Todos los bits a 0 = todo salidas
  Wire.endTransmission();

  Wire.beginTransmission(FilterAddr);  // Llamada I2C al expansor del filtro
  Wire.write(OutputPort);    // Acceso al registro de salidas
  Wire.write(Filter0);   // Puentear la resistencia y la bobina, desconectar todos los condensadores
  Wire.endTransmission();
}

void UMA_AEB_Class::filterConfig(bool _R1, int _C1, bool _L, int _R2, bool _C2){    //Esto reconfigura los filtros mediante parámetros discretos
  int _auxFilter=0;
  if(_R1==false) _auxFilter += 0x01; //Si R1 es cierto, se activa la resistencia del RC (R602)
  if(_C1==1)     _auxFilter += 0x02;     //Si C1 es 1 se activa el primer condensador de 4'7uF (C606)
  if(_C1==2)     _auxFilter += 0x04;     //Si C1 es 2 se activan 3 condensadores de 4'7uF (C607, C608, C609)
  if(_C1==3)     _auxFilter += 0x06;     //Si C1 es 3 se activan los 4 condensadores de 4'7uF
  if(_L==false)  _auxFilter += 0x08; //Si L es cierto, se activa la inductancia del RLC (L601)
  else{
    if(_R2==0)   _auxFilter += 0x80;     //Si R2 es 0 no se activa ninguna resistencia
    if(_R2==1)   _auxFilter += 0x40;     //Si R2 es 1 se activa la resistencia de 390 (R613)
    if(_R2==2)   _auxFilter += 0x10;     //Si R2 es 2 se activa la resistencia de 520 (R611 y R612)
    if(_R2==3)   _auxFilter += 0x50;     //Si R2 es 3 se activan en paralelo las dos resistencias anteriores
  }
  if(_C2)        _auxFilter += 0x20;       //Si C2 es cierto se activa el condensador final de 1uF (C610)

  Wire.beginTransmission(FilterAddr); //Esto configura las salidas del exp
  Wire.write(OutputPort);
  Wire.write(_auxFilter);
  Wire.endTransmission();
}

void UMA_AEB_Class::filterConfig(short int _config){  //Esto reconfigura los filtros mediante un único parámetro, se recomienda emplear las constantes predefinidas
  Wire.beginTransmission(FilterAddr);
  Wire.write(OutputPort);
  Wire.write(_config);
  Wire.endTransmission();
}

void UMA_AEB_Class::opAmpInit(void){    //Esto hace la configuración inicial necesaria para emplear el expansor del OpAmp
  Wire.beginTransmission(OpAmpAddr); //Esto configura el Exp como todo salidas
```

```
  Wire.write(Config); // Acceso al registro de configuración
  Wire.write(0x00);   // Todos los bits a 0 = todo salidas
  Wire.endTransmission();

   Wire.beginTransmission(OpAmpAddr); //Esto configura las salidas del exp como HIGH (desactivando los
switches)
  Wire.write(OutputPort);       // Acceso al registro de salidas
  Wire.write(OpAmp_No_Calib);     //Abrir todos los AS
  Wire.endTransmission();
 }

 void UMA_AEB_Class::opAmpConfig(short int calib_mode){      //Esto configura las salidas del exp según la
calibracion deseada
  Wire.beginTransmission(OpAmpAddr);
  Wire.write(OutputPort);       // Acceso al registro de salidas
  Wire.write(calib_mode);       // Reconfigurar todos los AS
  Wire.endTransmission();
 }

 void UMA_AEB_Class::opAmpCalib(void){    //Esto configura las salidas del exp según la calibracion deseada
  float _test=0;
  float value_pos=4;
  float value_neg=1;
  adcConfig(Channel1);

  opAmpConfig(OpAmp_Calib_1_0V);
  for (int _i=0;_i<10; _i++)adcRead();
  offsetIn=-(adcRead(true));

  opAmpConfig(OpAmp_Calib_1_pos);
  for (int _i=0;_i<10; _i++)adcRead();
  float value_1=adcRead(true);
  opAmpConfig(OpAmp_Calib_1_neg);
  for (int _i=0;_i<10; _i++)adcRead(true);
  float value_2=adcRead(true);

  gainIn=20/(value_1-value_2);

  opAmpConfig(OpAmp_Calib_2);
  do{
   dacFastVoltage(offsetOut);
   for (int _i=0;_i<10; _i++)adcRead(true);
   offsetOut=5-adcRead(true);
  }while(abs(opAmpIn())>0.1);

  do{
   dacFastVoltage(value_pos);
   for (int _i=0;_i<10; _i++)adcRead(true);
   value_pos=8-adcRead(true);
  }while(abs(opAmpIn()-9)>0.1);


 do{
  dacFastVoltage(value_neg);
  for (int _i=0;_i<10; _i++)adcRead(true);
  value_neg=2-adcRead(true);
 }while(abs(opAmpIn()+9)>0.1);


  gainOut=(value_pos-value_neg)/18;
  opAmpConfig(OpAmp_No_Calib);

 }

 float UMA_AEB_Class::opAmpIn(void){    //Esto configura las salidas del exp según la calibracion deseada
```

```
  return((adcRead(true)+offsetIn)*gainIn);
}

void UMA_AEB_Class::opAmpOut(float _data){      //Esto configura las salidas del exp según la calibracion deseada
  dacFastVoltage((_data*gainOut)+offsetOut);
}

void UMA_AEB_Class::sramConfig(byte sramModeRegister, byte mode){  //Este método sirve para configurar los registros de configuración de la SRAM
   SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0));  // Configuración de los parámetros del SPI, ocupación del bus
  digitalWrite(SRAM_CS, LOW);  // Activacion del pin que indica a la SRAM que va a comenzar la comunicacion
  SPI.transfer(sramModeRegister); // Acceso con el modo deseado deseado
  SPI.transfer(mode);            // Modificación del modo
  digitalWrite(SRAM_CS, HIGH);    // Indicar a la SRAM que la comunicación ha acabado
  SPI.endTransaction();     //Liberación del bus SPI
}

void UMA_AEB_Class::sramBegin(unsigned int sramRegister, short int opMode){     //Este método sirve para abrir el bus y configurar la siguiente comunicacion
//   SPI.beginTransaction(SPISettings(20000000, MSBFIRST, SPI_MODE0)); // Configuración de los parámetros del SPI, ocupación del bus
  digitalWrite(SRAM_CS, LOW);   // Activacion del pin que indica a la SRAM que va a comenzar la comunicacion
  SPI.transfer(opMode);   //Acceso al modo deseado
  SPI.transfer((byte)(sramRegister >> 8));   //Acceso al registro deseado, parte 1
  SPI.transfer((byte) sramRegister);             //Acceso al registro deseado, parte 2
}

void UMA_AEB_Class::sramEnd(void){   //Este método sirve para cerrar el bus
  digitalWrite(SRAM_CS, HIGH);      // Indicar a la SRAM que la comunicación ha acabado
  //SPI.endTransaction();        //Liberación del bus SPI

