UNIVERSITAT OBERTA DE CATALUNYA

Ingienería técnica de telecomunicaciones

Diseño de un extensor de entradas y salidas analógicas por MODBUS RTU sobre RS – 485

Memoria

Alumno: Jose Soriano Miras

Dirigido por: Carlos Alberto Pacheco Navas

CURSO 2012-13 (Febrero)

Agradecimientos

A los pocos días de comenzar el proyecto sufrí una caída, provocándome la fractura de una vértebra y la rotura del hueso del talón del pié izquierdo. Lo que supuso dos operaciones quirúrgicas, estar hospitalizado durante tres semanas y la limitación de movimientos en los meses siguientes.

Lejos de desanimarme en el desarrollo de este proyecto, vi la oportunidad para dedicarle más tiempo del previsto. Siempre y cuando tuviera el beneplácito de la UOC, por el retraso inducido en las entregas. Sabía que con esfuerzo, dedicación y sobre todo con el apoyo de mi familia recuperaría el tiempo empleado en la convalecencia.

Por tanto, mi primer agradecimiento es para quienes desde la UOC confiaron en mí, al ampliarme los plazos de entrega, especialmente al consultor, por su apoyo, ánimo y positiva valoración durante todo el proceso, gracias Carlos.

A mis padres y hermanas que siempre creyeron en mí a pesar de mis asperezas, pero sobre todo a mi mujer y mayor soporte, pues literalmente tira de mí, a pesar de los muchos ratos que le he regateado. Por empujarme a seguir cuando estoy cansado ó frustrado, por tener que soportar mis dolores y mal humor, por su continua sonrisa, y por muchas más cosas, gracias Choni.

Por último, agradecer a la personilla que da sentido a todo esto, siendo el motor que me impulsa, para no solo en la consecución de este proyecto, sino para finalizar la carrera. Espero que en el futuro, mi hija aprenda y valore tanto de su madre como de mí, que ante la adversidad no sirven los lamentos y solo hay un camino posible, levantarse y tirar para adelante, va por ti Marta.

Índice

Índice de tablas

Índice de figuras

1. Presentación

El control de un proceso productivo en la industria está dominado por los autómatas programables, PLC, *Programmable Logic Controller*, o también por PC´s dedicados. Estos sistemas de control necesitan sincronizar las distintas tareas, para lo que recibirán información del entorno industrial y enviarán señales para poder interactuar con el proceso de producción.

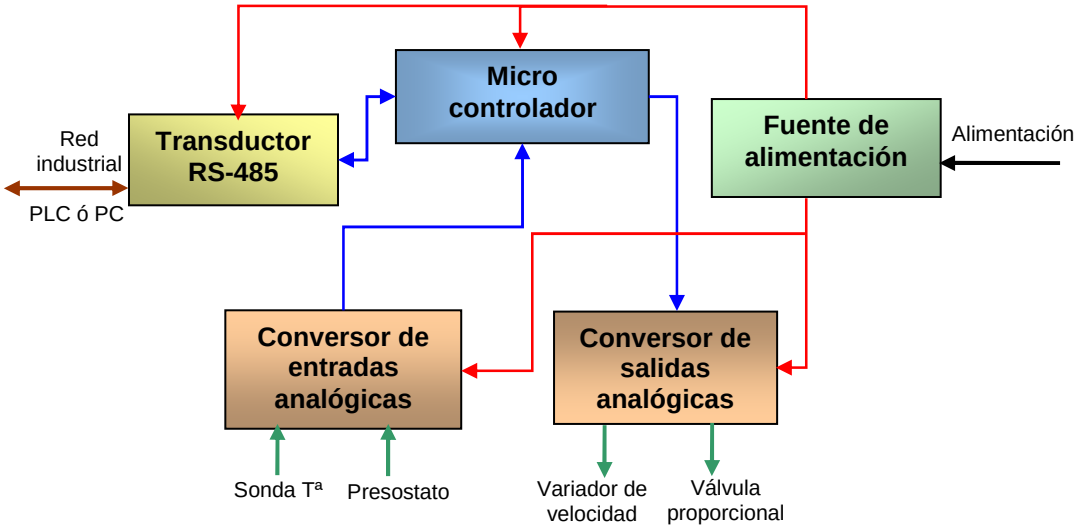
Esta interacción es necesaria protocolizarla y unificarla, para que las señales provenientes de múltiples y distintos sensores, con márgenes de medición de distinta amplitud, sean cuantificadas y procesadas de manera correcta. A su vez, se deben enviar a los distintos activadores unas señales previamente definidas para ejecutar acciones en el orden y maneras establecidas.

El bus RS-485 asigna la comunicación de forma física, a nivel eléctrico, para asegurar un blindaje ante interferencias y proteger la integridad de la información que se intercambia en la red. Con MODBUS se define el protocolo, que especifica el formato para el intercambio de datos entre los dispositivos.

El dispositivo maneja, por medio de un microcontrolador, el procesado de entradas y salidas analógicas, las cuales deben adaptarse para su tratamiento, con la ayuda de circuitos auxiliares. Las señales de la red de comunicación se resuelven a través de un transductor que implementa el bus RS-485. El circuito se completa con el diseño de su placa de circuito impreso ó PCB.

Mediante la codificación e implementación de un programa en lenguaje C se realizará la recepción, interpretación y ejecución de las órdenes recibidas a través de ModBus RTU, así como también de preparar los datos solicitados y transmitirlos para responder al dispositivo principal.

A continuación se muestra el diagrama de bloques simplificado del extensor:



Con línea roja se indica la alimentación y con línea azul se destacan las líneas de datos.

Figura 1: Diagrama de bloques del extensor

2. Objetivos

Los principales objetivos del presente proyecto son:

- Conocer uno de los buses más utilizados en las comunicaciones industriales, como es el RS-485.

- Analizar las tramas y funciones del protocolo MODBUS RTU.

- Estudiar tanto el *hardware* como el *software* del microcontrolador empleado, (16F877A) así como su entorno de programación y desarrollo.

- Practicar el diseño de acondicionadores de señales analógicas mediante el empleo de amplificadores operacionales, así como aplicar la técnica de modulación por ancho de impulso PWM para conversión D/A.

- Adquirir conocimientos sobre filtraje de ruido eléctrico y eliminación de armónicos mediante filtros paso bajos activos y analizando su respuesta mediante programas de simulación y prueba.

- Ampliar conocimientos sobre diseño de fuentes de alimentación y comparar entre distintos diseños.

- Codificar en lenguaje C las estructuras de control para posteriormente analizarlas a través de programas de simulación y comprobar el código implementado.

- Practicar con herramientas de desarrollo de placas de circuito impreso PCB (P*rinted Circuit Board*).

3. Enfoque y método seguido

A la hora de determinar los pasos a seguir en el desarrollo del presente proyecto se establece la necesidad de articularlos sobre el diagrama de bloques descrito por la figura 1. De este modo el trabajo que aquí se presenta tiene dos fases bien delimitadas:

1. Precisar y definir cada uno de los bloques de la figura 1.

2. Diseñar el programa de control del microcontrolador.

Por todo ello, primeramente se describen a nivel físico los buses y protocolos sobre los que trabaja este dispositivo para poder encuadrarlo y delimitar su espacio de trabajo dentro de un entorno de operación más amplio y al cual complementa.

Seguidamente se describe el tipo de componentes o dispositivos principales que serán necesarios así como las características que tendrán que cumplir para cubrir las especificaciones del enunciado del proyecto. Este análisis previo sirve para justificar la elección del modelo concreto de cada componente.

A partir de aquí se diseñan los circuitos específicos que resuelven cada bloque, seleccionando mediante los cálculos pertinentes los componentes discretos asociados a los dispositivos principales y simulando el circuito electrónico diseñado, para justificar el marco teórico y analítico previo.

Llegado este punto, se tiene el circuito electrónico completo que cumple con las especificaciones eléctricas requeridas. La siguiente tarea es implementar el programa de control que coordine y supervise cada uno de estos bloques desde el microcontrolador, para conseguir un dispositivo que responda a las operaciones solicitadas desde un equipo central.

Para la elaboración de este programa se analiza y diseña un organigrama de bloques que estructura las tareas y funciones a implementar. La mayoría de estas funciones se ha conseguido simular, para constatar su correcto funcionamiento e implementación.

Finalmente, se diseña la placa de circuito impreso que servirá para dar soporte físico a todo el dispositivo diseñado, cumpliendo unos requisitos determinados.

4. Tareas, planificación e incidencias

Se describen a continuación las tareas y actividades en las que se distribuye el presente trabajo.

4.1. Tareas y actividades.

Se muestran las siguientes tareas y actividades:

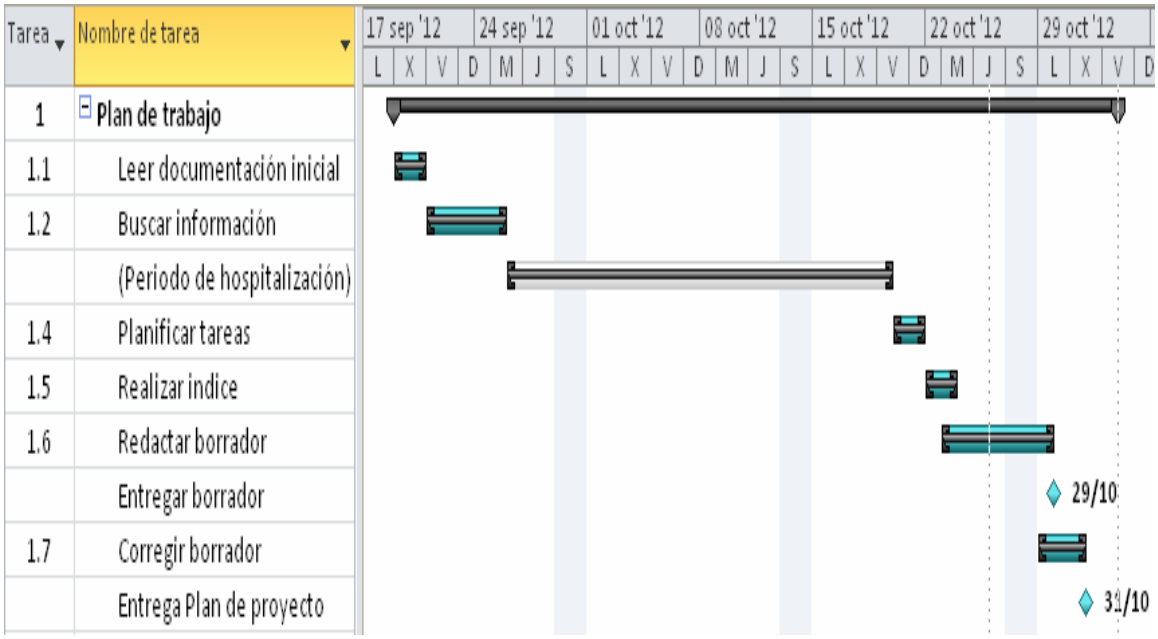
Tarea	Nombre	Descripción	Duración (Horas)	Prede-cesora
	Plan de trabajo		**52**	
1.1		Leer documentación inicial	2	
1.2		Buscar información	10	1.1
1.3		Planificar tareas	4	1.2
1.4		Realizar índice	3	1.3
1.5		Redactar borrador	25	1.4
1.6		Corregir borrador	8	1.5
	Pec 2		**96**	
2.1		Estudio protocolos e interfaces	12	1.5
2.2		Elección transductor	4	2.1
2.3		Elección procesador	8	2.2
3.1		Diseño E/S analógicos	12	2.3
3.2		Diseño f. alimentación	8	3.1
3.3		Definir esquema completo	8	3.2
3.4		Definir mapa de memoria	4	2.3 - 3.3
4.1		Redactar documentación	28	3.4
4.2		Corregir borrador	12	4.1
	Pec 3		**84**	
5.1		Análisis juego instrucciones	14	4.2
5.2		Estudio IDE programación	10	5.1
5.3		Diseñar organigrama control	8	5.2
5.4		Implementar programa control	24	5.3
6.1		Redactar documentación	16	5.3 – 5.4
6.2		Corregir borrador	12	6.1
	Memoria y presentación		**68**	
	Memoria		**40**	
7.1		Manejo entorno EDA	8	3.3
7.2		Generar *Netlist*	6	7.1
7.3		General Layout PCB	6	7.2
8.1		Redactar memoria	16	7.3
8.2		Redactar conclusiones	4	8.1
	Presentación		**28**	
9.1		Planificar presentación	4	8.2
9.2		Desarrollar presentación	12	9.1
9.3		Corregir borrador memoria	12	8.1
		Total horas	**300**	

Tabla 1: Tareas y actividades

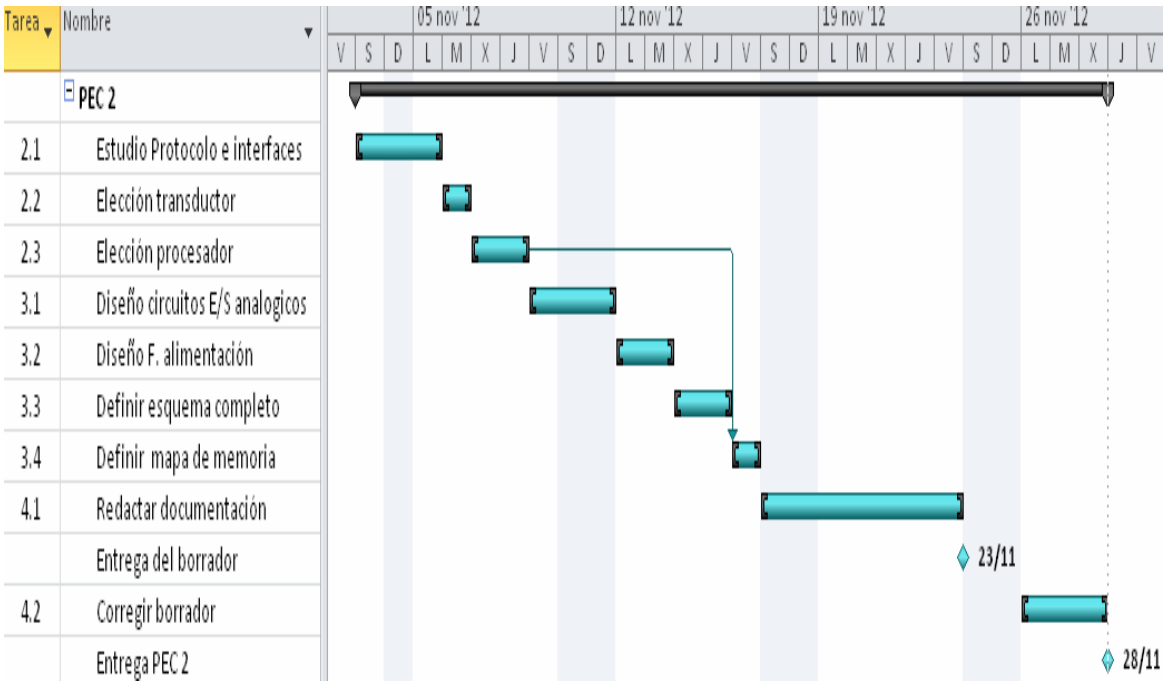
4.2. Planificación

La planificación del proyecto y de las respectivas entregas se representa mediante diagramas de Gantt.

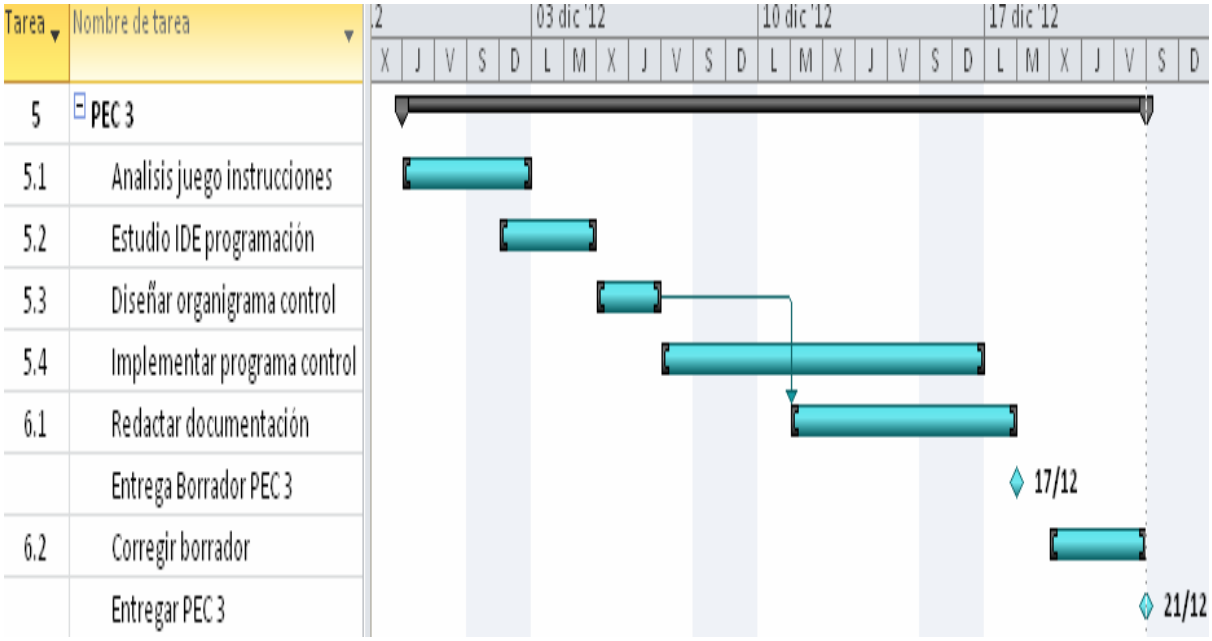
4.2.1. Diagrama del Plan de trabajo



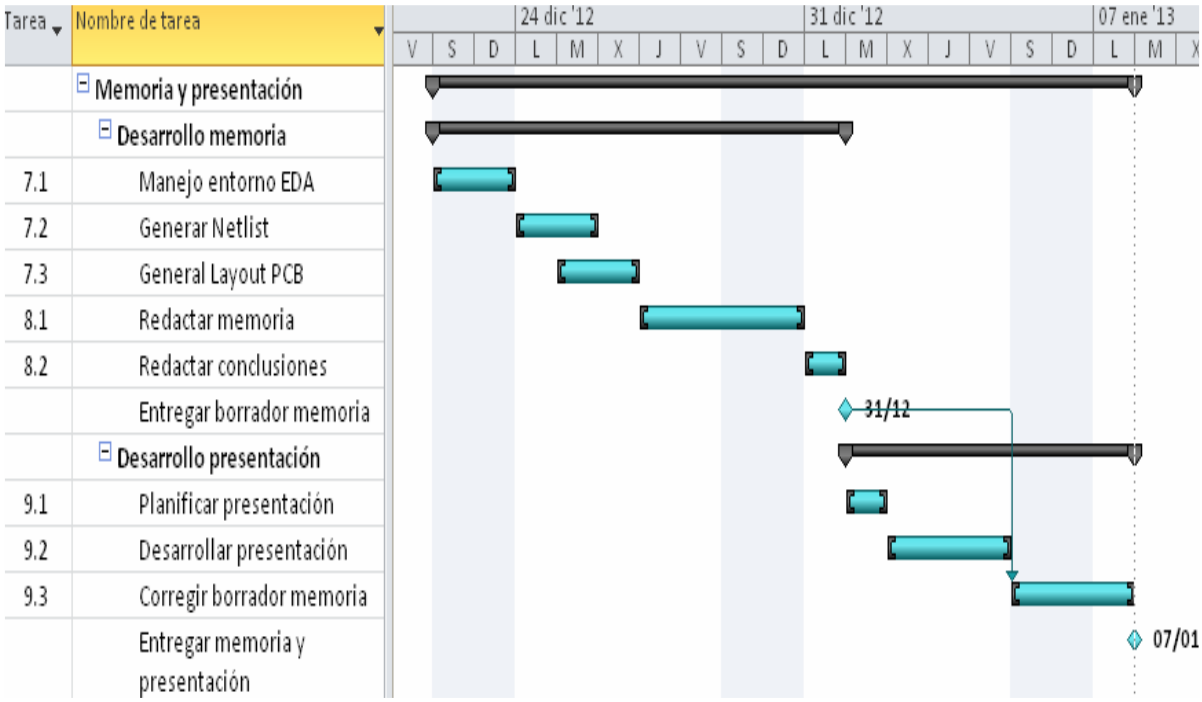
4.2.2. Diagrama de la Pec 2



4.2.3. Diagrama de la Pec 3



4.2.4. Diagrama de la Memoria y presentación



4.3. Incidencias, riesgos y plan de contingencia

Se exponen a continuación tanto las incidencias como los riesgos, que se encontraron durante el desarrollo del proyecto, así como las medidas tomadas para limitar sus efectos negativos.

4.3.1. Incidencias

Debido a la baja laboral que sufro desde el día 26 de septiembre, la disponibilidad hacia el proyecto ha estado determinada por los siguientes factores:

- Inferencia laboral nula, pues la baja irá más allá del mes de febrero.
- El tiempo de dedicación ha ido en aumento conforme ha evolucionado mi movilidad y ergonomía postural ante el ordenador.
- Horario variable según curas y rehabilitación, asegurando un mínimo de horas semanales, teniendo que llegar en algunos momentos a su ampliación, con el fin de mantener los objetivos marcados.
- De los tiempos dados se han descontado los dedicados a otra asignatura de la que estoy matriculado, siendo mínimo los solapamientos en las entregas.
- Cierta complicación en la cicatriz quirúrgica del pie, ha requiriendo más cuidados de los previstos, lo que ha provocando molestias que me han impedido ser constante algunos días, pero consiguiendo a medio plazo que el trabajo fuera uniforme.

4.3.2. Riesgos

En mi situación laboral los riesgos de trabajar temporalmente fuera de mi lugar de residencia son elevados, pero dada mi actual situación de baja laboral, el único riesgo es una nueva hospitalización, pero después del alta clínica recibida el día 22 de octubre y de una evolución favorable, las probabilidades de recaída, han ido descendiendo día a día y prácticamente son nulas.

4.3.3. Contingencias

- Fallo del ordenador: En menos de 24h se puede contar con un ordenador de repuesto y equipado con los programas necesarios, mientras se repara el averiado ó se adquiere uno nuevo.
- Fallo Disco duro: Actualmente el equipo trabaja con tres discos duros, uno de ellos externo, haciéndose 2 copias del proyecto varias veces al día, y otra copia diaria sobre una memoria USB al final de cada jornada.

5. Descripción del Bus y del protocolo empleado

A continuación se analizan las características principales que hacen de ellos una combinación muy utilizada en entornos industriales.

5.1. Bus RS-485

Este bus serie es uno de los más utilizados en las comunicaciones industriales, destacando tanto por su bajo coste como por su simplicidad, la cual facilita su instalación. Para su montaje se emplea un par trenzado que permite un enlace unidireccional, *semiduplex*, compartido tanto para transmitir como para recibir datos, pero no ambas operaciones a la vez.

