UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA TARJETA DE ADQUISICIÓN DE DATOS USB CON 4 CANALES ANALÓGICOS DE ENTRADA/SALIDA BASADA EN LA PLACA NUCLEO-F446RE DE ST, Y LIBRERÍAS PARA LabVIEW

AUTOR: GUILLERMO JOAQUÍN ALITE CEREZUELA

TUTOR: JUAN MANUEL HERRERO DURÀ

Curso Académico 2017-2018

AGRADECIMIENTOS

“Quiero agradecer todo el apoyo que he recibido de mis amigos y familia habéis sido un pilar fundamental, pero especialmente a mi madre, sé que hoy estaría muy orgullosa. Gracias por todas las velas que me pusiste durante la carrera”.

RESUMEN

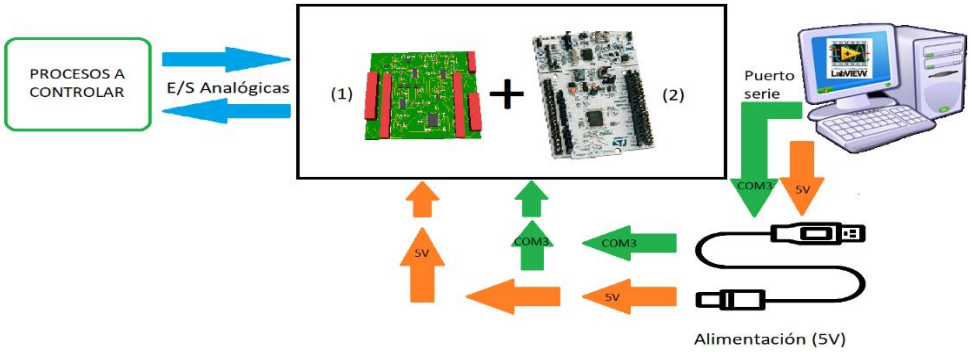
En este proyecto se aborda el diseño de todos los elementos necesarios para la creación de una tarjeta de adquisición de datos USB con 4 canales analógicos de entrada/salida. Se ha diseñado en base a la placa STM32F446RE, y por tanto este es el elemento que fija muchas de las condiciones que deben cumplir los demás componentes incluidos en este trabajo. Utilizando esta placa como cerebro de la tarjeta, se ha diseñado el *hardware* y el *software* requeridos.

El *hardware* diseñado es un circuito impreso que se acopla a la placa F446. Esta PCB contiene los circuitos eléctricos necesarios para: la conversión de rangos de voltajes de [-10, +10]V a rangos aceptados por el microcontrolador [0, 3.3]V, la alimentación diferencial de los amplificadores operacionales a [-12, +12]V, la adición de dos salidas analógicas por protocolo I2C y la adaptación de tensión del puerto serie.

Por otro lado, el *software* diseñado incluye: el programa insertado en el microcontrolador, STM32F446 (Cortex-M4), y las librerías de LabVIEW. El programa que contiene el microcontrolador está escrito en lenguaje C, y es el encargado de procesar y comprobar las ordenes que serán enviadas desde la aplicación en LabVIEW. Durante la ejecución del programa, se activarán los periféricos dependiendo de la orden que se reciba. Las librerías diseñadas en este proyecto son funciones para controlar los convertidores integrados en el sistema, y posteriormente se pueden utilizar en aplicaciones para el control de procesos, mediante el *software* LabVIEW. Además, se ha incluido un pequeño programa de pruebas, que sirve como ejemplo de aplicación de uso de la tarjeta y para realizar las pruebas del correcto funcionamiento de la misma.

La comunicación entre elementos se realiza por puerto serie virtual, mediante un cable USB se conectará el ordenador y el puerto virtual COM generado por la placa, este será el canal por el que los datos de órdenes serán enviados y recibidos. El cable USB también sirve para la alimentación a 5V de la placa y para reprogramar el micro. Esta es una de las características más importantes de este proyecto, la sencillez para reprogramar el microcontrolador, pudiendo darle otra funcionalidad, pero manteniendo el *hardware*. Podría servir, por ejemplo, para implementar hasta cuatro lazos de control, o como un simulador *Hardware-in-the-Loop (HIL)*, debido a esta característica, el proyecto no se queda encerrado en las funcionalidades descritas en este trabajo, sino que permite mejorar o cambiar estas funcionalidades. Otra característica destacable sería su reducido coste en comparación con tarjetas de adquisición de datos con características similares.

Como prueba de concepto, utilizando el prototipo realizado, se ha abordado el control de 4 motores independientes con 4 lazos de control distintos. Confirmando así la viabilidad y el buen funcionamiento del conjunto de elementos que componen este Trabajo de Fin de Grado.



RESUM

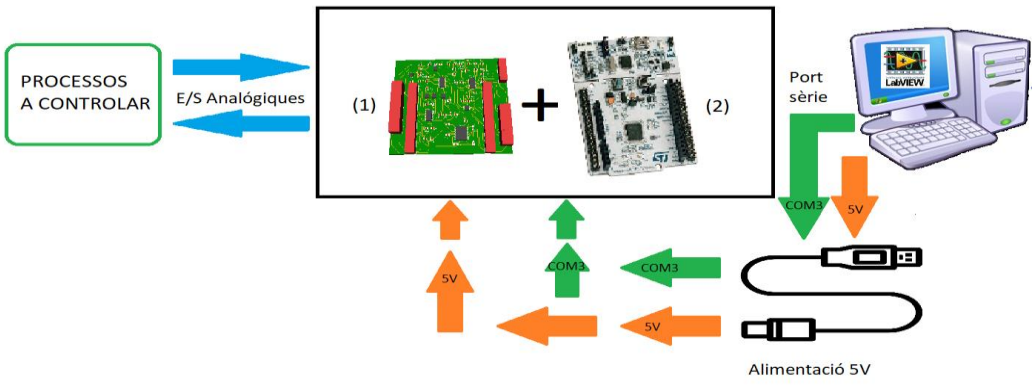
Aquest projecte aborda el disseny de tots els elements necessaris per a la creació d'una targeta d'adquisició de dades USB amb 4 canals d'entrada/sortida analògics. Està dissenyat basant-se en la placa STM32F446RE, i per tant aquest és l'element que imposa moltes de les condicions que han de complir els altres components inclosos en aquest treball. Utilitzant aquesta placa com el cervell de la targeta, s'ha dissenyat el *hardware* i *software* necessari.

El *hardware* dissenyat és una placa de circuit imprès que s'acobla a la placa F446. Aquesta PCB conté els circuits elèctrics necessaris per a: la conversió de l'interval de voltatge de [-10, +10] V a la franja acceptada pel microcontrolador [0, 3.3] V, l'alimentació diferencial a [-12, +12] V dels amplificadors operacionals, l'addició de dues sortides analògiques per protocol I2C i l'adaptació de tensió del port sèrie.

D'altra banda, el *software* dissenyat inclou: el programa inserit al microcontrolador, STM32F446 (Cortex-M4) i les llibreries de LabVIEW. El programa que conté el microcontrolador està escrit en llenguatge C i és el responsable del tractament i la consulta de comandes que seran enviades des de l'aplicació en LabVIEW. Durant l'execució del programa, s'activaran els perifèrics segons la comanda rebuda. Les llibreries dissenyades en aquest projecte són funcions per a controlar els convertidors integrats en el sistema i que posteriorment es poden utilitzar en aplicacions per al control de processos, utilitzant el *software* de LabVIEW. També s'ha inclòs un petit programa de proves, que serveix com a exemple d'aplicació d'ús de la targeta i per provar el correcte funcionament de la mateixa.

La comunicació entre elements es realitza per port sèrie virtual, mitjançant un cable USB que connectarà l'ordinador i el port COM virtual generat per la placa, aquest és el canal pel qual les ordres dades seran enviades i rebudes. El cable USB també serveix per a l'alimentació a 5V de la placa i per a reprogramar el micro. Aquesta és una de les característiques més importants d'aquest projecte, la senzillesa per a reprogramar el microcontrolador , proporcionant-li altres funcionalitats, però mantenint el *hardware*. Podria servir, per exemple, per implementar fins a quatre bucles de control, o com un simulador de Hardware-in-the-Loop (HIL), a causa d'aquesta característica el projecte no romandrà tancat dins de les característiques descrites en aquest treball, perquè permet millorar o canviar aquestes característiques. Una altra característica notable seria el baix cost que presenta, comparat amb targetes de adquisició de dades similars.

Com a prova de concepte, utilitzant el prototip realitzat, s'ha abordat el control de 4 motors independents amb 4 bucles de control diferents. Confirmant així la viabilitat i el correcte funcionament de tots els elements que conformen aquesta Treball de Fi de Grau.



ABSTRACT

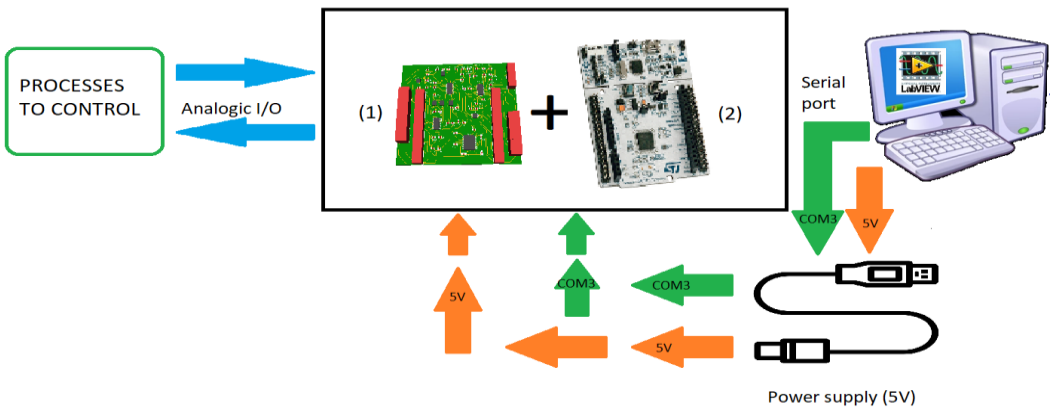
This project addresses the design of all the necessary elements for the creation of an USB data acquisition card with 4 analogue channels of input/output. It has been designed based on the STM32F446RE plate, and therefore this is the element that fixes many of the conditions that must be met by other components included in this work. Using this board as the card's brain, the required hardware and software has been designed.

The designed hardware is a printed circuit attached to the F446 plate. This PCB contains the electrical circuits needed for: the conversion of voltage ranges from [-10, + 10] V to the ones accepted by the microcontroller [0, 3.3] V, the differential power supply of operational amplifiers to [-12, + 12] V, the addition of two outputs Analogues by I2C protocol and the voltage adaptation of serial port.

On the other hand, the designed software includes: a program inserted in the microcontroller, STM32F446 (Cortex-M4), and LabVIEW libraries. The microcontroller contains a program which is written in C language and is in charge of processing and checking the orders that will be sent from the LabVIEW application. During the program execution, peripherals will be activated depending on the order received. The designed libraries in this project are functions to control the integrated converters in the system, and then, they can be used in applications for process control, using the LabVIEW software. In addition, a small test program has been included, as an example of howthe card could be operated, and also to test the correct functioning of the board.

Communication between elements is made by virtual serial port, through an USB cable that will connect the computer and the virtual port COM generated by the board, this will be the channel whereby the data of orders will be sent and received. The USB cable is also used for powering with 5V the board and for reprogramming the microprocessor. This is one of the most important features of this project, the simplicity to reprogram the microcontroller, being able to give it another functionality, but maintaining the hardware. It could be used, for example, to implement up to four control loops, or as a Hardware-in-the-Loop simulator (HIL). Because of this feature, the project is not locked in the functions described in this work, in fact, it allows us to improve or change these features. Another notable feature would be its reduced cost compared to data acquisition cards with similar features.

As a proof of concept, using the prototype made, the control of 4 independent engines has been addressed with 4 different control loops. This confirms the viability and the good functioning of the set of elements that compose this Final Project.



CONTENIDO DEL PROYECTO

- DOCUMENTO Nº1. MEMORIA DEL PROYECTO
- DOCUMENTO Nº2. PRESUPUESTO
- ANEXO I. MANUAL DE USUARIO
- ANEXO II. MANUAL DE DISEÑO DEL CIRCUITO IMPRESO
- ANEXO III. MANUAL DE PROGRAMACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA TARJETA DE ADQUISICIÓN DE DATOS USB CON 4 CANALES ANALÓGICOS DE ENTRADA/SALIDA BASADA EN LA PLACA NUCLEO-F446RE DE ST, Y LIBRERÍAS PARA LabVIEW

DOCUMENTO Nº1:

MEMORIA DEL PROYECTO

AUTOR: GUILLERMO JOAQUÍN ALITE CEREZUELA

TUTOR: JUAN MANUEL HERRERO DURÀ

Curso Académico 2017-2018

ÍNDICE DE CONTENIDO

1. OBJETO DEL PROYECTO 5
1.1 JUSTIFICACIÓN DEL PROYECTO 5
2. DIAGRAMA SOFTWARE/HARDWARE 5
3. CIRCUITO IMPRESO (PCB) 7
4. PROGRAMACIÓN DEL MICROCONTROLADOR 8
5. LIBRERÍAS PARA LabVIEW. 10
6. INTERFAZ GRÁFICA 11
7. PRUEBA DE CONCEPTO 12
8. TABLA DE CARACTERÍSTICAS 15
9. CONCLUSIONES 15
BIBLIOGRAFÍA 17

ÍNDICE DE FIGURAS

Figura 1. Esquema general de la comunicación software/hardware. (1) TAD_USB_32F4, Placa diseñada en el TFG. (2) Placa STM32F446RE. 6
Figura 2. Representación en 3D por el editor de PCB de DesignSpark 8.0. 8
Figura 3. Esquema de funcionamiento de la aplicación. 9
Figura 4. Variables de control y comprobación de “ESCRIBIR_DAC”. 10
Figura 5. Variables de control y comprobación de “LEER_ADC”. 11
Figura 6. Controles automáticos de la aplicación “tester.vi”. 12
Figura 7. Controles manuales de la referencia, la constante proporcional, el periodo y el puerto virtual COM. 13
Figura 8. Indicadores de posición, voltaje del motor y señal de control. 13
Figura 9. Secuencia establecida para las referencias en los motores. 13
Figura 10. Botones de Iniciar, Pausar y Stop, junto con los indicadores y controles del modo Reloj. 14
Figura 11. Gráfica de la referencia y la posición del motor 1. 14
Figura 12. Gráfica de la referencia y la posición del motor 2 14
Figura 13. Indicador gráfico de posición de los motores 1 y 2. 15

ÍNDICE DE TABLAS

Tabla 1. Representación de la cadena de bytes enviada al microprocesador 9
Tabla 2. Representación de la cadena de bytes enviada a la aplicación en LabVIEW, es caso de manejar un DAC. 10
Tabla 3.. Representación de la cadena de bytes enviada a la aplicación en LabVIEW, es caso de manejar un DAC (MSB y LSB contendrán el valor leído por el ADC). 10

Tabla 4. Tabla de características de la TAD_USB_32F4 15

1. OBJETO DEL PROYECTO

El proyecto tiene como objetivo el diseño de una Tarjeta de Adquisición de Datos (TAD), basada en la placa STM32F446 [1], con el fin de conseguir una tarjeta de cuatro entradas y cuatro salidas analógicas, utilizando comunicación por puerto serie.

Para ello se diseñará una placa de circuito impreso, que denominaremos TAD_USB_32F4. Que albergará (a modo de inserción) en su parte superior a la placa STM32F446, de forma que quedan acopladas. De esta manera dicha placa es capaz de comunicarse con la otra. Se usan los elementos de la TAD_USB_32F4 para recibir, adaptar y emitir señales, mientras que el procesador STM32F446 (Cortex-M4) [2] controla todos los procesos a realizar.

1.1 JUSTIFICACIÓN DEL PROYECTO

Este proyecto nace como un trabajo académico, donde el alumno que lo realiza se adentra en el diseño de circuitos impresos, así como en la programación y el control de microcontroladores y otros elementos electrónicos. Aplicando, y ampliando, los conocimientos aprendidos a lo largo del Grado de Ingeniería en Tecnologías Industriales (GITI).

Gracias al desarrollo de este proyecto se conseguirán conocimientos en el manejo de *softwares,* tales como: *DesignSpark PCB [3], STM32CubeMx [4], SW4STM32 [5] y LabView [6].*

Además, se pretende que el proyecto sirva como base para futuros TFGs, y como herramienta en los laboratorios de control, ya que el producto final será una TAD de bajo coste con cuatro entradas/salidas analógicas. Podría ser usada en lugar de las actuales, dotando así a los laboratorios de nuevo material, con un coste reducido, y con las características que se mencionarán a lo largo del proyecto.

Una característica que justifica este proyecto es su cantidad de salidas analógicas, la TAD_USB_32F4 está diseñada para albergar 4 canales de salida. Lo normal en el mercado actual es que las TAD incluyan tan solo 2 salidas analógicas, forzando en algunos casos, a la utilización de más de una tarjeta para conseguir el número de canales de control deseados. Además, esta característica se complementa con el hecho de que este proyecto tiene una reprogramación muy sencilla, permitiendo darle otro uso a la placa, manteniendo el mismo hardware. La TAD_USB_32F4 podría ser reprogramada para funcionar como un HIL (*Hardware In the Loop*) o como un controlador PID con cuatro lazos independientes. O incluso tener todas estas funcionalidades a la vez, programadas en el micro, y seleccionar una u otra.

2. DIAGRAMA SOFTWARE/HARDWARE

Los distintos bloques que este proyecto abarca aparecen esquematizados en la Figura 1, todos ellos se comunicarán entre sí con el fin de realizar las acciones para las cuales han sido diseñados. La unión de los elementos (1) y (2) es el centro de todo el proyecto, el conjunto de las dos placas será el encargado de adquirir las señales de salida y de escribir las señales de control de los procesos.

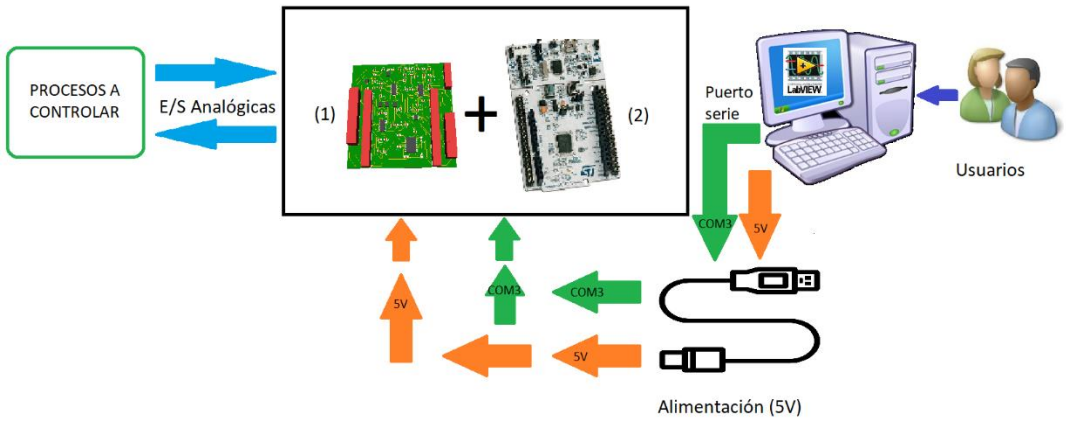


Figura 1. Esquema general de la comunicación software/hardware. (1) TAD_USB_32F4, Placa diseñada en el TFG. (2) Placa STM32F446RE.

El bloque de *software* comprende desde las librerías programadas en LabVIEW, hasta el programa que se ha insertado en el microcontrolador. Este bloque es el que permite al usuario comunicarse con el microcontrolador y realizar las operaciones que estas librerías permiten. Tales como medir el voltaje en cualquiera de las cuatro entradas analógicas, así como sacar voltaje por cualquiera de las cuatro salidas analógicas. El usuario solamente necesitará conocer el funcionamiento de estas librerías, explicado en el ANEXO III: MANUAL DE PROGRAMACIÓN. Así como conocer los rangos de entrada y salida de las E/S analógicas.

El bloque de *hardware* agrupa tanto la TAD_USB_32F4RE, que será explicada en el ANEXO II: MANUAL DE DISEÑO DE LA PCB, como la placa STM32F445RE. Estos dos elementos se unen formando uno solo, dotando al conjunto de mejores especificaciones. Utilizando los conectores repartidos por la placa TAD_USB_32F4 se conectarán los cables requeridos, tales como el de alimentación y comunicación por puerto serie, masas o señales de E/S analógicas. De manera lógica, necesitaremos también un proceso a controlar o monitorizar, un ordenador para ejecutar el *software* que abarca el proyecto y utilizarlo como fuente de alimentación de la placa.

El elemento (1) mostrado en la Figura 1, es la PCB diseñada en este proyecto, nombrada también como PCB TAD_USB_32F4, ya que es un accesorio que se adapta a los pines de la F446 (Elemento (2) de la Figura 1).