}

byte UMA_AEB_Class::sramRead(unsigned int sramRegister){   // Esto se usa para leer un único byte de la SRAM
  sramBegin(sramRegister, SRAMreadMode);
  byte data = SPI.transfer(0x00);           // Recepción del dato
  sramEnd();
  return(data);            //Devolución del dato recibido
}

void UMA_AEB_Class::sramContRead(unsigned int sramRegister, byte* dataArray, short int datanumbers){ // Esto se usa para realizar medidas continuas a un array
  sramBegin(sramRegister, SRAMreadMode);
  for(int i = 1 ; i < datanumbers ; i ++) *dataArray++ = SPI.transfer(0x00);       // Recepción del dato
  sramEnd();
}

void UMA_AEB_Class::sramWrite(unsigned int sramRegister, byte data){  // Esto se usa para escribir un único byte en la SRAM
  sramBegin(sramRegister, SRAMwriteMode);          //Acceso al registro deseado, parte 2
  SPI.transfer(data);            // Envio del dato
  sramEnd();
}

void UMA_AEB_Class::sramContWrite(unsigned int sramRegister, byte* dataArray, short int datanumbers){  // Esto se usa para realizar escrituras continuas desde un array
  sramBegin(sramRegister, SRAMwriteMode);
  for(int i = 1 ; i < datanumbers ; i ++) SPI.transfer(*dataArray++); // Envio del dato
  sramEnd();
}
```

```
void UMA_AEB_Class::testServo(void){ // Método para probar el Servomotor con el potenciómetro
analogWrite(SERV,map(analogRead(POT),0,1024,0,255));
}

void UMA_AEB_Class::conversorsTests(void){ // Método de prueba de DAC y ADC en serie
  static float _testData=0;
/*
  filterInit();
  adcConfig(Channel0);
*/
   dacFastVoltage(_testData);
   int _readData=adcRead();
   Serial.print("Tensión Generada: "); Serial.print((float)_testData,3);
   Serial.print("\t Tensión recibida: "); Serial.print((float)_readData*adcScale,3);
        Serial.print("\t  Error:  "); Serial.print(((((float)_testData)-((float)_readData*adcScale))/
((float)_testData))*100,3);Serial.print("%");
   Serial.println();
   _testData+=0.01;
   if(_testData>5)_testData=0;
}

void UMA_AEB_Class::conversorsCalib(void){ //Método de calibración de los conversores empleando la refe-
rencia 3V3
  adcConfig(Channel3); //Esto configura el ADC para leer los 3V3
  adcScale = 3.3/adcRead();
  filterInit();
  filterConfig(Filter0);
  dacFastData(3000);
  adcConfig(Channel0);
  for (int _i=0;_i<10; _i++)adcRead(false);
  dacScale=3000.0/(adcRead(true));
}

void UMA_AEB_Class::generalReset(void){   // Reiniciar todos los elementos del I2C
  Wire.beginTransmission(0x00);  // Esto realiza una llamada general
  Wire.write(0x06);              // Mensaje estandar de reinicio
  Wire.endTransmission();
}