El RS-485 solo define el medio físico, no establece el formato de tramas ni cuáles serán las señales de control ni el protocolo de enlace, por lo que bajo este estándar se encuentran implementados buses de distinta configuración. Este bus se describe como una arquitectura maestro-esclavo, donde el maestro inicia todas las comunicaciones y los esclavos envían datos al maestro solo cuando este lo indica. Su esquema se muestra a continuación:

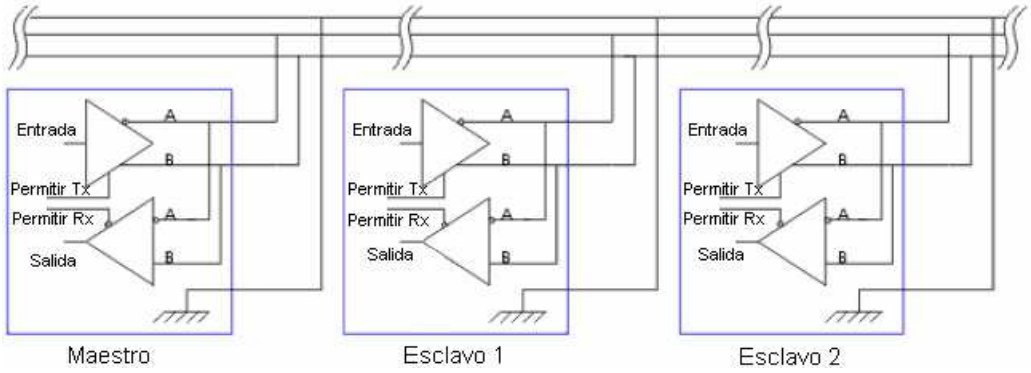


Figura 2: Arquitectura maestro esclavo

A través del par de cables trenzados se realiza el enlace físico, por donde son transmitidos los datos mediante un par de señales diferenciales. Una transmisión diferencial o balanceada no transmite los niveles lógicos como un valor de tensión con respecto a masa, ó 0V, sino que los codifica como la diferencia de potencial entre dos señales.

Al transmitir un ´1´ lógico, la señal A tiene una tensión +V y la señal B una –V, por lo que la diferencia entre las dos señales es: (A – B) = V – (-V) = +2V. En el caso de transmitir un ´0´ lógico se obtiene lo contrario: A es –V y B es +V, y la diferencia de potencial es: (A – B) = -V – (+V) = -2V.

Mediante una transmisión balanceada, se consigue cancelar las interferencias que afecten a la transmisión, pues cuando estas se producen, las dos líneas se ven afectadas por igual, y la diferencia entre las dos señales compensará el ruido. Este sistema consigue que la transmisión RS-485 pueda ser robusta y eficaz en un entorno tan ruidoso como es el industrial.

La figura siguiente muestra una transmisión balanceada, donde se ha inducido la misma interferencia a las dos señales. Al invertir una de ellas, los picos de ruido se cancelan entre si por poseer igual amplitud y polaridad invertida.

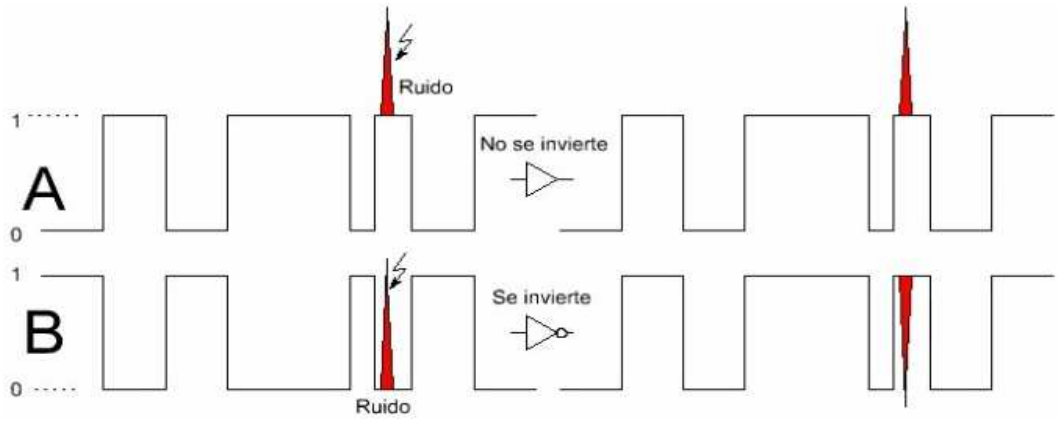


Figura 3: Transmisión balanceada

A continuación se muestra una tabla comparativa con diferentes características del estándar RS-485 con respecto a los estándares RS-232 y RS-422.

Características	RS 232	RS 422	RS 485
Diferencial	No	Si	Si
Nº de señales	De 6 a 13	4	2
Nº máximo de receptores	1	10	32
Modos de funcionamiento	Semi/full duplex	Semiduplex	Semiduplex
Topología de red	Punto a punto	Multipunto	Multipunto
Distancia máxima	15 m	1200 m	1200 m
Máx. velocidad a 12 m	20 kbs	10 Mbs	35 Mbs
Máx. velocidad a 1200m	1 kbs	100 kbs	10 kbs
Sensibilidad entrada receptor	±3 V	±200 mV	±200 mV
Rango entrada receptor	±15 V	±10 V	-7..12 V

Tabla 2: Comparativa entre protocolos serie

Se observa que aunque comparten similares características el RS-422 y el RS-485, este último es más sencillo de implementar por utilizar menos líneas de cableado entre dispositivos, permitiendo a la vez un mayor número de estos y facilitando el poder trabajar con un margen más amplio de tensiones.

En este proyecto se diseña un dispositivo que se comportará como un esclavo dentro de una red realizada mediante enlaces RS-485. Este podrá recoger y enviar señales analógicas, que una vez procesadas, podrán transmitirse o al dispositivo maestro, o hacia los receptores. El maestro podrá ser desde un PLC hasta un ordenador de control, dentro de una red aun mayor.

5.2. Protocolo MODBUS RTU

Se encarga de gestionar una comunicación del tipo cliente servidor entre distintos equipos conectados mediante un bus serie. Es muy empleado en entornos industriales gracias a su sencillez, versatilidad y especificaciones de acceso libre y gratuito, gestionadas por la *MODBUS Organization*. Es un protocolo de tipo Petición/Respuesta, en el cual existe un equipo maestro que puede acceder a varios equipos esclavos de una la red, donde cada uno de ellos tiene asignado una dirección única de dispositivo. El maestro puede hacer dos tipos de peticiones a un esclavo:

1. Enviar datos a un esclavo y esperar su respuesta de confirmación.
2. Pedir datos a un esclavo y esperar su respuesta con los datos.

Las peticiones de lectura y escritura que envía un maestro, llevan asociado un código de función, indicando la tarea que el esclavo debe ejecutar y que tipo de información espera recibir. Para gestionar este intercambio de peticiones y respuestas, los dispositivos de la red MODBUS organizan los datos en tramas, formadas por una secuencia de caracteres que puede interpretar el receptor.

En los dispositivos receptores, las tramas se sincronizan mediante la monitorización del intervalo de tiempo transcurrido entre caracteres recibidos. Los campos contenidos en el intercambio de tramas petición-respuesta son:

Figura 4: Intercambio de tramas entre maestro y esclavo

Según el formato de los datos, se encuentran dos versiones de este protocolo:

MODBUS RTU

En la transmisión MODBUS RTU (*Remote Terminal Unit*), los bytes se envían en su codificación binaria plana, sin ninguna conversión. De esta forma se consigue aprovechar el canal de comunicación y aumentar la velocidad de transmisión de datos, pero a cambio requiere una buena gestión de los tiempos entre bytes recibidos para delimitar el comienzo y fin de las tramas. La trama MODBUS RTU queda definida con los siguientes campos y longitudes:

Inicio	Dirección	Función	Datos	CRC	Final
Mínimo 3.5 caracteres de silencio	1 Byte	1 Byte	N Bytes (depende de la función)	2 Bytes	Mínimo 3.5 caracteres de silencio

Para gestionar las tramas, se envían caracteres de silencio, cuya duración es de un byte de datos enviado por el medio. Esta duración (T) depende de la velocidad (V_t) y del número de bits que se usen para su codificación (N) según $T = N / V_t$. El estándar MODBUS fija el tiempo entre tramas, para velocidades de hasta 19.200 bps, en un mínimo 3.5 veces la duración de un carácter.

Por ejemplo, para una configuración de velocidad serial de 19.200 bps, con un bit de inicio, un bit de parada y un bit de paridad (11 bit en total sumando los 8 bits de datos) se tiene: 3.5 • 11 / 19.200 = 2 ms.

MODBUS ASCII

Los datos se codifican como caracteres ASCII entre el “0” (30h) y el “9” (36h) y entre “A” (41h) y “F” (46h). También se utiliza el carácter “:” (3Ah) para marcar el comienzo de la trama y la pareja “CRLF” (0Dh, retorno de carro, 0Ah, salto de línea) para acotar el fin de la trama. La trama ASCII queda definida como:

Inicio	Dirección	Función	Datos	CRL	Final
Carácter “:” (3Ah)	2 Bytes	2 Bytes	N x 2 Bytes (depende de la función)	2 Bytes	caracteres CR y LF (0D0Ah)

La ventaja de este formato, es la facilidad de detectar el principio y fin de trama, con independencia de la velocidad de transmisión, en su contra tiene la poca eficacia que ofrece el envío de datos ASCII, frente a la transmisión binaria.

Campos de la trama

La descripción de cada campo en cualquiera de los dos formatos es la misma:

o **Dirección:** Indica el dispositivo al que va dirigido el mensaje. Cuando un dispositivo responde a un mensaje, debe enviar en primer lugar su dirección para que el maestro reconozca al emisor del mensaje. Utilizando la dirección cero se puede enviar mensajes a todos los dispositivos a la vez (*broadcast*).

o **Función:** Indica al dispositivo, indicado por el campo dirección, qué tipo de función o tarea debe realizar. Cuando se responde al maestro, este campo contendrá el mismo valor de la función.

o **Datos:** Este campo contiene la información necesaria para que los dispositivos puedan ejecutar la función solicitada, ó contener la información enviada por los dispositivos al maestro, como respuesta a una función.

o **Control de Errores:** Este campo permite al maestro y esclavos detectar errores de transmisión, evitando que se efectúen acciones erróneas ante alteraciones provocadas por ruido eléctrico. Los algoritmos de cálculo utilizados difieren en los dos formatos, para la trama RTU se utiliza el Control de Redundancia Cíclica, (CRC) y para la trama ASCII el Control de Redundancia Longitudinal, (CRL).

Funciones MODBUS

Hay disponibles más de 20 códigos de función para ModBus, de los que se describen los 8 más comunes, aunque en este proyecto solo se implementan 2.

o **Función 01 y 02**: Leen el estado de N entradas ó salidas discretas, respectivamente. La petición especifica el bit de inicio y la cantidad de bits a leer. En la respuesta, los datos se empaquetan en bytes, donde el primer bit solicitado ocupa el bit de menos peso del primer byte, continuando con los siguientes. Si no se completa un byte, se ponen a cero el resto de bits.

o **Función 03 y 04**: Leen el contenido binario de N registros mantenidos ó de entrada. Estos registros guardan los parámetros y variables del controlador. La petición especifica el registro de inicio y la cantidad de registros a leer. En la respuesta, los datos de cada registro son empaquetados con dos bytes.

o **Función 05**: Esta función permite forzar los valores lógicos de los bits del dispositivo indicado. Para activar el bit se debe enviar 00h, y se activa mediante 01h ó FFh, escribiéndolo en el byte más significativo. En la petición se especifica la referencia al bit ó bobina a modificar.

o **Función 06**: Permite modificar el contenido de un solo parámetro ó registro en el dispositivo indicado. La petición especifica la referencia del registro a alterar. La respuesta del dispositivo es una réplica de la petición realizada.

o **Función 15**: Fuerza el estado de una o más salidas, relacionadas como una secuencia de bits, a un estado lógico. La petición especifica las referencias de los bits y su secuencia en binario. La respuesta será la dirección del esclavo, el código de función, la dirección de inicio y número de bits forzados.

o **Función 16**: Establece valores en una secuencia de registros mantenidos. La petición indica las referencias de los registros a ser fijados y establece los valores en el campo de datos. La respuesta devuelve la dirección del esclavo, el código de función, la dirección de inicio y la cantidad de registros fijados.

Códigos de error

Cuando un dispositivo detecta un error por tener datos no válidos en la trama, la respuesta al maestro será la dirección del dispositivo, el código de la función, el código de error y el CRC. Los códigos de error son los siguientes:

Código	Tipo de error	Descripción
01	Función inválida	La función no esta permitida
02	Dirección inválida	La dirección está fuera del rango permitido
03	Dato inválido	El dato contiene un dato no válido
04	Fallo en el dispositivo	El controlador no responde
05	Reconocimiento (ACK)	Función aceptada y en proceso
06	Ocupado	Mensaje recibido, pero dispositivo ocupado
07	Reconocimiento negativo	La función no puede realizarse ahora

Tabla 3: Códigos de errores en ModBus

6. Descripción de los componentes empleados

A continuación se analizan las características principales que han de tener los componentes que integren este dispositivo.

6.1. Características del microcontrolador

Se debe tener claro, que la diferencia entre un microcontrolador (µC) y un microprocesador (µP) es funcional, pues para poder implementar cualquier proyecto mediante un microprocesador, es necesario combinarlo con otros circuitos auxiliares, y estos a su vez con memorias y periféricos.

Sin embargo, un µC se diseña con todos los componentes integrados, ahorrando espacio y tiempo en el montaje. Esto obliga a los fabricantes a disponer de un extenso catálogo de modelos adaptados a cada necesidad, y al proyectista a conocer las necesidades del proyecto para elegir el µC ideal.

Las características que se necesitan para nuestro proyecto son las siguientes:

- Entrada y salidas digitales.
- Puerto serie asíncrono para la comunicación RS-485.
- Conversor A/D para las 4 entradas analógicas.
- Conversor D/A para las 2 salidas analógicas.

Se analizará cada una de estas características para saber elegir más adelante el µC que mejor se adapte a estos requisitos.

Puertos de entrada y salida (E/S).

La utilidad de un µC empieza por la relación con otros dispositivos externos mediante señales discretas o binarias. Para ello, dispone de uno o más registros, denominados puertos, conectados a sus pines. La función de entrada o de salida de estos pines, se puede controlar independientemente.

Una característica importante, a tener en cuenta, en los pines de E/S es la corriente máxima que pueden entregar o recibir, estando comprendida normalmente entre 10 y 20 mA, suficiente para activar un LED.

Comunicación serie

Esta característica facilita la programación a la vez que consigue ahorrar espacio cuando se pretende que el µC pueda trabajar con sistema de comunicación *full duplex* o bidireccional asíncrono. Esto permite que cualquier proyecto se adapte a multitud de periféricos y dispositivos que transfieren información a través de varias normas como el RS – 232, el RS - 422 ó como el que compete a este proyecto, el RS – 485.

Algunos µC incorporan directamente dichas normas, pero el RS-485 no esta incorporado en ninguno, por lo que se deberá recurrir a circuitos auxiliares para terminar de implementar este protocolo.

Conversor Analógico Digital (A/D)

Uno de los objetivos del proyecto es cuantificar una señal analógica para transmitirla al equipo maestro. A través de una patilla del µC a la que le llega la señal analógica mediante variaciones de tensión, el convertidor analógico-digital se encargará de convertir las señales continuas en números digitales discretos, o sea, obtener un número binario a partir de un número real.

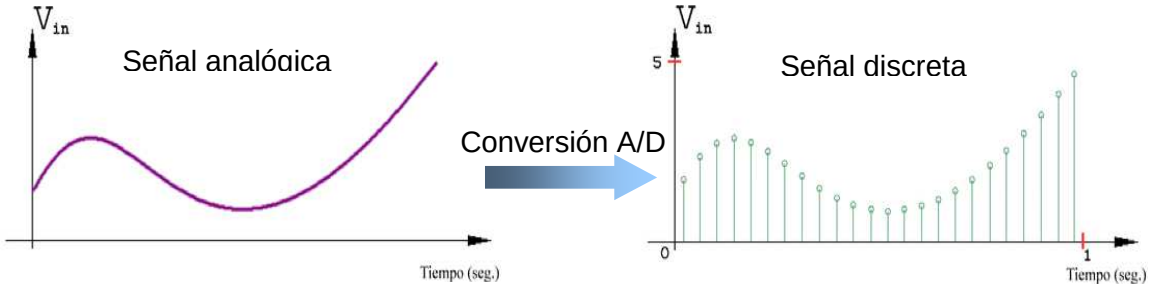


Figura 5: Conversión de señal Analógica a Digital

En la digitalización, la señal se tendrá que cuantificar con un mínimo de 10 bits, lo que proporciona 2^{10} = 1024 valores distintos, los cuales dividen el margen de tensiones proporcionado, entre 0 y 10V. Para conseguir hacer esta conversión con las entradas analógicas de corriente de 4..20mA, y con la misma precisión, la señal se tendrá que convertir previamente a tensión para el adecuado tratamiento por parte del conversor A/D.

Conversor Digital Analógico (D/A)

Otro de los objetivos del proyecto es entregar dos salidas analógicas a partir de los datos digitales del equipo maestro. Pero los µC no suelen disponer de esta posibilidad, por lo que se tendrá que recurrir o bien a la implantación de un doble circuito específico para realizar esta función, ó usar otra característica que sí esta disponible en mucho de ellos, como es la generación de un tren de impulsos cuya anchura se puede modular (PWM).

Modo PWM

Mediante esta característica se puede regular la frecuencia y duración del pulso, de forma que la potencia eléctrica que se transmite a un receptor será proporcional a la duración del pulso. Este mecanismo, en nuestro proyecto, servirá para generar señales de onda arbitraria, como una onda sinusoidal.

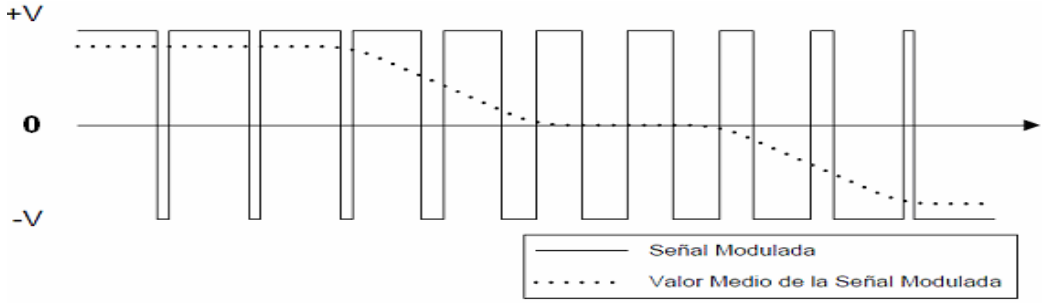


Figura 6: Modulación por anchura de pulsos PWM

Elección del microcontrolador

Además de las características básicas necesarias para el desarrollo del proyecto se deben tener en cuenta otros factores, como son el software de apoyo, el grado de divulgación e implantación, incidiendo en la cantidad de soluciones ya desarrolladas.

Visto lo anterior, se opta por escoger dentro del extenso catálogo del fabricante *MicroChip* y por la numerosa cantidad de desarrollos divulgados, un µC que reúne todos los requisitos planteados, como es el **16F877A**, cuyas características generales se describen a continuación:

- **Arquitectura RISC**: cuenta con repertorio de solo 35 instrucciones que se ejecutan en un ciclo de reloj, menos las de salto que emplean dos.
- **Frecuencia de operación**: hasta 20 MHz con un oscilador externo.
- **Oscilador interno**: rango de frecuencias desde 31KHz a 8MHz por software.
- **Alimentación**: dispone de un rango de alimentación desde 2V hasta 5.5V, dependiendo de la frecuencia, puede tener un consumo de hasta 300 mA.
- ***Brown-out Reset* (BOR)**: Por debajo de una tensión mínima, el µC se inicializa para evitar un funcionamiento errático.
- **Memoria**: ROM de 8K con tecnología FLASH, re-programable 10^5 veces
  EEPROM de 256 bytes, re-grabable 10^6 veces
  RAM de 368 bytes
- **Puertos E/S**: dispone de 33 pines divididos en 5 puertos, PORTA - PORTE, teniendo varias funciones, destacando la de entrada de interrupciones.
- **Convertidor A/D**: con 8 canales ó entradas y una resolución de 10 bits, utilizando para la conversión la técnica de “incremento y comparación”.
- **Temporizadores**: dispone de 3 de ellos, con diferentes funciones.
- **Comparadores**: incluye 2 comparadores analógicos de 200ns de resolución.
- **Módulo PWM**: incorpora dos módulos para variar al ancho de los pulsos.
- **Módulo USART**: Preparado para poder adaptarle comunicaciones seriales.
- ***WatchDog***: es un temporizador usado para vigilar bucles infinitos.

El diagrama de pines y las funciones principales se analizan a continuación:

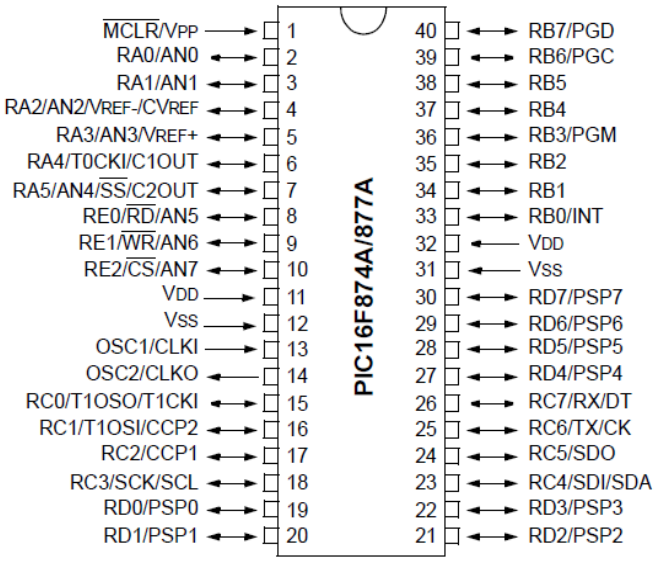


Figura 7: Diagrama de patillas del microcontrolador 16F877A

Se puede observar que para cubrir todas las funciones disponibles, es necesario que algunos de sus pines sean multipropósito.

Debido al gran número de estas funciones solo se analizaran las más relevantes para este proyecto.