La STM32F446RE es una plataforma de *hardware* libre, que ejecuta el programa que contenga el microcontrolador. Tiene multitud de periféricos en su interior que le permiten manejar distintas señales, pudiendo ser estos activados o desactivados desde el *software* STM32CubeMx. A lo largo del código implementado en la placa estos periféricos interaccionarán con las señales recibidas, con el fin de realizar la acción para la cual ha sido programada. La ventaja más evidente de esta placa es su sencillez para ser reprogramada, de manera que podría cambiar su utilidad si fuera necesario. En este proyecto ha sido realmente útil esta facilidad de reprogramar la placa, ya que se han realizado numerosas pruebas antes de conseguir el programa con todas las funcionalidades buscadas. Debido a que la F446 solo dispone de 2 salidas analógicas, ha sido necesario añadir el I2C.

El I2C es un periférico que está incluido en el elemento (1), su función es aportar 2 salidas analógicas más al conjunto del proyecto. Para llevar a cabo la comunicación microcontrolador-I2C se utilizan las señales I2C_SDA e I2C_SCL, todos los detalles de

esta comunicación quedan reflejados en El ANEXO II. MANUAL DE DISEÑO DE LA PCB.

3. CIRCUITO IMPRESO (PCB)

La PCB TAD_USB_32F4 es el elemento de estudio principal del proyecto, ya que es la que contiene los elementos necesarios para conseguir las características que serán descritas a lo largo del documento. La PCB incluye circuitos para la adaptación de las tensiones de: entrada y salida del microcontrolador, las tensiones del puerto serie y la tensión de alimentación de los operacionales.

Las especificaciones en base a las tensiones son las siguientes:

- Rango de tensiones del microcontrolador: [0, 3.3]V.
- Entradas/Salidas de la PCB: [-10, 10]V.
- Alimentación de la PCB TAD_USB_32F4 y la placa STM32F446 de 5V.

En base a estas especificaciones y al objetivo del proyecto, los circuitos electrónicos comprendidos en la TAD_USB_32F4 tendrán distintas funciones. A continuación, se enumeran todas ellas.

1. Adaptación de la tensión de alimentación de los amplificadores operacionales: Se convierten 5V en ±12V.
2. Adaptación de la tensión del puerto serie.
3. Adaptación de la tensión de entrada: De [-10, 10]V a [0, 3.3]V.
4. Adaptación de la tensión de salida: De [0, 3.3]V a [-10, 10]V
5. Adición de 2 DAC mediante protocolo I2C.

Debido a que la alimentación de la placa es de 5V, ha sido necesario utilizar un convertidor DC/DC para realizar la alimentación diferencial de los amplificadores operaciones de ±12V. Estos operacionales junto con las resistencias pertenecientes a cada entrada/salida forman los bloques sumador-amplificador que adapta las tensiones de entrada a tensiones que el microcontrolador es capaz de leer. Para la entrada se realiza la conversión de [-10, 10]V a [0, 3.3]V, y para la salida se realiza en orden inverso. Los convertidores utilizados son:

- Un ADC perteneciente al microcontrolador, que lee 4 canales distintos.
- Dos DAC pertenecientes al microcontrolador, componen 2 de las 4 posibles salidas.
- Dos DAC implementados con el chip AD5339ARMZ, mediante protocolo I2C.

La adición de DACs mediante protocolo I2C ha sido necesaria porque el microcontrolador no dispone de las 4 salidas requeridas para este trabajo. La resolución de estos convertidores es de 12 *bits*, permitiendo generar o utilizar números comprendidos entre [0, 4095].

Utilizando un cable USB se realiza la alimentación a 5 voltios del conjunto. El cable USB también sirve para la comunicación por el puerto virtual COM3 y para reprogramar el microcontrolador, utilizando el software *SW4STM32*. Además, se ha incluido un adaptador de tensión de puerto serie (MAX232), generando así otro canal adicional con el que podría realizarse otra comunicación, en una futura ampliación de la placa

Una vez diseñados los circuitos mediante la herramienta DesignSpark PCB, se procede a la ubicación de los componentes electrónicos y al enrutado de las pistas conductoras. Los componentes que se han utilizado son SMD (Surface Mounted Device), soldados en la superficie de la TAD_USB_32F4. Su colocación se ha realizado intentando que las pistas fueran lo más cortas posibles y que las vías pasantes fueran las mínimas posibles. También se ha procurado que ningún elemento colisione, debido a su altura, con la placa insertada en la parte superior, la STM32F446RE.

El resultado final es el que se muestra en la Figura 2, con unas dimensiones de 96.5mm de ancho y 86.5mm de largo. La placa se ha montado físicamente, es el modelo *NAKED* que ofrece *EUROCIRCUITS [7].* Se ha elegido este modelo porque es el más fácil de editar, teniendo en cuenta que la placa que se ha montado es el primer prototipo y ha sufrido distintos cambios.

Es un modelo que tiene las pistas al aire, pudiendo ser cortadas o empalmadas entre ellas. Se ha realizado el montaje mediante soldadura de estaño, fijando así los distintos componentes a la PCB.

El desarrollo del diseño de la PCB, tanto los circuitos electrónicos como su disposición espacial están detallados en el ANEXO III. MANUAL DE DISEÑO DEL CIRCUITO IMPRESO.

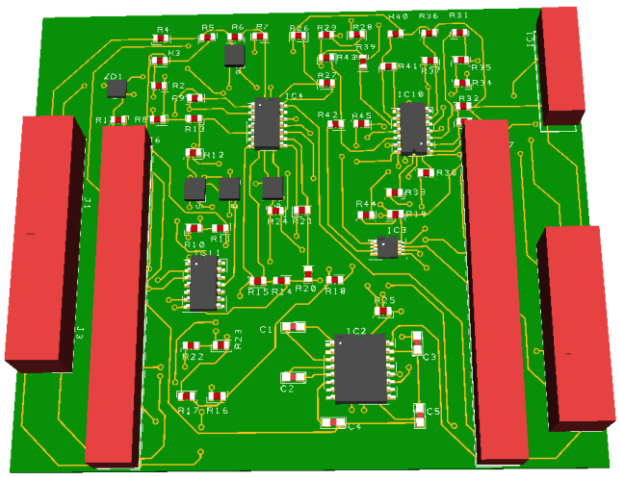


Figura 2. Representación en 3D por el editor de PCB de DesignSpark 8.0.

4. PROGRAMACIÓN DEL MICROCONTROLADOR

Utilizando los *softwares* STM32CubeMX y SW4STM32, se ha escrito una aplicación en lenguaje C, la cual contiene las rutinas que debe seguir el microcontrolador basándose en las señales que le lleguen. La aplicación consta de las funciones que permiten el manejo de los convertidores y aquellas que permiten la comunicación con la aplicación que maneja la placa PCB TAD_USB_32F4, en nuestro caso desarrollada mediante LabVIEW.

El programa, tras inicializar los conversores y puerto de comunicación, entra en un bucle infinito que constantemente está esperando recibir un dato por puerto USB. Se ha establecido un protocolo de interrupción para la recepción de datos, de manera que cuando se detectan *bytes* en el canal de recepción, el microcontrolador ejecuta una rutina, deteniendo el programa principal cada vez que se reciba un dato, iniciando el procesado de este dentro del bucle. Por tanto, se trata de una aplicación con un solo

modo de funcionamiento, de manera que la tarjeta estará esperando sin hacer nada hasta que una orden le indique la operación a realizar.

La petición de la operación a realizar llega por el puerto USB mediante una cadena de 6 *bytes*, cuya composición se muestra en la Tabla 1. A lo largo del programa se han utilizados valores hexadecimales para tratar cada una de las posiciones de la cadena, haciendo así más sencilla su comprensión. Cada uno desempeña una función, así que serán tratados de manera distinta.

0xFF	CMD	MSB	LSB	CHECK	0xFE

Tabla 1. Representación de la cadena de bytes enviada al microprocesador

A continuación, se explicará brevemente cada uno:

- 0xFF: Byte de inicio, siempre es constante.
- CMD: Byte de comando, la operación que realice el procesador dependerá de este valor. Este *byte* puede contener los siguientes valores:
  - Para los ADC: 0x01, 0x02, 0x03, 0x04.
  - Para los DAC: ox11, 0x12, 0x13, 0x14.
- MSB: Byte que representa la parte alta del dato que se quiere enviar o recibir.
- LSB: Byte que representa la parte baja del dato que se quiere enviar o recibir.
- CHECK: Byte reservado para comprobar que los datos se han enviado correctamente, representa la suma de los 3 *bytes* anteriores.
- 0xFE: Byte de fin, siempre es constante.

El valor de 12 *bits* que está incluido en MSB y LSB, se ha codificado de manera que nunca será portador del valor 0xFF, ya que este valor puede ser causa de un error en la transmisión. Para codificarlo se ha multiplicado por 4, de esta manera se evita que aparezca este valor en los *bytes* reservados para el dato. Esta característica queda reflejada con mayor detalle en el ANEXO III. MANUAL DE PROGRAMACIÓN.

Los bytes son recibidos de uno en uno, se procesan de manera consecutiva y en el orden especificado en la lista anterior. Cada uno establece una condición para los datos recibidos, si todas se cumplen se responderá con una cadena de estructura similar.

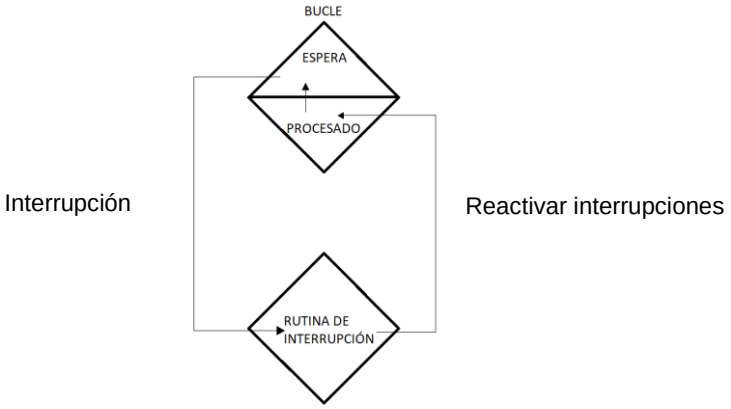


Figura 3. Esquema de funcionamiento de la aplicación.

Es en la etapa de procesado donde está el grueso del programa, en él se accede a diferentes funciones para habilitar y manejar los convertidores, tanto la lectura de los ADC como la escritura de los DAC. Cuando el procesado ya ha terminado y se ha ejecutado la orden del convertidor escogido, se realiza una transmisión por el cable USB, al igual que en el proceso de recepción de datos se transmite una cadena con estructura similar a la recibida, pero con los datos necesarios para confirmar que el proceso de comunicación ha sido exitoso.

0xFF	CMD	0x00	0x00	CMD	0xFE

Tabla 2. Representación de la cadena de bytes enviada a la aplicación en LabVIEW, es caso de manejar un DAC.

0xFF	CMD	MSB	LSB	CHECK	0xFE

Tabla 3.. Representación de la cadena de bytes enviada a la aplicación en LabVIEW, es caso de manejar un DAC (MSB y LSB contendrán el valor leído por el ADC).

En el ANEXO IV. MANUAL DE PROGRAMACIÓN se muestra cómo habilitar los componentes y sus pines, cómo generar el programa base sobre el cual se escribe el código principal, y se explica con detalle la estructura usada para el bucle principal, además se detallan las distintas funciones de la librería HAL que permiten manejar los componentes de la PCB.

La comunicación por puerto serie es a través de un cable USB, que además servirá como alimentación de la placa. De esta manera se reserva el MAX232 y sus conectores para futuras modificaciones del dispositivo. Debido a que el proyecto es fácilmente reprogramable, cualquier persona cualificada podría darle un uso distinto con tan solo cambiar el programa insertado en el micro. En el caso de esta placa, para realizar la comunicación por la entrada *"USB Mini B"* requiere habilitar el canal USART2, dejando así el USART1 para las señales que lleguen a los conectores exteriores del puerto serie posicionados sobre la TAD_USB_32F4", pero en este proyecto, el canal USART1 no se contempla.

Lo mencionado en el párrafo anterior da pie a comentar que el producto final de este proyecto se ha diseñado con una funcionalidad específica, la de una TAD. Pero el microcontrolador puede ser reprogramado con facilidad, permitiendo diseñar un dispositivo con un objetivo distinto, con tan solo cambiar el programa del microcontrolador. Disponiendo de todas las funcionalidades que aporta la PCB inferior.

5. LIBRERÍAS PARA LabVIEW.

Las funciones diseñadas en LabVIEW sirven para manejar los convertidores del micro, desde una interfaz gráfica. Las librerías de LabVIEW no solo contienen elementos de control, sino también de confirmación de envíos. De esta manera rápidamente podremos saber si el funcionamiento ha sido correcto.

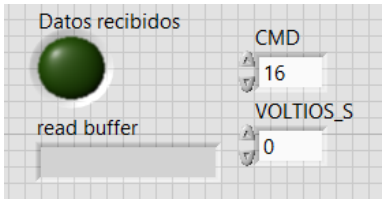


Figura 4. Variables de control y comprobación de "ESCRIBIR_DAC".

La Figura 1Figura 4 muestra los controles de la librería encargada de sacar voltaje a través del DAC escogido. Los dos controles de la derecha elegirán la salida y su voltaje, mientras que los de la izquierda mostrarán si los datos se han recibido de manera correcta, y que datos han llegado. El valor de CMD es con el que selecciona una salida u otra, con números comprendidos entre 16 y 19, equivalentes a las salidas de la 1 a la 4.

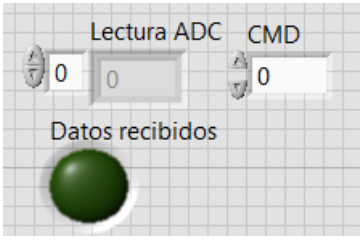


Figura 5. Variables de control y comprobación de "LEER_ADC".

De la misma manera tenemos el control del ADC y sus variables de comprobación, en este caso nos mostrará el valor leído por el convertidor. El CMD en este caso tendrá valores comprendidos entre 0 y 3, equivalente a las entradas de la 1 a la 4.

Además de estos controles e indicadores mostrados en la Figura 4 y Figura 5, las librerías contienen distintos indicadores para poder apreciar todos los procesos de formación de cadenas, unión de datos y trasformación de variables. De esta manera cada paso queda reflejado en un indicador distinto. Ha sido muy útil a la hora de crear el programa, porque da la posibilidad de conocer donde se produce el error. Además, para posteriores usuarios que utilicen las funciones de este proyecto, les será menos complejo entender las acciones que se llevan a cabo en las librerías.

Con estas librerías cualquier programador sería capaz de manejar los convertidores a antojo, tan solo tendría que aportar a las librerías todos los datos que necesita. No solo son importantes los valores del voltaje de salida y de CMD, las variables relacionadas con el puerto serie son de vital importancia para estas librerías. Es necesario inicializar el puerto serie con los valores especificados en el ANEXO I. MANUAL DEL USUARIO. En el anexo mencionado se explica cómo se inicializa el puerto serie, utilizando de ejemplo la aplicación "*tester.vi"* que complementa este proyecto.

6. INTERFAZ GRÁFICA

LabVIEW es una plataforma que permite al programador crear entornos gráficos, para conocer de manera muy sencilla como está funcionando el programa. Para este proyecto se ha diseñado una aplicación "tester.vi" (Virtual Instrument), que con unos pocos botones e interruptores permite al usuario ver el funcionamiento más básico de la TAD. Es un ejemplo de cómo usar las librerías diseñadas en este proyecto, a la vez que permite leer y escribir en los convertidores para realizar pruebas sobre la placa antes de utilizarla en un proceso real. En este proyecto el "tester.vi" ha sido muy útil durante el montaje de la placa y el diseño del programa del micro. Ya que cada vez que se añadían elementos había que probar si funcionaban correctamente.

Utilizando las librerías descritas en el apartado anterior un usuario es capaz de ver los valores que están gestionando los convertidores. Permitiendo tanto leer como escribir por cualquiera de los convertidores. Dispone de dos modos, manual y automático. El modo manual se basa en la ejecución de la librería asociada a cada acción una sola vez, mientras que el modo automático ejecutará en bucle las acciones que estén

seleccionadas mediante los interruptores. La manera en que se ha programado esta aplicación y como usarla queda reflejado en el ANEXO I. MANUAL DE USUARIO.

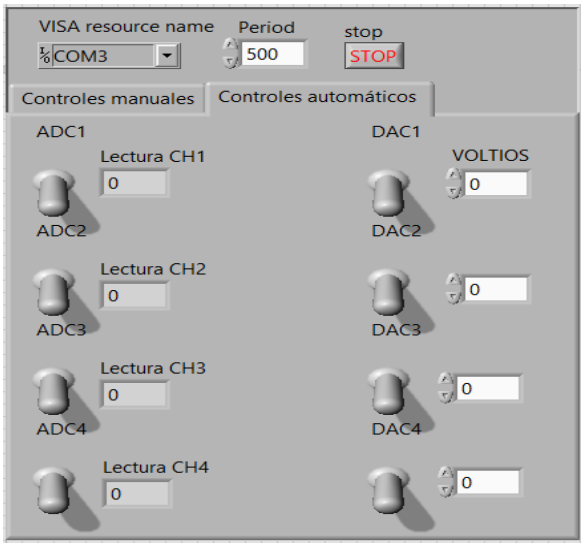


Figura 6. Controles automáticos de la aplicación "tester.vi".

Cabe destacar que los controles que se observan en la parte superior de la Figura 6 son controles especiales que no tienen que ver realmente con los datos de lectura ni de escritura. El primero de ellos *"VISA resource name"* especifica el puerto virtual COMX que vamos a utilizar para la comunicación por el cable USB. En este proyecto se ha establecido como valor por defecto el COM3. El siguiente dato es el periodo de repetición del bucle temporizado que mantiene en funcionamiento la aplicación. Y por último tenemos el botón de stop, control con el que seremos capaces de parar el bucle temporizado que estábamos ejecutando.

7. PRUEBA DE CONCEPTO

Finalmente se ha montado un prototipo funcional de la TAD_USB_32F4, y se ha realizado un control real para comprobar que el diseño es capaz de realizar el objetivo para el cual ha sido diseñado.

Esta prueba de concepto consiste en el control de posición de 4 motores, utilizando el prototipo y una aplicación en LabVIEW que calculaba las señales de control a partir de las señales de entrada. El objetivo de esta prueba no es realizar un control preciso, si no demostrar que la TAD_USB_32F4 es funcional y cumple los objetivos buscados en este trabajo.

Se ha establecido un control proporcional para realizar esta prueba, con una constante de valor K= -0.03. En la aplicación de LabVIEW se calcula la posición del motor utilizando el voltaje proporcionado por el motor, se calcula el error con la referencia establecida y se multiplica por la constante K para calcular la acción de control. De esta manera, en la aplicación de LabVIEW será necesario utilizar las librerías programadas en este proyecto para manejar los convertidores.

La aplicación diseñada para el control tiene dos modos de funcionamiento, referencia manual o automático. Manual sirve para probar los cuatro motores por separado, pudiendo cambiar la referencia a cada uno. A la derecha de estos controles aparecen los indicadores numéricos que representan distintas variables del proceso, como el voltaje que dan los motores, la posición del motor o la señal de control aplicada.

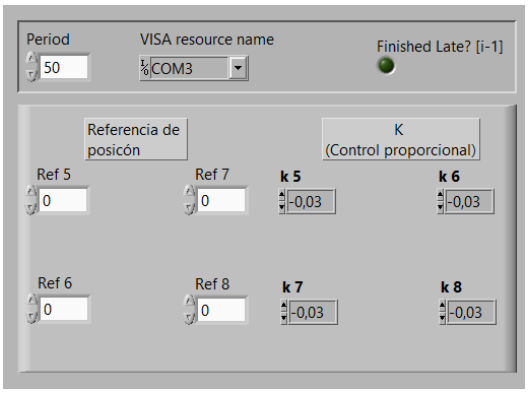


Figura 7. Controles manuales de la referencia, la constante proporcional, el periodo y el puerto virtual COM.

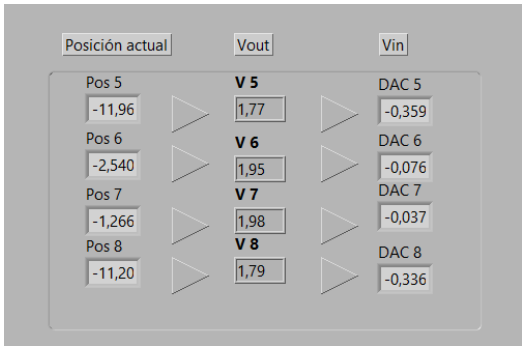


Figura 8. Indicadores de posición, voltaje del motor y señal de control.

Por otro lado, el modo automático utiliza la función "*PID Setpoint Profile VI*", la cual permite establecer una secuencia de puntos en la referencia, en distintos instantes. Es decir, en vez de indicar la referencia, esta cambiará con el tiempo según la secuencia indicada.