void UMA_AEB_Class::generalWakeUp(void){      // Despertar todos los elementos del I2C
  Wire.beginTransmission(0x00);  // Esto realiza una llamada general
  Wire.write(0x09);              // Mensaje estandar de activación
  Wire.endTransmission();
}
```

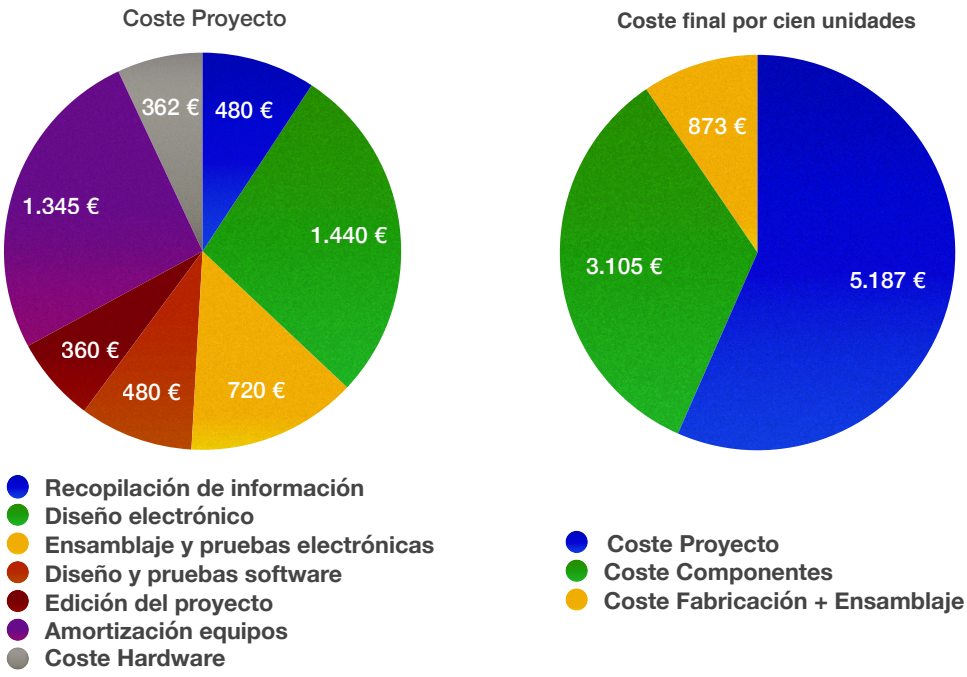
12.4 Análisis de costes

Costes de fabricación del escudo

Tipo	Ctd. total	Valor	Componente	Fabricante	Precio unitario	Precio envio	Componentes
R	100	1M2Ω	CRCW08051M20FKEA	Vishay	0,045 €	4,5 €	R205
	100	1M1Ω	CRCW06031M10FKEA	Vishay	0,034 €	3,4 €	R202
	400	120KΩ	ERJ3RED1203V	Panasonic	0,027 €	10,8 €	R702,R705,R708,R709
	200	100KΩ	CR0603-FX-1003HLF	Bourns	0,028 €	5,6 €	R203,R204
	400	47KΩ	MCR03EZPFX4702	ROHM	0,004 €	1,6 €	R318,R320,R401,R807
	400	20KΩ	MCR03EZPFX2002	ROHM	0,004 €	1,6 €	R703,R704,R707,R710
	400	5K1Ω	MCR03EZPFX5101	ROHM	0,004 €	1,6 €	R701,R718,R712,R713
	300	1KΩ	CRG0603F1K0	TE Connectivity	0,022 €	6,6 €	R101,R102,R601
	1300	470Ω	MCR03EZPFX4700	ROHM	0,003 €	3,9 €	R603-610,R714-717,R801
	100	390Ω	CRG0805F390R	TE Connectivity	0,018 €	1,8 €	R613
	900	300Ω	MCR03EZPFX3000	ROHM	0,003 €	2,7 €	R201,R402-405,R612,R706,R711,R802
	1700	220Ω	MCR03EZPFX2200	ROHM	0,004 €	6,8 €	R301-316,R611
	100	100Ω	MCR03EZPFX1000	ROHM	0,004 €	0,4 €	R602