Relación de pines con las funciones más relevantes para el proyecto:

Nº Pin	Función	Descripción Multipropósito
1	**MCLR** **Vpp**	Pin de reinicio. Un nivel bajo reinicia el microcontrolador Voltaje de programación
2	**RA0** **AN0**	E/S de propósito general en el puerto PORTA Entrada del canal 0 del convertidor A/D
3	**RA1** **AN1**	E/S de propósito general en el puerto PORTA Entrada del canal 1 del convertidor A/D
4	**RA2** **AN2** **Vref-**	E/S de propósito general en el puerto PORTA Entrada del canal 2 del convertidor A/D Entrada de referencia negativa de voltaje de convertidor A/D
5	**RA3** **AN3** **Vref+**	E/S de propósito general en el puerto PORTA Entrada del canal 3 del convertidor A/D Entrada de referencia positiva de voltaje de convertidor A/D
6	**RA4**	E/S de propósito general en el puerto PORTA
7	**RA5** **AN4**	E/S de propósito general en el puerto PORTA Entrada del canal 4 del convertidor A/D
8	**RE0** **AN5**	E/S de propósito general en el puerto PORTE Entrada del canal 5 del convertidor A/D
9	**RE1** **AN6**	E/S de propósito general en el puerto PORTE Entrada del canal 6 del convertidor A/D
10	**RE2** **AN7**	E/S de propósito general en el puerto PORTE Entrada del canal 7 del convertidor A/D
11 y 32	**Vdd**	Suministro de voltaje positivo
12 y 31	**Vss**	Tierra (ground – GND)
13	**OSC1**	Entrada del oscilador de cristal
14	**OSC2**	Salida del oscilador de cristal
15	**RC0**	E/S de propósito general en el puerto PORTC
16	**RC1** **CCP2**	E/S de propósito general en el puerto PORTC Salida del módulo PWM 2
17	**RC2** **CCP1**	E/S de propósito general en el puerto PORTC Salida del módulo PWM 1
18	**RC3**	E/S de propósito general en el puerto PORTC
19 - 22	**RD0 – RD3**	E/S de propósito general en el puerto PORTD
23 -24	**RC4 – RC5**	E/S de propósito general en el puerto PORTC
25	**RC6** **TX**	E/S de propósito general en el puerto PORTC Salida asíncrona del módulo USART
26	**RC7** **RX**	E/S de propósito general en el puerto PORTC Entrada asíncrona del módulo USART
27 - 30	**RD4 – RD7**	E/S de propósito general en el puerto PORTD
33	**RB0** **INT**	E/S de propósito general en el puerto PORTB Interrupción externa
34 - 40	**RB1 – RB7**	E/S de propósito general en el puerto PORTB

Tabla 4: Descripción de las patillas del microcontrolador 16F877A

Arquitectura interna

Este µC tiene una arquitectura *Harvard*, la cual se dispone de dos buses de datos diferentes. Uno es de 8 bits que conecta la CPU con la RAM. El otro consiste en varias líneas de 14 bits conectando la CPU con la ROM. Como el bus de datos utilizado tiene mas líneas que el tamaño de los datos, tanto la instrucción como los datos se pueden leer simultáneamente. Además, al estar separadas ambas memorias, permite ejecutar dos instrucciones a la vez.

Microcontroladores alternativos

Dentro de la misma familia PIC16F87XA de Microchip al que pertenece este µC se podrían emplear los chips: 16F873A, 16F874A y 16F876A. La diferencia entre ellos radica en el número de entradas y salidas disponibles, la capacidad de memoria, y el numero de patillas o encapsulado.

La razón de elegir el 16F877A radica en unos parámetros básicos, pero condicionados por admitir un crecimiento de las funciones o tareas a realizar. Ante nuevos requisitos por parte del programa, este chip permite desarrollar más líneas y tener disponibles las suficientes entradas y salidas discretas para afrontar futuras ampliaciones.

Las soluciones que ofrecen otros fabricantes con características parecidas son:

- Motorota: MC68HC908MR16, con hasta 6 canales PWM
- Analog Devices: ADUC812, el cual integra 2 convertidores D/A

Existen un número mayor de microcontroladores pero la mayoría de ellos tiene limitada la resolución para PWM a 8 bits, incluyendo la plataforma de hardware libre Arduino Dicimila basada en una placa con el microcontrolador Atmega168.

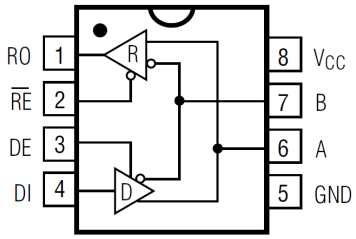
6.2. Características del transductor RS-485

Como se ha visto en las características del µC 16F788A se dispone de un módulo para facilitar las comunicaciones seriales asíncronas, pero no incorpora directamente el protocolo RS-485, por lo que para convertir los niveles lógicos a este protocolo se recurre a implementarlo mediante un circuito especializado.

Varios fabricantes incorporan estas soluciones en forma de circuitos integrados (CI) en sus catálogos, como el SN75176B de *Texas Instruments*, el DS3695 de *National Semiconductor*, el LTC485 de *Linear Technology* y el MAX485 de *Maxim*. Todos estos CI tienen las entradas en las líneas A y B del bus RS-485 y una entrada con lógica TTL que habilita el modo recepción o transmisión.

Concretamente el MAX485 cumple con los requisitos necesarios, siendo un transductor de baja potencia para comunicaciones RS-485 y RS422, que puede transmitir y recibir datos a velocidades de hasta 2.5 Mbps, aunque sea en modo *half-duplex*, permitiendo conectar hasta 32 dispositivos. En reposo, las salidas del *drive*r y del receptor se hallan en estado de alta impedancia.

A continuación se muestran las características eléctricas más destacables:



Voltaje alimentación (Vcc): ……12V
Voltaje control entrada ($\overline{RE}$, DE): .de -0.5V a 12.5V
Voltaje entrada del driver (DI): ……..de -0.5 a 12.5V
Voltaje salida del driver (A, B):…… -8V a 12.5V
Voltaje entrada del receptor: (A, B):… -8.5V a 12.5V
Voltaje salida del receptor (RO):……. -0.5 a 12.5V
Intensidad máxima cortocircuito……..……...250mA

Figura 8: Diagrama de patillas del transductor MAX 485

El diagrama de pines y su descripción se realizan a continuación:

Nº Pin	Función	Descripción
1	RO	Receptor de Salida. Si A>B en 200mV ➔ RO= '1' sino RO='0'
2	$\overline{RE}$	Permiso RO. Si $\overline{RE}$ ='0' ➔ RO permitido sino RO =' ↑Z'
3	DE	Permiso DI. Si DE ='1' ➔ DI permitido sino RI =' ↑Z'
4	DI	*Driver* de entrada.
5	GND	Conexión a masa
6	A	Entrada receptora no invertida y salida del *driver* no invertida
7	B	Entrada invertida del receptor y salida invertida del driver
8	Vcc	Alimentación positiva: 4.75V ≤ Vcc ≤ 5.25V

Tabla 5: Descripción de las patillas del transductor MAX485

Al transmitir información hay que colocar los pines 2 y 3, $\overline{RE}$ y DE, a nivel alto, de esta forma se habilita el integrado para enviar información por la patilla 4, DI, y se bloquea para recibir información por la patilla 1, RO. Para recibir información hay que colocar los pines 2 y 3, $\overline{RE}$ y DE, en nivel bajo, con lo que se bloquea en envío de información por la patilla 4, DI, y se activa la recepción por la patilla 1, RO.

Según las curvas del *datasheet* del MAX485, figuras siguientes, su consumo puede llegar hasta 140 mA, dependiendo del voltaje, estado alto o estado bajo, que entregue a la salida.

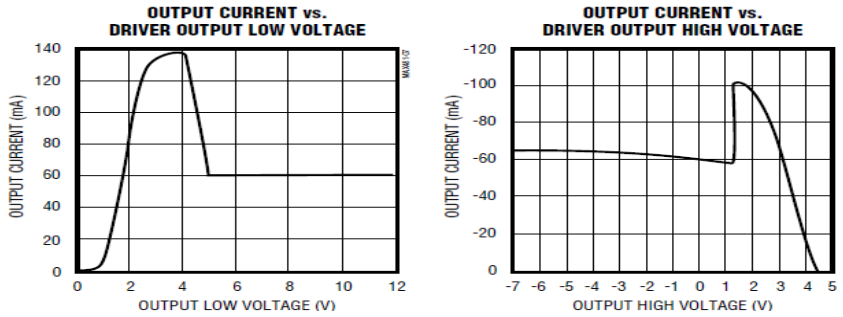


Figura 9: Curvas de consumo del MAX485

Otra característica que proporciona el MAX485 es que la distancia entre dispositivos puede llegar hasta 1 Km sin presentar atenuaciones de señal.

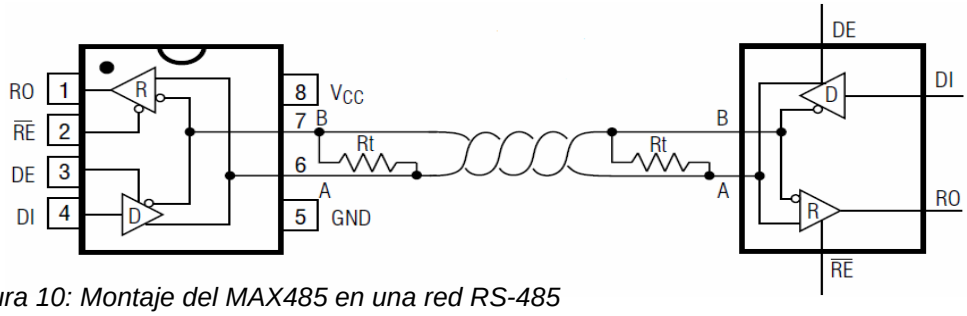


Figura 10: Montaje del MAX485 en una red RS-485

Se debe tener en cuenta para su montaje físico la importancia de conectar entre los pines A y B una resistencia de 120Ω, cuando el dispositivo esté en el extremo del par trenzado. Esta resistencia terminal reduce las reflexiones de voltaje que podrían provocar una mala lectura de los datos, eliminando además las reflexiones haciendo que las corrientes inicial y final sean iguales.

7. Diseño del esquema electrónico del circuito

Se procederá a analizar y diseñar cada uno de los bloques vistos en la *figura 1*, para obtener el circuito completo que cumpla con el objetivo del proyecto.

7.1. Diseño esquema básico microcontrolador

El diseño básico para que el µC esté preparado para funcionar de manera correcta debe constar de los siguientes elementos:

- Alimentación
- Circuito de reinicio
- Circuito de reloj

Alimentación

Este µC, como se ha visto en sus características, se puede alimentar con rango de 2V hasta 5.5. En este proyecto trabajará a una tensión de 5V, para lo que se diseñará una fuente de alimentación, que proporcionará esta tensión. El diseño de esta fuente se estudiará más adelante, pero dada su importancia se deja indicado en el esquema básico.

Circuito de reinicio

Para el correcto funcionamiento del µC se debe colocar un “1” lógico en la patilla MCLR, mientras que un “0” lógico en esta patilla causa un reinicio de forma inmediata. En este circuito se incluye un botón de RESET que al pulsarlo lleva el la patilla MCLR a un voltaje de 0V, reiniciando el µC lo que provocará que la ejecución del programa comience desde el principio. Mediante la resistencia de 10 KΩ se evita cortocircuitar a tierra cuando se pulsa este botón.

Este pulsador no es totalmente necesario, sin embargo en este circuito se ha incorporado para permitir la habilitación, y recuperar el funcionamiento normal, en caso de que ocurra un error que lleve a fallar al µC.

Circuito de reloj

Aunque este µC tiene un oscilador interno, para conseguir una mayor precisión y estabilidad en la frecuencia, se recurre a utilizar un oscilador externo para hacerlo más robusto ante cambios de temperatura y voltaje. Este oscilador externo se conecta a las patillas OSC1 y OSC2. La inhabilitación de esta señal detiene al µC, funcionando en el mismo punto después de recuperarse la señal.

Este oscilador podrá estar formado por algunos de estos componentes: cristal de cuarzo, resonador cerámico o un circuito RC. Dependiendo del componente elegido se configurará el µC para trabajar en uno de los siguientes modos:

- **Modo LP**: Se utiliza solo para trabajar con cristales de cuarzo de baja frecuencia, como puede ser 32.768 Hz. En este modo el consumo de corriente será menor que en resto de modos.

o **Modo XT**: Se utiliza para cristales de cuarzo de frecuencias intermedias de hasta 8MHz, teniendo un consumo medio en comparación con los otros modos.

o **Modo HS**: Utilizado cuando el cristal trabaja a una frecuencia de entre 8MHz y 20MHz, a costa de un mayor consumo que los demás modos.

El modo de funcionamiento se selecciona durante el proceso de escritura del programa en el µC, que mediante los bits apropiados y junto a otros bytes, forman la denominada Palabra de configuración, la cual se almacena en la ROM junto a otros registros especiales de acceso nulo.

Para este proyecto se ha elegido trabajar con un oscilador externo formado por un cristal de cuarzo de 4MHz el cual no requiere espera para estabilizar la frecuencia. Además del cristal también se conectan los condensadores C1 y C2, cuya capacidad estará entre 15pF y 22pF.

A continuación se muestra el esquema básico para el correcto funcionamiento del µC, a partir del cual se irá ampliando:

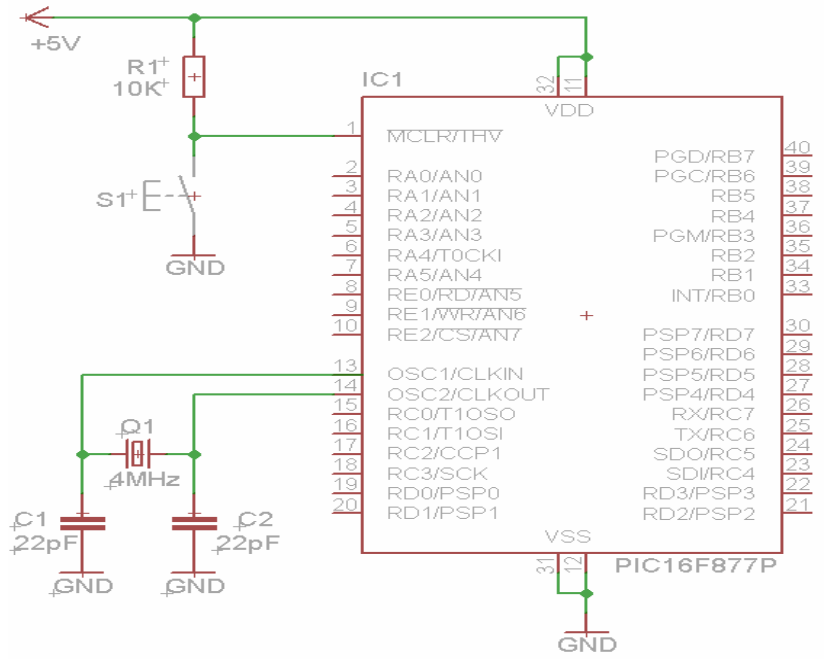


Figura 11: Esquema básico del microcontrolador

7.2. Diseño del esquema de la interfaz RS-485

Observando la figura 8, el MAX485 tiene dos partes diferenciadas. Por un lado tiene las patillas DI, RO, DE y RE para conectar al µC, y por el otro, dispone de los terminales A y B para enlazar al bus RS485, bornas X1-1 y X1-2.

Este tipo de dispositivos se van conectando entre ellos a través de un cable trenzado, siendo necesario conectar en el último de ellos una resistencia de 120Ω entre los terminales A y B. De esta manera se pueden conectar hasta 32 dispositivos ó nodos, si se quisiera ampliar esta cifra habría que recurrir al MAX487 que permite la conexión de hasta 128 nodos.

En este proyecto se ha incluido un micro-interruptor, S1-1, para seleccionar la conexión de una resistencia R5 de 120Ω por si el dispositivo está ubicado en un extremo del par trenzado. Esta resistencia, junto con una doble conexión al BUS RS-485, bornas X1-3 y X1-4, facilitan el montaje y permiten aislar al resto de equipos que le siguen hasta el final del cable para buscar problemas de red.

También se deben conectar una resistencia R1 desde el terminal A hasta Vcc que y otra resistencia R2 entre B y GND, ambas de un valor de 560Ω, para cuando el bus no está ni transmitiendo, ni recibiendo. Los diodos *zener* D1 y D2 de 13V protegen el circuito al limitar las subidas de tensión, su corriente mínima de polarización se asegura mediante las resistencias R4 y R5 de 47Ω.

El control de la dirección de los datos se realiza mediante la patilla RD7 del µC, teniendo en cuenta que al ser de colector abierto, cuando se escriba en RD7 un `1` lógico, indicando recepción de datos al habilitar $\overline{RE}$, en esta patilla se obtendrá 0V. Cuando se escriba un `0` lógico en RD7, indicando transmisión, en su pastilla se obtendrá una diferencia de potencial debido a la corriente que circula por la resistencia R6 de 10KΩ conectada a 5V.

Para fijar en el dispositivo los valores de paridad, velocidad de transmisión y dirección del dispositivo se recurrirá a ocho micro-interruptores, S1-2 a S1-9, que definirán estos parámetros de la siguiente manera:

o **Paridad**: Se establece mediante el estado de un interruptor:

S1 – 2	Paridad
OFF	PAR
ON	IMPAR

Tabla 6: Selección de la paridad serie

o **Velocidad**: Se establece mediante el estado de dos interruptores:

S1 – 3	S1 – 4	Velocidad en bps
OFF	OFF	2.400
OFF	ON	4.800
ON	OFF	9.600
ON	ON	19.200

Tabla 7: Selección de velocidad serie

o **Nº de dispositivo**: Se establece mediante el estado de 5 interruptores:

S1-5	S1-6	S1-7	S1-8	S1-9	Dirrec.	S1-5	S1-6	S1-7	S1-8	S1-9	Dirrec.
OFF	OFF	OFF	OFF	ON	1	ON	OFF	OFF	OFF	ON	17
OFF	OFF	OFF	ON	OFF	2	ON	OFF	OFF	ON	OFF	18
OFF	OFF	OFF	ON	ON	3	ON	OFF	OFF	ON	ON	19
OFF	OFF	ON	OFF	OFF	4	ON	OFF	ON	OFF	OFF	20
OFF	OFF	ON	OFF	ON	5	ON	OFF	ON	OFF	ON	21
OFF	OFF	ON	ON	OFF	6	ON	OFF	ON	ON	OFF	22
OFF	OFF	ON	ON	ON	7	ON	OFF	ON	ON	ON	23
OFF	ON	OFF	OFF	OFF	8	ON	ON	OFF	OFF	OFF	24
OFF	ON	OFF	OFF	ON	9	ON	ON	OFF	OFF	ON	25
OFF	ON	OFF	ON	OFF	10	ON	ON	OFF	ON	OFF	26
OFF	ON	OFF	ON	ON	11	ON	ON	OFF	ON	ON	27
OFF	ON	ON	OFF	OFF	12	ON	ON	ON	OFF	OFF	28
OFF	ON	ON	OFF	ON	13	ON	ON	ON	OFF	ON	29
OFF	ON	ON	ON	OFF	14	ON	ON	ON	ON	OFF	30
OFF	ON	ON	ON	ON	15	ON	ON	ON	ON	ON	31
ON	OFF	OFF	OFF	OFF	16						

Tabla 8: Selección de la dirección ModBus

Estos ocho micro-interruptores, del S1-2 al S1-9, irán conectados al puerto B del µC, PORTB, de manera que al iniciarse el programa y se lean estos parámetros, quede fijado su valor para el resto de la ejecución.

Se muestra a continuación el esquema del µC con los componentes necesarios para comunicarse, con otros dispositivos, mediante conexión al BUS RS-485:

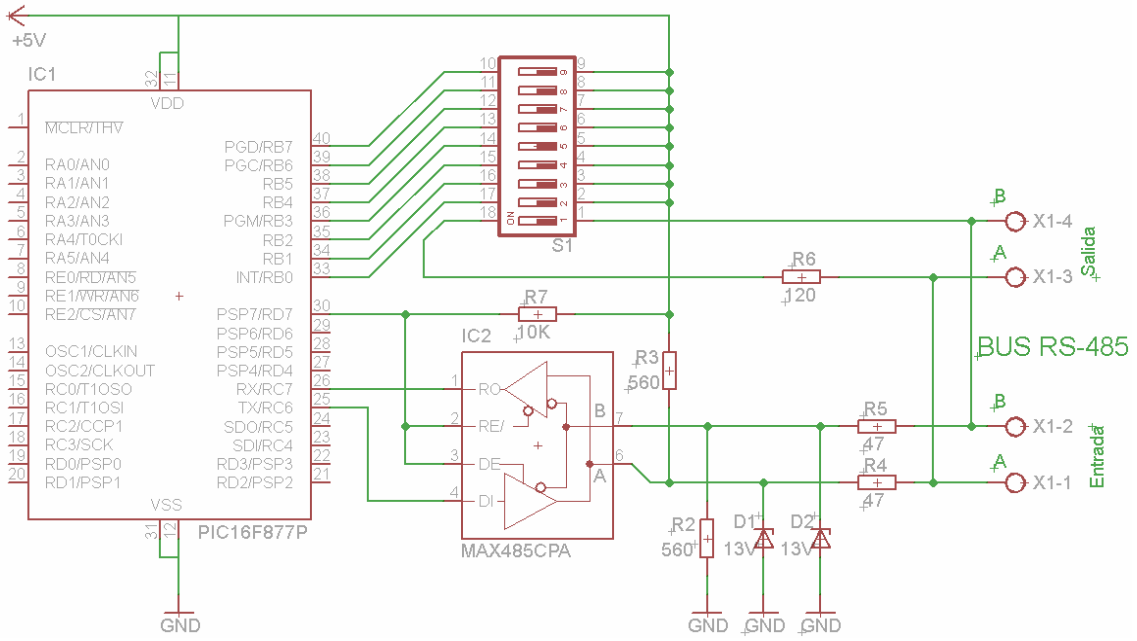


Figura 12: Esquema comunicación con bus RS-485

Una alternativa al uso de micro-interruptores es realizar la configuración de dichos parámetros mediante ModBus. El uso de este método se realizaría en los siguientes pasos:

1. Asignar cada uno de los parámetros a una dirección del mapa de memoria del dispositivo, en este caso del µC, paso que se haría mediante programa.

2. Definir estas direcciones de memoria o parámetros con unos valores por defecto, tales como dirección 1, velocidad 9600 bps y sin paridad. Estos valores también se establecen al iniciarse el programa por primera vez.

3. Al conectarlo por primera vez, al maestro se le indica que se comunique con el dispositivo número 1, a la velocidad y paridad establecidas por defecto en el nuevo dispositivo, para modificar las direcciones de memoria definidas en los pasos anteriores y poder reconfigurar al nuevo dispositivo.

4. Una vez transmitida la trama anterior, el maestro volverá a configurarse a la velocidad y paridad que tenga establecidas para comunicarse con toda la red de manera estándar.

A partir de aquí el nuevo dispositivo se podrá comunicar con el maestro con los mismos parámetros que el resto de la red y con un número de dispositivo establecido automáticamente, sin que haga falta ninguna configuración externa.