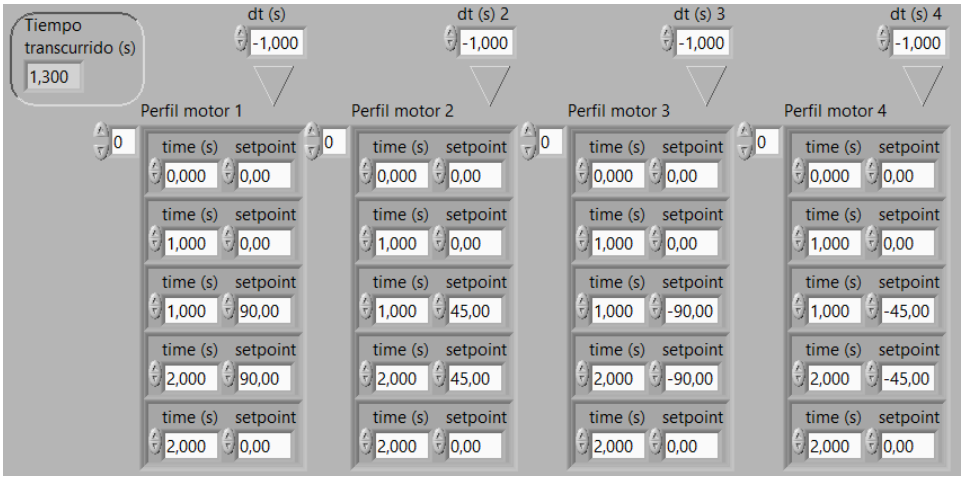


Figura 9. Secuencia establecida para las referencias en los motores.

Para ambos casos existen los botones de: Pausar, Iniciar y Stop. El botón de Pausar detiene el procesado de datos, es decir que solo detiene la adquisición y escritura de datos. Iniciar pone en marcha el procesado de datos a través de la placa. Y el botón Stop detiene la aplicación directamente.

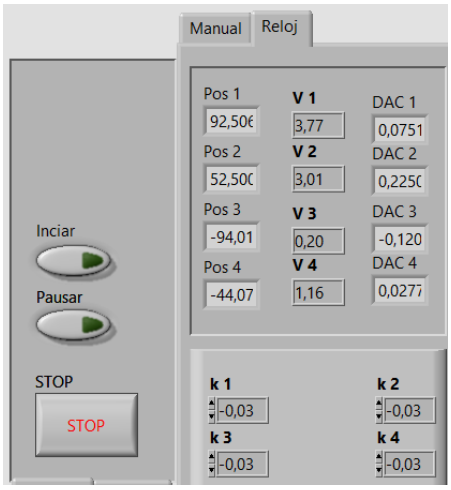


Figura 10. Botones de Iniciar, Pausar y Stop, junto con los indicadores y controles del modo automático.

Por último, tenemos la parte de representación gráfica, donde podemos observar las señales que se están generando. En las gráficas lineales veremos en color blanco el valor de la referencia, y en color rojo el valor de la posición del motor.

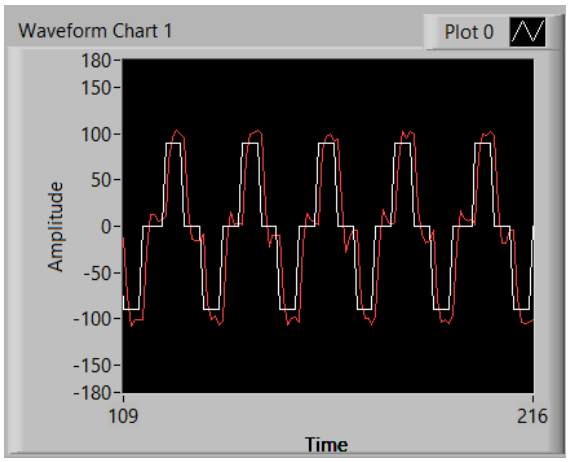


Figura 11. Gráfica de la referencia y la posición del motor 1.

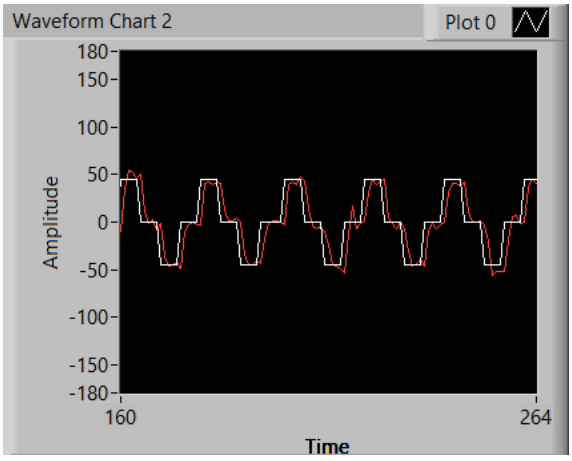


Figura 12. Gráfica de la referencia y la posición del motor 2

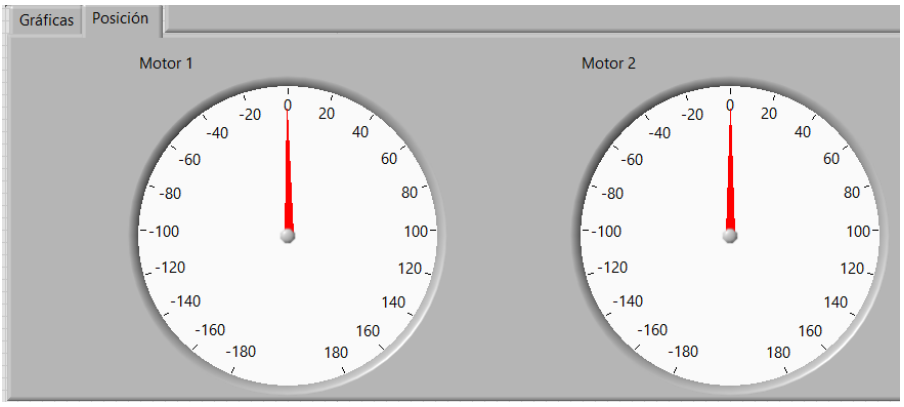


Figura 13. Indicador gráfico de posición de los motores 1 y 2.

8. TABLA DE CARACTERÍSTICAS

	Valor	Unidad
Tiempo de adquisición ADC	2	ms
Tiempo de escritura DAC	2	ms
Rango de entrada	[-10, 10+]	V
Impedancia de entrada	70	KΩ
Impedancia de salida	15.63	Ω

Tabla 4. Tabla de características de la TAD_USB_32F4

9. CONCLUSIONES

Una vez llegados a este punto, podemos concluir que se han cumplido los objetivos establecidos para este proyecto. El resultado obtenido es una TAD de bajo coste con capacidad para cuatro entradas analógicas y cuatro salidas analógicas en el rango de ±10V, que permitirá realizar el control de cuatro lazos distintos. Finalmente se ha conseguido montar un prototipo totalmente funcional, a pesar de que han sido necesarios diversos parches para completar su funcionamiento, ha sido con este prototipo con el que se ha realizado la prueba de concepto explicada en el apartado anterior. Además, la adición de un programa de pruebas, como es el *"tester.vi"* le aportan valor añadido al proyecto, haciendo más sencilla su comprensión. Por tanto, el producto final es la combinación de *software* y *hardware* necesaria para conseguir el funcionamiento explicado a lo largo de este documento.

Este proyecto sirve como base para crear nuevos diseños, utilizando las librerías que se aportan en este trabajo, otra persona podrá crear un programa que realice, por ejemplo, el control de un proceso complejo de cuatro variables, o controlar cuatro procesos distintos. Todas estas combinaciones se dejan al libre albedrío del programador. Además, este proyecto puede ser reprogramado con relativa facilidad, permitiendo que otros programadores utilicen este diseño para darle un enfoque distinto, como podría ser una simulación de "*Hardware in the loop"* o un controlador tipo PID con cuatro lazos de control.

El diseño de las aplicaciones a través de los distintos *software* ha requerido de una planificación previa. Esto ha servido a la hora de la creación de programas, dado que, si se desarrolla la aplicación de un software concreto, sin tener en cuenta cómo será la comunicación entre ellas, o como se comprenderán entre ellas, provocará errores y pérdidas de tiempo innecesarias. Por eso se estableció una estructura de envío y recepción de datos con unas condiciones concretas desde un principio. Como ejemplo de lo mencionado, tenemos el protocolo de comunicación por puerto USB, o la configuración de los componentes electrónicos presentes en el microcontrolador.

Como es normal, se han cometido errores en la placa prototipo que se ha montado, pero gracias a estos, el alumno ha consolidado conocimientos y se ha mejorado el producto final cada vez que se subsanaba un error. Debido a la falta de experiencia en el diseño de PCB, y que ha sido el primer prototipo, la placa ha sufrido distintos cambios, si se comenzara de nuevo el diseño, con todos los conocimientos adquiridos, podría mejorarse el resultado.

BIBLIOGRAFÍA

[1] STMicroelectronics. STM32 Nucleo-64 development board with STM32F446RE MCU, supports Arduino and ST morpho connectivity, 2018.

- http://www.st.com/en/evaluation-tools/nucleo-f446re.html [Consultado:junio 2018]

[2] *ARM. Cortex-M4 processor, 2018.*

- https://developer.arm.com/products/processors/cortex-m/cortex-m4 [Consultado: junio 2018]

[3] *RS Componentes. DesignSpark, 2018.*

- https://www.rs-online.com/designspark/home [Consultado: junio 2018]

[4] *STMicroelectronics. STM32Cube initialization code generator, 2018.*

- http://www.st.com/en/development-tools/stm32cubemx.html [Consultado: junio 2018]

[5] *STMicroelectronics. System Workbench for STM32: free IDE on Windows, Linux and OS X*

- http://www.st.com/en/development-tools/sw4stm32.html [Consultado: junio 2018]

[6] National Instruments. NI LabVIEW 2018, 2018.

- http://www.ni.com/es-es/shop/labview/labview-details.html [Consultado: junio 2018]

[7] EUROCIRCUITS. NAKED Proto, 2018.

- https://www.eurocircuits.com/blog/NAKED-proto/ [Consultado: junio 2018]

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA TARJETA DE ADQUISICIÓN DE DATOS USB CON 4 CANALES ANALÓGICOS DE ENTRADA/SALIDA BASADA EN LA PLACA NUCLEO-F446RE DE ST, Y LIBRERÍAS PARA LabVIEW

DOCUMENTO Nº2:

PRESUPUESTO

AUTOR: GUILLERMO JOAQUÍN ALITE CEREZUELA

TUTOR: JUAN MANUEL HERRERO DURÀ

Curso Académico 2017-2018

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN 5
2. CUADRO DE PRECIOS Nº1: MANO DE OBRA 5
3. CUADRO DE PRECIOS Nº2: MATERIALES Y AMORTIZACIONES 5
4. CUADRO DE PRECIOS Nº3: PRECIOS PARCIALES 5
5. CUADRO DE PRECIOS Nº4: PRECIOS DESCOMPUESTOS 6
6. RESULTADO FINAL 7
7. ESTUDIO DE VENTA 7

ÍNDICE DE TABLAS

Tabla 1. Salario de la mano de obra. 5
Tabla 2. Coste del material utilizado y sus amortizaciones. 5
Tabla 3. Tabla de precios parciales. 5
Tabla 4. Tabla de precios descompuestos, Capítulo 1. 6
Tabla 5. Tabla de precios descompuestos, Capítulo 2. 6
Tabla 6. Tabla de precios descompuestos, Capítulo 3. 6
Tabla 7. Tabla de precios descompuestos, Capítulo 4. 6
Tabla 8. Tabla de precios de producción del hardware. 7

1. INTRODUCCIÓN

El presupuesto contempla el coste monetario que sería necesario para diseñar y realizar este proyecto. Incluirá por tanto los costes relacionados con el diseño del *hardware* TAD_USB_32F4 y de las aplicaciones de *software* pertenecientes al microcontrolador y al entorno de LabVIEW. Por último, se ha realizado un estudio del coste por unidad fabricada de la TAD_USB_32F4, que solo incluirá los costes de fabricación de las placas.

2. CUADRO DE PRECIOS Nº1: MANO DE OBRA

Código	Descripción	Precio (€/h)
MO.1	Graduado en Ingenieria en Tecnologías Industriales (GITI)	20,00
MO.2	Técnico de laboratorio	12,00

Tabla 1. Salario de la mano de obra.

Se han considerado 221dias/año como jornada laboral, contando así con 1768h/año con una jornada diaria de 8 h.

3. CUADRO DE PRECIOS Nº2: MATERIALES Y AMORTIZACIONES

Código	Descripción	Precio (€)	Precio (€/año)	Precio (€/h)
Software				
MAT.S1	STM32CubeMX	X	0,00	0,00
MAT.S2	SW4STM32	X	0,00	0,00
MAT.S3	Library Loader	X	0,00	0,00
MAT.S4	NI LabVIEW 2018	X	3451	1,95
MAT.S5	DesignSparck PCB 8.0	X	0,00	0,00
Hardware				
MAT.H1	Ordenador portátil	1055	263,75	0,15
MAT.H2	Placa de pruebas STM32F445RE	13.44	X	X
MAT.H3	PCB de diseño (Modelo NAKED)	10,79	X	X
MAT.H4	Componentes electrónicos	41,37	X	X

Tabla 2. Coste del material utilizado y sus amortizaciones.

Se ha considerado una vida útil del MAT.H1 ordenador portátil de 4 años.

4. CUADRO DE PRECIOS Nº3: PRECIOS PARCIALES

Código	Unidades	Descripción	Medición	Precio	Importe
Capítulo 1. Diseño del circuito impreso (PCB TAD_USB_32F4)					
UO1	Uds.	Diseño del circuito eléctrico y colocación de los elementos electrónicos sobre la placa.	1	1.644,24	1.644,24
		Total Capítulo 1			**1.644,24 €**
Capítulo 2. Desarrollo de aplicaciones en base a los *softwares* utilizados					
U02.1	Uds.	Diseño de la aplicación destinada al microcontrolador	1	1438,71	1438,71
U02.2	Uds.	Diseño de las librerias mediante LabVIEW	1	1352,52	1352,52
		Total Capítulo 2			**2.791,23 €**
Capítulo 3. Ensayos sobre el dispositivo					
U03	Uds.	Montaje de la placa, ensayo sobre modelos reales y recopilación de datos	1	1190,632	1190,632
		Total Capítulo 3			**1.190,63 €**
Capítulo 4. Tratamiento de la información recogida en el proyecto.					
UO4	Uds.	Redacción de los documentos.	1	616,59	616,59
		Total Capítulo 4			**616,59 €**

Tabla 3. Tabla de precios parciales.

5. CUADRO DE PRECIOS Nº4: PRECIOS DESCOMPUESTOS

Capítulo 1. Diseño del circuito impreso (PCB TAD_USB_32F4)					
UO1		Diseño del circuito eléctrico y colocación de los elementos electrónicos sobre la placa.			
Comprende la implementación de los circuitos eléctricos, la distribución de todos los componentes electrónicos y su conexionado.					
Código	**Unidades**	**Descripción**	**Rendimiento**	**Precio**	**Importe**
MO.1	h	Graduado en Ingeniería en tecnologías industriales	80,00	20,00	1600,00
MAT.S3	h	Library Loader	80,00	0,00	0,00
MAT.S5	h	DesignSpark PCB 8.0	80,00	0,00	0,00
MAT.H1	h	Ordenador portátil	80,00	0,15	12,00
	%	Costes directos complementarios	0,02		32,24
		Coste Total de la UO1			**1.644,24 €**

Tabla 4. Tabla de precios descompuestos, Capítulo 1.

Capítulo 2. Desarrollo de aplicaciones en base a los *softwares* utilizados					
UO2.1	Unidades	Diseño de la aplicación destinada al microcontrolador			
Incluye la programación del algortimo principal y el desarrollo de la estructura de comunicación.					
Código	**Unidades**	**Descripción**	**Rendimiento**	**Precio**	**Importe**
MO.1	h	Graduado en Ingeniería en tecnologías industriales	70,00	20,00	1400,00
MAT.S1	h	STM32CubeMX	70,00	0,00	0,00
MAT.S2	h	SW4STM32	70,00	0,00	0,00
MAT.H1	h	Ordenador portátil	70,00	0,15	10,50
	%	Costes directos complementarios	0,02		28,21
		Coste Total de la UO2.1			**1.438,71 €**
UO2.2	Unidades	Diseño de las librerias mediante LabVIEW			
Abarca la programación de las librerías que controlan los convertidores.					
Código	**Unidades**	**Descripción**	**Rendimiento**	**Precio**	**Importe**
MO.1	h	Graduado en Ingeniería en tecnologías industriales	60,00	20,00	1200,00
MAT.S1	h	NI LabVIEW 2018	60,00	1,95	117,00
MAT.H1	h	Ordenador portátil	60,00	0,15	9,00
	%	Costes directos complementarios	0,02		26,52
		Coste Total de la UO2.2			**1.352,52 €**
		Coste total de la UO2			**2.791,23 €**

Tabla 5. Tabla de precios descompuestos, Capítulo 2.

Capítulo 3. Ensayos sobre el dispositivo					
UO3	Unidades	Montaje de la placa, ensayo sobre modelos reales y recopilación de datos			
Incluye el montaje de la PCB, las pruebas hechas para comprobar errores y los ensayos de prueba para la recopilación de datos.					
Código	**Unidades**	**Descripción**	**Rendimiento**	**Precio**	**Importe**
MO.1	h	Graduado en Ingeniería en tecnologías industriales	40,00	20,00	800,00
MO.2	h	Técnico de laboratorio	20,00	12,00	240,00
MAT.S2	h	SW4STM32	60,00	0,00	0,00
MAT.S4	h	NI LabVIEW 2018	60,00	1,95	117,00
MAT.H1	h	Ordenador portátil	60,00	0,15	9,00
MAT.H2	h	Placa de pruebas STM32F445RE	1,00	13,44	13,44
MAT.H3	h	PCB de diseño (Modelo NAKED)	1,00	10,79	10,79
MAT.H4	h	Componentes electrónicos	1,00	41,37	41,37
	%	Costes directos complementarios	0,02		24,63
		Coste Total de la UO1			**1.190,63 €**

Tabla 6. Tabla de precios descompuestos, Capítulo 3.

Capítulo 4. Tratamiento de la información recogida en el proyecto.					
UO4	Unidades	Redacción de los documentos.			
Abarca la redacción de todos los documentos del TFG: memoria, presupuesto y anexos.					
Código	**Unidades**	**Descripción**	**Rendimiento**	**Precio**	**Importe**
MO.1	h	Graduado en Ingeniería en tecnologías industriales	30,00	20,00	600,00
MAT.H1	h	Ordenador portátil	30,00	0,15	4,50
	%	Costes directos complementarios	0,02		12,09
		Coste Total de la UO1			**616,59 €**

Tabla 7. Tabla de precios descompuestos, Capítulo 4.

6. RESULTADO FINAL

Capítulo 1. Diseño del circuito impreso (PCB TAD_USB_32F4)……1644.24€

Capítulo 2. Desarrollo de aplicaciones en base a los softwares utilizados……2791.23€

Capítulo 3. Ensayos sobre el dispositivo……..……1190.63€

Capítulo 4. Tratamiento de la información recogida en el proyecto……..616.59€

PRESUPUESTO TOTAL DE EJECUCIÓN MATERIAL……**6242.69€**

Gastos generales (12%)……...….749.12€

Beneficio industrial (6%)……...419.5€

PRESUPUESTO TOTAL DE INVERSIÓN……..…..7411.32€

IVA (21%)……..……1556.37€

PRESUPUESTO BASE DE LICITACIÓN……8967.69€

7. ESTUDIO DE VENTA

Se ha estudiado el caso de querer reembolsar el coste de este proyecto. Para lograr este objetivo se ha establecido una producción de 1000 unidades del *hardware* que este proyecto presenta, la unión de la placa STM32F446RE y de la TAD_USB_F446. Los costes quedan reflejados en la siguiente tabla.

Elemento	Precio unitario	Total
Componentes electrónicos	23,99	23.990,00
Circuito impreso	5,9	5.900,00
Soldar componentes	12	12.000,00
STM32F446RE	11,93	11.930,00
Total	53,82	**53.820,00 €**

Tabla 8. Tabla de precios de producción del hardware.

El coste de adquirir cada unidad del producto cuesta 53,82€, y por tanto para poder recuperar lo invertido en el diseño del proyecto y en fabricar las 1000 unidades, el precio al que se debería vender este producto es de 116.6€.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA TARJETA DE ADQUISICIÓN DE DATOS USB CON 4 CANALES ANALÓGICOS DE ENTRADA/SALIDA BASADA EN LA PLACA NUCLEO-F446RE DE ST, Y LIBRERÍAS PARA LabVIEW

ANEXO I.

MANUAL DEL USUARIO

AUTOR: GUILLERMO JOAQUÍN ALITE CEREZUELA

TUTOR: JUAN MANUEL HERRERO DURÀ

Curso Académico 2017-2018

ÍNDICE DE CONTENIDO

1. REQUISITOS TÉCNICOS 5
2. CONEXIONES ENTRE DSPOSITIVOS 5
3. APLICACIÓN *"tester.vi"* 6
4. LIBRERIAS 11
BIBLIOGRAFÍA 13

ÍNDICE DE FIGURAS

Figura 1. Esquema de conexionado de la PCB. 6
Figura 2. Función de la paleta VISA que inicializa el puerto serie. 6
Figura 3. Control manual de la aplicación "tester.vi" 7
Figura 4. Aplicación "tester.vi", diagrama de bloques del control manual. 8
Figura 5.Aplicación "tester.vi", control automático. 9
Figura 6.Aplicación "tester.vi", diagrama de bloques del control automático. 10
Figura 7. Función "ESCRIBIR_DAC" de la librería. 11
Figura 8. Función "LEER_ADC" de la librería 12

1. REQUISITOS TÉCNICOS

Tanto para utilizar las librerías en nuevas aplicaciones, como para poder ejecutar la aplicación *"tester.vi"*, es necesario disponer de un ordenador con los requisitos que la aplicación LabVIEW necesita. Estas características son distintas para cada sistema operativo, por tanto, es recomendable revisar la página web [1] y consultar que se cumplen las especificaciones.