	100	47mΩ	ERJL06KF47MV	Panasonic	0,212 €	21,2 €	R805
	900	0Ω	CR0603-J/-000ELF	Bourns	0,01 €	9 €	R103,R317,R319,R321-323,R803,R804,R806
RV	100	10KΩ	RK09K1110A0J	Alps Electric	0,48 €	48 €	RV301
C	100	220uF (E)	UWT0J221MCL1GB	Nichicon	0,184 €	18,4 €	C201
	700	4'7uF	1206YG475ZAT1A	AVX	0,038 €	26,6 €	C208,C211,C213,C606-609
	100	2'2uF	12103C225KAT2A	AVX	0,487 €	48,7 €	C605
	100	1uF	222278115663	Phycomp	0,134 €	13,4 €	C610
	2600	1uF (A)	GRM21BF51C105ZA01L	Murata	0,025 €	65 €	C202,C205,C214,C216,C401,C501,C601,C603,C619-626 ,C715,C716,C717-720,C801,C802,C806,C901
	700	100nF	CGA3E2X7R1H104K080AA	TDK	0,078 €	54,6 €	C210,C703,C704,C711-714
	2500	10nF (B)	06031C103KAT2A	AVX	0,008 €	20 €	C203,C206,C215,C217,C402,C502,C602,C604,C611-618 ,C701,C702,C707-710,C803,C807,C902
	700	100pF (C)	06031A101J4T2A	AVX	0,026 €	18,2 €	C204,C207,C212,C705,C706,C804,C805
DZ	100	Zener 2V5	LM285D-2.5G	ON Semiconductor	0,42 €	42 €	D701
	200	Zener 5V6	MMBZ5232BLT1G	ON Semiconductor	0,014 €	2,8 €	D201,D203
	200	Zener 10V	LM4040DIM3-10.0/NOPB	Texas Instruments	0,302 €	60,4 €	D704,D705
DL	100	LED Verde	LP T670-G2J2-1	OSRAM Opto Semiconductors	0,10 €	10,3 €	D202
	500	LED Naranja	KPH-1608SECK	Kingbright	0,059 €	29,5 €	D706-709,D801