Esta opción obliga a que no exista ningún dispositivo asignado a la dirección 1, pues un en caso de que dos dispositivos compartan la misma dirección se produciría un error. Por tanto, la dirección 1 solo existe un breve instante, justo cuando se conecta y reconfigura un dispositivo nuevo.

7.3. Diseño del esquema de las entradas analógicas

Debido a las especificaciones de diseño, se trabaja con dos tipos distintos de entradas analógicas, requiriendo un tratamiento diferente para cada caso:

- Variaciones de tensión de 0..10 V
- Variaciones de intensidad de 4..20 mA,

7.3.1. Diseño de las entradas analógicas 0..10 V

Este µC para la conversión A/D dispone de ocho canales para conversión A/D preparados para recibir voltajes de hasta un valor de 5V, por tanto para trabajar con señales cuyo rango varía entre 0 y 10V es necesaria su adaptación previa.

Para conseguir esta reducción se recurre a un divisor de tensión mediante dos resistencias iguales, en cuya intersección se coloca un seguidor de tensión implementado mediante un amplificador operacional (OP). Esta configuración proporciona la misma tensión a la entrada que a la salida, con independencia de la impedancia de los canales A/D del µC y del divisor de tensión.

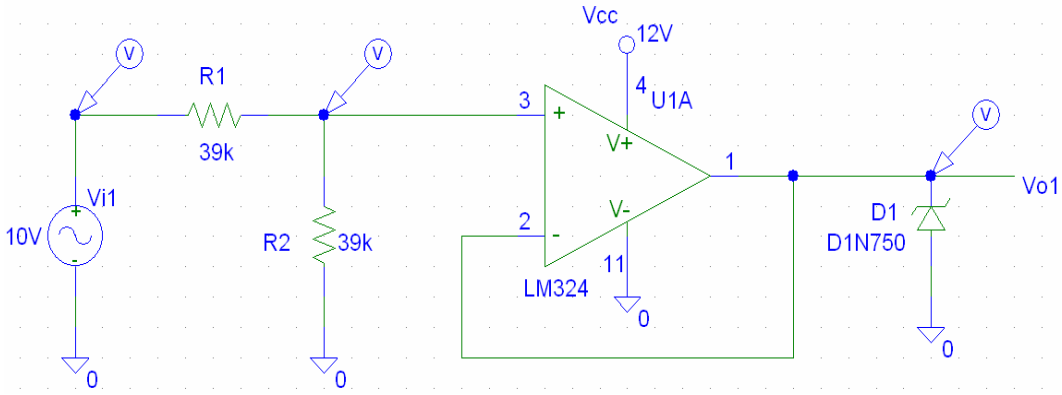


Figura 13: Esquema entrada analógica de 0..10 V

Para evitar pérdidas de señal debido a la baja impedancia de la sonda de entrada, se coloca otro seguidor de tensión a la entrada del divisor de tensión, pues las entradas de un OP, tienen una impedancia muy elevada, provocando que entra ambas entradas, tanto positiva como negativa, la intensidad sea igual a cero, $I_+ = I- = 0$. La tensión entre el terminal positivo y negativo es el misma, $V_- = V_+$, haciendo que la corriente necesaria para alimentar el canal A/D la proporcione el OP, aislando la entrada de la salida.

Para elegir el OP se ha optado por el LM324N, un CI con un amplio rango de alimentación sin necesidad de que sea simétrica o de doble polaridad, lo que facilita su alimentación y en cuyo encapsulado se integran 4 de estos OP. Las resistencias se elegirán de un valor de 39KΩ, para conseguir una alta impedancia de entrada junto a la entrada positiva del OP.

Este circuito dispondrá de dos entradas de 0..10 V, las cuales se conectaran en la patilla 2 (AN0) y en la patilla 3 (AN1) respectivamente, las cuales se protegerán mediante un diodo *zener* de 5.6v a la salida del OP, en caso de que la entrada analógica sobrepase los 10V, por efecto de alguna perturbación.

Se ha realizado la simulación en DC con *PSpice Schematics* 9.1 del circuito de la figura con el siguiente resultado:

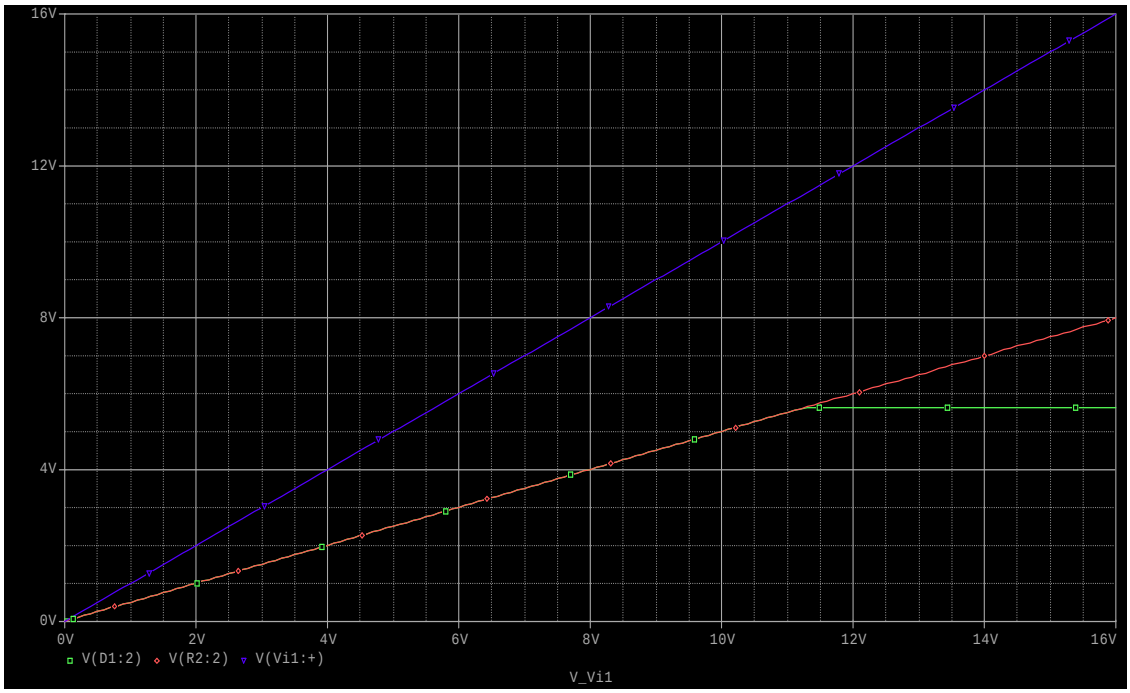


Figura 14: Simulación de la entrada analógica 0..10V

Se observa como la entrada positiva del OP, (color rojo), es justo la mitad de la señal de entrada (color azul). La salida del OP (color verde), es también la mitad de la tensión de alimentación, mostrando además el efecto del diodo *zener* al limitar a 5.6 voltios una subida de voltaje en la entrada.

El esquema completo de la entrada analógica de 0..10V, sería el siguiente:

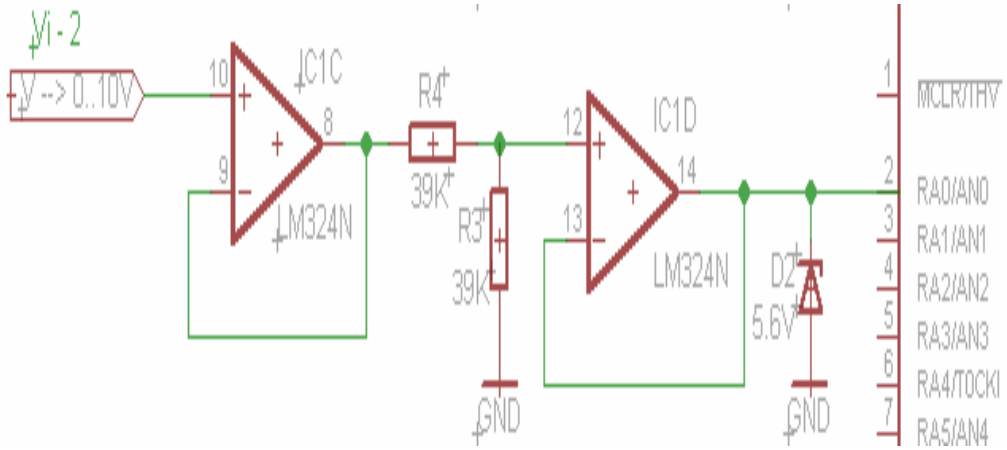


Figura 15: Conexión de la entrada analógica al microcontrolador

7.3.2. Diseño de las entradas analógicas de 4..20 mA

Los canales de conversión A/D del µC trabajan con niveles de voltaje en el rango de 0 a 5V, por lo para el tratamiento de una señal analógica donde la amplitud que varia es la intensidad que circula, será obligatorio una conversión previa. Si esta corriente se pasa por una resistencia para provocar una diferencia de tensión, de manera que su valor máximo coincida con la corriente máxima, se podrá calcular el valor de esta resistencia:

$$R = \frac{V_{max}}{I_{max}} = \frac{5\,V}{20\;\;mA} = 250\;\;\Omega$$

Con este valor se obtiene una conversión lineal de intensidad a voltaje. Sin embargo, al calcular el valor de la tensión cuando esta resistencia es recorrida por la corriente mínima de entrada se obtiene el siguiente valor:

$$V_{min} = R \cdot I_{min} = 250\;\Omega \cdot 4\,mA = 1V$$

El rango ha quedado restringido desde 1V a 5V, perdiéndose un 20% de resolución. Igualmente, cuando se acople a la impedancia de entrada de los canales A/D, el valor de carga quedaría alterado. Para corregir esto, el valor de tensión se llevará a la entrada positiva de un OP, el cual no provocará perdidas de tensión al no existir circulación de corriente a través suya.

Para que la señal oscile dentro de los valores deseados, se coloca un divisor de tensión en la entrada negativa del OP, cuyas resistencias tendrán un valor entra ellas proporcional al rango de la señal de entrada y conectadas a la tensión de referencia requerida de +5V.

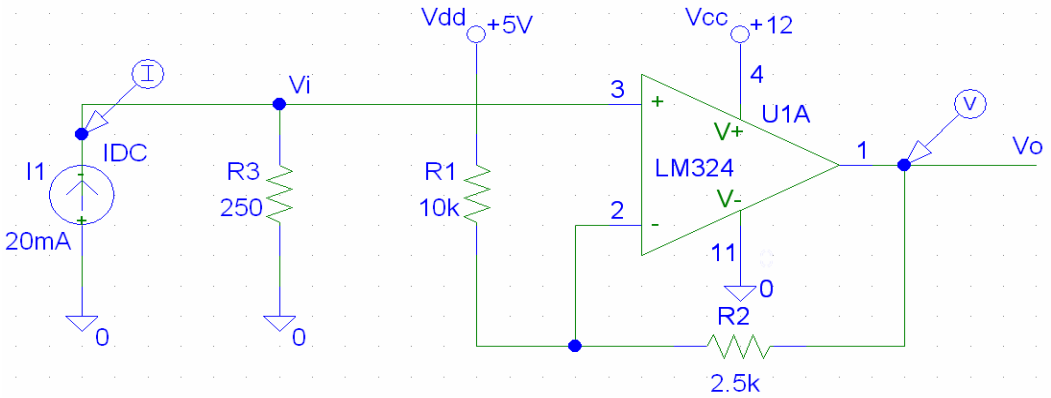


Figura 16: Esquema entrada analógica 4..20 mA

El análisis de este circuito está representado por la siguiente ecuación:

$$V_o = \frac{R\,2 + R\,1}{R\,1} \cdot V_i - \frac{R\,2}{R\,1} \cdot V_{dd}$$

Como para una tensión de entrada Vi = 1 la tensión de salida debe ser Vo = 0

$$0 = \frac{R2}{R1} + 1 - \frac{R2}{R1} \cdot 5 \quad \Rightarrow \quad 0 = 1 - 4 \cdot \frac{R2}{R1} \quad \Rightarrow \quad R1 = 4 \cdot R2$$

Si se elige para R1 un valor de 10KΩ, para R2 el valor quedará fijado en 2K5Ω. Es importante que el valor de 250Ω para R3 deba mantenerse, y aunque comercialmente no existe, se podrían asociar dos resistencias de 100Ω y 150Ω en serie. De la misma manera, para R2 también se puede conseguir su valor asociando 1KΩ y 1K5Ω en serie, que si están disponibles comercialmente.

Mediante el programa *PSpice Schematics* 9.1 se ha realizado la simulación del circuito de la figura con el siguiente resultado:

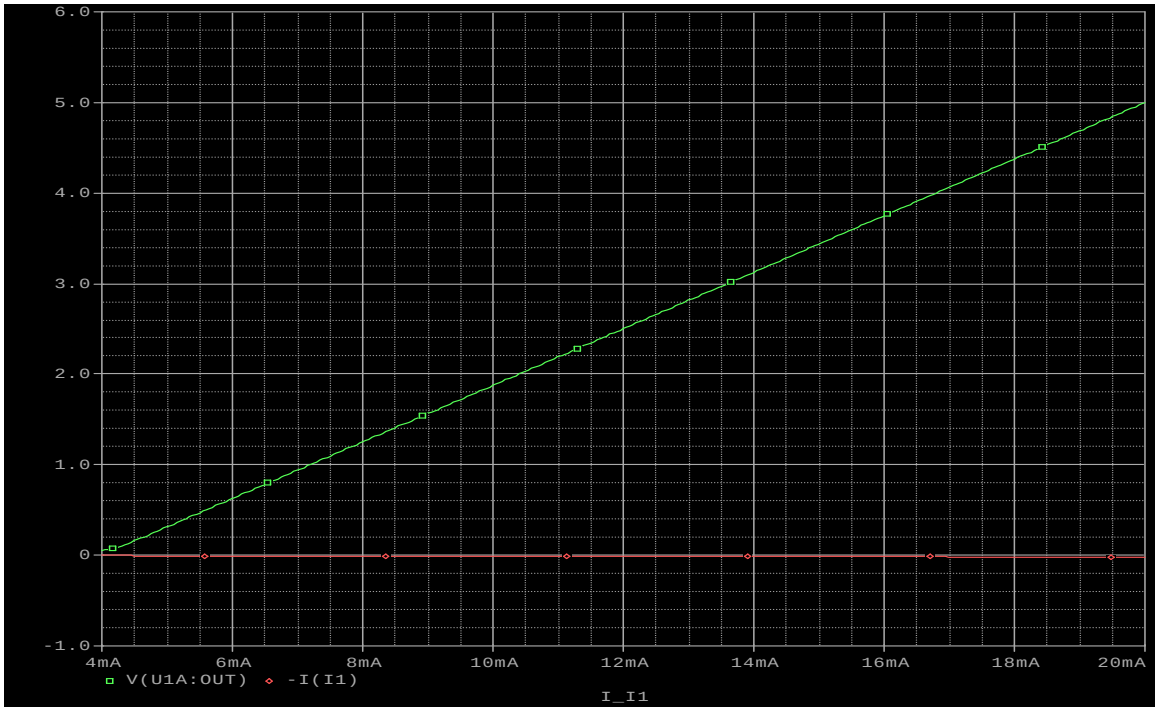


Figura 17: Simulación de la entrada analógica 4..20 mA

Se observa la linealidad y correspondencia entre el valor de intensidad definido por la norma 4..20 mA y el valor de la tensión dentro del rango definido 0..5 V.

Esta señal adaptada, se introduce al µC a través de las patillas AN2 y AN3 respectivamente, por ser dos las entradas requeridas de este tipo de señal.

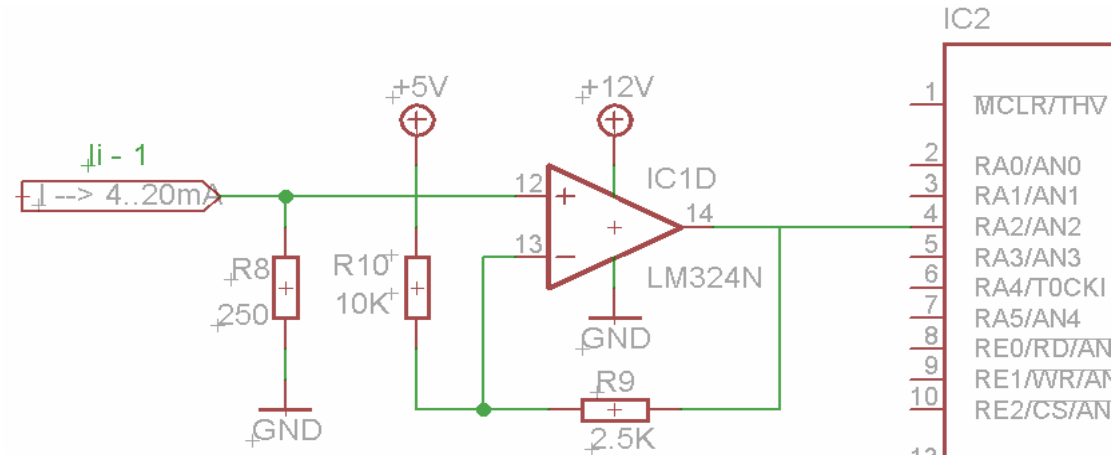


Figura 18: Conexión de la entrada analógica 4..20 mA al microcontrolador

7.4. Diseño de las salidas analógicas de 0..10 V

El µC 16F877A no tiene implementados conversores D/A en su interior, pero dependiendo de la resolución requerida la solución pasa por aprovechar la modulación por ancho de impulso PWM, cuyos generadores si integra.

En una señal PWM la frecuencia base está fijada, pero el ancho del pulso es variable, o sea, que el ciclo de trabajo se puede hacer oscilar entre el 0% y el 100% de acuerdo a la amplitud de la señal original. En la figura siguiente se observa una señal PWM y su transformación como una señal modulada.

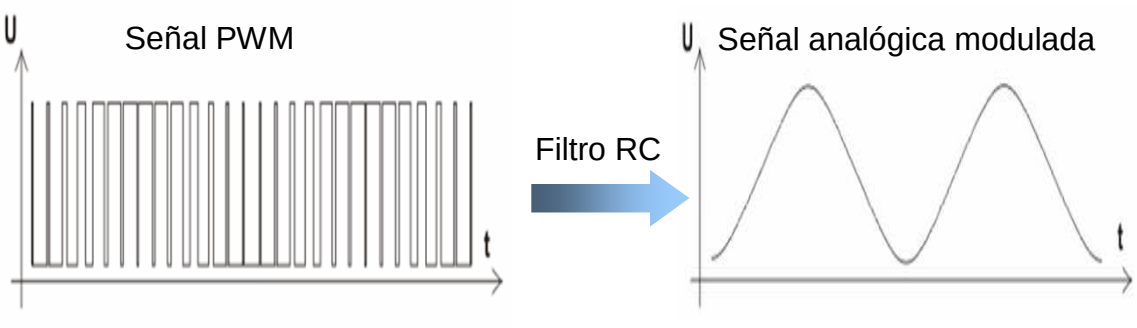


Figura 19: Conversión de una señal PWM en analógica

Un análisis de *Fourier* de una señal PWM muestra la existencia de un pico principal a la frecuencia Fn = 1/T, mostrando además la existencia de otros picos destacables a F = K/T, donde K es un entero. Los picos con K ≥ 2 son armónicos de la componente principal, los cuales son innecesarios y deben ser eliminados. Esto requiere que la señal PWM debe ser filtrada mediante filtros pasa bajos que puedan cancelar este ruido inherente a la propia señal.

En la figura siguiente se muestra el espectro de frecuencia de una señal PWM.

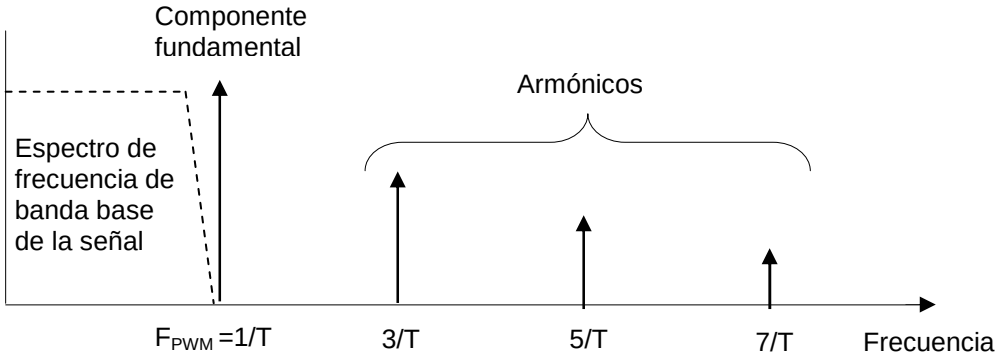


Figura 20: Espectro de frecuencia de la señal PWM

Este espectro muestra que el ancho de banda de la señal buscada podría ser $F_{bw} \leq (F_{PWM} = 1 / T)$. Pero esto requeriría el diseño de un filtro complejo y caro pues requiere la respuesta de una función rectangular. En la práctica se escoge F_{bw} con un valor más reducido. O sea $F_{PWM} = K \bullet F_{bw}$, donde K >> 1. Este valor de K será elegido dependiendo del valor en dB que se quiera rechazar, del propio ruido, de la componente fundamental.

Por tanto, para filtrar el ruido con un elevado dB en una señal PWM, se podría optar por filtros pasa bajos complejos de gran pendiente, pero lo más práctico y barato es que el µC module la señal PWM a una frecuencia elevada.

Para determinar este valor, se debe tener en cuenta la frecuencia de trabajo del µC, que se ha fijado en 4MHz, y la resolución de 10 bits que se quiere alcanzar. La resolución establece el número de posibles estados diferentes, que se determina mediante la expresión 2^{10} = 1024, que será la cifra en la que se dividirá la señal de reloj para determinar la frecuencia de modulación PWM.

$$f_{\min} = \frac{4\,MHz}{2^{10}} = \frac{4\cdot 10^{6}}{1024} = 3906.25\,Hz \qquad \Rightarrow \qquad T = \frac{1}{f} = \frac{1}{3906\ Hz} = 256\ \mu s$$

La forma de establecer este periodo se realiza por medio del registro PR2 del temporizador 2 junto con un registro de pre-escala, de la siguiente forma:

Periodo PWM = (PR2 + 1) • 4T_{OSC} • Valor de pre-escala del Timer2

- PR2, es un valor definible para fijar un valor de referencia. En este proyecto se establecerá a $255_{10} = FF_{16}$.
- T_{OSC}, es el periodo de oscilación del reloj: 4MHz = 1 / 4 • 10^6 = 250 ns.
- Valor de pre-escala del Timer2, hace referencia a unos bits que forman parte del registro TMR2 para variar el periodo, en este caso se pondrá a 1.