2. CONEXIONES ENTRE DSPOSITIVOS

El dispositivo tiene distintos tipos de conexiones, para efectuar las comunicaciones harán falta los siguientes cables:

- Un cable de tipo *"USB Type A to Mini B"* [2] para la alimentación del dispositivo, que sirve también para la comunicación y programación del micro de la placa.
- Un juego de cables con al menos una terminación macho, para realizar las conexiones de las señales de entrada, salida y masas, a poder ser de distintos colores para distinguir las entradas con mayor facilidad.

Estos cables junto con la siguiente lista de elementos son el *hardware* necesario para realizar este proyecto:

- Un ordenador que cumpla los campos explicados en el apartado de requisitos técnicos de este manual.
- La placa STM32F446RE.
- El circuito impreso que se ha diseñado en este proyecto.

Para realizar una conexión segura se recomienda seguir estos pasos:

1) Descargar e instalar los *drivers* necesarios para el control de la placa STM32F446.
2) Conexionado de E/S:
   - Conectar las masas de las entradas con las de la PCB. (Cable **negro)**
   - Conectar las señales de entrada a los conectores de entrada analógicos de la PCB. (Cables de colores)
   - Conectar las salidas de los DAC elegidos a las señales de control del proceso. (Cables de colores)
3) Conexión de la alimentación (Cable "*USB Type A to Mini B")*:
   - Conectar el extremo tipo A del cable a una fuente de alimentación que proporcione 5V de tensión continua.
   - Conectar el extremo tipo *"Mini B"* al puerto USB de la placa F446.

En la Figura 1 se muestra un esquema de cómo sería una conexión con todos los canales de entrada y salida ocupados. Es recomendable conectar el cable USB el último, debido a que es la alimentación. De esta manera antes de conectar el USB se comprueba que todos los demás cables estén conectados donde deben y que no existe contacto entre ellos. Una vez hecho esto se conecta la alimentación a la placa, que instantáneamente ejecutará el programa que contiene, es importante recordar que existe un led en la placa STM32F446 que parpadea cuando la placa está en funcionamiento.

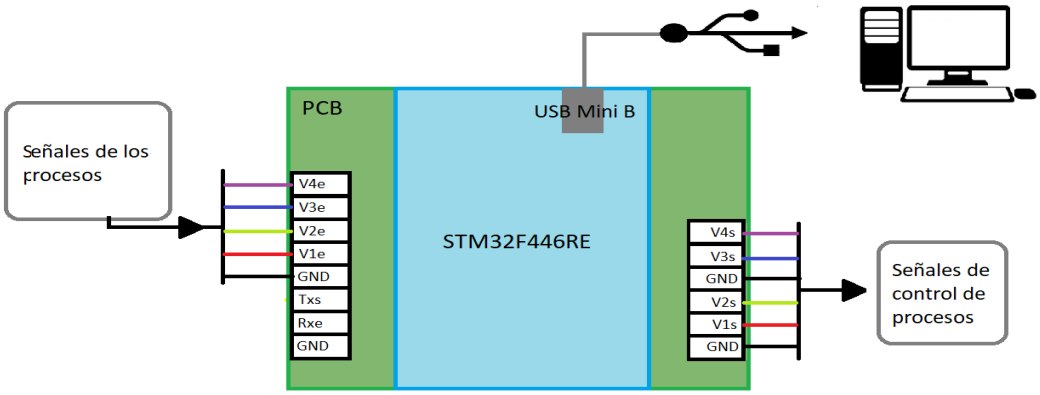


Figura 1. Esquema de conexionado de la PCB.

3. APLICACIÓN *"tester.vi"*

Esta aplicación diseñada en LabVIEW, que está contenida en el producto final de este proyecto, sirve como ejemplo de aplicación de las librerías diseñadas, así como de interfaz gráfica para la comprobación del correcto funcionamiento de la TAD.

Contiene dos modos de funcionamiento, manual y automático. En el modo manual aparecen ocho botones que ejecutan una sola vez, cualquiera de las ocho ordenes programadas. Mientras que en el modo automático podemos ejecutar hasta ocho ordenes de manera continua.

De la misma manera, sirve como ejemplo de inicialización de un puerto serie en LabVIEW, mediante la paleta VISA. Para la aplicación "*tester.vi*" los valores de configuración del puerto serie son los siguientes.

- Enable termination char = False.
- Visa resource name = COM3.
- Baud rate = 115200 bps.
- Data bits = 8.
- Parity = none.
- Stop bit = 1.0.
- Flow control = 0.

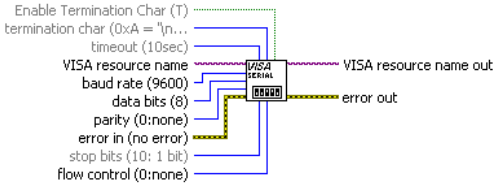


Figura 2. Función de la paleta VISA que inicializa el puerto serie.

Para cambiar de modo manual a automático, y viceversa, bastará con cambiar la pestaña del *tab control*, que controla un *case structure*, este control queda reflejado en la Figura 4.

Los primeros pasos en la aplicación deben ser los siguientes:

- Se elige un periodo.

- Se designa el puerto por el que se va a realizar la comunicación con la placa.
- Elegimos modo manual o automático.

En el caso del modo manual, para leer el voltaje de un canal tendremos que pulsar el botón correspondiente a este. Y en el indicador adyacente aparecerá el valor, en un rango de [-10, 10], en ese instante. Para sacar voltaje por las salidas se procede de manera análoga, pero en este caso escribiendo el valor escogido, en el rango [-10, 10], se pulsa el botón correspondiente a la salida elegida, y si no ha habido ningún fallo, un led verde se encenderá brevemente indicando que los datos han sido recibidos satisfactoriamente, finalmente aparecerá una tensión a la salida.

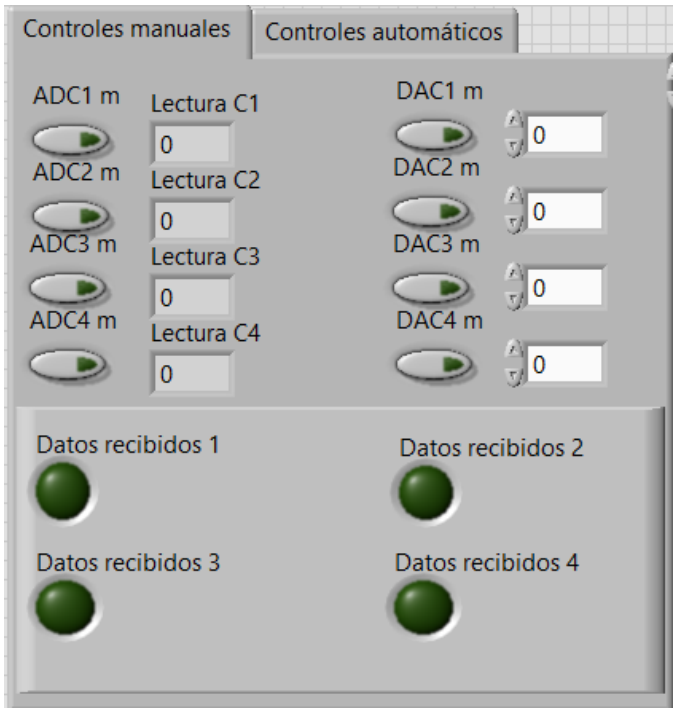


Figura 3. Control manual de la aplicación "tester.vi".

En la Figura 4, se aprecia como en función de los botones establecidos en el control manual, se accederá a un caso u otro del *event structure*, esta parte del funcionamiento se ha diseñado con un *Timeout* de 500ms, reseteando el estado de los leds en el evento que se ejecuta al pasar este tiempo. Todos los botones realizan la misma operación, lo único que cambia es el canal al que hacen referencia.

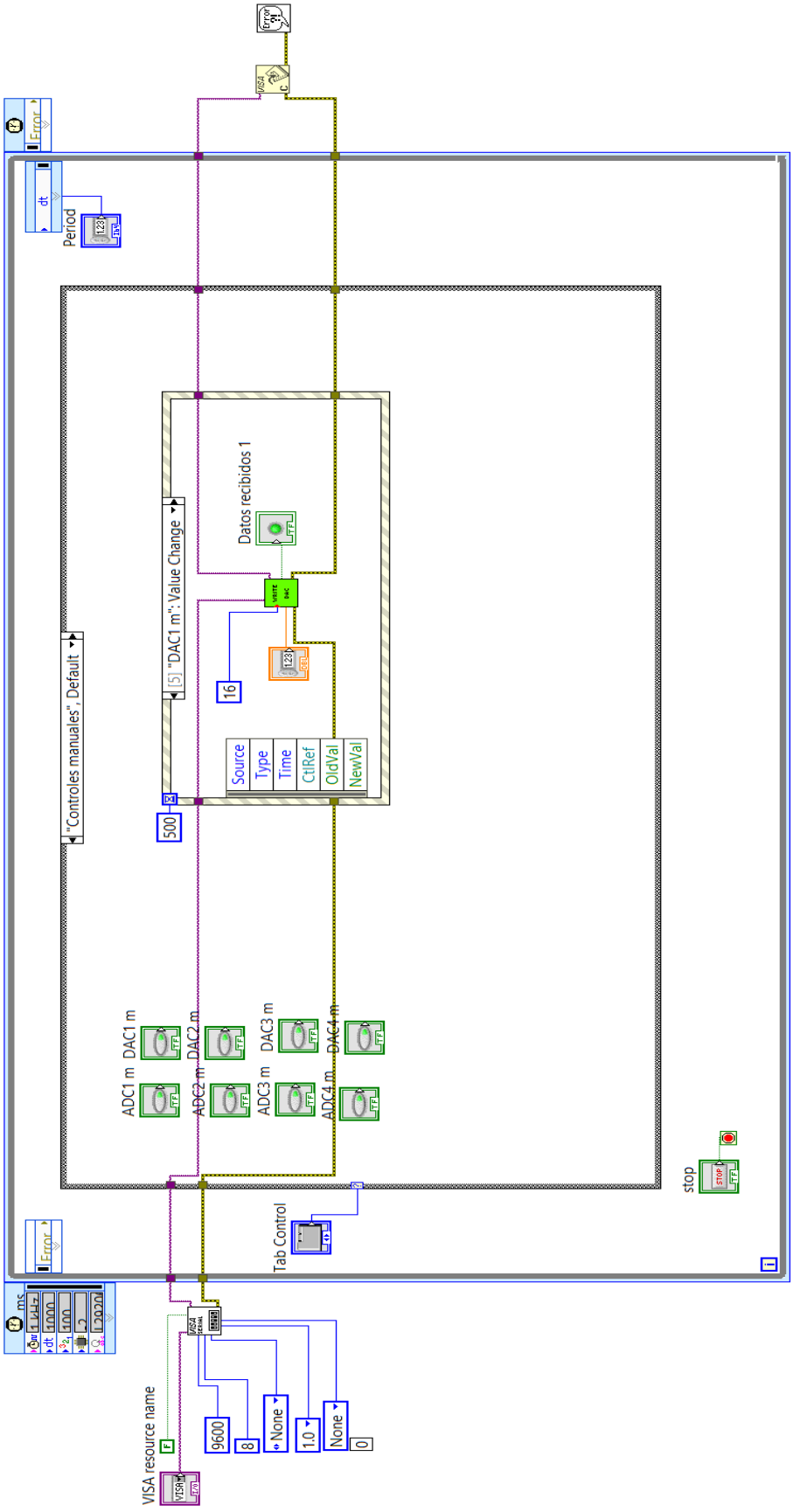


Figura 4. Aplicación "tester.vi", diagrama de bloques del control manual.

Por otro lado, el control automático tiene un funcionamiento parecido en cuanto a las opciones que nos proporciona. La diferencia es que en este caso nos encontramos con interruptores, que al ser activados ejecutarán constantemente la acción indicada. Es decir, con solo levantar el interruptor de un canal del ADC podemos ver cómo cambia a tiempo real el voltaje de entrada, así como activar un DAC e ir cambiando el voltaje de salida mediante el control numérico.

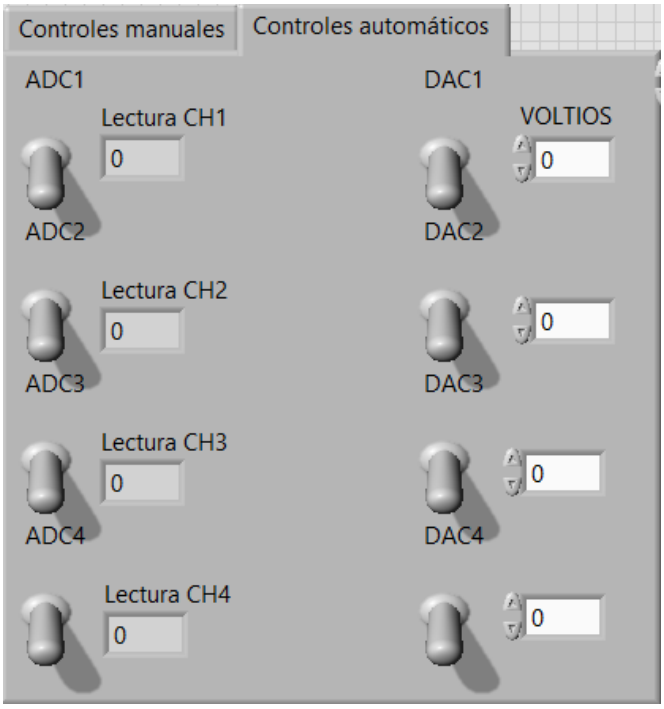


Figura 5.Aplicación "tester.vi", control automático.

Para el caso de los controles automáticos se ha optado por una estructura que va consultado el estado de los interruptores uno por uno. En caso de que el interruptor esté encendido ejecutará la acción relacionada con este, cuando acabe la operación pasará a comprobar el siguiente caso. Si el interruptor está apagado simplemente avanzará al siguiente. Cuando llega al último comenzará de nuevo, y el programa seguirá así hasta que se cambie el control a manual o se detenga el proceso.

Para realizar esta estructura se han utilizado dos *case structure,* el primero se encarga de avanzar a la siguiente orden, controlado por un valor entre [0, 7], reflejado en *Numeric,* Figura 6. Dentro de este está programado el *case structure* que transforma el valor *booleano* del interruptor en 1 o 0, y lo compara con una constante. Si la comparación es correcta se ejecutará el caso *True*.

Es recomendable probar primero los controles manuales porque permiten conocer de un vistazo si la transmisión ha sido correcta gracias al cambio en los leds. Y seguidamente usar los controles automáticos, para probar distintos periodos y escoger el que más se adecúe según la dinámica de las señales.

La manera en que las funciones de escritura y lectura forman las ordenes que serán enviadas a la PCB queda detallado en el ANEXO III. MANUAL DE PROGRAMACIÓN, así como la comprobación de las condiciones establecidas en el protocolo de comunicación.

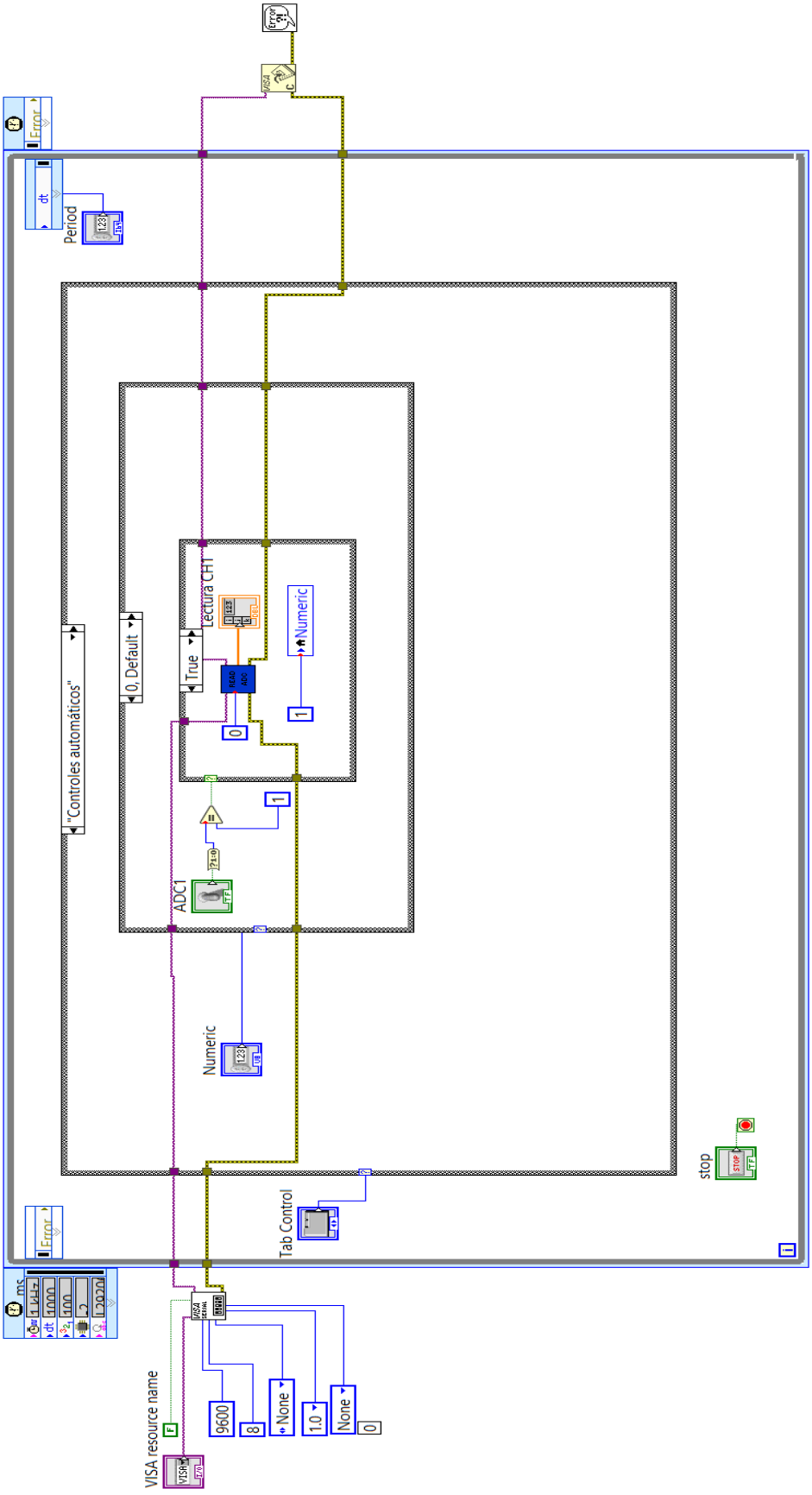


Figura 6.Aplicación "tester.vi", diagrama de bloques del control automático.

4. LIBRERIAS

En este apartado se explicará cada una de las entradas y salidas de las funciones programadas en LabVIEW que constituyen la librería de manejo de la placa TAD_USB_32F4. Hay dos funciones:

- ESCRIBIR_DAC: En esta función se forma el vector, con una estructura como la mostrada en la Tabla 1 del DOCUMENTO Nº1. MEMORIA DEL PROYECTO, que contendrá los *bytes* necesarios para completar una escritura en uno de los DAC disponibles. Los *bytes* que contienen la instrucción son: CMD (segunda posición de la cadena) y el valor a escribir (tercera y cuarta posición de la cadena), codificado en 12 de los 16 *bits* que se dispone para el dato. Es importante remarcar que el dato que se envía desde la aplicación de LabVIEW está multiplicado por 4 para solventar errores, como se explica en el ANEXO III.MANUAL DE PROGRAMACIÓN. Estos 12 *bits* al ser enviados de vuelta en la cadena que se transmite desde el micro al ordenador, contendrán todo 0. Las demás posiciones del vector son datos para comprobar que no se produzca ningún error en la comunicación. Una vez generada la cadena, se encarga de transmitirla a través del cable USB, esperar la respuesta y comprobar que la comunicación se ha ejecutado con éxito.

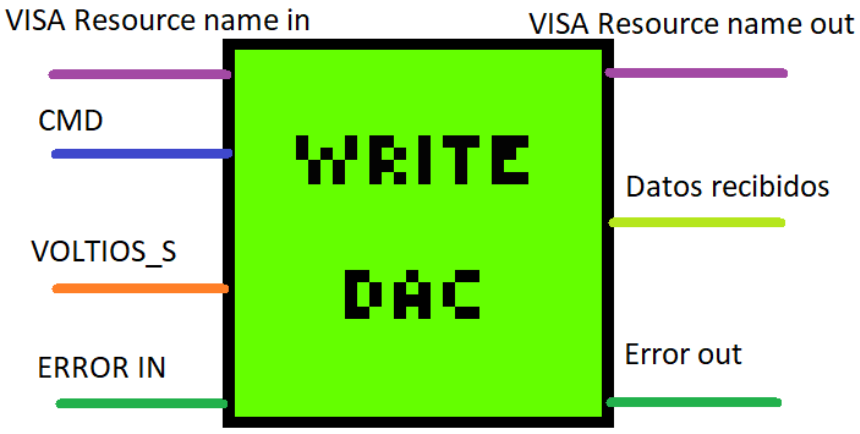


Figura 7. Función "ESCRIBIR_DAC" de la librería.