	1600	LED Rojo	KP-1608SRC-PRV	Kingbright	0,053 €	84,8 €	D301-308,D603-610
DS	500	Schottky Doble	DB5S308K0R	Panasonic	0,087 €	43,5 €	D204,D601,D702,D703,D802
SW	300	SPST-NA	2-1437565-7	TE Connectivity	0,088 €	26,4 €	SW101,SW301,SW302
F	200	1A	MICROSMD075F-2	Littelfuse	0,299 €	59,8 €	F201,F202
L	100	47mH	RL181S-473J-RC	Bourns	1,008 €	100,8 €	L601
	200	10uH	LQM18FN100M00D	Murata	0,112 €	22,4 €	L203,L204
L_Ferrita	500	1MHz	742792609	Wurth Elektronik	0,115 €	57,5 €	L201,L202,L205,L206,L801
Q	100	MOSFET N	DMG3420U-7	Diodes Zetex	0,064 €	6,4 €	Q801
–	100	Display 7seg	KCSC02-106	Kingbright	1,214 €	121,4 €	S301
–	100	AO 100mA	NCV20081SQ3T2G	ON Semiconductor	0,30 €	30 €	U602
–	100	AO Quad ±15V	LM324D	ST Microelectronics	0,079 €	7,9 €	U701
–	100	AO Shunt	NCS199A2SQT2G	ON Semiconductor	0,386 €	38,6 €	U801
–	100	Boost Doble IC	LT3463AEDD#PBF	Linear Technology	2,19 €	219 €	U201
–	100	Memoria RAM	23LC512-I/SN	Microchip	1,234 €	123,4 €	U401
–	100	DAC	MCP4725A1T-E/CH	Microchip	0,638 €	63,8 €	U501
–	100	ADC	AD7995YRJZ-1500RL7	Analog Devices	3,35 €	335 €	U901
–	200	Expansor E/S	PCA9554BS3,118	NXP	0,97 €	194 €	U601,U702
–	800	Analog Switch	TS5A3166	Texas Instruments	0,211 €	168,8 €	U603-610
–	100	Quad Analog Switch	DG411DY-T1-E3	Vishay	0,942 €	94,2 €	U703