Al sustituir, se obtiene un periodo *PWM = (255 + 1) • 4 • 250 • 10^{-9} • 1 = 256 µs*, que corresponde a la frecuencia de 3906 Hz, confirmando el valor previsto. Con la siguiente expresión del *datasheet*, se cumple la resolución planteada:

$$\frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)}\,bits = \frac{\log\left(\frac{4\cdot 10^{6}}{3906}\right)}{\log(2)} = 10\ bits$$

Para realizar el filtrado paso bajo con un elevado corte, o sea con una pendiente pronunciada, se implementará un filtro *Butterworth* de 2º orden, con lo que se alcanza los -40dB, empleando un OP, como se muestra en el esquema siguiente:

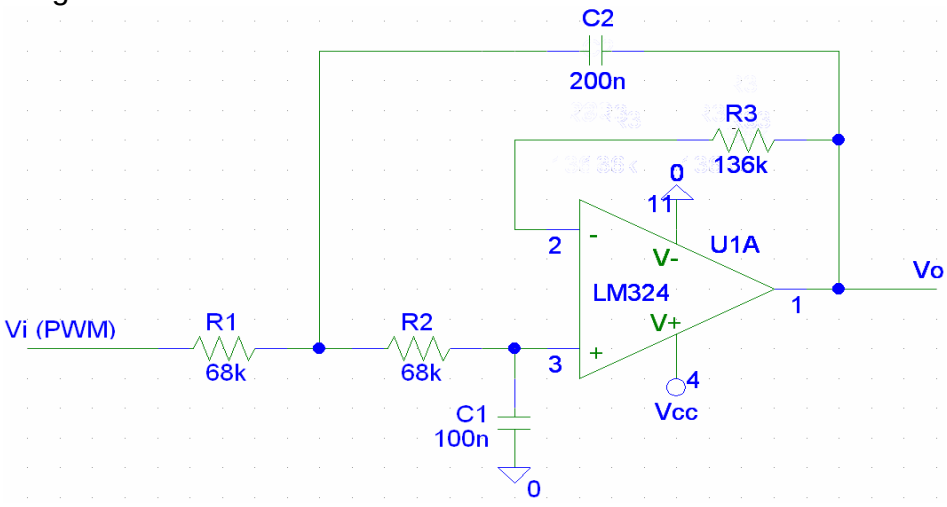


Figura 21: Esquema filtro paso bajo para PWM

El circuito del amplificador operacional es básicamente un seguidor de voltaje, donde la ganancia es unitaria, por lo que el voltaje a través de C1 es igual al voltaje de salida Vo. El procedimiento de diseño para este tipo de filtro es:

1. Seleccionar la frecuencia de corte ω_c o bien f_c. En este caso se escoge una frecuencia de corte de 16.6 Hz, para tener la seguridad de filtrar la frecuencia de 50 Hz en el tercer armónico, pues al ser la frecuencia de suministro de la red eléctrica es fácil que provoque ruido por inducción.

2. Escoger C1, seleccionando un valor entre 100 pF y 0.1 µF. Al ser la frecuencia de corte tan baja se escoge el valor más alto de 0.1 µF, para evitar obtener un valor muy alto de R, como se vera en el punto 4.

3. El valor del condensador C2 = 2 • C1. Por tanto, C2 = 0.2 µF

4. El valor de R se calcula mediante la siguiente expresión:

$$R = \frac{0.707}{\omega_c \cdot C} = \frac{0.707}{2\pi \cdot 16.6 \cdot 0.1 \cdot 10^{-6}} = 67.784\ \Omega \approx 68\,K\Omega$$

5. Se selecciona el valor de las resistencias cumpliendo estas condiciones: R = R1 = R2 y Rf = 2R, por tanto: *R1 = R2 = 68 KΩ y Rf = 136 KΩ*

Para obtener un rango de variación de 0..10 V, se coloca a la salida del filtro paso bajo otro OP en configuración de no inversor, para que amplifique por un factor de 2 y a la vez sea capaz de entregar una corriente superior a los 200 mA, mediante el OP LM12. Este OP tiene alimentación simétrica, pero como solo va a trabajar sobre el rango positivo, se puede poner a tierra la alimentación negativa y sabiendo que Vo = 2 Vi, su comportamiento será:

$$Vo = Vi\left(1+\frac{R1}{R2}\right) \Rightarrow 2\cdot Vi = Vi\left(1+\frac{R1}{R2}\right) \Rightarrow 2 = 1+\frac{R1}{R2} \Rightarrow 1 = \frac{R1}{R2}$$

La relación final indica que el valor de las dos resistencias debe ser igual, eligiéndose un valor elevado de 10 K Ω, para limitar la corriente a entregar. El circuito siguiente recoge el esquema completo de la salida analógica por medio de PWM y mediante *schematics* se realizan las gráficas de simulación:

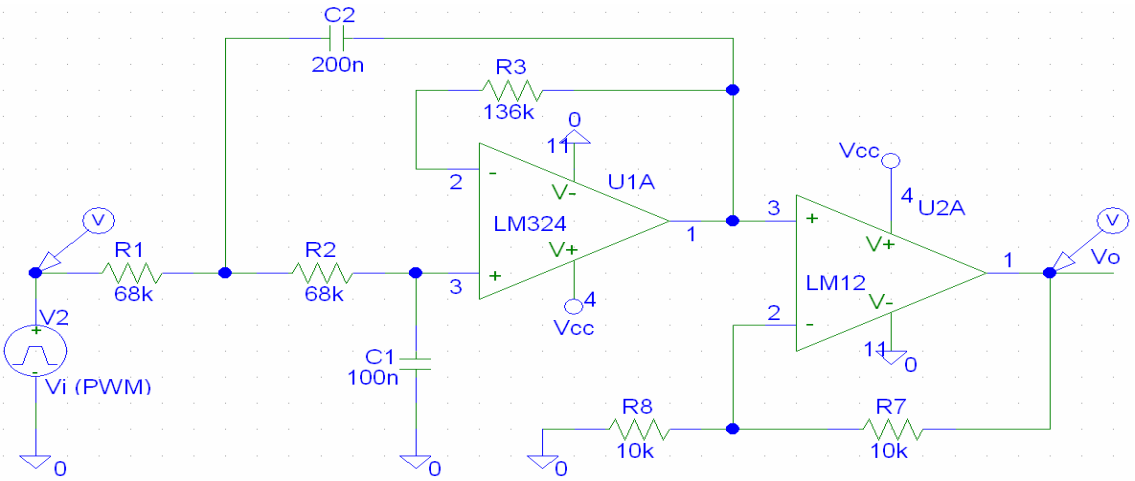


Figura 22: Esquema amplificador de tensión

La gráfica siguiente muestra la salida de la fuente *VPulse* con los siguiente valores en sus parámetros: DC = 0, AC = 0, V1 = 0, V2 = 5V, TD = 64u, TR = 0.1u, TF = 0.1u, PW = 192u y PER = 256u, los cuales caracterizan una señal PWM con un ciclo en estado alto del 75%:

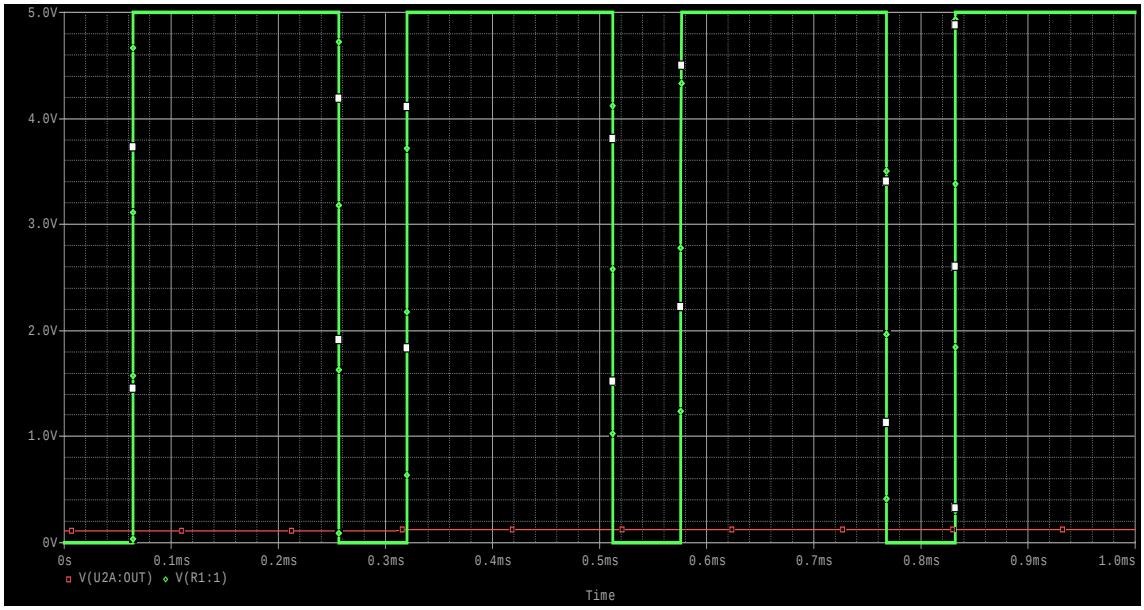


Figura 23: Gráfica de la señal PWM al 75% de estado en alto

Se puede observar, en la gráfica que sigue, como el voltaje de salida, en rojo, se sitúa en los 7.5V, como corresponderían a una señal PWM del 75% de ciclo en estado alto. En verde, se compara con la señal de entrada PWM.

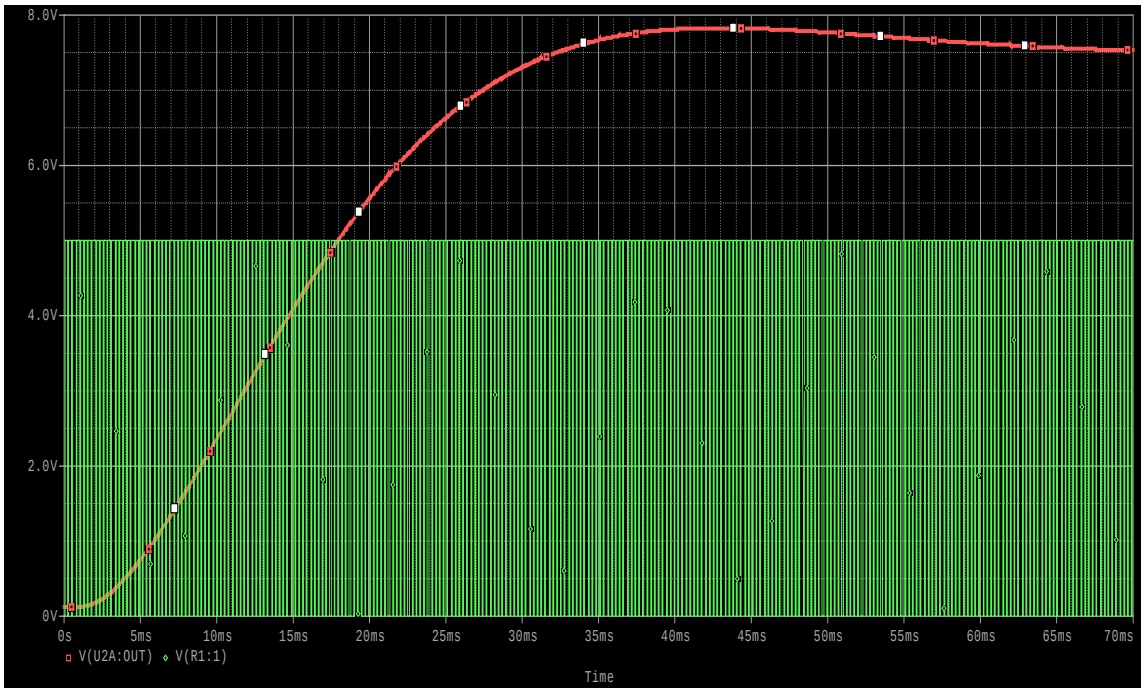


Figura 24: Gráfica de salida modulada del conversor PWM de 0..10 V

En la gráfica siguiente se puede observar el espectro de frecuencias de la señal modulada así como sus armónicos, en rojo, y como la señal de salida no ha sufrido perturbaciones debido a ellas, gracias al filtro paso bajo.

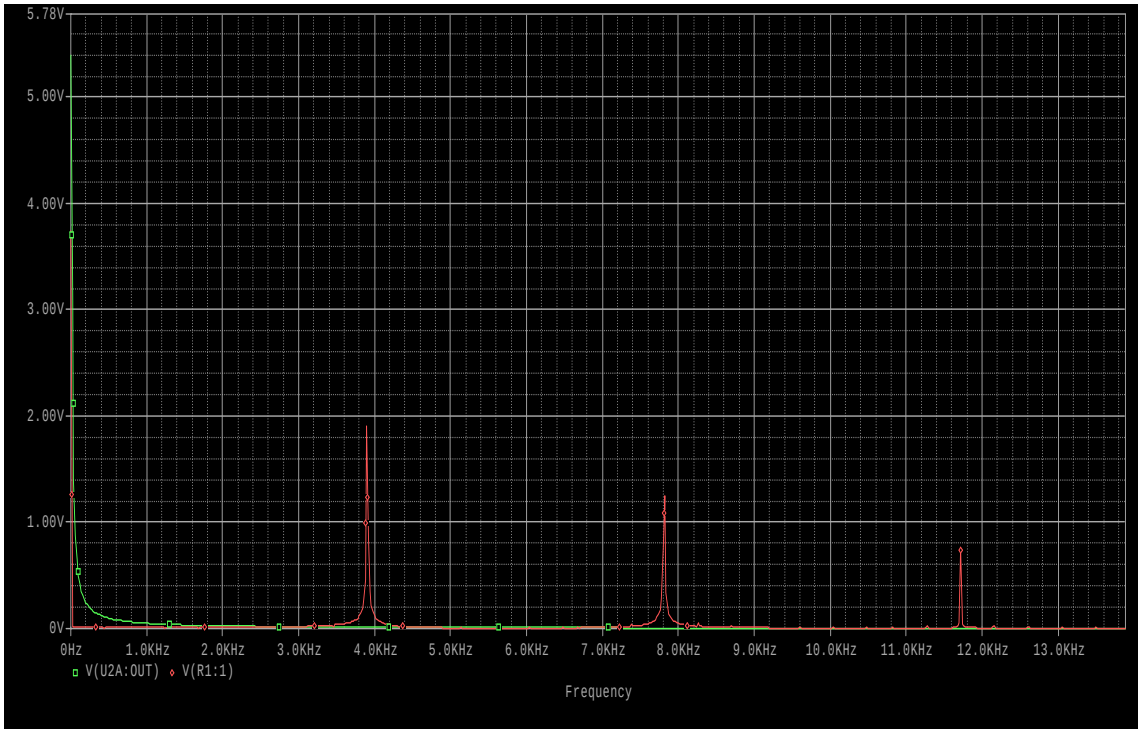


Figura 25: Gráfica del espectro de frecuencias de la señal PWM

El esquema completo del adaptador de la salida PWM a una salida analógica de 0..10 V sería el siguiente, conectándose en las patillas CCP1 (17) y CCP2 (16) respectivamente, cada una de las dos salidas de las que se dispone:

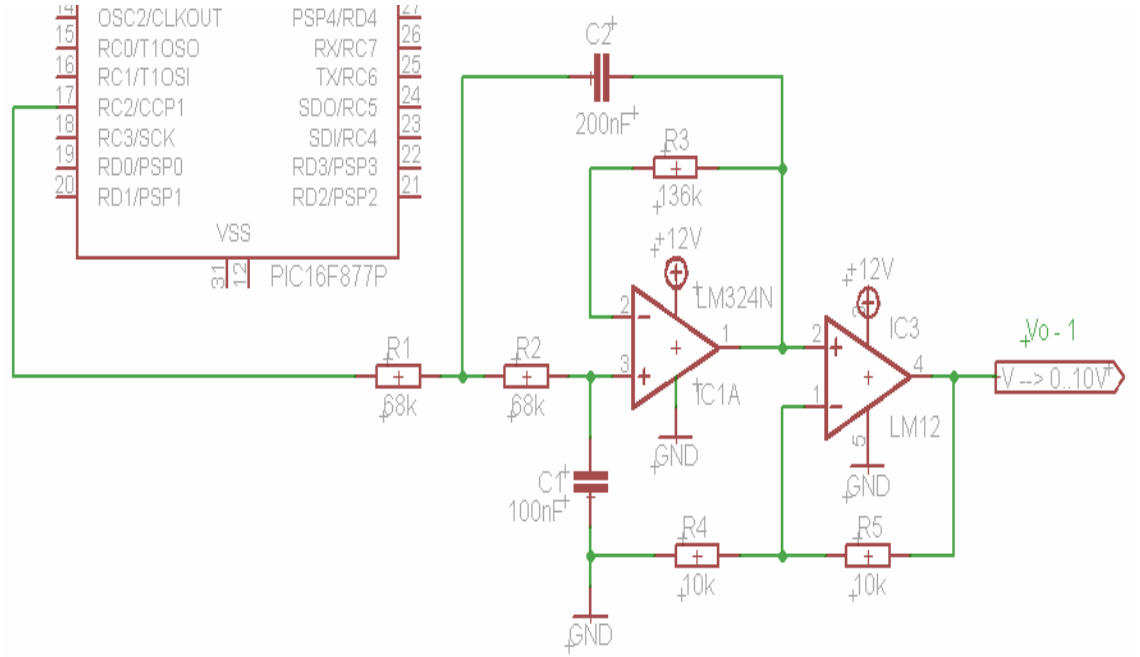


Figura 26: Esquema completo de la salida analógica 0..10 V

7.5. Diseño de la Fuente de alimentación

El diseño tiene en cuenta las tensiones y corrientes necesarias para proporcionar la potencia demandada por el circuito, cubriendo además el amplio rango de alimentación, requerido por el proyecto.

Se analizó el desarrollo de una fuente conmutada, por su alta eficiencia y reducido consumo de potencia en forma de calor, pero este tipo de fuentes producen bastante ruido debido a su alta velocidad de conmutación y su tratamiento y estabilización requiere más espacio en la placa. Por lo que, finalmente se recurre a reguladores integrados por su baja generación de ruido.

Diseño de la entrada de alimentación en continua

Dado el amplio margen de alimentación se diseñara un circuito que a su salida se obtenga siempre la misma tensión con independencia del voltaje que se introduzcan por la entrada, mientras se suministra la corriente necesaria. El circuito que proporciona este comportamiento se analiza a continuación:

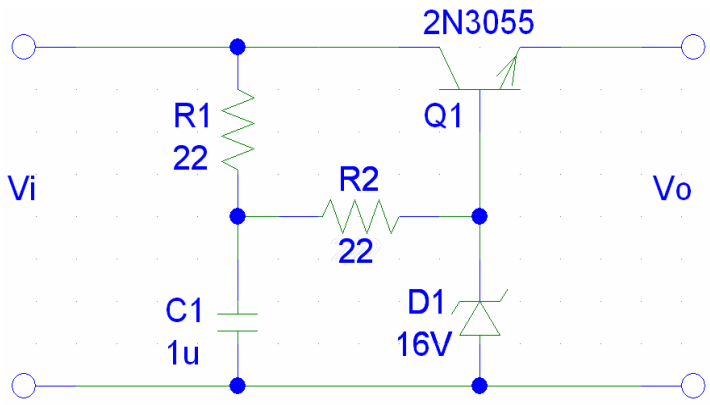


Figura 27: Etapa reductora de la fuente de alimentación

La tensión de salida Vo esta fijada mediante el diodo D1 cuya tensión *zener* Vz, es de 16 V menos la caída de voltaje entre base y emisor, Vbe, del transistor Q1. Vo = Vz – Vbe = 16V – 0.7 = 15.3 V, valores suficientes para que los reguladores de tensión integrados de la siguiente etapa, operen correctamente.

El resto de la tensión se queda en el transistor Q1, el cual regula la corriente, I_L que alimenta al circuito, teniendo que ser capaz de proporcionar hasta 1 A y aguantar la tensión máxima entre colector y emisor Vce, de 32 V cuando este conectada a una fuente de 48 VDC, siendo este el caso más desfavorable al tener que disipar una potencia de 32 W en forma de calor, escogiéndose el transistor 2N3055 por ser capaz de entregar estas prestaciones.

Esta potencia transformada en calor, debe mantenerse dentro de unos límites funcionales. La facilidad de eliminar este calor en un encapsulado viene determinada por dos factores, la resistencia a la disipación dentro del encapsulado, Rtjc y por otro la resistencia a la disipación desde el encapsulado hacia el exterior, la suma de estos dos valores se expresa como Rtja.

El transistor 2N3055 viene encapsulado con un formato TO-3, cuyo valor Rtja típico es de 35ºC/W, con este dato se puede calcular la potencia máxima de trabajo sin disipador, la cual esta determinada por la relación entre el rango de temperatura de trabajo y la resistencia a disipar este calor:

$$Pmax = \frac{Tjmax - Ta}{Rtja}$$

Donde Tjmax es la temperatura máxima que puede alcanzar la unión semiconductora y Ta es la temperatura ambiente. Del *datasheet* del fabricante se obtiene que Tjmax = 200 ºC. Por tanto, considerando Ta a 25ºC:

$$Pmax = \frac{200^{\circ}C - 25^{\circ}C}{35^{\circ}C/W} = 5\ Watios$$

Por tanto a partir de 5W se debe poner un disipador de calor, en este caso, siendo 32 W la potencia a disipar, con mayor razón será necesario el facilitar la disipación de calor a través de este elemento.

Si se prescindiera del disipador de calor, la temperatura alcanzada por la cápsula del transistor se calcularía como la suma de la temperatura ambiente ta, y el poder de disipación de la cápsula, el cual está determinado por el producto de la potencia de trabajo, Pt, por la resistencia a su disipación, Rtja:

$$Tc = Ta + Pt \cdot Rtja = 25^{\circ}C + 32\,W \cdot 35^{\circ}C/W = 1.145^{\circ}C$$

Este valor tan elevado evidencia la necesidad de colocar un disipador de calor, pues al llegar a los 200 ºC se destruiría el transistor. El radiador reduce la resistencia del todo el conjunto, con lo que se consigue desahogar más calor.

El transistor en esta configuración de seguidor de emisor es un amplificador de corriente, sirviendo además para adaptar impedancias, pues al tener una impedancia de entrada mucho más alta que la de salida, los circuitos anexos se verán menos afectados.

Las resistencias R1 y R2 proporcionan la corriente de polarización de la base de Q1 y de D1, asegurándose de cumplirlo para la tensión mínima posible. El valor de esta resistencia se divide entre dos para intercalar entre ellas un condensador C1 que cumpla Xc << R1 para eliminar corrientes inducidas. El cálculo de estas dos resistencias esta definido por siguiente expresión, donde h_{fe} es la ganancia del transistor:

$$R1 + R2 = \frac{Vi - Vz}{I_{Z\min} + I_{B\max}} = \frac{Vi - Vz}{I_{Z\min} + \frac{I_{L\max}}{h_{fe}}} = \frac{17V - 16V}{2mA + \frac{1A}{50}} = 45\,\Omega$$

El condensador C1 se calcula para eliminar el ruido provocado por el convertidor PWM en el intervalo de frecuencia de 1.22 KHz, y poder evitar su inducción hacia la alimentación, estimándose una impedancia Xc < 0.2 << R1.

$$C = \frac{1}{2\pi \cdot f \cdot Xc} = \frac{1}{2\pi \cdot 1220\ Hz \cdot 0.2\,\Omega} = 652.27\ \mu F$$

La capacidad comercial será de 680 µF, el condensador será de tipo electrolítico y al menos de 63 V para soportar los valores altos de alimentación.