- LEER_ADC: De la misma manera que en el caso anterior en esta función se genera la cadena con la orden que el micro debe realizar. En este caso la instrucción está codificada únicamente en el *byte* CDM, ya que solo será necesario conocer el canal escogido para leer. Por tanto, se ha decidido que los 12 *bits* reservados para el valor, a la hora del envío, serán 0. En el caso del envío de datos desde el microcontrolador a la aplicación del ordenador, estos *bits* contendrán codificado el dato que se ha adquirido en un espacio de 12 bits, y como en la función anterior este dato está multiplicado por 4 para prevenir funcionamientos erróneos. Por último, cuando el vector está completo, se procede a enviar los *bytes* por el cable USB, esperar una respuesta del micro y comprobar que no existan errores en la transmisión y recepción de datos.

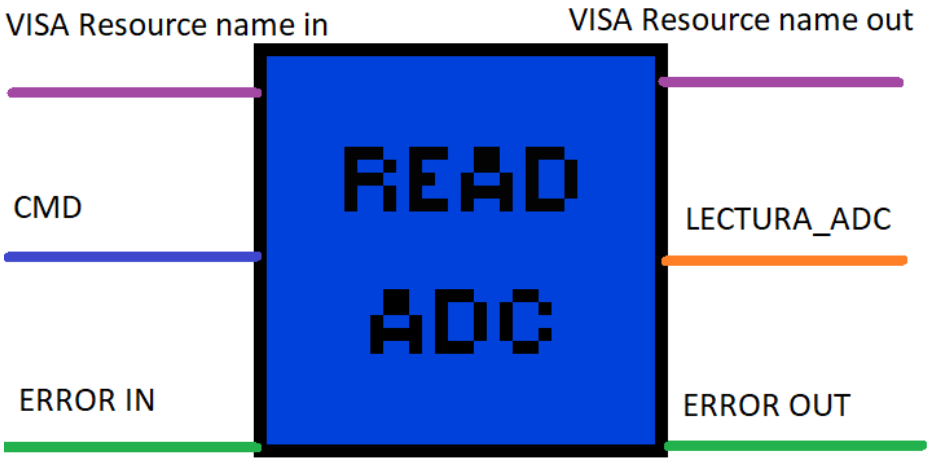


Figura 8. Función "LEER_ADC" de la librería.

Para ambos casos las entradas superiores (color morado) son las referidas al puerto escogido para hacer la comunicación. Bastará con ir al administrador de dispositivos del sistema operativo para conocer el nombre del elemento. De la misma manera la entrada inferior (color verde oscuro) es la misma en ambos casos, y contiene los códigos de error que pueden suceder durante la ejecución del programa. Las entradas en azul representan la variable encargada de seleccionar uno de los cuatro posibles convertidores, los valores reservados para los ADC son 0, 1, 2 y 3, mientras que para los DAC son 16, 17, 18, y 19. La salida de color naranja indican valores de voltajes. En el caso de escritura será el valor del voltaje de salida, y en el caso de lectura será el valor que el convertidor ha calculado. En la función "*ESCRIBIR_DAC"* se ha añadido una salida *booleana* (color verde), valor 1 o 0, que indica si la escritura se ha realizado con éxito.

BIBLIOGRAFÍA

[1] *Nationa Instruments. System Requirements for LabVIEW Development Systems and Modules, 2018.*

- http://www.ni.com/white-paper/53740/en/ [Consultado: junio 2018]

[2] *USB CableConnector Type Guide, 2018.*

- *https://www.newnex.com/usb-connector-type-guide.php* [Consultado: junio 2018]

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA TARJETA DE ADQUISICIÓN DE DATOS USB CON 4 CANALES ANALÓGICOS DE ENTRADA/SALIDA BASADA EN LA PLACA NUCLEO-F446RE DE ST, Y LIBRERÍAS PARA LabVIEW

ANEXO II.

MANUAL DE DISEÑO DEL CIRCUITO IMPRESO

AUTOR: GUILLERMO JOAQUÍN ALITE CEREZUELA

TUTOR: JUAN MANUEL HERRERO DURÀ

Curso Académico 2017-2018

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS 3
1. GENERALIDADES 5
2. DISEÑO DE LA CIRCUITERÍA 5
2.1 ADAPTACIÓN DE LA TESNIÓN DE LOS AMPLIFICADORES OPERACIONALES 5
2.2 ADAPTACIÓN DE LA TENSIÓN DEL PUERTO SERIE 6
2.3 ADAPTACIÓN DE LAS TENSIONES DE ENTRADA ANALÓGICAS 6
2.4 ADAPTACIÓN DE LA TENSIÓN DE SALIDA 8
2.5 ADICIÓN DE 2 DAC MEDIANTE I2C 10
2.6 CONECTORES HEMBRA DE ACOPLE CON LA PLACA STM32F446. 10
2.7 LISTA DE COMPONENTES 11
3. DISEÑO DE LA PCB 12
BIBLIOGRAFÍA 14

ÍNDICE DE FIGURAS

Figura 1. Unidad de adaptación de la tensión de alimentación de los amplificadores operacionales. 6
Figura 2. Unidad de adaptación de la tensión del puerto serie. 6
Figura 3. Unidad de adaptación de las tensiones de entrada analógicas. 7
Figura 4. Amplificador operacional LM124DRG4 (IC4) de 4 canales y conectores para las señales de entrada. 7
Figura 5. Unidad de adaptación de las señales de salida. 8
Figura 6. LM123DRG4 (IC11). 9
Figura 7. Conectores de las señales de salida y amplificador operacional del bloque de adaptación de salida. 9
Figura 8. Unidad de conversión DAC mediante I2C. 10
Figura 9. Adaptadores hembra que acoplan con la placa STM32F446. 10
Figura 10. Esquema realizado en el editor de PCB de DesignSpark PCB 8.0 12
Figura 11. Vista 3D de la PCB proporcionada por el software DesignSpark PCB 8.0. 13

ÍNDICE DE TABLAS

Tabla 1. Lista de componentes utilizados. 12

1. GENERALIDADES

Para el diseño del circuito impreso, o PCB (Printed Circuit Board) TAD_USB_32F4, se han utilizado distintos *software,* en todos los casos se usa una versión gratuita de los programas.

- *DesignSparkPCB 8.0*: plataforma para diseñar los circuitos eléctricos necesarios en el proyecto, que además añade un entorno gráfico para distribuir los elementos electrónicos y sus conexiones de una manera sencilla [1].

- *Library Loader*: Aplicación que nos lleva, a través de un navegador web, al buscador de componentes electrónicos de RS-Components, donde podemos buscar y descargar los *schematics*, que automáticamente se añadirán a las librerías de DesignSparkPCB [2].

- STM32CubeMX: plataforma de entorno gráfico que permite configurar microcontroladores de manera sencilla, generar código para generar un proyecto base sobre la configuración establecida, y conocer las direcciones físicas de los pines que el microcontrolador va a usar.

2. DISEÑO DE LA CIRCUITERÍA

En este apartado se explican los distintos circuitos implementados y sus elementos. Dado que el objeto de diseño tiene que ir acoplado a la placa STM32F446, la PCB tiene unas premisas fijadas.

- Los adaptadores hembra de la PCB diseñada tienen que estar a la misma distancia que las salidas de la placa STM32F446.
- La PCB irá acoplada por la parte inferior, esto obligará a tener precaución con los componentes que tengan una cierta altura.
- Los conectores de las entradas analógicas se situarán en un lado de la PCB y los conectores de las salidas en el extremo opuesto.

Además, el proyecto tiene como objetivo propio tener disponibles cuatro entradas y cuatro salidas analógicas. Con todo esto y teniendo en cuenta la longitud de las conexiones y el número de vías, siempre intentando que sea el menor posible, se ha diseñado la placa.

2.1 ADAPTACIÓN DE LA TESNIÓN DE LOS AMPLIFICADORES OPERACIONALES

Los amplificadores operacionales están integrados en los circuitos que adaptan las tensiones de entrada/salida en ciertos rangos. Están alimentados a ±12V, se hace uso del TMA1205S [3], que amplifica la tensión de 5V hasta los ±12V necesarios.

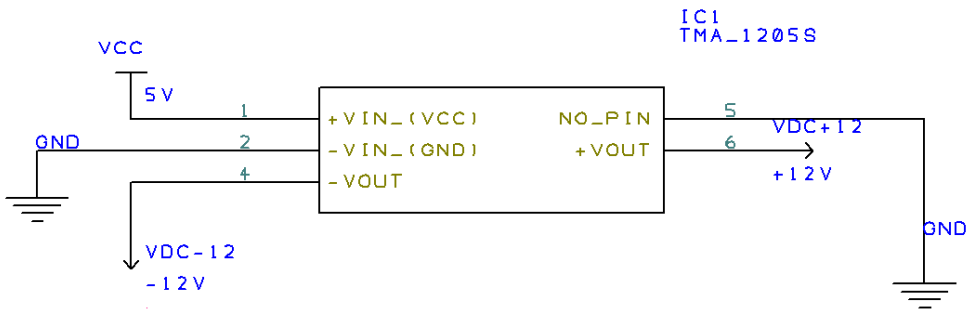


Figura 1. Unidad de adaptación de la tensión de alimentación de los amplificadores operacionales.

2.2 ADAPTACIÓN DE LA TENSIÓN DEL PUERTO SERIE

Los canales de transmisión y recepción del puerto serie trabajan a voltajes distintos del rango aceptado por el microcontrolador. Por tanto, hará falta una etapa de adaptación de estas señales, para que estas puedan ser tratadas por el micro.

Este proceso se lleva a cabo mediante el componente MAX232AEWE [4] que junto con los conectores permitirán la comunicación con el dispositivo.

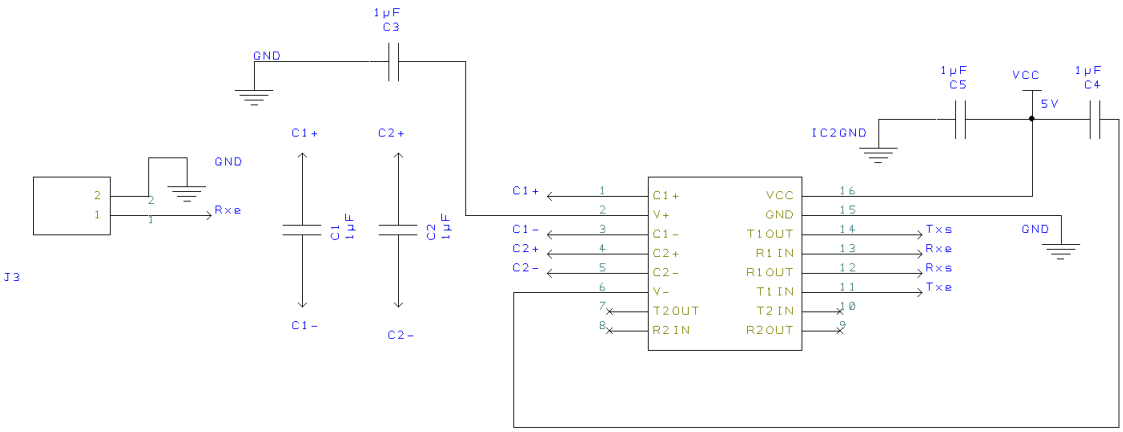


Figura 2. Unidad de adaptación de la tensión del puerto serie.

2.3 ADAPTACIÓN DE LAS TENSIONES DE ENTRADA ANALÓGICAS

Las entradas varían en un rango de ±10V, y dado que el microcontrolador acepta tensiones de 0V a 3.3V, para adaptarlas se ha empleado un circuito sumador-amplificador con las siguientes características de diseño:

- Si V1e=-10V, entonces la salida debe cumplir 1OUTe>0.
- Si V1e=10v, entonces la salida debe cumplir 1OUTe<3.3v

Teniendo en cuenta estas restricciones, se ha llegado a la solución de la Figura 3 y Figura 4, esta es una de las infinitas soluciones que existen variando el valor de las resistencias. Este circuito sigue la expresión (1):

$$1OUTe=1.22(0.123*V1e + 1.25) \quad (1)$$

Sustituyendo los valores máximo y mínimo de tensión, la conversión queda tal que:

- Si V1e=10V ⟶ 1OUTe=3.05V
- Si V1e=-10V ⟶ 1OUTe=0V

El proceso consta de cuatro entradas, que siguen el mismo esquema que la entrada V1e, las salidas de este circuito van conectadas al adaptador hembra de la PCB, que a su vez se conecta con el microprocesador de la placa STM32F446.

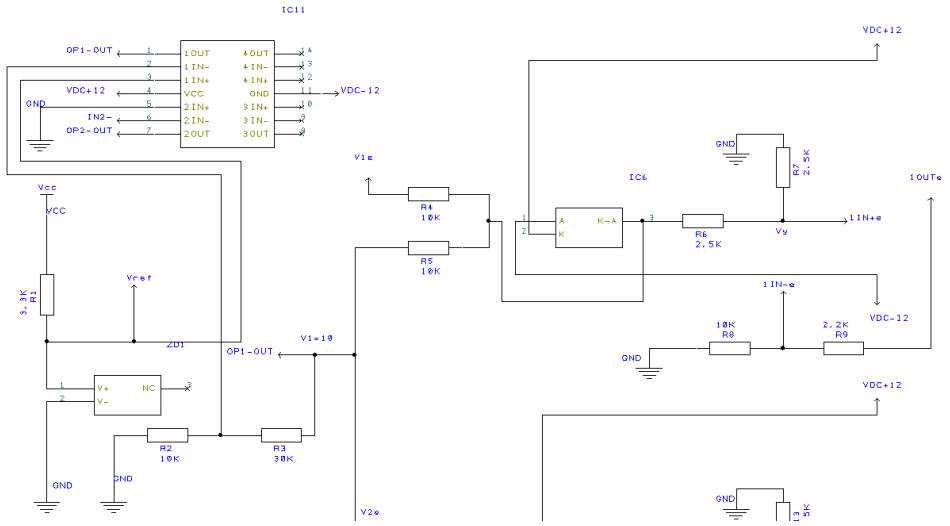


Figura 3. Unidad de adaptación de las tensiones de entrada analógicas.

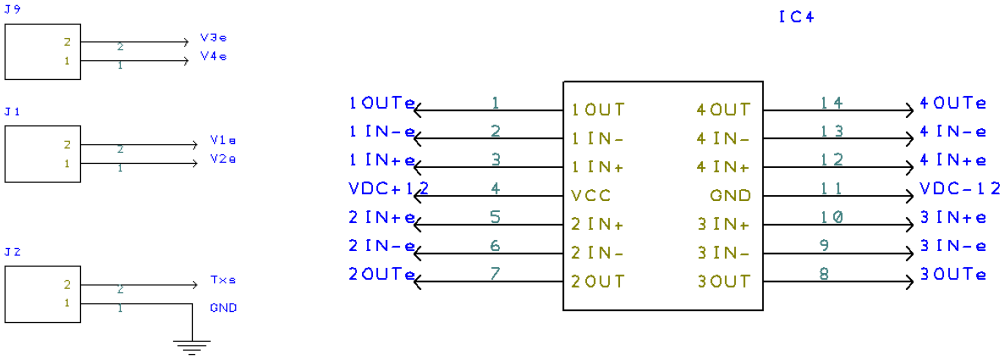


Figura 4. Amplificador operacional LM124DRG4 (IC4) de 4 canales y conectores para las señales de entrada.

En la Figura 3 se aprecia 1 de las 4 salidas del amplificador mostrado en la Figura 4, en estas figuras se pueden distinguir distintos componentes electrónicos:

- Conectores J9/J1/J2: conectores para las señales de entrada y sus masas.
- Amplificadores LM124DRG4 [5]: se distinguen 2 amplificadores distintos:
  - IC11: Mantiene una tensión fija de 10V a su salida.

- IC4: Bloque sumador-amplificador que adapta las entradas “V1e”, ”V2e”,”V3e”,”V4e” a valores de tensión en el rango del microcontrolador.

- Par de diodos de protección MMBD1203 [6]: Protegen los amplificadores operacionales de ser dañados, funcionarán solo en caso de que la tensión supere el valor de ±12V con la que se alimentan los amplificadores.
- Diodo Zener ADR5041ARTZ-R7 [7]: elemento que fija una tensión de 2.5V, “Vref”, permitiendo la calibración de los ADC del microcontrolador, con el objetivo de tener un resultado más preciso al independizarse del rango del microcontrolador [0.3, 3V].

2.4 ADAPTACIÓN DE LA TENSIÓN DE SALIDA

De manera análoga a las entradas analógicas, las salidas analógicas del microcontrolador han de ser adaptadas para llevarlas a la salida. Por tanto, las señales han de pasar por una etapa de amplificación siguiendo ciertas restricciones.

- Si DAC1=0V ⟶ 1OUTs>-10V.
- Si DAC1=3.3V ⟶ 1OUTs<10V.

Respetando estas condiciones se llega al circuito mostrado en la Figura 4, que sigue la expresión:

$$1OUTs=12.12(DAC1-1.65) \quad (2)$$

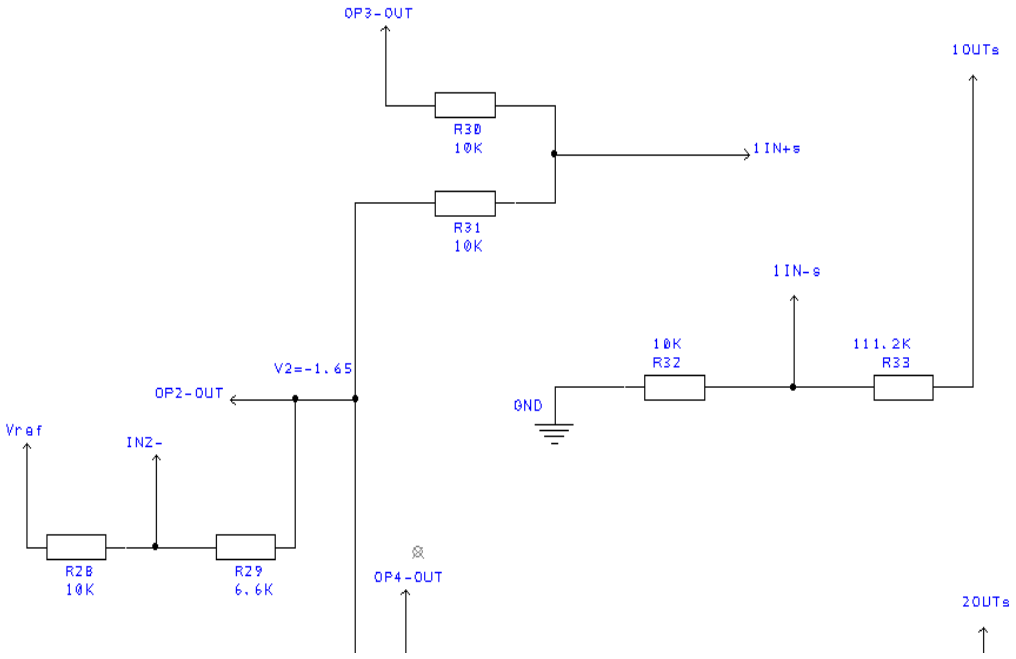


Figura 5. Unidad de adaptación de las señales de salida.

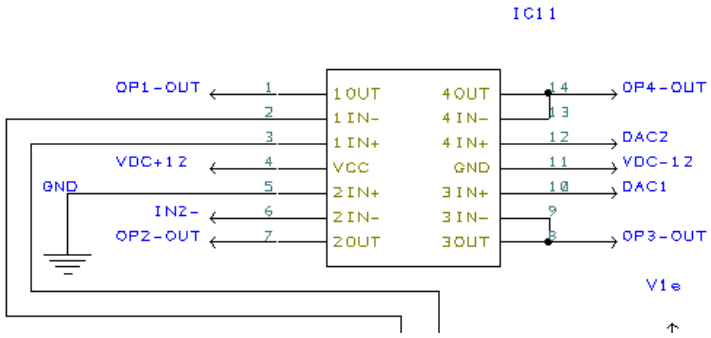


Figura 6. LM123DRG4 (IC11).

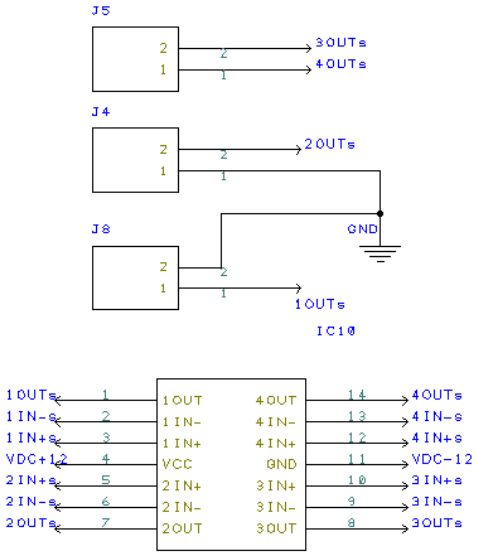


Figura 7. Conectores de las señales de salida y amplificador operacional del bloque de adaptación de salida.

De las figuras anteriores se distinguen distintos elementos electrónicos:

- Elementos J5/J4/J8: Conectores para las salidas y sus masas
- Amplificadores operacionales LM124DRG4: se aprecian 2 operacionales distintos:
  - IC11: Mantiene una tensión de -1.65V a su salida y actúa como seguidor de las señales DAC1 y DAC2.
  - IC10: Bloque sumador-amplificador que adapta las entradas de “DAC1”,”DAC2”,”DAC3” y ”DAC4” a valores de tensión dentro del rango necesario a la salida [-10,10]V.