J	100	Pines arduino	ARD-0056	Sparfunk	1,9 €	190 €	J101
	300	Pines servo	251-8092	RS Pro	0,129 €	38,7 €	J801-803
						3104,618 €	
		Componentes	**Fabricación**	**Montaje**	**Coste por unidad**		
		3104,618 €	98 €	775 €	**39,77618 €**		

Costes del proyecto

COSTE DEL PROYECTO	€/h	HORAS	COSTE
Realización del proyecto			**3.480 €**
Recopilación de información	8 €	60	480 €
Diseño electrónico	12 €	120	1.440 €
Ensamblaje y pruebas electrónicas	12 €	60	720 €
Diseño y pruebas software	12 €	40	480 €
Edición del proyecto	9 €	40	360 €
Amortización equipos			**1.345 €**
Ordenador	3 €	280	840 €
Licencias de software	0 €	200	0 €
Uso equipos laboratorio	4 €	120	480 €
Utiles de soldadura	5 €	5	25 €
Coste Hardware			**362 €**
Fabr. placa prototipo			24 €
Coste componentes prototipos			40 €
Coste placa final			132 €
Coste componentes placa final			166 €
Coste Proyecto			**5.187 €**
Coste de 100u			**3.978 €**
Coste Componentes			3.105 €
Coste Fabricación + Ensamblaje			873 €
Total Fabricación 100u			**9.165 €**
Coste por unidad			92 €



12.5 UMA_AEB_V1.1.0

dimensiones que ésta: 68.6 mm x 53.4 mm [2].

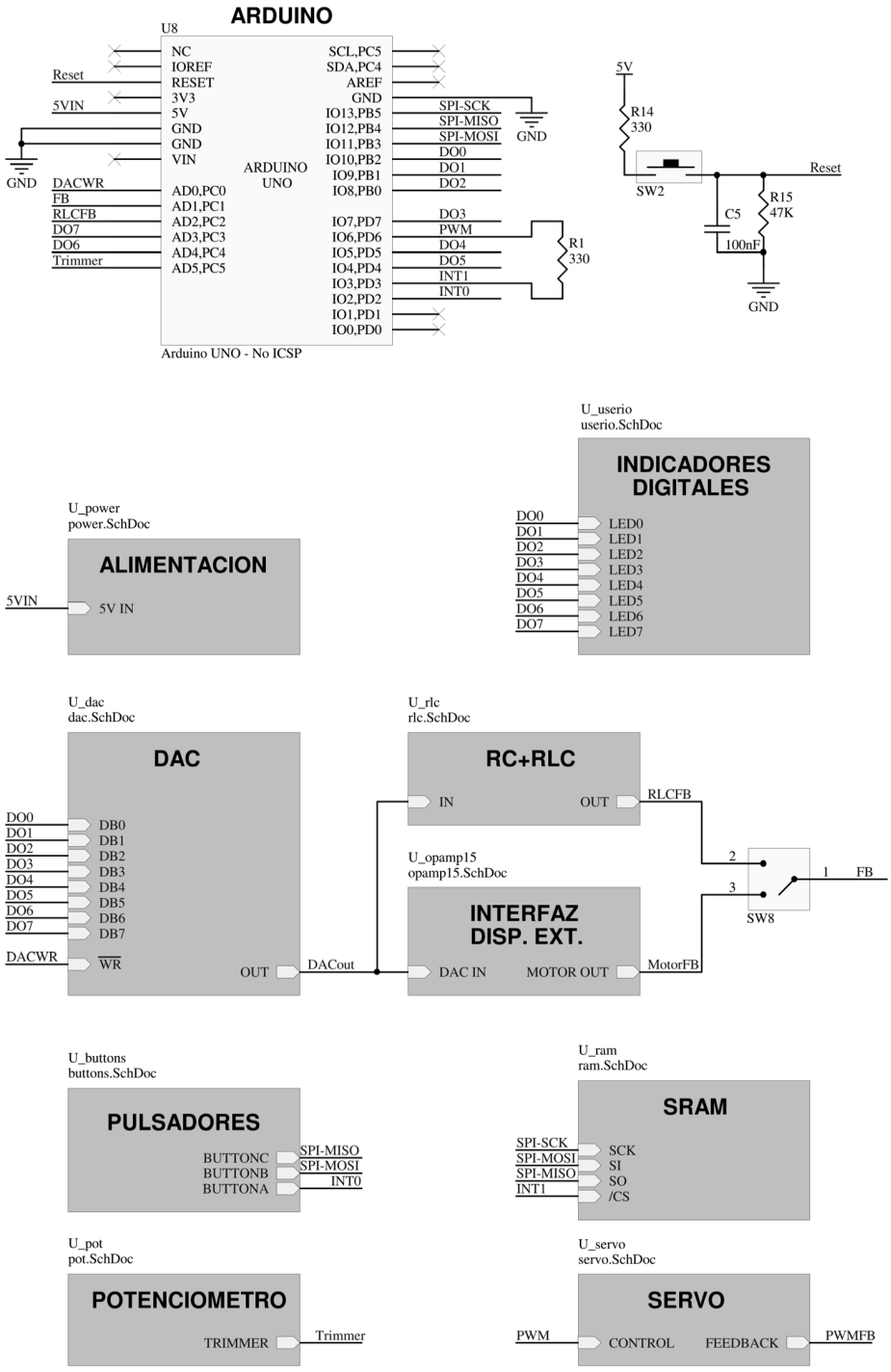