Diseño de los reguladores de tensión para 5 V y 12 V

Como se ha visto en los anteriores apartados se necesitan dos tensiones fijas:

- **+5 V**: Para alimentar el µC 16F877A y con un consumo total de 0.5 mA.

- **+12 V**: Tensión del MAX 485 consumiendo 60 mA, los 2 C. I. LM324 tomando 150 mA cada uno y los dos 2 OP LM12 donde el consumo se dispara hasta los 0.6 A, totalizando casi 1A de consumo total para 12 V.

Para implementar ambas tensiones se recurre a los reguladores de tensión 7805 y 7812 para suministrar las tensiones de +5V y +12V respectivamente. El siguiente esquema describe el circuito completo de la fuente de alimentación:

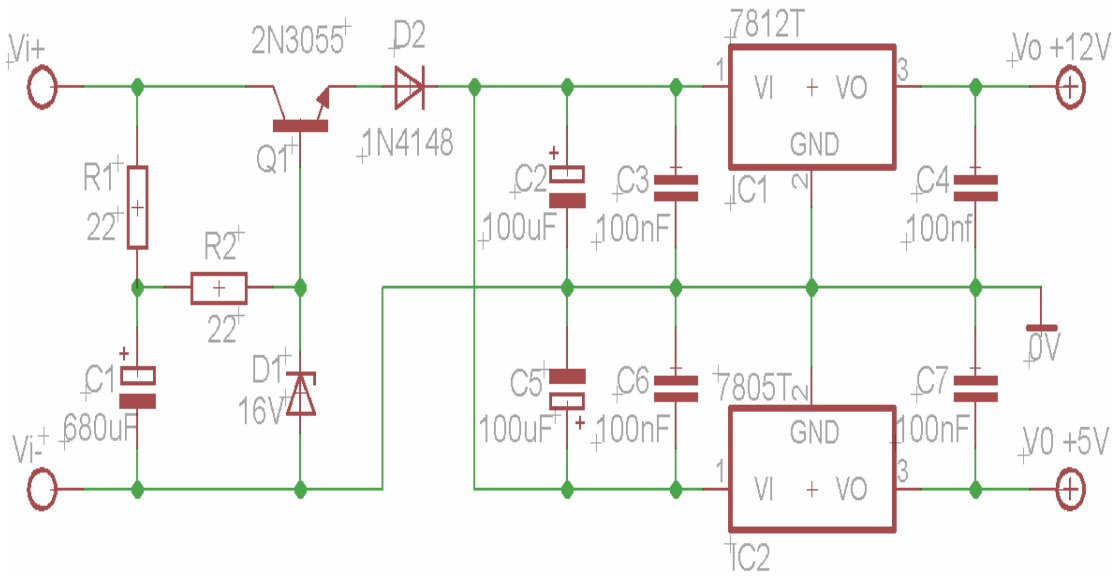


Figura 28: Esquema completo de la fuente de alimentación

El diodo D2 se incorpora para prevenir al circuito de posibles inversiones de polaridad en la entrada. El regulador 7805 tendrá que disipar una mayor potencia al tener una mayor diferencia de potencial entre la entrada y la salida. Los condensadores de 100uF filtran el ruido por variaciones de corriente en los componentes y los de 100nF estabilizan antes y después del regulador.

8. Implementación del programa de control

En los capítulos anteriores se han analizado las necesidades del proyecto, después se han elegido los componentes cuya función cubriera estos requisitos y finalmente se le ha dado forma a un circuito completo cuyas especificaciones eléctricas faciliten la implementación del software de control del conjunto.

Para facilitar el desarrollo de este programa de control, se estudiaran las distintas tareas que tendrá que resolver, para lo que se dividirán y organizarán en forma de bloques con el fin de poder estructurarlo en forma de organigrama y realizar la implementación mediante lenguaje C.

Las principales tareas que tendrá que resolver son las siguientes:

1. Leer los parámetros de configuración de paridad, velocidad de transmisión y número de dispositivo a través de las entradas del puerto B.

2. Leer los datos, convertidos en binario, de las cuatro entradas analógicas recibidas por los canales AN1, AN2, AN3 y AN4.

3. Modificar los datos que permiten variar las dos salidas analógicas por medio de pulsos PWM a través de las salidas CCP1 y CCP2.

4. Tratamiento del protocolo ModBus, para reconocimiento de las tramas que se reciben del maestro, y emisión de los mensajes de respuesta a este.

5. El uso del bit de paridad por el incremento de complejidad que tiene el poder manejar un noveno bit.

El tratamiento de errores consistente en el envío al maestro de un mensaje respuesta donde se indica el error, se ha simplificado, de forma que cuando se reciba un mensaje erróneo no se enviará ninguna respuesta, por lo que pasado un tiempo, definido en el maestro, este volverá a mandar el mensaje. Ese tiempo dependerá de la velocidad de transmisión y de la longitud del cable.

Para poder gestionar estas funciones desde el maestro se tendrá que identificar las direcciones estándar del protocolo ModBus con las correspondientes direcciones de memoria específicas de este µC mediante el siguiente mapa de memoria:

Dirección de memoria	Descripción	Rango de medida
30001	Entrada analógica 1	0..10 V
30002	Entrada analógica 2	0..10 V
30003	Entrada analógica 3	4..20 mA
30004	Entrada analógica 4	4..20 mA
40001	Salida PWM analógica 1	0..10 V
40002	Salida PWM analógica 2	0..10 V

Tabla 9: Mapa de memoria

8.1. Análisis del programa de control

La misión principal de este dispositivo es estar a la espera de recibir un mensaje, a través de la comunicación serie por medio del protocolo ModBus RTU. Este mensaje o trama podrá consistir en una de las dos tareas siguientes:

o Leer las entradas analógicas solicitadas a través de la función 04h de ModBus y enviarlas como mensaje de respuesta al dispositivo maestro.

o Generar una señal analógica a partir de los datos digitales enviados a través de la función 06h de ModBus y responder con una replica del mensaje recibido.

Se describen los mensajes de petición y de respuesta para ambas funciones:

Función ModBus 04h

Lee el contenido binario de los registros de entrada (referenciados como 3xxxx) en el esclavo. El *broadcast* no es soportado.

El mensaje de petición contiene la dirección del dispositivo destino, la función y especifica el registro de inicio y cantidad de registros a ser leídos. Los registros 1…n son *direccionados* en las posiciones 0…n-1. Los dos últimos bytes contienen la comprobación de redundancia cíclica, CRC.

En el caso de este dispositivo, cuya dirección sea la 13h, el mensaje de petición del maestro está formado por 8 bytes y tiene el siguiente formato:

Pos	Nombre del campo	Valor *hex.*
0	Dirección del dispositivo	13
1	Función	03
2	Dirección de inicio alto	00
3	Dirección de inicio bajo	00
4	Número de registros alto	00
5	Número de registros bajo	04
6 - 7	Comprobación del error CRC (2 bytes)	XXXX

Tabla 10: Formato petición función 04h ModBus

En este ejemplo, los bytes 2 y 3 contienen la dirección 0000h, indicando que las direcciones a leer son a partir de la 30001, mientras que los bytes 4 y 5 definen que los registros a leer son 4.

El mensaje de respuesta está formado por la dirección del esclavo, la función solicitada y el número de bytes involucrados para transmitir los datos solicitados. Los datos son empaquetados como dos bytes por registro.

La respuesta para la petición anterior está formada por 13 bytes y tendrá el siguiente formato, suponiendo que los datos contenidos desde 30001 a 30004 son 00FEh, 01DCh, 02BAh, 0398h y que se corresponden a la lectura de los cuatro canales analógicos.

Pos	Nombre del campo	Valor hex.
0	Dirección del esclavo	13
1	Función	04
2	Número de bytes	8
3	Dato alto primer registro 30001	00
4	Dato bajo " " "	FE
5	Dato alto segundo registro 30002	01
6	Dato bajo " " "	DC
7	Dato alto tercer registro 30003	02
8	Dato bajo " " "	BA
9	Dato alto cuarto registro 30004	03
10	Dato bajo " " "	98
11 - 12	Comprobación del error CRC (2 bytes)	XXXX

Tabla 11: Formato respuesta función 04h ModBus

El número de bytes necesarios para enviar los datos se coloca en la posición 2 de la trama de respuesta.

Función ModBus 06h

Carga un valor a un solo registro (referenciados como 4xxxx). Cuando se produce *Broadcast*, la función carga el mismo valor de referencia en todos los dispositivos esclavos de la red.

El mensaje de petición contiene la dirección del dispositivo destino, la función y especifica la referencia del registro a ser configurado. Los registros son direccionados de manera que el registro 1 hace referencia a la posición 0.

Para este dispositivo, cuya dirección designada es la 13h, y al que se quiere modificar el valor del registro 30001, con el valor 0123h, el mensaje de petición del maestro estará formado por 8 bytes y tendrá el siguiente formato:

Pos	Nombre del campo	Valor *hex.*
0	Dirección del dispositivo	13
1	Función	06
2	Dirección de registro alto	00
3	Dirección de registro bajo	00
4	Configurar dato alto	01
5	Configurar dato bajo	23
6 - 7	Comprobación del error CRC (2 bytes)	XXXX

Tabla 12: Formato petición-respuesta función 06h de ModBus

La dirección completa 0000h indica que las dirección a leer es la 40001, que corresponde a la salida analógica asociada a la salida PWM 1. La dirección 0001h indicaría la salida analógica PWM 2.

El mensaje de respuesta es una replica del mensaje de petición, que el programa reenvía una vez completada la operación solicitada.

Organigrama general de inicialización y sincronización de tareas

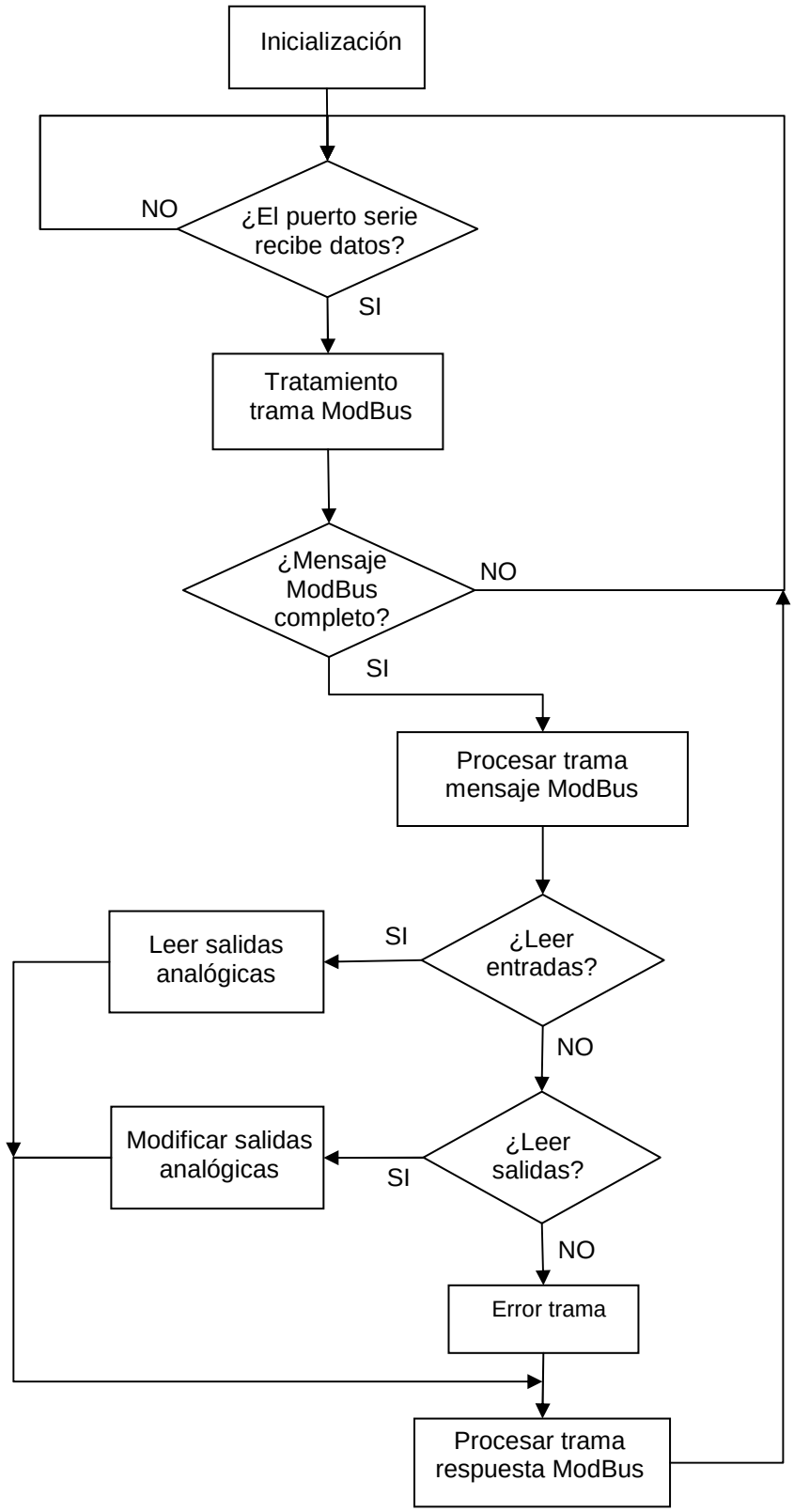


Figura 29: Organigrama básico del programa de control

8.2. Programa de Inicialización

Esta formado por la combinación de una cabecera que define los parámetros generales, y por una función principal que actúa como eje central. Además, se describe una función que sirve para limpiar el buffer de recepción-transmisión.

Cabecera del programa

Se incluye el archivo <16F877A.h> para manejar este µC concreto mediante su fichero de definición. Por medio de la directiva #FUSES se describen los parámetros de inicialización de la palabra de configuración, también define las librerías necesarias para el desarrollo del programa tales como el la velocidad de reloj y el manejo del protocolo RS-232, para indicar los pines de transmisión y recepción, así como la patilla de habilitación de estos modos al C.I. MAX485.

```
#include <16F877A.h>
#device adc = 10

#FUSES NOWDT                    // No Watch Dog Timer, sin perro guardian
#FUSES XT                       // Velocidad fijada con cristal de cuarzo
#FUSES NOBROWNOUT               // No brownout reset
#FUSES NOLVP                    // No low voltage prgming, used for I/O

#use delay(clock = 4000000)     // Frecuencia de reloj = 4 MHz
#use RS232(XMIT = PIN_C6, RCV = PIN_C7, ENABLE = PIN_D7)
```

Dentro de la cabecera se declaran las variables que definen el comportamiento del puerto B que servirá para modificar externamente valores de configuración. También se definen los registros de control de la comunicación serie para tener acceso al noveno bit cuando se configure una transmisión con paridad.

```
#byte trisB = 0x86              //registro trisB   en dirección 86h
#byte portB = 0x06              //    "      portB en      "      06h
#byte rcsta = 0x18              //    "      rcsta   en      "      18h
#byte txsta = 0x98              //    "      txsta   en      "      98h
```

Siguiendo con la cabecera, se definen las constantes y variables globales que se manejan en todo el programa, tales como el número de dispositivo y todas las variables relacionadas con la transmisión serie.

```
byte num_dispos;                //guarda número de este esclavo
int1 paridad;                   //recoge la configuración de paridad
int const tama_buffer = 13;     //tamaño buffer comunicación serie
byte buffer_datos[tama_buffer]; //contiene la trama MODBUS recibida
int indice_buffer = 0x00;       //siguiente dato en buffer
int1 buffer_paridad[8];         //contiene bits paridad de cada byte trama
int1 trama_ok;                  //trama recibida para este dispositivo ok
int1 error_paridad;             //indica error paridad en trama recibida
int1 trama _ error;             //indica error CRC en la trama recibida
```

Función principal del programa, función *main*.

En este apartado solo se comentan las tareas básicas relacionadas con la inicialización, dejándose para los apartados siguientes, donde se describen las operaciones implementadas, las explicaciones respectivas realizadas en la función principal.

Dentro de la función principal se declaran e inicializan los valores necesarios para el manejo de las funciones involucradas en cada parte del programa, como son las de interpretación de tramas, tratamiento de las comunicaciones y la lectura-escritura de los valores analógicos.

En la función se declaran y definen las variables que recogen la configuración externa leída a través del puerto B, como son los parámetros para inicializar la velocidad de comunicación serie y la paridad, así como la definición de variables auxiliares.

También se define el comportamiento del puerto B, para que todos sus pines trabajen como entradas, para poder leer los parámetros configurables externamente mediante micro-interruptores vistos en el apartado 7.2.

```
byte valor_puerto_B;          //guarda todos valores de configuración
byte val_veloc;               //guarda el valor de la velocidad
byte temp;                    //variable auxiliar
byte indice;                  //utilizada de contador dentro de bucles
int1 par;                     //guarda valor paridad calculado por byte
trisb = 0b11111111;           //todo entradas en el puerto B
```

Mediante divisiones y productos se consigue desplazar a derecha e izquierda respectivamente para poder aislar cada uno de los valores de configuración y poder tratarlos adecuadamente en el resto del programa.

```
valor_puerto_B = portB;              //copia estado del puerto B
paridad = bit_test(valor_puerto_B, 1); //obtiene tipo de paridad
temp = valor_puerto_B / 2;           //numero dispositivo + velocidad
num_dispos = valor_puerto_B / 8;     //obtiene numero de dispositivo
val_veloc = temp - 4*num_dispos;     //obtiene tipo velocidad
```

A partir del valor de la paridad se selecciona si existe o no bit de paridad, mediante el uso de la directiva RS232. En caso de que la paridad sea utilizada, esta será por defecto paridad par.

```
if (paridad)                         //Según paridad se activa la
  {
    #use RS232(PARITY = E)           //directiva E = par ó
  }
  else
  {
    #use RS232(PARITY = N)           //directiva N = sin paridad
  }
```

A partir del valor de los dos bits que definen la velocidad se selecciona la velocidad correspondiente en baudios, así como el tiempo máximo entre tramas para cada una de las velocidades escogidas, como se vio en el apartado 5.2., dedicado al protocolo ModBus RTU.

```
switch (val_veloc)                    //convierte tipo de velocidad
  {                                   //en velocidad transmisión
    case 0:
      set_uart_speed(2400);           //'00b' = 2.400 bps
      #use RS232 (TIMEOUT = 16)       //ms mínimo entre tramas =3.5*11/2400
      break;

   case 1:
      set_uart_speed(4800);           //'01b' = 4.800 bps
      #use RS232 (TIMEOUT = 8)        //ms mínimo entre tramas =3.5*11/4800
      break;

   case 2:
      set_uart_speed(9600);           //'10b' = 9.600 bps
      #use RS232 (TIMEOUT = 4)        //ms mínimo entre tramas =3.5*11/9600
      break;

   case 3:
      set_uart_speed(19200);          //'11b' = 19.200bps
      #use RS232 (TIMEOUT =2)         //ms mínimo entre tramas =3.5*11/19200
      break;
  }
```

Función para limpieza del buffer

La siguiente función permite vaciar el *array Buffer_datos* cada vez que se termine la transmisión del mensaje de respuesta, para que esté preparada para recibir otra trama y no pueda inducir a errores con valores previos.

Mediante un bucle se recorre el *array* y se van poniendo a cero cada posición de manera secuencial, incluyendo la variable que indica la recepción de una función y la de error en la trama.

```
Void vacia_buffer(void)
{
  int indice;
  for(indice=0; indice > tama_buffer; indice++)      //bucle que pone a cero
  {
    buffer_datos[indice] = 0x00;                     //todos los datos del buffer
  }
  indice_buffer = 0x00;                              //inicialización para recibir
  trama_ok = 0;                                      //la siguiente función
  trama_error = 0;
}
```

8.3. Programa de lectura de las entradas analógicas

Dentro del programa principal se inicializa el modo de trabajo de los canales analógicos. El modo permitido más aproximado al requerido es donde trabajan los cinco canales AN0 – AN4. Además se define el origen interno de sincronización de reloj para realizar el muestreo de las señales.

```
setup_adc_ports(AN0_AN1_AN2_AN3_AN4);      // define patillas analógicas
setup_adc(ADC_CLOCK_INTERNAL);             // define velocidad de muestreo
```

Para manejar y procesar las entradas analógicas se define la siguiente función que lee los canales indicados por la función 04h. Mediante un bucle que se inicia en la dirección indicada por la posición 3 y por la cantidad de registros que señala la posición 5 del *array buffer_datos*, se selecciona primero el canal y pasado un intervalo de 20µs, para dar tiempo a cargar el condensador, se muestrea el canal ajustado.

Estos valores se guardan en las posiciones correspondientes del *array buffer_datos*, convirtiendo los variables de 16 bits a bytes mediante desplazamientos de 8 bits, el cual almacena la trama respuesta para la función 04h de ModBus, para ser enviada al dispositivo maestro.

Terminado el proceso se le da formato a la respuesta indicando el número total de bytes a transmitir, se calcula el valor de los dos bytes del CRC y se transmite la trama indicando el total de bytes a transmitir.

```
void leer_entra_analog(void)
{
  byte dir_inicio;                          //dirección desde se comienza a leer
  byte dir_final;                           //número de registros a leer
  byte canal;                               //contiene canal analógico a leer
  byte buffer_Pos;                          //posición buffer donde se guardan datos
  int16 dato_aux;                           //variable para conversión int16 -> byte

  dir_inicio = buffer_datos[3];             //inicializa con valor primera dirección
  dir_final = dir_inicio + buffer_datos[5]; //inicializa con la última dirección a leer
  buffer_pos = 3;                           //posición donde empiezan a guardar datos
  for (canal = dir_inicio; canal < dir_final; canal++)
  {
    set_adc_channel(canal);                 //selección del canal
    delay_us(20);                           //retardo de 20us
    dato_aux = read_adc();                  //lee y guarda dato digital del canal
    buffer_datos[buffer_pos++] = dato_aux >> 8;   //desplaza 8 bits guarda altos
    buffer_datos[buffer_pos++] = dato_aux;        //guarda los 8 bits bajos
  }
  buffer_datos[2] = buffer_pos - 3;         //pone en buffer total datos a transmitir
  calculo_crc(buffer_pos + 2);              //calcula el crc de la trama a enviar
  trans_respuesta(buffer_pos + 3);          //transmite el buffer con tamaño adaptado
}
```

8.4. Programa de escritura de las salidas analógicas

Para la escritura en las salidas CCPX para conformar la señal PWM que servirá como señal analógica de salida se declaran las siguientes instrucciones que configuran los registros involucrados en el programa principal, los cuales servirán para conformar la onda cuadrada de pulso variable y que los filtros paso bajo que tienen a la salida convertirán en una señal continua analógica.