Cabe destacar que en la Figura 5 sólo se muestra una de las 4 salidas, pero todas funcionan de la misma manera. Además, debido a la falta de capacidad, por parte de los DAC pertenecientes al micro, de mantener los voltajes a su salida, se ha optado por desactivar el buffer, que viene activo por defecto, y utilizar las salidas 3 y 4 del IC11 como buffer de estas señales. El objetivo de este circuito es el de un seguidor, es decir que mediante un amplificador operacional se mantiene la señal proveniente de los DAC, pero sin pérdidas de voltaje.

2.5 ADICIÓN DE 2 DAC MEDIANTE I2C

Con la finalidad de añadir 2 salidas más a la TAD, se ha implementado el elemento AD5339ARMZ [8]. Para realizar la comunicación de este componente con el microcontrolador se utilizan las señales “I2C_SCL” e “I2C_SDA”. Este es uno de los elementos más delicados de la placa. Debido a su pequeño tamaño, tiene los pines muy cerca unos de otros y podía darse el caso de que se cortocircuitaran.

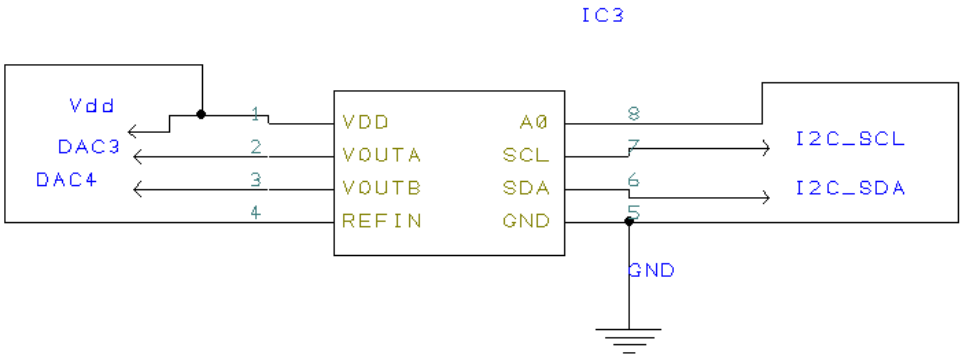


Figura 8. Unidad de conversión DAC mediante I2C.

2.6 CONECTORES HEMBRA DE ACOPLE CON LA PLACA STM32F446.

Una de las condiciones establecidas en el diseño de esta placa era la distancia que debía haber entre los conectores hembra, para que al acoplarse con la placa STM32F446 todos los pines encajaran.

Para ello, los conectores se han situado paralelos y a una distancia de 54.25 mm uno del otro. Este componente es el enlace entre la PCB diseñada, el microcontrolador y el resto de los procesos de la placa F446, con lo que se consigue reproducir las posiciones de la placa F446 sobre la PCB TAD_USB_32F4. De este modo todas las señales que necesiten pasar por elementos de la placa superior primero deberán pasar por estos conectores.

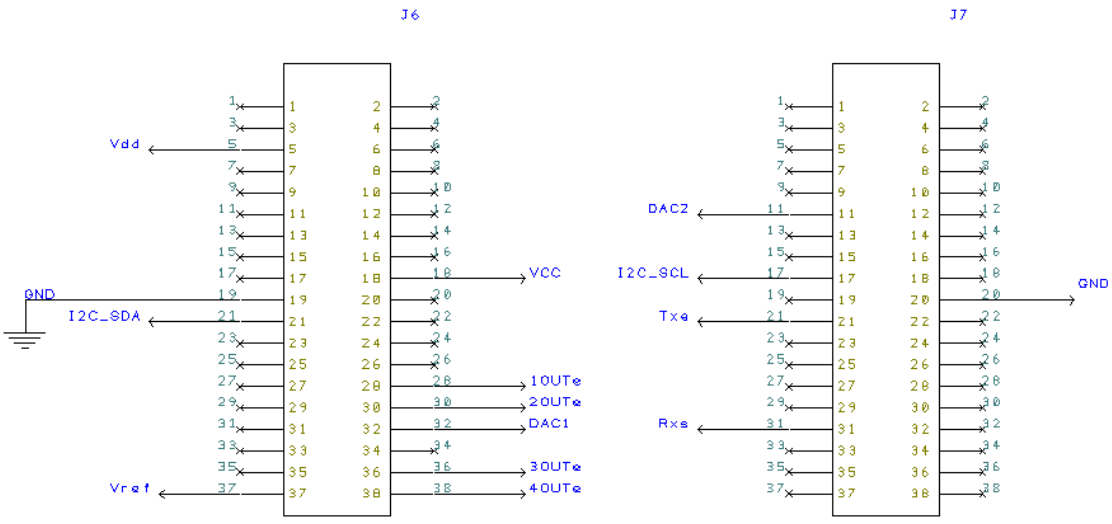


Figura 9. Adaptadores hembra que acoplan con la placa STM32F446.

- Tensiones de entrada, “1OUTe”, “2OUTe”, “3OUTe”, “4OUTe”: Se conectan al ADC (pines PA0,PA1,PC0,PC1).
- Tensiones de salida, “DAC1”, ”DAC2” ,”I2C_SCL”, “I2C_SDA”: Las dos primeras las suministra los DAC del microprocesador (pines PA4,PA5). Las dos últimas señales se utilizan para que el elemento AD5339ARMZ implemente otros 2 DAC, que serían las señales “DAC3” y “DAC4”, obteniendo así 4 posibles salidas.
- Canales de transmisión y recepción del puerto serie Tx y Rx: Situados en los pines PA9 y PA10.
- Tensión de calibración de convertidores: “Vref” en el pin PC3 mantiene una tensión fija de 2.5V para la calibración de los ADC.

2.7 LISTA DE COMPONENTES

Etiqueta	Valor	Encapsulado	Descripción
C1	1µF	0805(SMD)	Condensadores
C2	1µF		
C3	1µF		
C4	1µF		
C5	1µF		
J1	-	1729018(SHDR2W110P0X500_1X2_1000X810X1)(TH)	Conectores entrada/salida
J2			
J3			
J4			
J5			
J8			
J9			
J6	-	7-534998-0(SHDR40W85P254_2X20_5588X670X10)(TH)	Conectores hembra
J7			
ZD1(ADR5041ARTZ-R2)	2,5V	SOT95P237X112-3N(SMD)	Diodo zener, regulador de tensión
IC1(TMA_1205S)	-	NMAXX_SIL(TH)	Conertidor de tensión
IC2(MAX232AEWE+)	-	SOIC127P1032X265-16N(SMD)	Adaptador de tensiones de comunicación por puerto serie
IC3(AD5339ARMZ)	-	SOP65P490X110-8N(SMD)	Convertidor digital analógico de 12 bits
IC4(LM124DRG4)	-	SOIC127P600X175-14N	Amplificadores operacionales de 4 canales
IC10(LM124DRG4)			
IC11(LM124DRG4)			
IC5(MMBD1203)	-	SOT96P237X111-3N	Diodos de protección
IC6(MMBD1203)			
IC8(MMBD1203)			
IC9(MMBD1203)			

R1	3,3KΩ	0805(SMD)	Resistencias
R2	10KΩ		
R3	30KΩ		
R4	10KΩ		
R5	10KΩ		
R6	2,5KΩ		
R7	2,5KΩ		
R8	10KΩ		
R9	2,2KΩ		
R10	10KΩ		
R11	10KΩ		
R12	2,5KΩ		
R13	2,5KΩ		
R14	10KΩ		
R15	2,2KΩ		
R16	10KΩ		
R17	10KΩ		
R18	2,5KΩ		
R19	2,5KΩ		
R20	10KΩ		
R21	2,2KΩ		
R22	10KΩ		
R23	10KΩ		
R24	2,5KΩ		
R25	2,5KΩ		
R26	10KΩ		
R27	2,2KΩ		
R28	10KΩ		
R29	6,6KΩ		
R30	10KΩ		

R31	10KΩ		
R32	10KΩ		
R33	111.2KΩ		
R34	10KΩ		
R35	10KΩ		
R36	10KΩ		
R37	111.2KΩ		
R38	10KΩ		
R39	10KΩ		
R40	10KΩ		
R41	111.2KΩ		
R42	10KΩ		
R43	10KΩ		
R44	10KΩ		
R45	111.2KΩ		

Tabla 1. Lista de componentes utilizados.

*SMD (Surface Mount Device): Encapsulados de superficie.

*TH (Through Hole): Encapsulados de agujero pasante

3. DISEÑO DE LA PCB

La placa diseña consta de 2 capas, la superior (top) y la inferior (bottom), por las cuales discurren las conexiones de la placa. Para pasar de una a otra se utilizan agujeros pasantes (vías), lo que permite que los cables no se entrecrucen, teniendo así una conexión más sencilla.

Todos los componentes utilizados están situados sobre la capa superior de la PCB, y solo se utilizará la capa inferior para el enrutado. El ancho de las pistas se ha dejado por defecto y tiene un valor de 0.15mm, así como el valor del espesor de conductor en las vías, de 0.125mm.

Ha sido necesario poner elementos, tales como el TMA_1205S y los conectores de entrada/salida a una distancia de 4.8mm de los conectores hembra. Esto es debido a que son elementos con una altura considerable, y si estuvieran colocados debajo de la STM32F446 impedirían el acople de ambas placas.

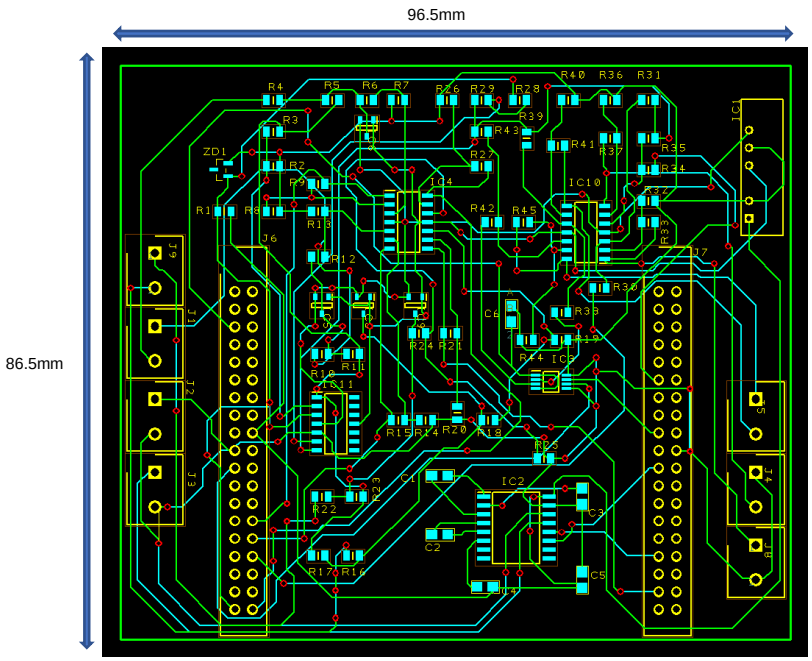


Figura 10. Esquema realizado en el editor de PCB de DesignSpark PCB 8.0.

(Amarillo→Componentes, Azul celeste→Pines de componentes, Cian→ Capa inferior, Verde→Capa superior).

Con todo lo que se ha comentado en este anexo, se ha llegado a la solución que se muestra en la Figura 10. En ella se representa de forma esquemática la distribución de los componentes y sus conexiones por ambas capas.

Además, DesignSpark nos permite hacer una previsualización en 3D de la Figura 10, representada en la Figura 11.

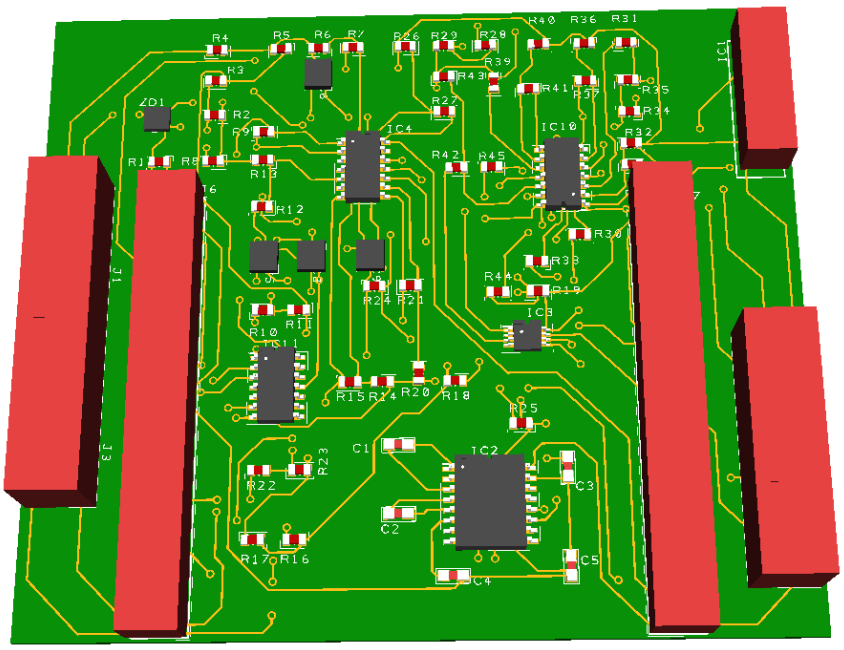


Figura 11. Vista 3D de la PCB proporcionada por el software DesignSpark PCB 8.0.

BIBLIOGRAFÍA

[1] *RS Components. DesignSpark PCB software, 2018.*
- https://www.rs-online.com/designspark/pcb-software [Consultado: junio 2018]

[2] RS Components. *Library loader download and installation, 2018.*
- https://www.rs-online.com/designspark/pcb-part-library-download-and-installation [Consultado: junio 2018]

[3] *Traco Power. DC/DC Converters TMA Series, 1 Watt*
- https://www.mouser.es/datasheet/2/687/tma-519409.pdf [Consultado: junio 2018]

[4] *MAXIM INTEGRATED. MAX220–MAX249 +5V-Powered, Multichannel RS-232 Drivers/Receivers*
- https://www.mouser.es/datasheet/2/256/MAX220-MAX249-1307854.pdf [Consultado: junio 2018]

[5] *TEXAS INSTRUMENTS.* LMx24, LMx24x, LMx24xx, LM2902, LM2902x, LM2902xx, LM2902xxx Quadruple Operational Amplifiers.
- http://www.ti.com/lit/ds/symlink/lm124.pdf [Consultado: junio 2018]

[6] *FAIRCHILD SEMICONDUCTOR. MMBD1201/. MMBD1202/. MMBD1203/. MMBD1204/. MMBD1205/ Small signal diodes.*
- https://www.mouser.es/datasheet/2/308/MMBD1203-1120967.pdf [Consultado: junio 2018]

[7] *ANALOG DEVICES. Precision Micropower Shunt Mode Voltage References, 2018.*
- https://www.mouser.es/datasheet/2/609/ADR5040_5041_5043_5044_5045-878983.pdf [Consultado: junio 2018]

[8] *Analog Devices. 2.5 V to 5.5 V, 250 µA, 2-Wire Interface, Dual Voltage Output, 8-/10-/12-Bit DACs.*
- https://www.mouser.es/datasheet/2/609/AD5337_5338_5339-877140.pdf [Consultado: junio 2018]

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA TARJETA DE ADQUISICIÓN DE DATOS USB CON 4 CANALES ANALÓGICOS DE ENTRADA/SALIDA BASADA EN LA PLACA NUCLEO-F446RE DE ST, Y LIBRERÍAS PARA LabVIEW

ANEXO III.

MANUAL DE PROGRAMACIÓN.

AUTOR: GUILLERMO JOAQUÍN ALITE CEREZUELA

TUTOR: JUAN MANUEL HERRERO DURÀ

Curso Académico 2017-2018

ÍNDICE DE CONTENIDO

1. GENERALIDADES 6
2. HABILITAR COMPONENTES DEL MICROCONTROLADOR 7
3. PROGRAMACIÓN DEL MICROCONTROLADOR 8
3.1 DECLARACIÓN DE VARIABLES 9
3.2 ESTRUCTURA DEL PROGRAMA PRINCIPAL 9
3.3 PROTOCOLO DE COMUNICACIÓN POR PUERTO SERIE 13
3.4 FUNCIONES PARA EL MANEJO DE LOS ADC Y DAC 13
3.4.1 Change_Channel 14
3.4.2 LECTURA DE ENTRADAS MEDIANTE ADC 14
3.4.3 ESCRITURA DE SALIDAS MEDIANTE DAC 15
4. PROGRAMACIÓN EN LABVIEW 16
4.1 GENERALIDADES 16
4.2 LIBRERIAS SECUNDARIAS 17
4.2.1 FUNCIÓN ESPERAR 17
4.2.2 FUNCIÓN LEER 18
4.3 LIBRERÍA “LEER_ADC” 19
4.4 LIBRERÍA “ESCRIBIR_DAC” 20
BIBLIOGRAFÍA 22

ÍNDICE DE FIGURAS

Figura 1. STM32F446RE. 6
Figura 2. Esquema de distribución de las entradas y salidas de las señales del microcontrolador. 7
Figura 3. Configuración del puerto serie USART1. 8
Figura 4. Ejemplo de inicialización del puerto serie en LabVIEW 2018. 17
Figura 5. Ejemplo de cierre del puerto serie en LabVIEW 2018. 17
Figura 6. Función esperar. 18
Figura 7. Función leer. 18
Figura 8. Función StringToVolt. 19
Figura 9. Formación de la cadena, para la librería “LEER_ADC”, que será enviada por el puerto serie. 20
Figura 10. Envío, comprobación y conversión de datos durante la comunicación con el microcontrolador. 20
Figura 11. Formación de la cadena, para la librería “ESCRIBIR_DAC”, que será enviada por el puerto serie. 21

ÍNDICE DE TABLAS

Tabla 1. Estructura de datos durante el envío de datos. 10

ÍNDICE DE CÓDIGOS

Código 1. Variables globales 9
Código 2. Switch de estados. 11
Código 3. Encendido de los DAC y activación de las interrupciones (previo al bucle). 11
Código 4. Switch de instrucciones (Casos 0, 1 y 2). 12
Código 5. Caso 5 del switch de instrucciones. 13
Código 6. Rutina de interrupción. 13
Código 7. Función Change_Channel. 14
Código 8. Primera parte de la función LEER_ADC. 15
Código 9 Función ESCRIBIR_DAC 16

1. GENERALIDADES

Para el diseño de este apartado se han utilizado 3 *software* diferentes:

- STM32CubeMX: Programa de licencia gratuita donde podemos establecer los elementos del microprocesador que queremos usar, así como establecer el pin físico por el cual se realizará la comunicación de cada componente. Nos permitirá a su vez generar un código base sobre el cual programar las funcionalidades que se deseen [1].
- SW4STM32: Programa de licencia gratuita basado en *Eclipse*, que permite crear proyectos en lenguaje C y depurar el código para resolver los errores que puedan aparecer [2].
- NI LABVIEW 2018: Programa con licencia de pago, en este caso se ha usado la licencia académica proporcionada por la UPV. Permite, mediante lenguaje G programar sistemas. Es una herramienta muy potente en el campo del control y la automatización.

El *hardware* sobre el que se monta la PCB de diseño se trata del modelo STM32F446RE [3] con Nucleo-64, que contiene un microcontrolador Cortex-M4, se muestra en la Figura 1.

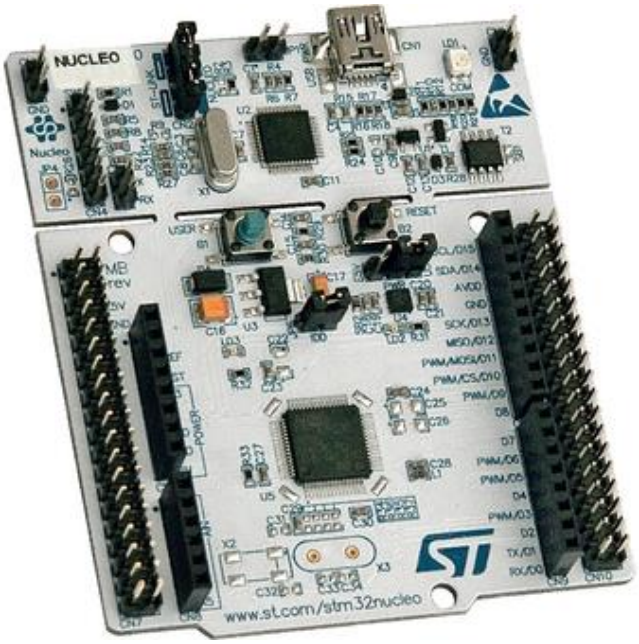


Figura 1. STM32F446RE.

Los requisitos para hacer la programación del micro de la placa son los siguientes:

- Sistema operativo: Windows (XP,7,8,10), Linux 64-bit o Mac OS X
- Cable USB: Conexión PC-placa mediante *"USB type A to Mini-B"*
- Drivers: ST-LINK/V2 necesario para configurar el periférico.

Además de todo esto, sería conveniente disponer de un manual con las funciones HAL (*Hardware Abstraction Layer*) [4] y su modo de uso, ya que se usarán en repetidas ocasiones y nos serán muy útiles a la hora de programar los elementos que se hayan habilitado en el microcontrolador.