Fig. 13. Esquemático general de la realización preferida y de la parte del subsistema misceláneo que no tiene que ver con la alimentación de la invención. En esta figura aparecen las 5interconexiones de los distintos subsistemas (mostrados en las figuras siguientes), así como las

existentes entre los mismos y el microcontrolador ATmega328P de la placa Arduino/Genuino UNO.

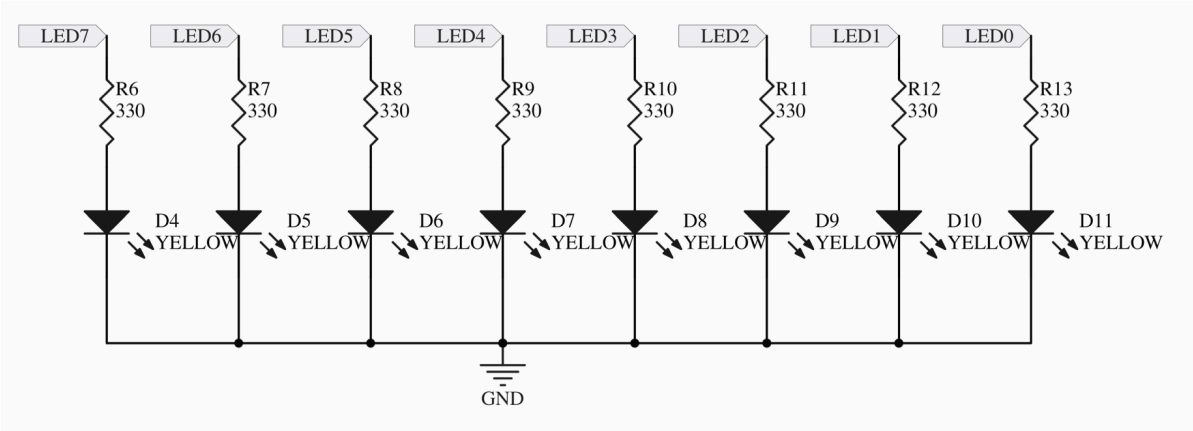


Fig. 14. Esquemático con la realización preferida del subsistema de indicadores led.

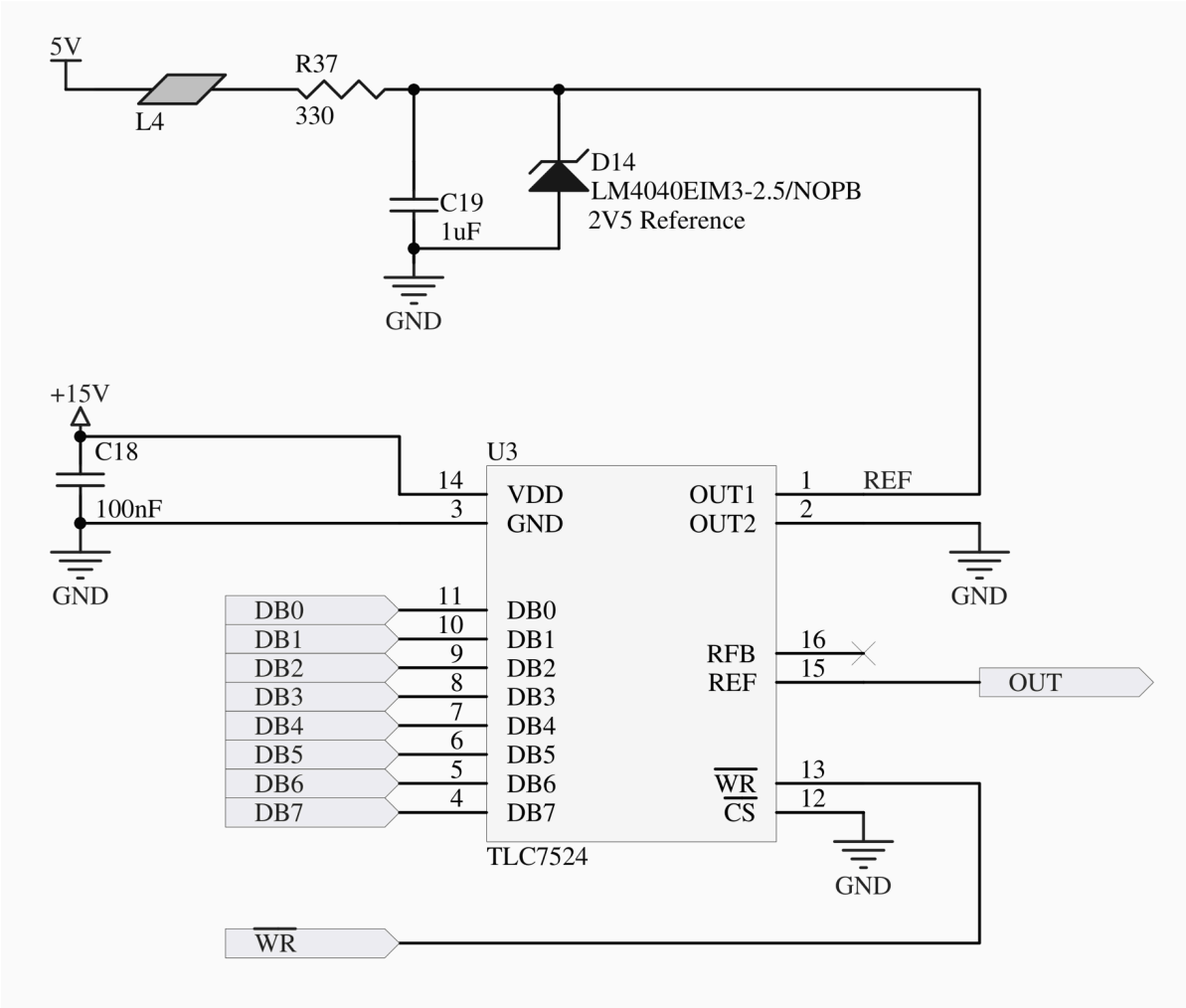


Fig. 15. Esquemático con la realización preferida del subsistema de conversión digital-analógica.

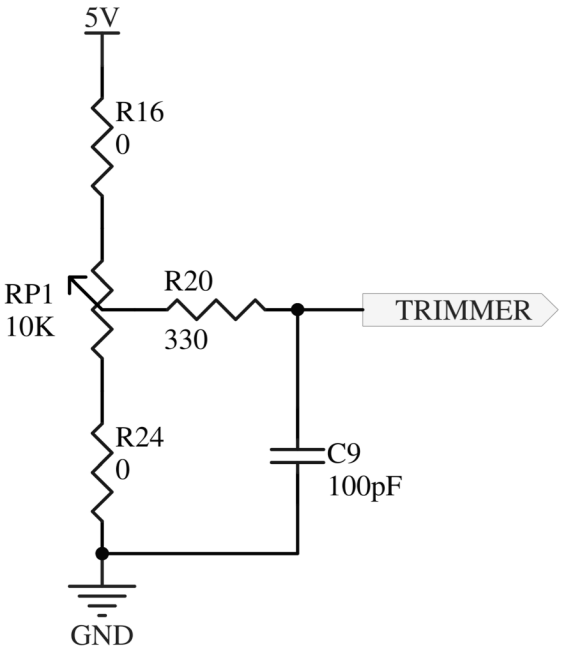


Fig. 19. Esquemático con la realización preferida del subsistema de potenciómetro manual.

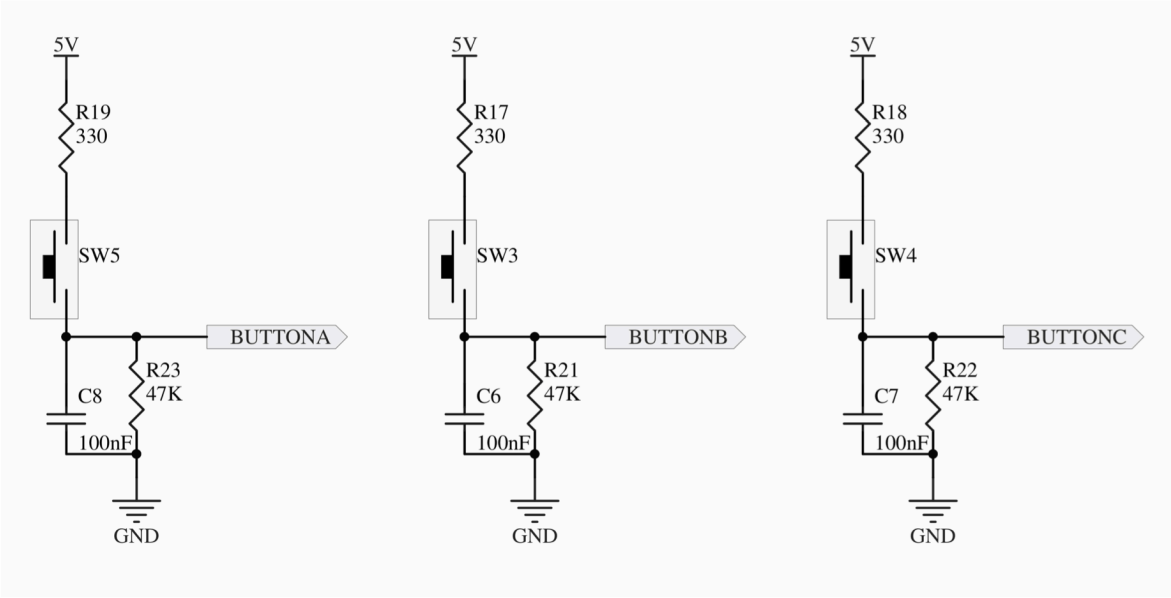


Fig. 20. Esquemático con la realización preferida del subsistema pulsadores.

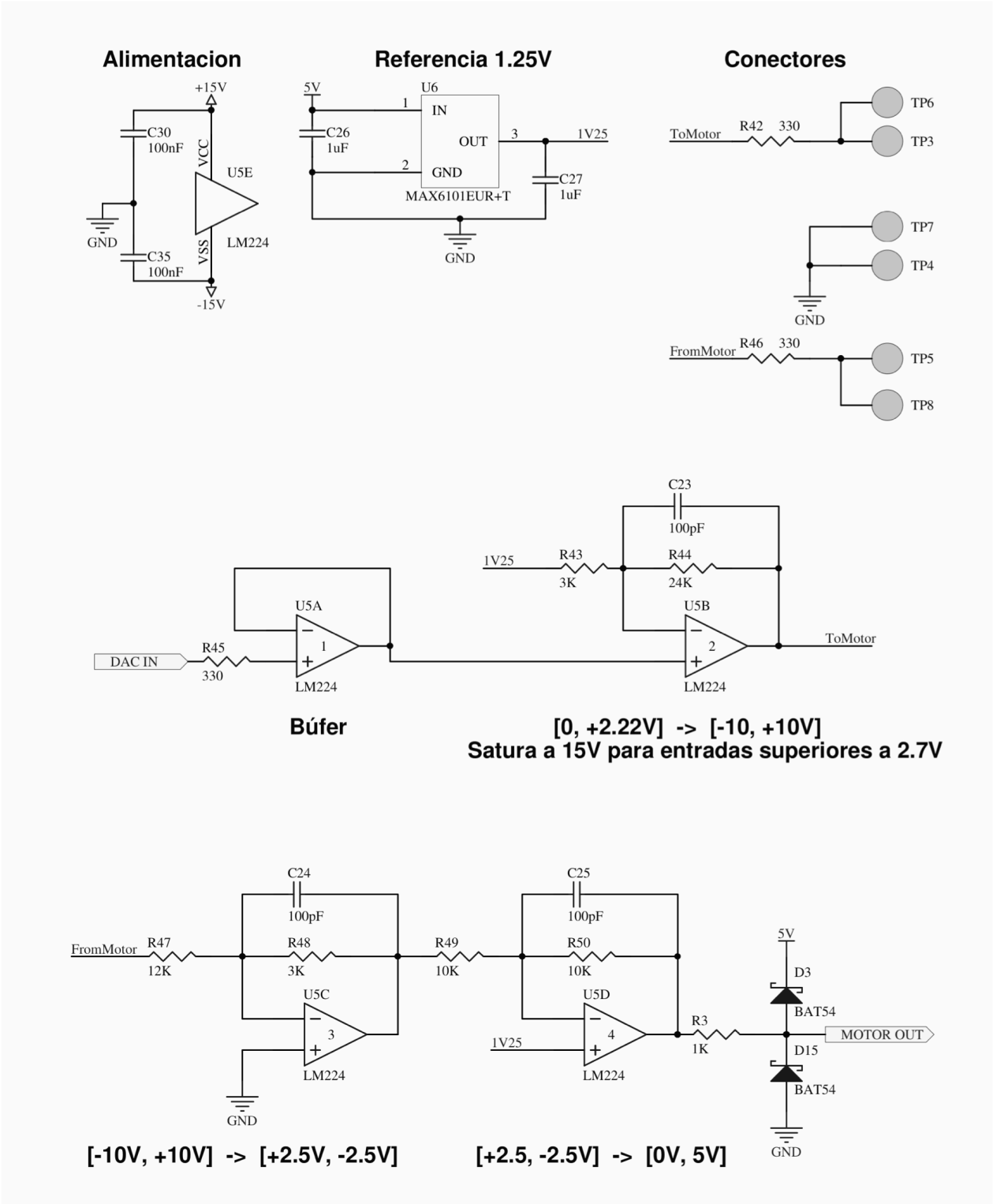


Fig. 17. Esquemático con la realización preferida del subsistema de interfaz con dispositivos externos.

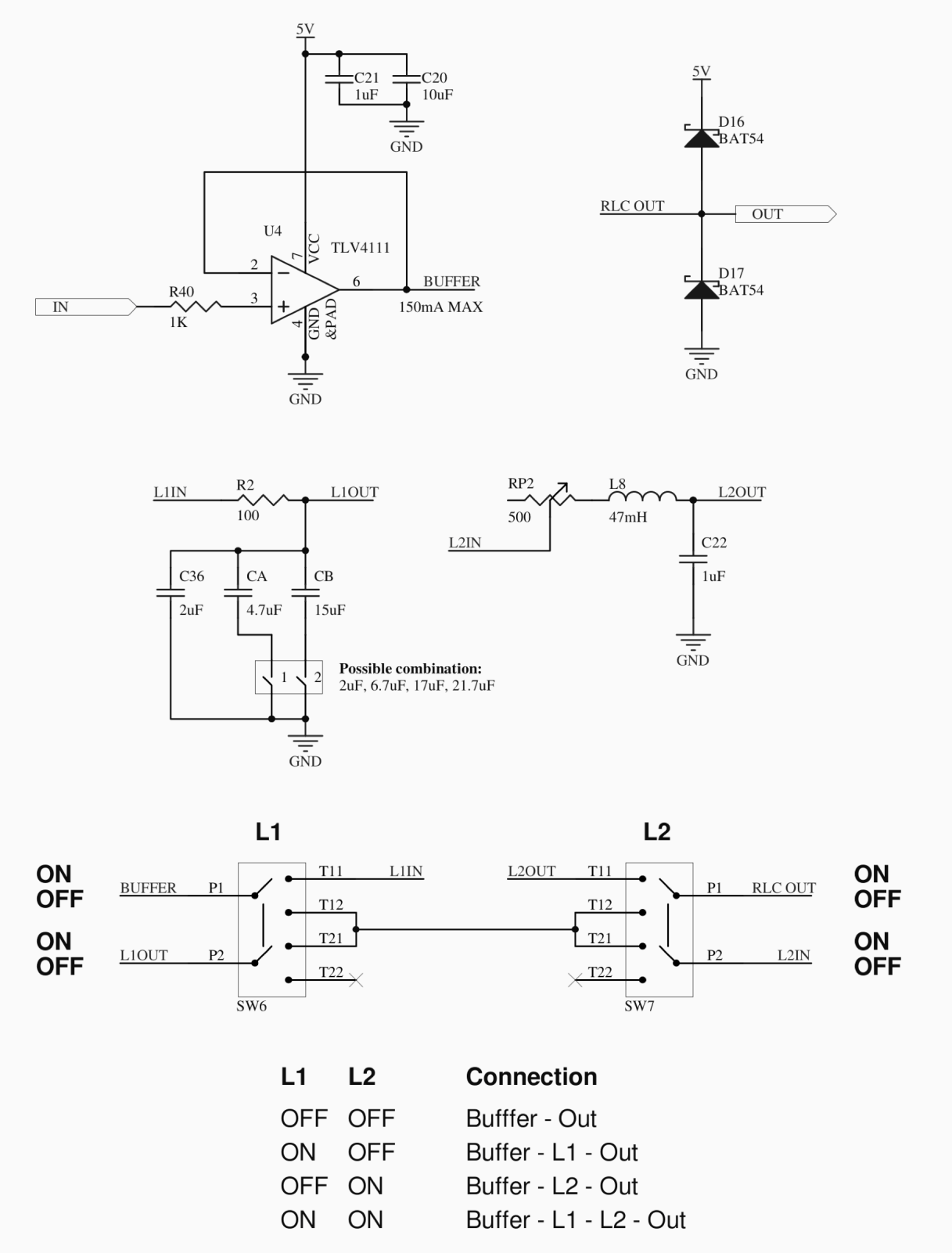


Fig. 16 Esquemático con la realización preferida del subsistema RC+RLC.

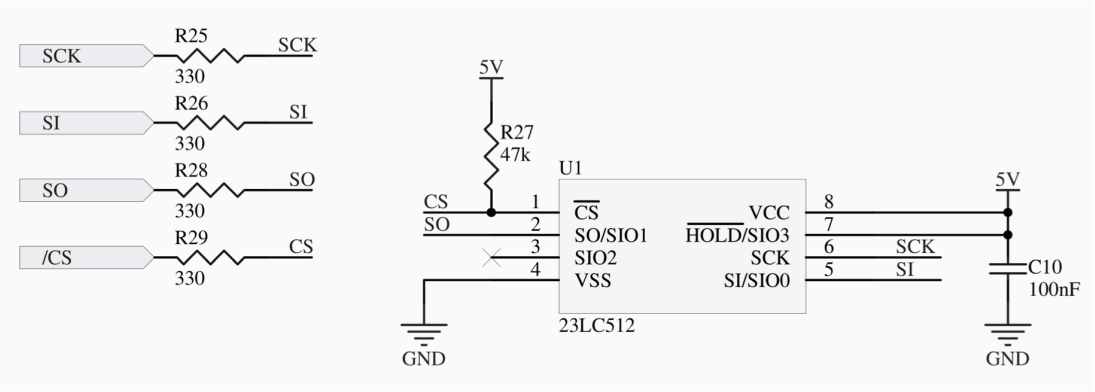


Fig. 21. Esquemático con la realización preferida del subsistema SRAM.

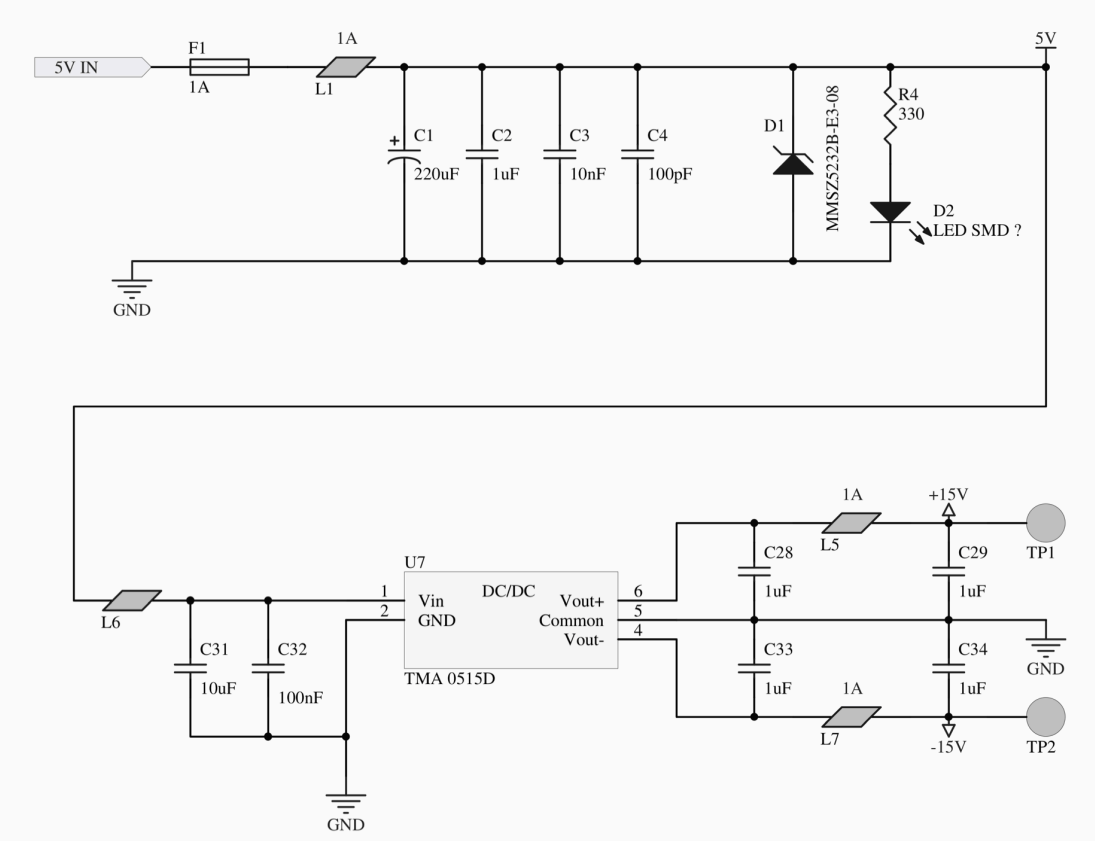