Se establece la frecuencia de salida del modulador del pulso de ondas PWM, mediante la programación del temporizador T2, cuyo valor se fija en 3906 Hz que corresponde a un periodo de 256 µs. A continuación se define el modo de trabajo de los dos módulos CCP para que se comporten como generadores PWM.

```
setup_timer_2(T2_DIV_BY_1, 255, 1);      // PR2 = 255, T = 256us, F = 3906Hz
setup_ccp1(CCP_PWM);                     // CCP1 configurado como PWM
setup_ccp2(CCP_PWM);                     // CCP2 configurado como PWM
```

Para convertir los valores digitales en la duración correspondiente de los pulsos PWM se recurre a la siguiente función. Donde mediante la posición baja correspondiente de la trama ModBus, posición 3, se determina el canal PWM a escribir. La dirección 30001 queda fijada por el valor 0 y a la dirección 30002 se establece por el valor 1.

Mediante un producto de 2^8 = 256 y una suma se consigue pasar los 2 bytes que se encuentran en las posiciones 4 y 5 del *array buffer_datos* a una sola variable de 16 bits que determina el valor analógico de salida.

Al final de la función se trasmite la respuesta al maestro, la cual será una replica del mensaje recibido para la función 06h ModBus.

```
void modif_sal_analog(void)
{
   int16 dato_aux;                       //une dos bytes datos en uno solo 16 bits

   dato_aux = buffer_datos[0x04] * 256 + buffer_datos_s[0x05];
   switch (buffer_datos[0x03])           //testea dirección baja para determinar
  {
     case 0x00:                          //si se escribe en el canal PWM 1
       set_pwm1_duty(datos_aux);         //coloca el dato en PWM 1
       break;

     case 0x01:                          //ó se escribe sobre el canal PWM 2
       set_pwm2_duty(datos_aux);         //coloca el dato en PWM 2
       break;
  }
  trans_respuesta(8);                    //transmite el buffer con tamaño 8
}                                        //replica del mensaje del maestro
```

8.5. Interrupción de control de la comunicación serie

El programa deberá estar continuamente a la espera de recibir mensajes dirigidos hacia él para poder realizar y transmitir las respuestas solicitadas, y cuando termine la transmisión volver a habilitar el modo de recepción.

Para estar preparado para recibir mensajes sin supervisar continuamente la comunicación se recurre al manejo de interrupciones, pues permiten que cualquier suceso interior o exterior interrumpa la ejecución del programa y salte a ejecutar una rutina de atención de interrupciones, previamente definida.

Mediante la directiva *init_rda* se especifica que la función que le sigue es una función de interrupción que se activará cuando reciba un dato RS232. En el programa se define la función *serial_isr()* que permite almacenar en el *array buffer_datos* los bytes que se van leyendo con el comando *getc*. Cuando esta rutina toma el comando *kbhit()* permite la lectura de la línea de recepción cada vez que llega un nuevo dato, para evitar nuevas interrupciones.

Si la paridad esta activada, esta rutina también se encarga de leer este noveno bit a través del bit 0 del registro *rcsta* almacenándolo en el *array buffer_paridad* para analizarlo y verificándolo dentro del programa mediante otra función.

Una vez recibidos los 8 bytes que completan toda la trama ModBus, se comprueba que el destinatario del mensaje coincida con en número de dispositivo configurado externamente, poniendo a 1 la variable *trama_ok* cuando coinciden y a 0 cuando no es el destinatario, para que el programa principal atienda o no el mensaje.

```
#int_rda
void serial_isr()
{
  if (kbhit())                                        //por cada dato disponible
  {
    buffer_datos[indice_buffer++] = getc();           //lo lee y añade al buffer
    If (paridad)                                      //si paridad esta activada
      buffer_paridad[indice_buffer] = bit_test(rcsta, 0);  //añade bit paridad
  }                                                   //en buffer_paridad
                        //si destino = config y octavo byte recibido (trama completa)
  If ((buffer_datos[0x00] == num_dispos) && (indice_buffer == 8))
    trama_ok = 1;                                     //la trama se da por buena
}
```

Dentro del programa principal se habilitara esta interrupción y todas las demás para estar preparadas para cualquier evento, aunque previamente se habrá vaciado el buffer que contendrá la trama.

```
enable_interrupts(int_rda);          //habilita interrupción RDA
enable_interrupts(global);           //habilita resto de interrupciones
vacia_buffer();                      //borra el buffer;
```

8.6. Programa de reconocimiento y tratamiento ModBus

Dentro del programa principal, y después de analizar los datos recibidos, contenidos en el *array buffer_datos* como una trama correcta, se pasa a interpretar la función requerida para manipular y procesar los datos.

Si esta activada la paridad, lo primero que se hace es verificar su correcto envío mediante un bucle que recorre el *array buffer_array* donde se guardan los bits de paridad y se comparan con los bytes correspondientes del *array buffer_datos* mediante el calculo empleado en la función cal_paridad().

```
while(trama_ok)
  {
    if (paridad)                                  //si paridad esta activada
      for (indice = 0; indice > 8; indice++)
      {
        par = cal_paridad(buffer_datos[indice]);  //se calcula la paridad
        if (par == buffer_paridad[indice])        //compara con la recibida
          error_paridad = 0;                      //sin errores = 0
        else
          error_paridad = 1;                      //con errores = 0
      }
    if ((comprobar_crc(8)) && (!error_paridad))   //trama correcta
    {
      switch(buffer_datos[1])                     //según función se realiza la
      {                                           //la secuencia programada
        case 0x04:                                //Función leer registros entradas
          leer_entra_analog();
          break;
        case 0x06:                                //Función escribir registros salida
          modif_sal_analog();
          break;
      }
      vacia_buffer();                             //borra el buffer
    }
    else
     trama_error = 1;
  }
}                                                 //cierre del programa principal
```

Una vez superada tanto la comprobación del CRC como la de paridad se pasa a analizar la posición 1 del buffer que contiene una de las dos funciones que se debe realizar, llamando a la función encargada de realizar esa tarea. Una vez realizadas las tareas se vacía el buffer y el programa queda a la espera de una nueva interrupción.

En el caso de que tanto la paridad como el CRC den error, el programa quedará en espera de que el maestro, pasado un tiempo sin respuesta, reenvíe la trama de nuevo.

8.7. Programa para la transmisión de datos serie

Uno de los objetivos principales de todo el proyecto es poder transmitir al dispositivo maestro los datos recogidos en los canales analógicos, por lo que una vez formateada la trama en el *array buffer_datos* se llama a esta función.

Como parámetro de la función se pasa el tamaño del buffer, pues este varía de los 13 bytes para la respuesta de la función 04h a los 8 bytes que contiene la respuesta a la función 06h de ModBus. Mediante un bucle se realiza el recorrido de este array, con el tamaño especificado, para extraer su contenido y transmitir byte a byte la secuencia de datos, mediante el comando *putc()*.

```
void trans_respuesta(int tama)
{
  int indice;
  int1 par;

  for (indice = 0; indice < tama; indice++)          //recorre todo el buffer
  {
    if (paridad)                                     //si existe paridad
    {
      par = cal_paridad(buffer_datos[indice]);       //calcula el valor
      if (par)                                       //y modifica el bit 0
        bit_set(txsta, 0);                           //del registro txsta
      else                                           //con dicho valor
        bit_clear(txsta, 0);                         //antes del envío del dato
    }
    putc(buffer_datos[indice]);                      //enviando data a dato
  }
}
```

Si el bit de paridad esta activado, bit 6 del registro TXSTA, se calcula mediante la función *cal_paridad* el valor de la paridad y con este valor devuelto se actualiza el bit 0, denominado TX9D, del registro *TXSTA*, encargado del estado de la transmisión y control, antes de realizar la transmisión. Al hacer esto dentro del bucle se asegura que cada uno de los bytes este acompañado del noveno bit de paridad correspondiente. La figura siguiente muestra los registros implicados en la transmisión serie.

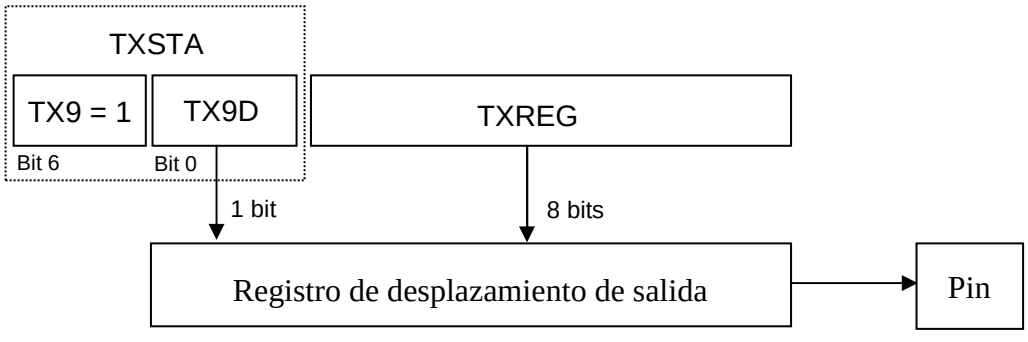


Figura 30: Registros implicados en tratar la paridad

8.8. Programa para el cálculo de la paridad

El uso del bit de paridad proporciona un método simple de detección de errores, basado en enviar junto a los datos un bit más para indicar si el número de bits con valor 1 es par o impar. Es un método que aunque detecta errores no proporciona técnicas para corregirlos. Además si durante la transmisión se produce un doble cambio de bits, la paridad no cambiará y se darán por buenos los datos corrompidos.

En un código de paridad impar el bit de paridad será 1 si el número total de 1 a transmitir es impar. Utilizando un código de paridad par, el bit de paridad será 0 si el número total de 1 es par. Cuando se active la paridad mediante el micro-interruptor colocado externamente, será este último el método empleado para calcular la paridad.

Esta función recibe como parámetro el byte cuya paridad se ha de calcular. Dentro de un bucle, a este byte, se le va desplazando un bit hacia la derecha de manera que se pueda comprobar el valor del último bit mediante una operación AND e incrementar la variable *num_bits* cada vez que detecta un 1.

Una vez terminado el bucle se divide por 2 y mediante el módulo de esta división se obtiene directamente el valor del bit de paridad, que será devuelto mediante el comando *return*.

```
int1 cal_paridad(int8 dato)
{
  int8 indice, num_bits, aux_dato;

  num_bits = 0;
  aux_dato = dato;
  for (indice = 0; indice < 8; indice++)        //recorre bit a bit el dato
  {
    if (aux_dato & 0x01)                        //comprueba si el bit 0 es un 1
      num_bits++;                               //en cuyo caso lo cuenta
    aux_dato = aux_dato >> 1;                   //rota a la derecha una posición
  }

  if (num_bits % 2 == 0)                        //si el número de unos es par
    return 0;                                   //se devuelve un 0
  else
    return 1;                                   //sino se devuelve un 1
}
```

Todos los dispositivos conectados a una red ModBus deben ser configurados con el mismo método de verificación de paridad.

Si no se ha especificado el uso del bit de paridad se transmitirá un bit de parada adicional para rellenar el carácter del mensaje.

8.9. Programa para comprobar y calcular el CRC

Los mensajes incluyen un campo de comprobación de error que está basado en un método Comprobación de Redundancia Cíclica (CRC). El campo CRC es de dos bytes, cuyo valor binario de 16 bits es calculado por el dispositivo emisor, que lo añade al mensaje. El dispositivo receptor calcula el CRC al recibir el mensaje y lo compara con el valor recibido en el campo CRC. Si ambos valores no coinciden, existirá error.

Para calcular el valor CRC para ModBus se inicializa un registro de 16 bits, con el valor de FFFFh. A continuación se inicia un proceso que toma los bytes sucesivos del mensaje, los opera con el contenido del registro y actualiza éste con el resultado obtenido. Sólo los 8 bits de datos son utilizados para generar el CRC. Los bits de inicio, parada y paridad no entran en el cálculo del CRC.

Para la generación del CRC, se efectúa una operación XOR a cada dato de 8 bits con el contenido del registro, a este resultado se le aplica un desplazamiento de un bit en la dirección del bit menos significativo (LSB), rellenando la posición del bit más significativo (MSB) con un cero.

El LSB es extraído y analizado, actuando de diferente manera según su valor:

- Si es un 1, se realiza un XOR entre el registro y el valor A001h determinado por el polinomio generador CRC16, $1 + x^2 + x^{15} + x^{16}$, que es el que se aplica para ModBus.

- Si es un 0, no se efectúa la operación XOR en el valor actual del registro y el proceso se repite con ocho desplazamientos más, como se ha descrito más arriba y así con cada uno de los bytes del mensaje.

Cuando todos los bytes del mensaje han sido procesados, el contenido final del registro es el valor del CRC.

En la implementación de este método, utilizado para encontrar errores producidos durante la transmisión, se han utilizado dos funciones, una para comprobar que el CRC recibido es correcto y otra para calcular el CRC de la trama a transmitir.

Aunque ambas funciones trabajan de la misma manera para calcular el CRC, el tratamiento al final de cada rutina es diferente, según sea para comprobar que el CRC recibido es correcto, ó para guardar el valor del CRC calculado en el *array buffer_datos* antes de su transmisión.

Programa de comprobación del CRC

```
int1 comprobar_crc(int n)
{
  unsigned int16 crc, indice_1, indice_2, acarreo, aux_crc;

  crc = 0xFFFF;                                   //inicialización del registro
  for (indice_1 = 0; indice_1 < n; indice_1++)    //recorrido del buffer_datos
  {
    crc = crc ^ buffer_datos;                     //registro crc XOR dato
    for (indice_2 = 0; indice_2 < 8; indice_2++)  //bucle para desplazar bytes
    {                                             //a la derecha
      aux_crc = crc;                              //copia del registro crc
      acarreo = aux_crc & 0x0001                  //se extrae el valor del LSB
      crc = crc >> 1;                             //desplaza 1 bit a la derecha
      if (acarreo == 1)                           //si LSB = 1
        crc = crc ^ 0xA001;                       //registro XOR polinomio
    }
  }
  if ((buffer_datos[n + 1] != (crc >> 8)) || (buffer_datos[n] != (crc & 0xFF)))
    return 1;                                     //Se compara CRC calculado
  else                                            //con el recibido
    return 0;
}
```

Programa para el cálculo del CRC

```
void calculo_crc(int n)
{
  int16 crc, aux_crc, acarreo;
  int8 indice_1, indice_2;

  crc = 0xFFFF;                                   //inicialización del registro
  for (indice_1 = 0; indice_1 < n; indice_1++)    //recorrido del buffer_datos
  {
    crc = crc ^ buffer_datos;                     //registro crc XOR dato
    for (indice_2 = 0; indice_2 < 8; indice_2++)  //bucle para desplazar bytes
    {                                             //a la derecha
      aux_crc = crc;                              //copia del registro crc
      acarreo = aux_crc & 0x0001;                 //se extrae el valor del LSB
      crc = crc >> 1;                             //desplaza 1 bit a la derecha
      if (acarreo == 1)                           //si LSB = 1
        crc = crc ^ 0xA001;                       //registro XOR polinomio
    }
  }
  buffer_datos[n + 1] = crc >> 8;                 //Se guarda  CRC calculado
  buffer_datos[n] = crc & 0xFF;                   //al final del array
}
```

8.10. Simulación de las entradas y salidas analógicas

Mediante el programa Proteus-Isis se han simulado las entradas y salidas digitales, así como la presentación en un *display* LCD de la configuración relativa al número de dispositivo y tipo de velocidad escogida. Se ha introducido dos señales a las entradas AN0 y AN1, cuyo voltaje es variable mediante sus respectivos potenciómetros, como se observa en este esquema:

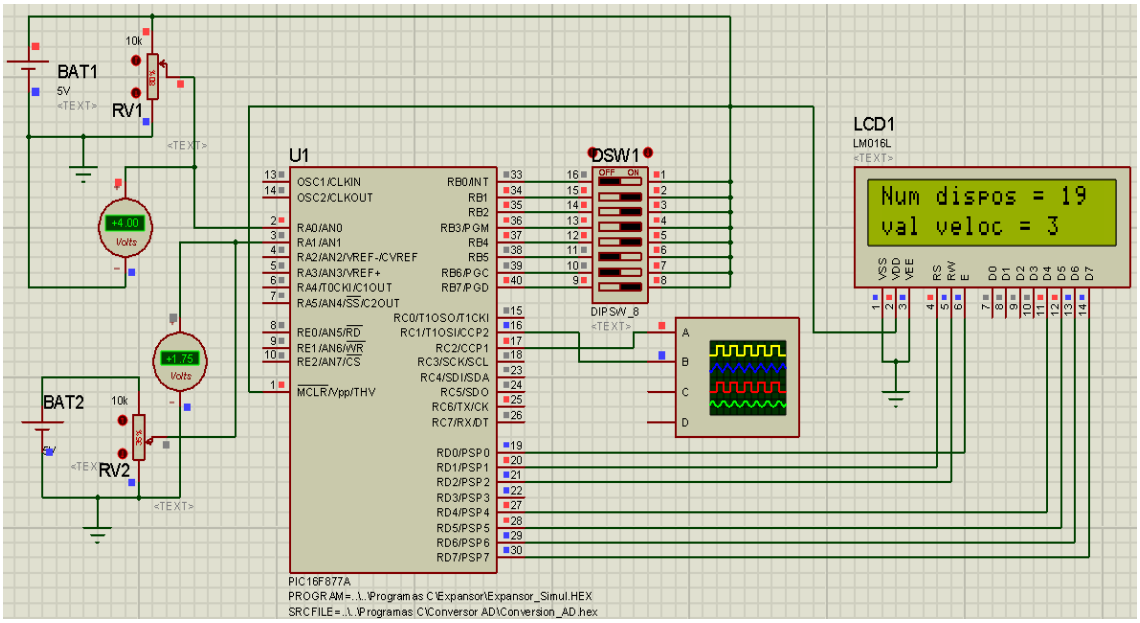
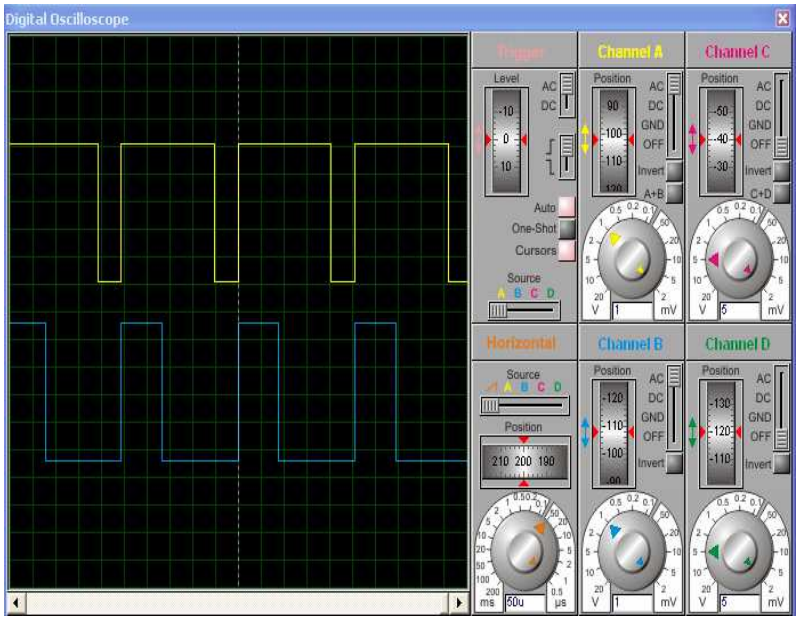


Figura 31: Esquema para la simulación ADC - PWM

Las dos entradas analógicas convertidas en tramas se han reintroducido a las salidas PWM a través del *array buffer_datos* para comprobar como variando el valor de los potenciómetros se obtiene una variación del ancho de los pulsos.



En la entrada AN0 se entregan 4V, lo que supone el 80% de ancho de pulso a la salida PWM1, como se puede comprobar en la señal amarilla.

En la entrada AN1 la diferencia de potencial es de 1.75V, que se corresponde con un 35% de ancho de pulso en la salida PWM2, como se observa en la señal azul del osciloscopio.

Figura 32: Salidas en osciloscopio de las señales PWM simuladas

9. Diseño de la placa PCB

A partir de los circuitos electrónicos del extensor de E/S, estudiados en el punto 7, se realizará el diseño de la placa PCB o layout. A continuación se muestra el circuito electrónico completo, donde las conexiones desde la fuente de alimentación no se muestran, dándose por implícitas, con el fin de facilitar la comprensión del circuito.

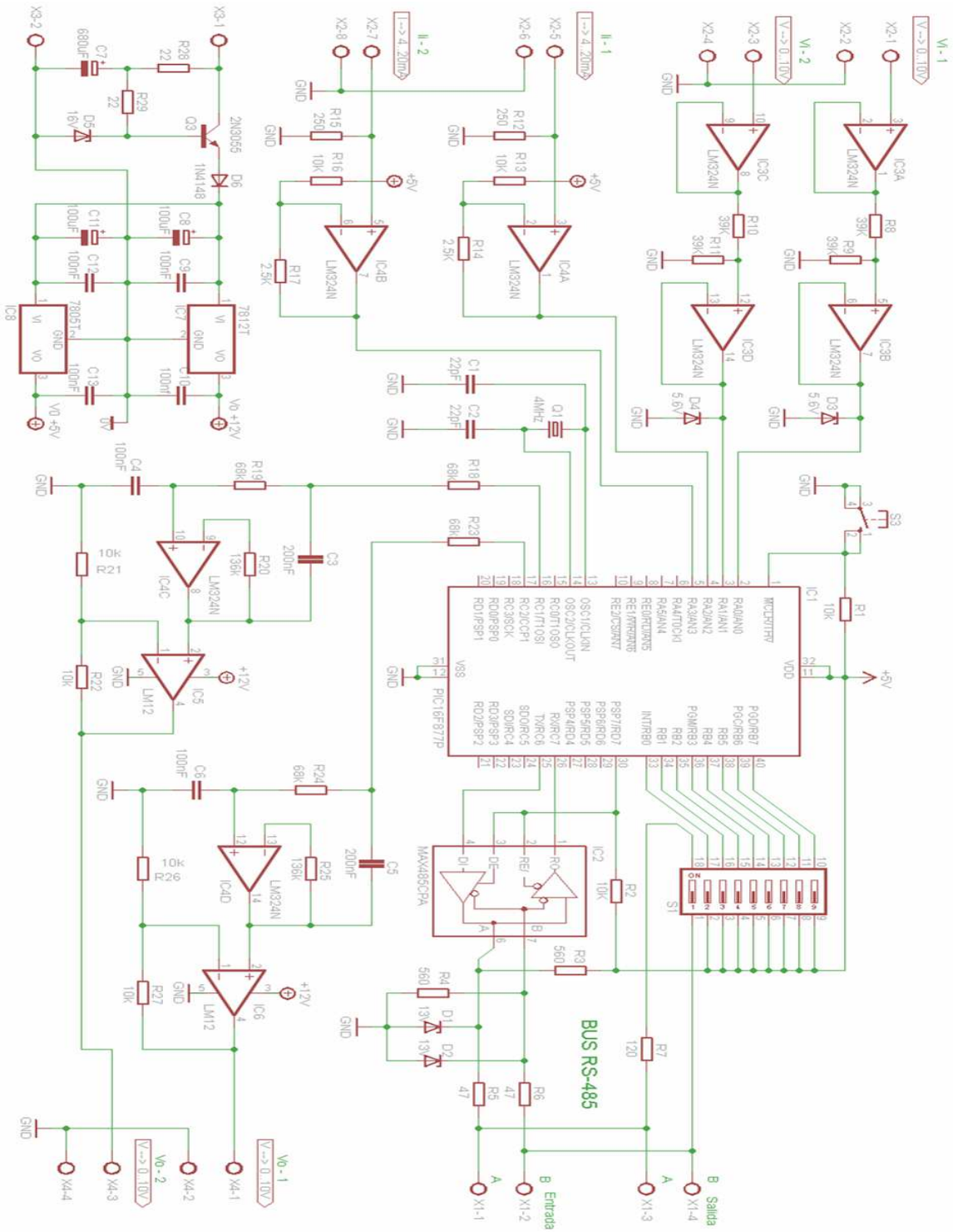


Figura 33: Esquema electrónico completo del extensor

9.1. Programa para el diseño de la PCB

Para facilitar el diseño de la PCB se recurrirá al programa Eagle 6.1.0., el cual consta tanto de un editor de circuitos como otro para distribuir los componentes, disponiendo de una amplia librería de componentes electrónicos y variedad de encapsulados para adaptarse a cualquier requisito y necesidad.