2. HABILITAR COMPONENTES DEL MICROCONTROLADOR

Esta parte del diseño se realiza con STM32CubeMX, el cual nos permite generar un proyecto y elegir el modelo de microcontrolador que manejamos. Para habilitar componentes bastará con usar el desplegable que aparece en la parte izquierda del programa e ir activando todos los componentes que necesitemos.

Cada componente puede usar distintos pines, que pueden entrar en conflicto entre ellos, por lo que se ha procedido con cuidado de no solaparlos.

En la Figura 2 se muestra la distribución de los elementos habilitados, así como la dirección física del pin.

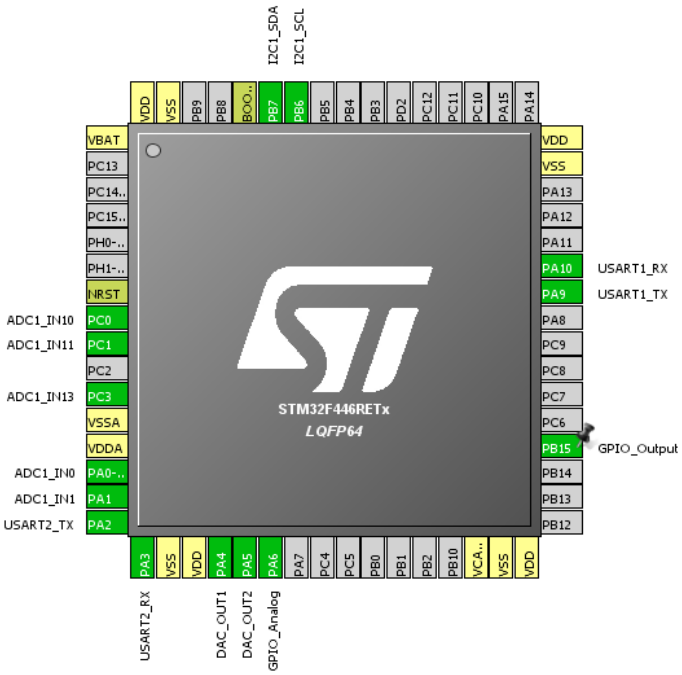


Figura 2. Esquema de distribución de las entradas y salidas de las señales del microcontrolador.

En este caso los pines habilitados han sido:

- ADC: PA0, PA1, PA2 y PC0 son los pines en los que el ADC podría tomar una tensión y realizar la conversión, son las entradas de la PCB
- DAC: PA4 y PA5 son las OUT1 Y OUT2 del DAC, son dos de las salidas de la PCB.
- USART/*Mode Asynchronous:* USART1 pines PA9 y PA10, transmisión (Tx) y recepción (Rx) respectivamente. USART2 pines PA4 y PA5, equivalentes a (Tx) y (Rx).
- I2C: PB7 Y PB6 señales de comunicación por protocolo I2C.
- PB15: Led de usuario de la placa STM32F446RE.

Además, en el programa STM32CubeMX, es necesario modificar la configuración, para que el puerto serie tenga activadas las interrupciones. Para ello nos ubicamos en la pestaña de *Configuration* y haciendo clic en USART1 y USART2, accederemos a sus configuraciones. Por último, en la pestaña *NVIC Settings* hay que activar el modo de interrupciones.

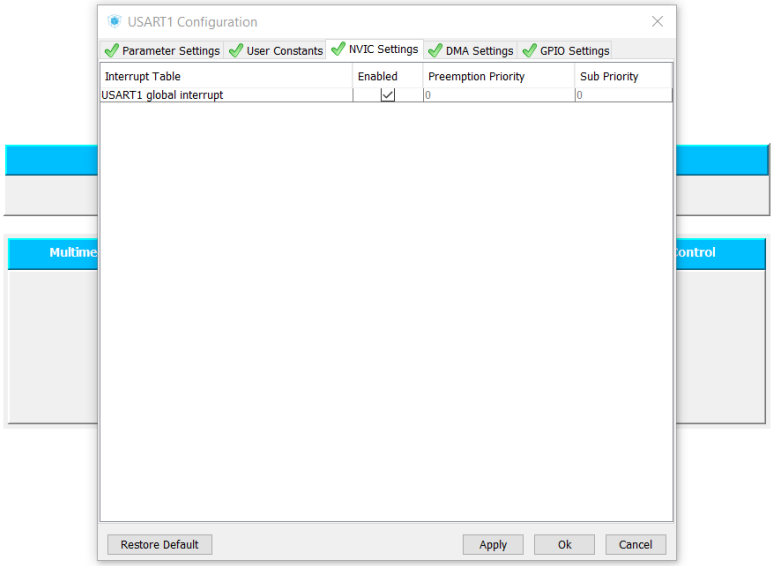


Figura 3. Configuración del puerto serie USART1.

Los valores que se han fijado, en la pestaña *Parameter Settings*, para los parámetros de configuración del puerto serie son: 115200 bps, 8 bits, sin paridad y *"stop bit"* con valor de 1.

Con todos los elementos bien configurados, se procede a generar el código base que será modificado y depurado en *SW4STM32*. Para ello en la pestaña de *Project/Project Settings,* se rellenan los campos sobre el proyecto, es importante seleccionar la aplicación que posteriormente se usará para depurar el código en lenguaje C, por lo tanto, en el campo de *Toolchain/IDE* se seleccionará *SW4STM32*, al hacer clic en OK se generará el código base que ya puede ser tratado, y sobre el cual se escribirán todas las funciones que el microcontrolador debe ejecutar.

3. PROGRAMACIÓN DEL MICROCONTROLADOR

Con ayuda del *software SW4STM32* basado en eclipse, se modifica el proyecto generado por la otra herramienta usada en este apartado, *STM32CubeMX.* De todos los archivos generados solo será necesario modificar el "main.c", es en este archivo donde se escribirán las funciones y el bucle que debe ejecutar el microcontrolador.

El código generado distingue dos zonas de escritura:

- Escritura del usuario: Es el espacio donde el usuario deberá escribir las declaraciones de variables, funciones, etc… que quiera añadir al programa. Este espacio queda determinado por la siguiente expresión.

```
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
```

- Escritura reservada para configuración: Es la zona en la que se generan líneas de código escritas por *CubeMX,* donde se inicializan los componentes habilitados y sus configuraciones establecidas previamente en el programa, así como distintas funciones propias del microcontrolador.

3.1 DECLARACIÓN DE VARIABLES

Las variables globales que se han implementado son las siguientes:

```
    //Variables del programa principal
char datoenvia[6]={0xFF,0x00,0x00,0x00,0x00,0xFE};
char flag=0,cmd,error,checksum,estado='0';
char datorecibe;
long tiempo, tiempoled;
union{
        int dato;
        uint8_t cadena[4];
}u;
    union{
        int datodac;
        uint8_t cadenadac[2];
}vdac;
```

Código 1. Variables globales

Estas son las variables que se usan en el bucle principal. Mediante las interrupciones producidas al recibir *bytes* por el puerto serie, la variable "flag" cambiará su valor. Ejecutándose así el bucle y permitiendo que las variables sean procesadas, durante la recepción de datos, el vector "datoenvia" se prepara para ser enviado por el puerto virtual, como respuesta a los bytes recibidos.

La estructura *"union"* permite almacenar el mismo valor en variables de distintos tipos, de manera que cuando una cambia de valor, modificará el valor de las demás. Ha sido útil a la hora de desglosar y rearmar los datos que tienen un tamaño considerable. Las demás variables inicializadas se usan para guardar valores necesarios durante el procesado de los datos.

3.2 ESTRUCTURA DEL PROGRAMA PRINCIPAL

El archivo "main.c" consta de 3 partes diferenciadas:

- Inicialización de variables y periféricos: Son las primeras líneas de código y contienen las variables declaradas tanto por el usuario como por el programa, además ejecuta las funciones para poner en funcionamiento los periféricos seleccionados.
- Bucle infinito: Son las líneas de código que se ejecutarán constantemente y forman la estructura principal del programa.
- Funciones: Es la última zona del programa, donde se encuentran las funciones definidas por *CubeMX* y por el usuario. Estas funciones son declaradas con anterioridad en la zona de declaración de variables.

Dentro del bucle se ha establecido una estructura con dos *switch*, uno se encarga de comprobar todas las condiciones y de realizar las operaciones necesarias (*switch* de instrucciones), mientras el otro actualiza la variable que controla ambos *switch* (*switch* de estado). Pero antes del bucle es necesario hacer dos cosas:

- Utilizar la función *"HAL_DAC_Start"*, que enciende los DAC que se han establecido en el microcontrolador. Ya que, aunque estén inicializados, si no se especifica el canal que se quiere activar, no se obtiene el resultado buscado.

- Iniciar el protocolo de recepción por interrupción, de manera que cada vez que se reciba un dato se ejecute la misma rutina.

Una vez hecho esto, el bucle comenzará, y no hará nada hasta que la variable "flag", sea igual a 1. Esto se consigue gracias a la rutina de interrupción, donde se actualiza esta variable permitiendo al bucle ejecutar los *switch* y reiniciar "flag*"*.

En el Código 4 se puede observar los tres primeros casos del *switch* de instrucciones. Cada caso comprueba si el dato recibido es un "0xFF". En el *case 0,* esa es la condición necesaria para avanzar al siguiente estado, debido a que el byte inicial se ha establecido con ese valor. Pero por esa misma razón en los siguientes casos se comprueba la misma condición. De esta manera si hubiera algún fallo en alguna transmisión y se vuelve a enviar el *byte* de inicio el programa es capaz de detectar el error y reiniciar todas las variables.

Durante todo el *switch* de instrucciones se van comprobando las condiciones establecidas por el orden de la estructura de *bytes* descrita en la **¡Error! No se e ncuentra el origen de la referencia.** del DOCUMENTO Nº1: MEMORIA.

Por tanto, un bucle completo realizará la comprobación de las condiciones en el siguiente orden:

1. El primer dato que se recibe es 0xFF;
2. Los tres siguientes datos no son 0xFF.
3. El quinto dato que se recibe es la suma de los tres anteriores.
4. El sexto dato no es un 0xFF.
5. El sexto dato es 0xFE.

De manera también preventiva se ha establecido un tiempo máximo entre el procesado de un dato y el siguiente en una cadena completa, ya que si un dato tardara demasiado en ser procesado podría significar que ha habido algún tipo de error. Este tiempo tiene una duración definida de 20 ms y ha sido implementado gracias a la función *"HAL_GetTick",* que proporciona una cuenta iniciada al principio del programa. Es decir, nos permite conocer el instante en el que se ha ejecutado la función.

Otro posible error es que alguno de los *bytes* reservados para el dato (tercer y cuarto *byte)*, tenga el valor 0xFF, valor que podría significar que la transmisión se ha interrumpido y ha empezado una nueva, como se ha comentado anteriormente. Ya que este error ocurre en el momento de la transmisión de datos, para subsanarlo, el dato que se envía desde el ordenador está multiplicado por cuatro, pero cuando el microcontrolador lo recibe, lo divide entre cuatro. Impidiendo que los bytes enviados o recibidos contengan 0xFF. Esta precaución es análoga a la hora de enviar un dato desde el micro al ordenador, pero de manera inversa, el microcontrolador es el que multiplica por cuatro el dato enviado, y el ordenador quien divide entre cuatro. Esta operación equivalente a desplazar dos *bits* a la izquierda el dato, dejando la estructura del vector enviado de la siguiente manera:

1111 1111 (0xFF)	CMD	00XX XXXX	XXXX XX00	CHECKSUM	1111 1110 (0xFE)

Tabla 1. Estructura de datos durante el envío de datos.

Por tanto, si ocurre alguna cosa fuera del funcionamiento normal, se cambiará el valor de la variable *"error"*, y si es necesario también cambiará el valor de *"estado"*. El dato de *"error"* se usa en el *switch* de estados, impidiendo el avance cuando ha ocurrido algún fallo.

```
switch(estado){

case '0': if(error==0){
            estado='1';
            }
          break;
case '1': if(error==0){
            estado='2';
            }
          break;
case '2': if(error==0){
            estado='3';
            }
          break;
case '3': if(error==0){
            estado='4';
            }
          break;
case '4': if(error==0){
            estado='5';
            }
          break;
case '5': if(error==0){
            estado='0';
            }
          break;
}
```

Código 2. Switch de estados.

En el Código 4 se muestran los primeros casos del *switch* de instrucciones, los demás casos se basan en comprobar las condiciones comentadas con anterioridad y de ir almacenando los bytes recibidos en distintas variables, para más tarde formar la cadena que será enviada por el puerto serie. También es el encargado de llamar a las funciones que permitirán leer los ADC y escribir tensiones en los DAC.

```
 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 HAL_DAC_Start(&hdac,DAC_CHANNEL_1);
 HAL_DAC_Start(&hdac,DAC_CHANNEL_2);
if(HAL_UART_Receive_IT(&huart2,(uint8_t*)&datorecibe,1)!=HAL_OK){
          _Error_Handler(__FILE__, __LINE__);
}
```

Código 3. Encendido de los DAC y activación de las interrupciones (previo al bucle).

Mientras la placa esté funcionando el led de usuario de la STM32F446RE permanecerá parpadeando, de este modo se podrá saber que la placa está funcionando y el programa se está ejecutando correctamente. Este led ha sido asignado al pin PB15 separando el puente SB20, y uniéndolo al SB21 de la F446.

```
tiempoled=HAL_GetTick();
  while (1)
  {
      if((HAL_GetTick()-tiempoled)>500){
          HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_15);
          tiempoled=HAL_GetTick();
      }

       if(flag){
           flag=0;
           switch(estado){

           case '0':if(datorecibe==0xFF){
                       tiempo=HAL_GetTick();
                       error=0;
                       }
                   else{
                       error=1;
                   }
                   break;
           case '1': if(datorecibe==0xFF){
                       tiempo=HAL_GetTick();
                       error=1;
                   }
                   else if((HAL_GetTick()-tiempo)<20){
                       tiempo=HAL_GetTick();
                       cmd=datorecibe;
                       error=0;
                   }
                   else{
                       estado='0';
                       error=1;
                   }
                   break;

           case '2': if(datorecibe==0xFF){
                       tiempo=HAL_GetTick();
                       estado='1';
                       error=1;
                   }
                   else if((HAL_GetTick()-tiempo)<20){
                       tiempo=HAL_GetTick();
                       vdac.cadenadac[1]=datorecibe;
                   }
                   else{
```

Código 4. Switch de instrucciones (Casos 0, 1 y 2).

Cuando todas las condiciones se cumplen, llega el *case 5*, es en esta parte donde todos los datos recibidos que se han almacenado son usados para describir la acción a realizar. En el Código 5 se muestra como dependiendo del valor de “cmd” leerá o escribirá. Por un lado, la función ESCRIBIR_DAC necesita tanto el “cmd” como el dato que se quiere escribir, considerado en este caso como un número entre 0 y 4095, o dicho de otra manera un numero de 12 bits. Por otro lado, la función LEER_ADC necesita también el “cmd”, pero en este caso la función devuelve el valor que el ADC ha detectado.

Por último, se crea la cadena con la respuesta a la orden recibida, en este caso “cadenaenvia*”*, y mediante la función “*HAL_UART_Transmit”*, enviará los datos por el puerto serie. Cabe destacar que, si la orden que recibe el microcontrolador es de escritura, los bytes reservados en *“*cadenaenvia*”* para el valor a enviar será 0.

```
case '5':  if(datorecibe==0xFF){
             tiempo=HAL_GetTick();
             estado='1';
             error=1;
             }
         else if(((HAL_GetTick()-tiempo)<20)&&(datorecibe==0xFE)){
         if((cmd==0x01)||(cmd==0x02)||(cmd==0x03)||(cmd==0x04)){
                 u.dato=LEER_ADC(cmd)*4;
             }
         if((cmd==0x11)||(cmd==0x12)||(cmd==0x13)||(cmd==0x14)){
             ESCRIBIR_DAC((int)(vdac.datodac/4),cmd);
             u.dato=0;
             }
             datoenvia[1]=cmd;
             datoenvia[2]=u.cadena[1];
             datoenvia[3]=u.cadena[0];
             datoenvia[4]=cmd+u.cadena[1]+u.cadena[0];
             HAL_UART_Transmit(&huart1,(uint8_t*)datoenvia,6,100);
         }
         else{
             estado='0';
             error=1;
         }
         break;
```

Código 5. Caso 5 del switch de instrucciones.

3.3 PROTOCOLO DE COMUNICACIÓN POR PUERTO SERIE

Como se ha explicado en el apartado anterior, la recepción de datos por parte del micro se ha realizado por interrupción, lo que significa que cada vez que el microcontrolador reciba un dato se activará una rutina, establecida en la función "*HAL_UART_RxCpltCallback*".

```
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){

    flag=1;

    if(HAL_UART_Receive_IT(&huart2,(uint8_t*)&datorecibe,1)!=HAL_OK){
            Error_Handler();
        }
}
```

Código 6. Rutina de interrupción.

Es en esta función donde "flag", la variable encargada de que el bucle avance hasta los *switch*, cambia su valor. También se habilitan de nuevo las interrupciones para que el siguiente dato siga la misma rutina.

La otra comunicación por puerto serie que se produce es cuando el bucle finaliza correctamente, en ese momento el microcontrolador transmite la cadena correspondiente a "cadenaenvia", la manera en que se estructura la cadena enviada está descrita en el Código 5.

3.4 FUNCIONES PARA EL MANEJO DE LOS ADC Y DAC

En este apartado se definen las funciones capaces de adquirir datos de un canal del ADC y de escribir un voltaje en una de las salidas de los DAC. Para llevar a cabo esta tarea se han creado tres funciones, dos para la lectura del ADC, y una para los DAC. A continuación, serán brevemente comentadas:

- Change_Channel: Función que selecciona uno de los posibles canales de lectura del ADC.
- LEER_ADC: Permite activar el convertidor, esperar la conversión, adquirir el dato, y apagar el convertidor. Es un tipo de función que devuelve un dato de tipo entero.
- ESCRIBIR_DAC: Función que permite dotar de potencial a una de las salidas de la tarjeta, además tiene incluidas las funciones que configuran y habilitan el I2C.

3.4.1 Change_Channel

El convertidor ADC que se ha usado, en concreto el ADC1, tiene habilitados distintos pines, de manera que es capaz de leer de distintas entradas. Para poder elegir el canal que tiene que leer el convertidor se ha implementado una función que edita la configuración establecida del ADC en función del comando que se recibe. Esta función queda representada en el Código 7.

```
void Change_Channel(int num){

  ADC_ChannelConfTypeDef sConfig;


    /**Configure for the selected ADC regular channel its co
    */
  if(num==0){
      sConfig.Channel = ADC_CHANNEL_0;
  }
  if(num==1){
      sConfig.Channel = ADC_CHANNEL_1;
  }
  if(num==2){
      sConfig.Channel = ADC_CHANNEL_11;
  }
  if(num==3){
      sConfig.Channel = ADC_CHANNEL_10;
  }
  if(num==4){
      sConfig.Channel = ADC_CHANNEL_VREFINT;
   }
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK){
    _Error_Handler(__FILE__, __LINE__);
  }

}
```

Código 7. Función Change_Channel.

3.4.2 LECTURA DE ENTRADAS MEDIANTE ADC

Para realizar la operación de lectura del ADC, primero se lee la tensión de 2,5V proveniente de Vref, esta tensión sirve para calibrar con mayor precisión el dato tomado. Seguidamente se establece el canal que se tiene que leer y usando librerías HAL se inicia el convertidor, se espera a adquirir el dato y se deshabilita el convertidor. Para las cuatro entradas la función es equivalente, pero seleccionando el canal correspondiente.

```
float LEER_ADC(char cmd){
    float i_volt=0, i_entero;
    uint32_t datoanalogico=0, dato_ref=1;

    //Calibrar ADC con entrada a 2'5V
    Change_Channel(4);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,100);
    dato_ref=HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);

    if(dato_ref==0){
        dato_ref=3090;
    }
    if(cmd==0x01){
        //ADC_1
        Change_Channel(0);
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1,100);
        datoanalogico=HAL_ADC_GetValue(&hadc1);
        HAL_ADC_Stop(&hadc1);
    }
```

Código 8. Primera parte de la función LEER_ADC.

Una vez el dato ha sido correctamente adquirido se procede a su tratamiento, se calcula el entero de 12 bits, utilizando el dato adquirido como referencia, y ese valor será el que la función devuelva.

3.4.3 ESCRITURA DE SALIDAS MEDIANTE DAC

Por último, tenemos la función encargada de generar voltajes en los DAC de salida. Para llevar a cabo su objetivo necesita dos datos: el valor a escribir y la salida escogida. Será importante discernir en esta función los dos DAC provenientes del microcontrolador y los DAC del I2C [6].

Mientras que los DAC del micro utilizarán la función *"HAL_DAC_SetValue",* los DAC del I2C [5] necesitarán de la función "*HAL_I2C_Master_Transmit*" y también de un tratamiento de los datos ya que este componente necesita una estructura de datos muy concreta para que funcione.