Fig. 22. Esquemático con la realización preferida de la parte del subsistema misceláneo 5relativa a la alimentación de la invención.

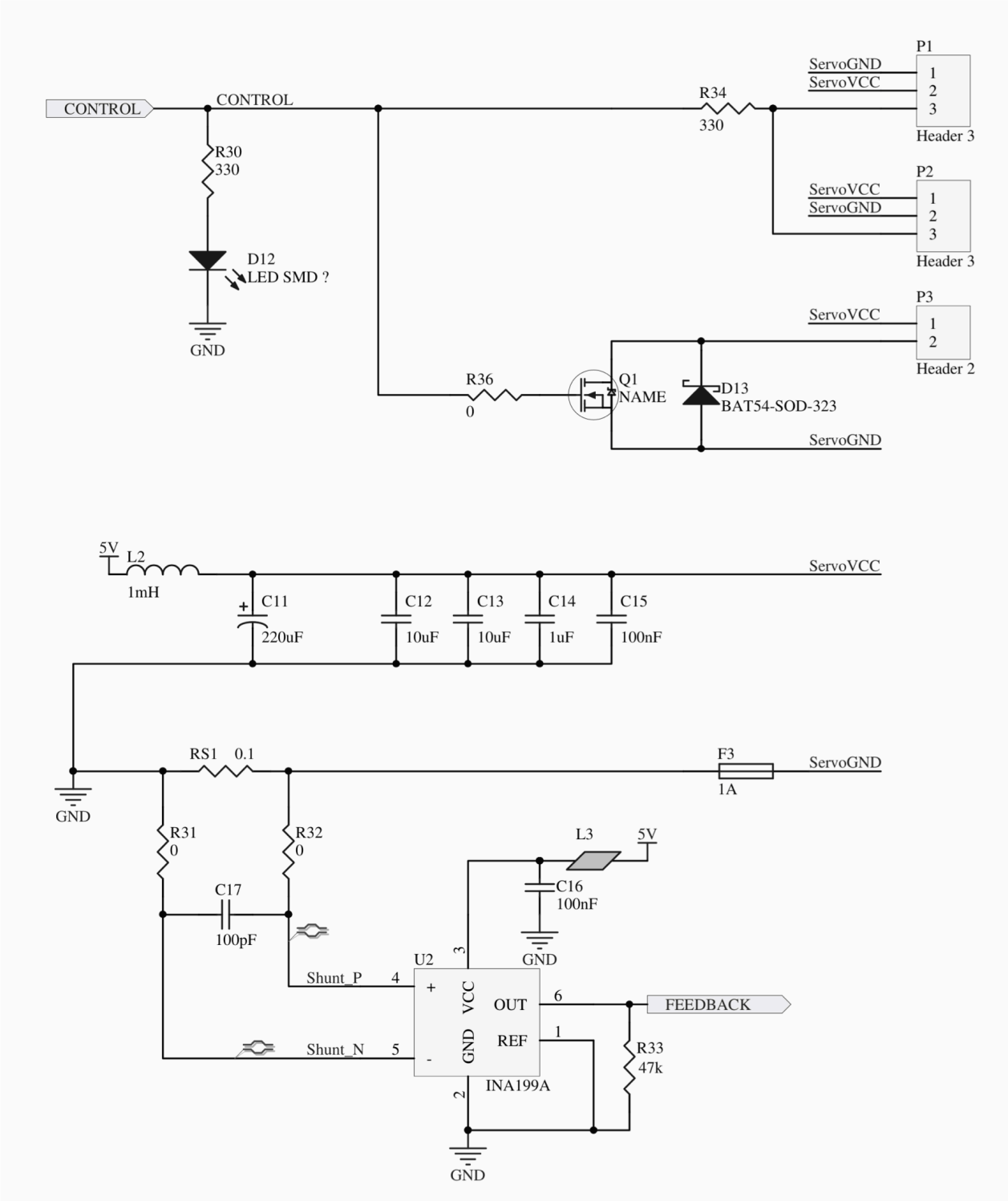


Fig. 18. Esquemático con la realización preferida del subsistema de control de servomotores de radio control y cargas que necesiten potencia.

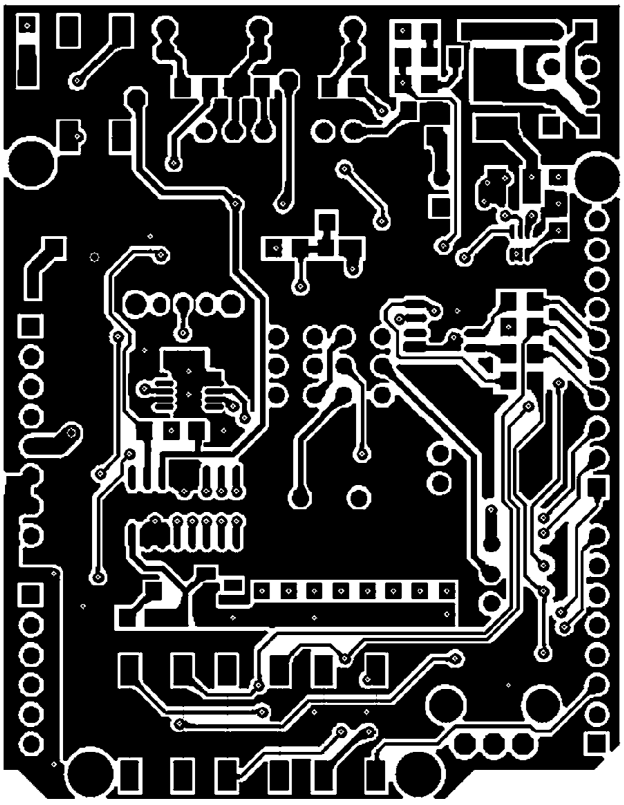


Fig. 23. Cara de pistas superior de la PCB para la realización preferida.

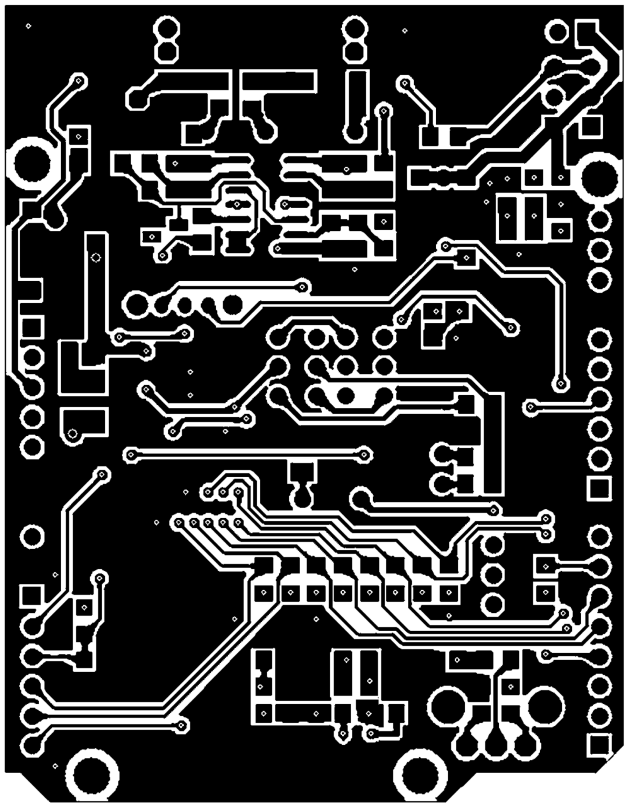


Fig. 24. Cara de pistas inferior de la PCB para la realización preferida.

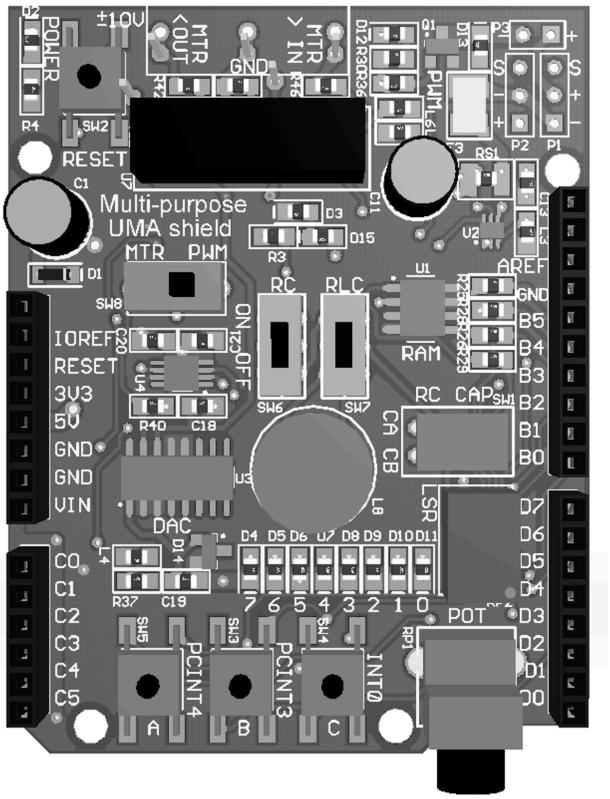


Fig. 25. Aspecto simulado de la cara superior de la PCB para la realización preferida, una vez soldados los componentes, donde se aprecia el etiquetado diseñado para la invención, que en los conectores laterales replica el nombre de los pines del microcontrolador ATmega328P, algo no disponible en la placa Arduino/Genuino UNO.

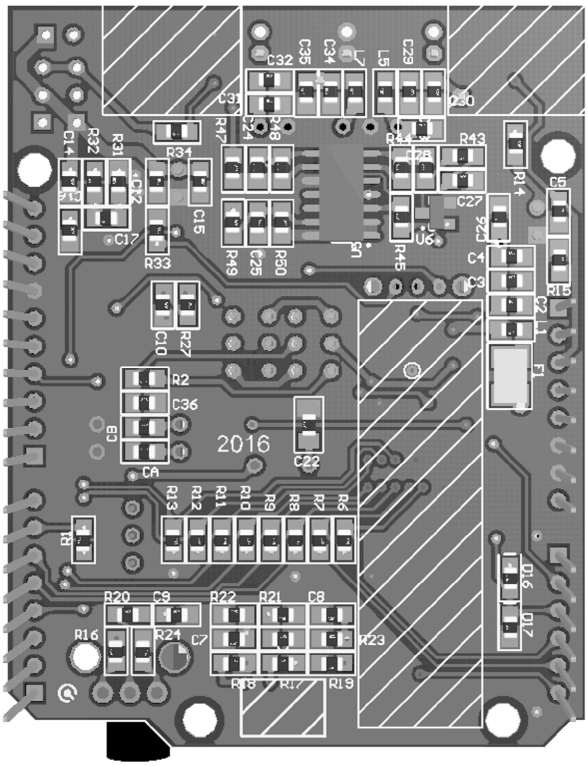


Fig. 26. Aspecto simulado de la cara inferior de la PCB para la realización preferida, una vez soldados los componentes. Se observan, apuntando hacia fuera de la imagen, los pines de 5conexión que unen la invención con la placa Arduino/Genuino UNO.