Una vez realizado el diseño completo del esquema, como el de la figura 33 se procede al diseño de la placa PCB, donde junto a los terminales de los componentes se muestran las líneas que comparten con otros elementos, con el fin de tener una visión global y aproximarlos lo máximo posible para reducir la longitud de las pistas.

Para optimizar la distribución de los componentes en la placa PCB se tendrán en cuenta los siguientes requisitos:

o Colocar los componentes emisores de calor como son los reguladores de tensión, el transistor de potencia y los O.P. de potencia en un lado de la placa para facilitar la colocación de un disipador de calor con pocas modificaciones.

o Distanciar lo máximo posible los circuitos sensibles como son el µC, el transductor y los O.P. de baja potencia de las fuentes de calor anteriores para evitar distorsiones por temperatura.

o Acercar lo máximo posible el cristal de cuarzo al µC para no sufrir atenuaciones por distancia, en la señal de sincronismo.

o Colocar a un lado de la placa las entradas tanto de las señales analógicas como de alimentación y en lado contrario las salidas de las señales y la conexión a la red RS-485.

o Ubicar los micro-interruptores en un lado, para evitar manipulaciones en el interior de la placa y aislar al usuario de contactos directos con los componentes bajo tensión.

o Posicionar el pulsador de *Reset* en una esquina de la placa para facilitar su rápida localización en el caso necesario.

o Distribuir el resto de componentes, resistencias, condensadores y diodos de manera uniforme, respetando una separación mínima para disipar el calor generado. Las líneas de alimentación del dispositivo y las salidas analógicas se realizan con un ancho mayor de pista para permitir una corriente más elevada.

Realizada la distribución según las especificaciones anteriores se pasa al diseño de las pistas que servirán para la serigrafía de la placa. Esto se puede ejecutar tanto de manera manual, pista a pista, o de modo automático. En el desarrollo de presente proyecto se ha realizado de forma automática, pues debido a la complejidad del esquema se requiere el uso de una placa de montaje de doble cara, que hace aún más complicado el diseño de las pistas.

9.2. Distribución de componentes en la PCB

En la figura siguiente se muestra la distribución de componentes realizada con los requisitos propuestos y mostrando sus referencias en una placa de forma rectangular con un largo de 15.8 cm y un ancho de 9 cm.

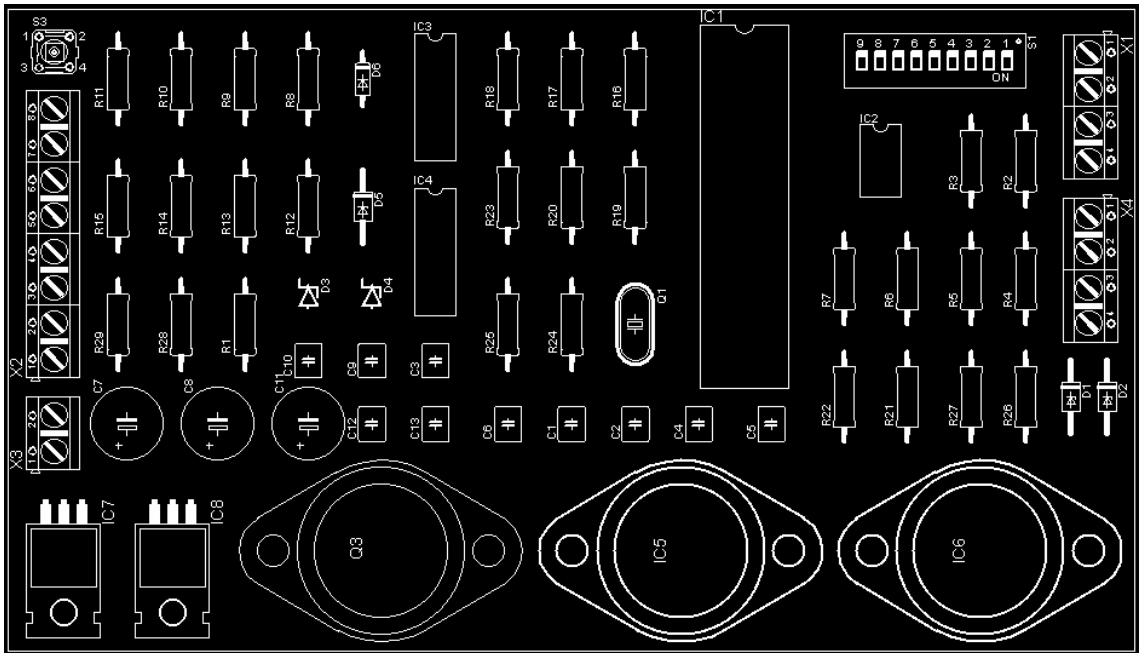


Figura 34: Distribución de componentes en la PCB

A partir de esta distribución se realiza el trazado automático de las pistas mediante la función *autoruoter* y su configuración, lo que permite obtener el siguiente resultado donde se observa el layout a doble cara, así como las referencias de los componentes y sus valores.

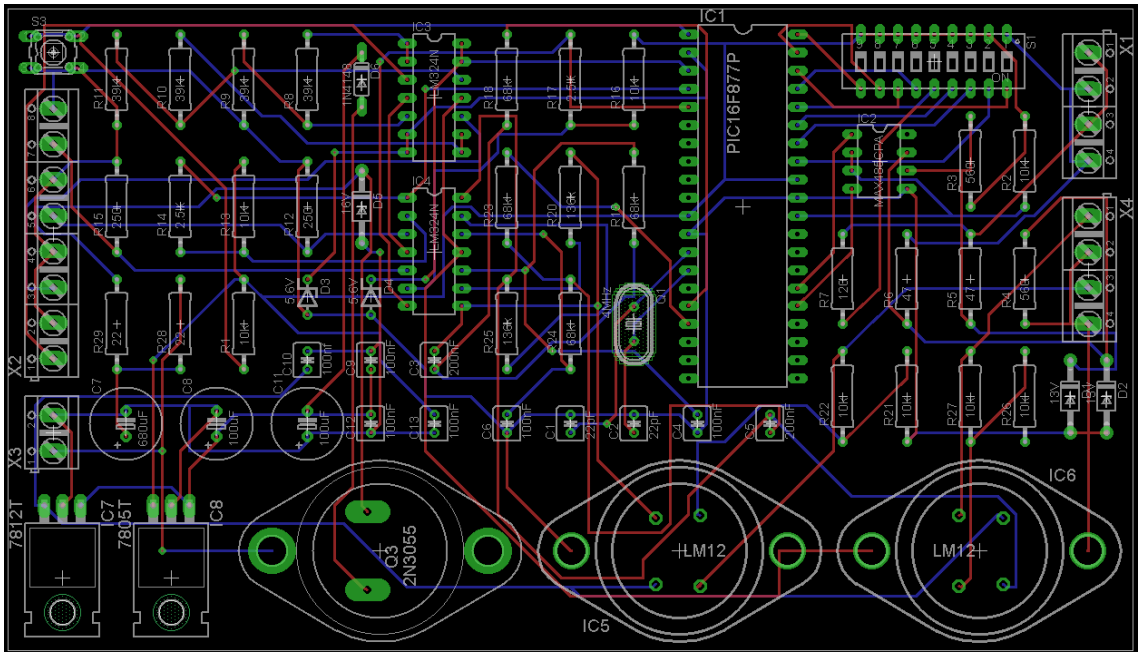


Figura 35: Placa PCB completa por las dos caras

9.3. Diseño de las pistas o layout de la PCB

A través de la función de inhibición de capas se pueden aislar las pistas. Además mediante las funciones *Poligon* y *Ratsnest* y un aislamiento entre pistas de 0.016 pulgadas se cubren los espacios vacíos con el plano de masa con el objetivo de reducir la impedancia del nodo GND y asegurar una tensión constante. Adicionalmente se representan las referencias de los componentes.

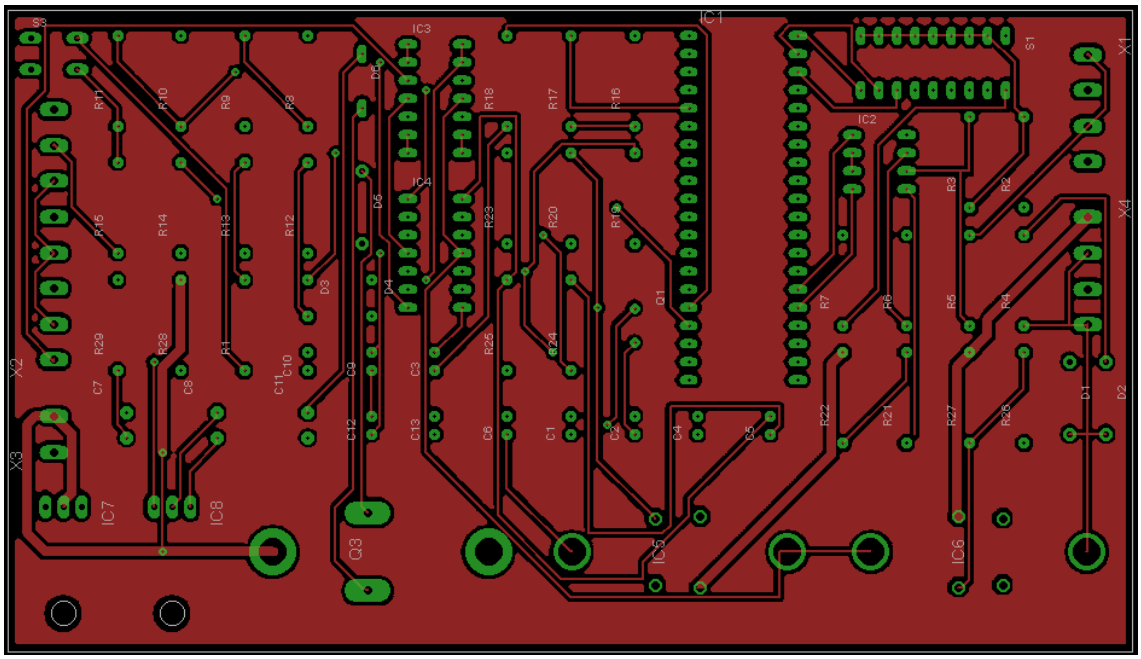


Figura 36: Cara de los componentes en la PCB

Aplicando la misma técnica, pero en la cara opuesta, se obtiene el siguiente layout con las pistas en azul las pistas y en verde los orificios.

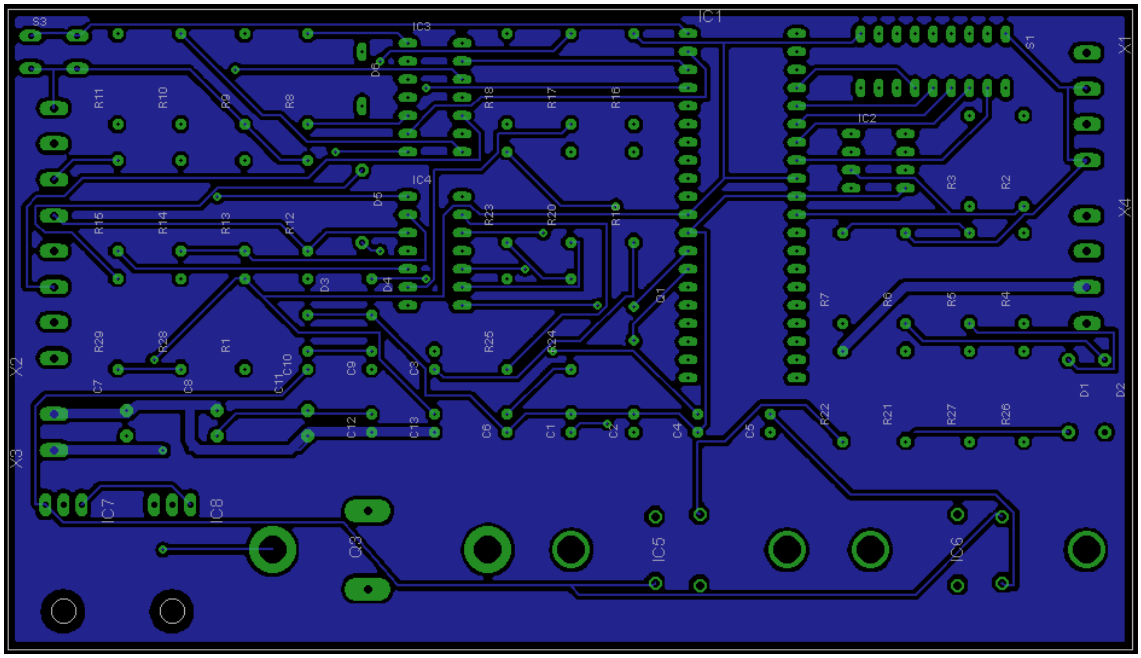


Figura 37: Cara contraria a los componentes en la PCB

10. Valoración económica

Para hacer una valoración del coste de realizar físicamente el proyecto se muestra el listado de componentes junto con su referencia, acompañado de su valor económico en euros.

Referencia	Valor	Precio	Referencia	Valor	Precio
C1	22 pF	0.31	**C2**	22 pF	0.31
C3	200 nF	0.384	**C4**	100 nF	0.43
C5	200 nF	0.384	**C6**	100 nF	0.43
C7	680 uF	8.17	**C8**	100 uF	0.43
C9	100 nF	0.43	**C10**	100 nF	0.43
C11	100 uF	2.58	**C12**	100 nF	0.43
C13	100 nF	2.58	**D1**	Zener 13V	0.314
D2	Zener 13V	0.314	**D3**	Zener 5.6V	0.023
D4	Zener 5.6V	0.023	**D5**	Zener 16V	0.314
D6	1N4148	0.156	**IC1**	16F877A	4.62
IC2	MAX485CPA	4.15	**IC3**	LM324N	1.08
IC4	LM324N	1.08	**IC5**	LM12	30
IC6	LM12	30	**IC7**	7812T	1.28
IC8	7805T	1.10	**Q1**	4MHz	1.35
Q3	2N3055	1.55	**R1**	10K Ω	0.015
R2	10K Ω	0.015	**R3**	560 Ω	0.015
R4	560 Ω	0.015	**R5**	47 Ω	0.015
R6	47 Ω	0.015	**R7**	120 Ω	0.015
R8	39K Ω	0.063	**R9**	39K Ω	0.063
R10	39K Ω	0.063	**R11**	39K Ω	0.063
R12	250 Ω	0.06	**R13**	10K Ω	0.015
R14	2.5K Ω	0.034	**R15**	250 Ω	0.06
R16	10K Ω	0.015	**R17**	2.5K Ω	0.015
R18	68K Ω	0.088	**R19**	68K Ω	0.088
R20	136K Ω	0.024	**R21**	10K Ω	0.015
R22	10K Ω	0.015	**R23**	68K Ω	0.088
R24	68K Ω	0.088	**R25**	136K Ω	0.024
R26	10K Ω	0.015	**R27**	10K Ω	0.015
R28	22 Ω	0.046	**R29**	22 Ω	0.046
S1	DIP09YL	1.57	**S2**	10-XX	1.35
X1	AK500/4	1.188	**X2**	AK500/8	1.60
X3	AK500/2	0.67	**X4**	AK500/4	1.188
PCB cobre	2 caras	8.78	**Zócalo eyec.**	40 pines	11.82
Total Euros					**123.82**

Tabla 13: Listado de componentes y su coste

Se puede observar que el costo no es muy elevado, se trata de componentes muy comunes y fáciles de encontrar, siendo los OP de potencia los que encarecen más la valoración.

11. Conclusiones

La diversidad de áreas y temáticas tratadas ha sido muy variada y amplia, lo que ha obligado a utilizar varias aplicaciones para el diseño, codificación y simulación durante el desarrollo del presente proyecto. Para seleccionar los programas a utilizar se han testado otras aplicaciones con similares características. Esto ha proporcionado una gran experiencia práctica, que servirá en futuros proyectos para elegir la aplicación más idónea y adaptada al objetivo planteado.

De igual modo, se han estudiado varias alternativas para determinar la solución que mejore y a la vez simplifique el circuito en su conjunto, lo que ha llevado a ampliar los conocimientos previos. Por otra parte, mediante la simulación de gran parte de los circuitos y algoritmos implementados se ha conseguido afianzar nuevas habilidades.

En la comprobación de los circuitos analógicos ha resultado muy enriquecedor profundizar en la interpretación de las graficas y datos obtenidos mediante la simulación, y así comprobar la coincidencia con los datos calculados teóricamente, tanto para la adaptación de las entradas analógicas como para el tratamiento y filtrado de las señales PWM.

Idéntica sensación se ha obtenido codificando el programa de control donde, utilizando tramas predefinidas, se han comprobado y simulado gran parte de los algoritmos codificados, la lectura de las entradas analógicas, la generación PWM, la interpretación de las funciones ModBus y la preparación de las respectivas respuestas.

El diseño del *layout* de la placa PCB también ha servido para conocer un área importante como es el soporte físico y que normalmente se pasa por alto en el diseño final.

Para concluir, se proponen dos líneas a desarrollar en el futuro:

1. Una distribución de la alimentación en tensión alterna, más eficaz para ser repartida. Según mi experiencia, considero que la alimentación en tensión continua, formulada en el enunciado, provoca una caída de tensión en su distribución, y con ello una perdida de eficacia. Esta opción disminuye las pérdidas de tensión por un lado y ayuda a que las fuentes con reguladores integrados, elegidas para este proyecto, aumenten su eficiencia. Añadir un transformador con salida de 18 V y una potencia de 40 W reduce el margen de tensión de alimentación, con la consiguiente reducción en la emisión de calor.

2. Incorporar, como mínimo, dos relés que sean controlados desde las salidas del µC para disponer de contactos libres de potencial con los que poder activar o desactivar la alimentación a otros dispositivos. Al mismo tiempo, disponer de varias entradas binarias adaptadas a dos rangos distintos de alimentación, señales TTL y 24 VAC, para recibir valores provenientes de dispositivos externos. Estas entradas y salidas digitales se controlarían mediante ModBus.

12. Bibliografía y Software

12.1. Bibliografía

La bibliografía utilizada ha estado formada por la combinación de libros, páginas Web de diferentes fabricantes y proveedores para obtener los *data Sheet* de los componentes utilizados, diversos video-manuales en *youtube* además de consultar en foros de electrónica y de programación:

Información sobre el protocolo ModBus RTU:

- *The MODBUS organization*. Web [fecha de consulta: 3 de noviembre del 2012]. <www.MODBUS.org/tech.php>
- Axon Group. Web [Fecha de consulta: 3 de noviembre del 2012] <www.axongroup.com.co/pages/modbus>
- Autómatas programables, editorial Marcombo, autores J. Balcells, J.L.Rom.

Elección del microcontrolador:

- Microchip. Web [Fecha de consulta: 7 de noviembre del 2012] <www.microchip.com>
- MikroElektronika. Web [Fecha de consulta: 6 de noviembre del 2012] <www.mikroe.com/products/view/285/book-pic-microcontrollers-programming-in-c/>

Elección Transductor:

- Maxim integrated. Web [Fecha de consulta: 5 de noviembre del 2012] <www.maximintegrated.com/products/interface/transceivers/rs-485.cfm>
- Analog Devices. Web [Fecha de consulta: 5 de noviembre del 2012] <www.analog.com/en/interface-isolation/rs-485-rs-422/products/index.html>

Diseño de los circuitos de tratamiento de datos analógicos:

- Microelectrónica, Editorial Hispano Europea. Jacob Millman, Harbin Graven
- Amplif. Operac. y circ. Integrados lineales. Robert F. Coughlin
- Consulta de foros de electrónica

Diseño de la fuente de alimentación:

- ParaElectrónicos. Web. [Fecha de consulta: 10 de noviembre del 2012] <http://www.paraelectronicos.com.ar/p/index.php?id=elec_fuentes>
- Consulta de foros de electrónica

Programación en CCS y simulación con Proteus – Isis

- Compil. C CCS y simul. Proteus para PIC, E. Garcia Breijo, Marcombo

Diseño de circuitos electrónicos y PCB con Eagle

- Coaguila, D. (2012, 23 de junio). "Eagle 6.2 Tutorial pasa a paso – Cap. I". Youtube. [Fecha de consulta: 10 de noviembre del 2012] <http://www.youtube.com/watch?v=1oRQShk9S2U>

Valoración del coste del montaje de la placa y componentes del proyecto

- RS Components. Web. [Fecha de consulta: 28 de diciembre del 2012] <http://es.rs-online.com/web/>

12.2. Software empleado

El software empleado ha sido el siguiente, alguno de los cuales han precisado de tiempo y pericia para poder aplicarlo a las necesidades según evolucionaba el desarrollo del proyecto:

- Windows XP: Sistema operativo
- MS Word: Edición de textos
- MS Project: Diseño diagramas de *Gantt*
- Schematics: Diseño y simulación de circuitos analógicos
- CCS C: Programación en lenguaje C del microcontrolador
- Eagle: Diseño de circuitos y desarrollo del PCB
- Proteus – ISIS: Simulación del microcontrolador a partir de CCS C