Los I2C necesitan 4 *bytes* para realizar una operación:

- (1º) Es la dirección del componente, este *byte* se escribe en la propia función que envía datos al I2C. En nuestro caso vale: 0x0C<<1.
- (2º) Establece que salida se activará, la salida 1 o la 2.
- (3º) Está dividido en 2 partes de 4 bits. Los primeros 4 establecen modos de funcionamiento, en el cual se ha inicializado un 0x02, lo que quiere decir que funciona en modo normal, la señal CLR (Señal negada) está a 1 y LDAC (Señal negada) a 0. Los 4 bits restantes del tercer *byte* representan la parte alta del conjunto de los 12 bits restantes, reservados para el valor que se quiere escribir.
- (4º) El *byte* restante representa la parte baja del dato, son los 8 *bits* que faltan para completar los 32.

```
void ESCRIBIR_DAC(int o_volt, char cmd){
    union{
    int datodigital;
    uint8_t cadena[4];
    }d;
    d.datodigital = 0;
    char i2c_envia[3]={0x00,0x20,0x00};

    //(V-->dato bits)
    d.datodigital=o_volt;

    //Comprobación del rango
    if(d.datodigital>=4095){
        d.datodigital=4095;
    }
    if(cmd==0x11){
        //DAC conversion

        HAL_DAC_SetValue(&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,d.datodigital);
    }
    if(cmd==0x12){
        //DAC conversion
        HAL_DAC_SetValue(&hdac,DAC_CHANNEL_2,DAC_ALIGN_12B_R,d.datodigital);
    }
    if(cmd==0x13){
        i2c_envia[0]=0x01; //DACA
        d.cadena[1]=d.cadena[1]&0x0F; //4 Últimos bytes MSB
        i2c_envia[1]=i2c_envia[1]|d.cadena[1]; //Union MSB
        i2c_envia[2]=d.cadena[0]; //LSB
        HAL_I2C_Master_Transmit(&hi2c1, 0x0C<<1, (uint8_t*)i2c_envia,3, 100);
    }
```

Código 9 Función ESCRIBIR_DAC.

Para la cuarta salida (cmd=0x14), las líneas de código son equivalentes a la parte que controla la tercera salida (cmd=0x13), pero cambiando el valor de la cadena por i2c_envia[0] = 0x02, de esta manera seleccionamos la salida “B” del I2C, equivalente a DAC4.

4. PROGRAMACIÓN EN LABVIEW

En este caso la versión de LabVIEW utilizada es “*NI LabVIEW 2018 (32-bits)”*, con este software se pretende diseñar las librerías necesarias para que el microcontrolador del proyecto pueda comunicarse con un ordenador.

Las funciones principales que se han creado en LabVIEW son *“LEER_ADC”* y *“ESCRIBIR_DAC”*, que a su vez contendrán otras funciones de carácter más secundario. Es en estas librerías donde se realizan las conversiones de rangos y se prepara la cadena de *bytes* que será enviada por el puerto serie, con la estructura explicada en la Tabla 1. Estas funciones requieren de una inicialización del puerto serie para funcionar, para ello se usará la paleta VISA de LabVIEW, que permite al programador declarar el puerto que se va a usar y sus características. Además, la paleta VISA nos permitirá ejecutar la trasmisión y la recepción de datos.

4.1 GENERALIDADES

Las librerías diseñadas siguen un patrón muy parecido, ambas reciben un dato que indica cual de todos los ADC o DAC utilizar, en el caso concreto de *“ESCRIBIR_DAC”* también será necesario especificar el valor que se quiere escribir. Utilizando los datos iniciales, la función elegida, va construyendo la cadena que contendrá la orden final que

será enviada cuando esté lista. Una vez el microcontrolador responde, las librerías comprobarán que la acción ha sido realizada con éxito.

Como se ha comentado antes, existen otras librerías de carácter secundario, estas funciones son: *"FUNCIÓN LEER"*, *"FUNCIÓN ESPERAR"* y *"StringToVolt"*. Estas funciones permiten compactar el programa y hacerlo más sencillo a la vista.

La comunicación es por puerto serie, y por tanto cada vez que se quiera usar las librerías primero habrá que iniciar el puerto serie y cerrarlo al final del programa. La manera de hacer esto queda reflejado en la Figura 4 y Figura 5 .

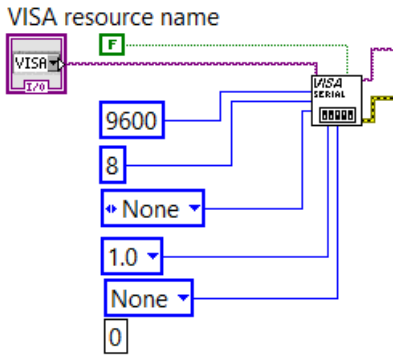


Figura 4. Ejemplo de inicialización del puerto serie en LabVIEW 2018.

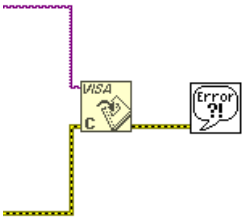


Figura 5. Ejemplo de cierre del puerto serie en LabVIEW 2018.

4.2 LIBRERIAS SECUNDARIAS

En primer lugar, se explicarán las funciones de menor peso, ya que son más sencillas y servirán para comprender posteriormente las funciones más importantes. Están separadas de las grandes porque realizan una tarea muy concreta, y si no estuvieran encapsuladas en estas funciones secundarias, harían que el entendimiento del programa principal fuera más enrevesado.

4.2.1 FUNCIÓN ESPERAR

Este es el caso de una función dedicada a esperar durante un cierto tiempo, comprobando el número de *bytes* que están en el puerto serie esperando a ser leídos, cuando su valor sea igual a 6, se procederá al siguiente paso en la estructura del programa. Sin embargo, si pasan más de 2,5 segundos y no se detectan los 6 *bytes*, se activará una señal de *"Timeout"*, proporcionando un error e impidiendo que los *bytes* sean leídos.

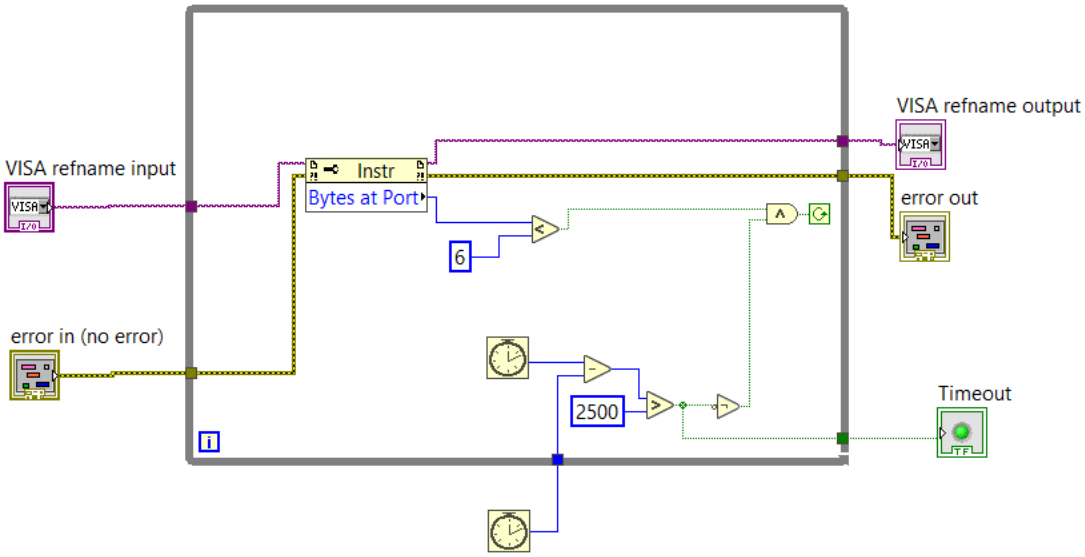


Figura 6. Función esperar.

4.2.2 FUNCIÓN LEER

Una vez la función esperar ha concluido sin errores, se activa la función leer, que realizará comprobaciones parecidas a las que realiza el microcontrolador.

- El primer *byte* es 0xFF.
- La suma del segundo, tercer y cuarto *byte* es igual al valor registrado en el quinto *byte.*
- El último valor es 0xFE

Cumpliendo estás condiciones se considera que se ha realizado una recepción de datos válida, y encenderá un led indicando que no ha habido ningún problema.

Hay que remarcar que, si pasa mucho tiempo sin que la “Función esperar” reciba los datos, la “Función leer” no realizará su operación, sino que mandará un error por pantalla.

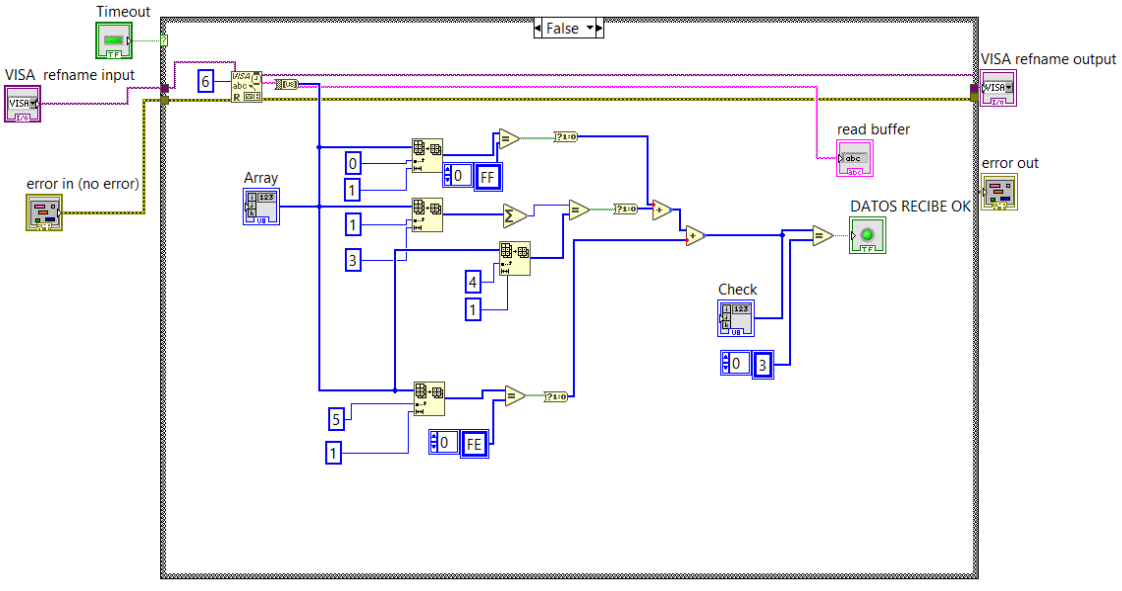


Figura 7. Función leer.

4.2.3 StrinngToVolt

Esta es una pequeña función que solo se usa en la librería *"LEER_ADC"*, permite convertir los caracteres almacenados en una cadena, a datos de tipo decimal. Además, adapta el rango de lectura a [-10, 10]V.

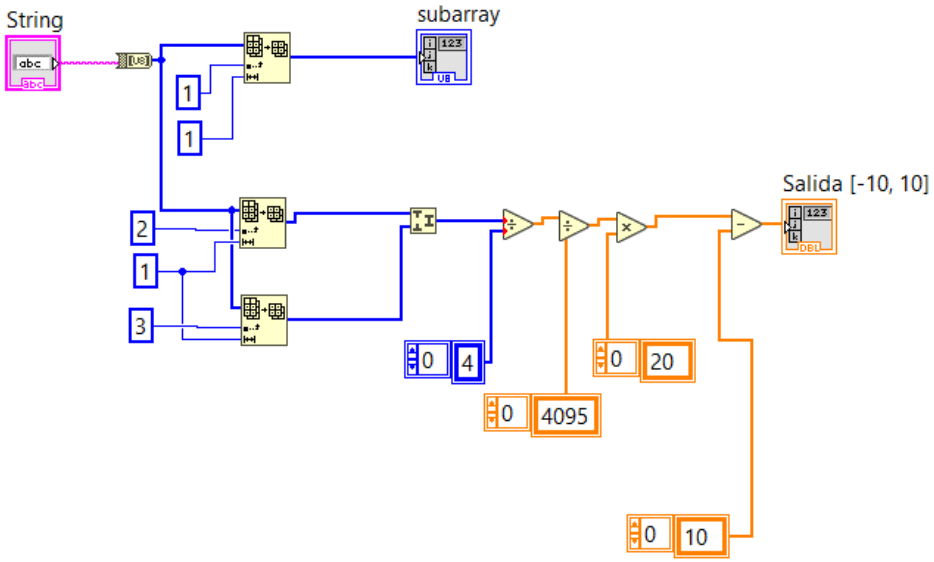


Figura 8. Función StringToVolt.

4.3 LIBRERÍA "LEER_ADC"

Esta es la función que permite controlar el ADC y elegir de entre sus posibles canales. Para ello dispone de la entrada *"CMD"*, en este apartado se escribirá un valor comprendido entre [0, 3]. Esto es debido a que cada canal que puede leer el ADC ha sido codificado con un número en hexadecimal, desde 0x01, hasta 0x04. Como se muestra en la Figura 9, el *"CMD"* aumenta su valor en 1 y es añadido a la cadena que forma la orden, para posteriormente ser enviada al micro a través del puerto serie.

Seguidamente se suman los valores requeridos para obtener el valor de *"CHECKSUM",* el resultado se añade a la cadena, y puesto que ya está lista para ser enviada, se transforma el *array* en una variable de tipo *string*.

Ha sido necesario proceder así porque la función de la paleta VISA *"VISA Write Function"*, requiere un dato de tipo *string*. Es esta función la encargada de transmitir la cadena, transformada en caracteres, por el puerto serie hasta el micro de la TAD. Cuando esta función finaliza su trabajo, se ejecutan las funciones secundarias descritas en el apartado anterior. El programa esperará los *bytes* en el puerto, encenderá un led si la recepción ha sido correcta y mostrará el valor leído. La ejecución de estas últimas funciones queda plasmada en la Figura 10.

La cadena que se recibe desde el micro contendrá la estructura de *bytes* que se ha explicado hasta ahora, incluyendo en el espacio reservado para el dato, el valor de la lectura de convertidor. Como en el caso anterior, al enviar un dato por puerto serie es necesario que sea de tipo *string*, y será la función *"StringToVolt"* la encargada de hacer la conversión de caracteres a voltios.

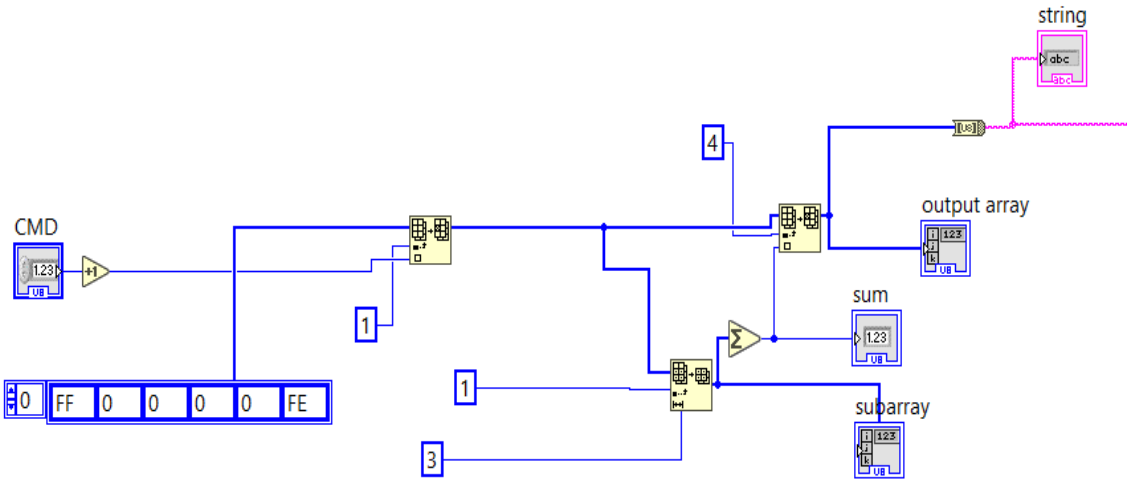


Figura 9. Formación de la cadena, para la librería "LEER_ADC", que será enviada por el puerto serie.

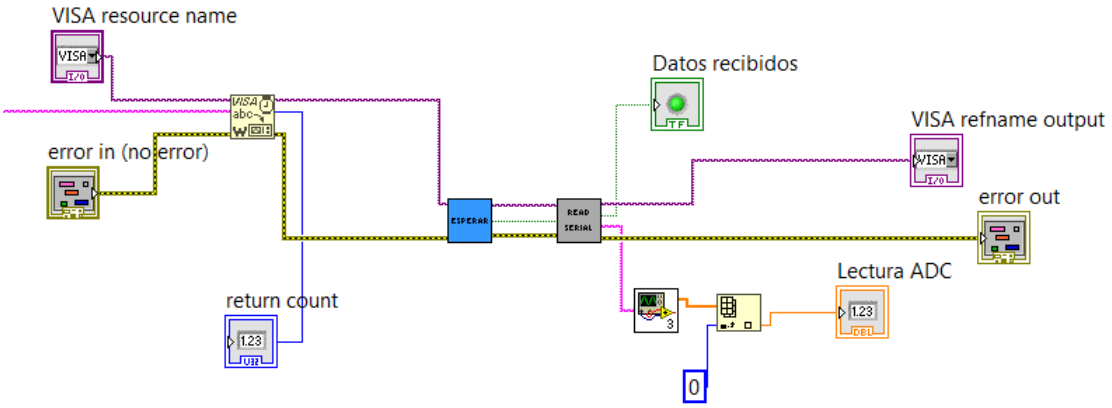


Figura 10. Envío, comprobación y conversión de datos durante la comunicación con el microcontrolador.

4.4 LIBRERÍA "ESCRIBIR_DAC"

Mediante esta librería es posible controlar la selección de uno de los 4 DAC habilitados en la placa y su voltaje. A diferencia de la anterior esta función requiere dos datos, el *"CMD"* y el voltaje que se desea en la salida. El rango de los valores de *"CMD"* en este caso es [16, 19]. En hexadecimal se han definido los DAC, con los valores comprendidos entre [0x11, 0x14]. Y el valor del voltaje estará comprendido entre [-10, 10]V.

Esta librería opera de manera muy parecida a la anterior, primero conforma la cadena que va a ser enviada, después la transforma a *string*, la envía, y comprueba las condiciones establecidas para la respuesta. A diferencia del caso anterior, en la cadena enviada desde el ordenador al micro, los dos *bytes* reservados para el valor contendrán un número entero de 12 *bits,* [0, 4095], sin embargo, en *"LEER_ADC"* estos bits son todo ceros.

Ha sido necesario calcular la ecuación de la recta que relaciona el rango [-10, 10] con [0, 3.3]. Esta ecuación queda representada en (3).

(VOLTIOS_S [-10, 10]) *0,165+1,65=VOLTIOS_S[0, 3.3] (3)

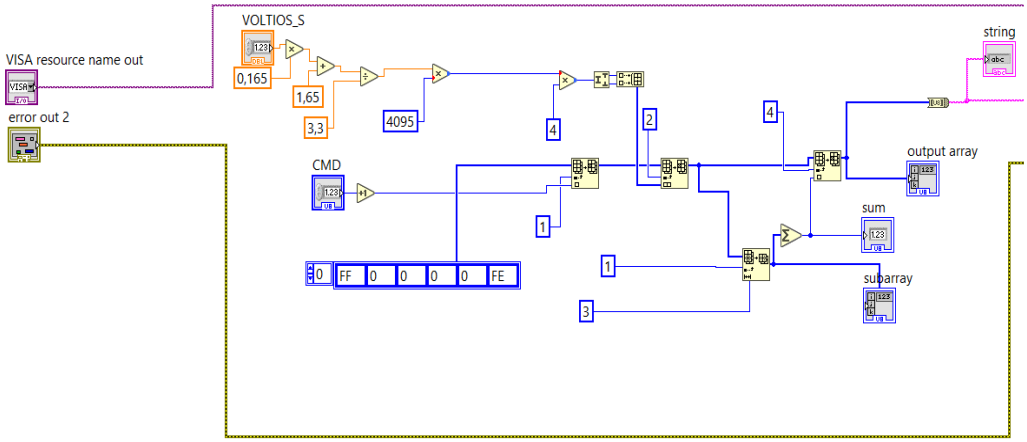


Figura 11. Formación de la cadena, para la librería "ESCRIBIR_DAC", que será enviada por el puerto serie.

El resto de la función *"ESCRIBIR_DAC"* es igual que la estructura que aparece en la Figura 10. Se encarga de: escribir por el puerto serie, esperar la respuesta y comprobar que la respuesta es correcta.

BIBLIOGRAFÍA

[1] *STMicroelectronics. STM32CubeF4, 2018*

- http://www.st.com/en/embedded-software/stm32cubef4.html [Consultado: junio 2018]

[2] *OpenSTM32. free Integrated Development Environment for STM32 microprocessors developped by Ac6 Tools,2018.*

- *http://www.openstm32.org/HomePage/* [Consultado: junio 2018]

[3] *STMicroelectronics. STM32 Nucleo-64 development board with STM32F446RE MCU, supports Arduino and ST morpho connectivity, 2018.*

- *http://www.st.com/en/evaluation-tools/nucleo-f446re.html* [Consultado:junio 2018]

[4] *STMicroelectronics. UM1725 User Manual. Description of STM32F4xx HAL drivers, 2018.*

- *http://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf* [Consultado: junio 2018]

[5] *ANALOG DEVICES. 2.5 V to 5.5 V, 250 μA, 2-Wire Interface, Dual Voltage Output, 8-/10-/12-Bit DACs, 2018.*

- *http://www.analog.com/media/en/technical-documentation/data-sheets/AD5337_5338_5339.pdf* [Consultado: junio 2018]