UNIVERSITAT OBERTA DE CATALUNYA

Ingeniería Técnica de Telecomunicación

(Especialidad en Telemática)

Diseño de un extensor de entradas y salidas analógicas

por ModBus RTU sobre RS-485

Alumno: David Olmedo Díaz

Dirigido por: Carlos Alberto Pacheco Navas

CURSO 2012-13 (Febrero)

AGRADECIMIENTOS

A mi sobrina, Daniela, que alegra mi vida demostrándome que las ganas de aprender cosas nuevas no deben perderse jamás, *independientemente de la edad que tengas*.

A mi mujer, por compartir los buenos y malos momentos que nos han tocado vivir durante estos últimos 10 años juntos.

A mis hermanas, por su apoyo en los momentos difíciles y por compartir conmigo los felices.

A mis amigos, sin los cuales no podría haber encontrado la motivación suficiente ni las ganas de seguir en aquellos momentos en los que piensas en arrojar la toalla.

A mis padres, por la paciencia infinita, la comprensión y el apoyo que me han ofrecido durante estos largos y complicados años que he dedicado a mi formación universitaria. Por inculcarme los valores y pilares en los que se sustenta mi vida: educación, respeto, compromiso, perseverancia, etc.

A Daniel Prado, Ingeniero de Telecomunicaciones y excelente persona; y a su familia por ofrecerme y compartir conmigo tantas cosas durante mi estancia en Málaga, sin pedir nunca nada a cambio.

A todo aquel que me ha ayudado y me ha servido de apoyo, motivación o inspiración, especialmente a la persona que hace que mi vida brille cada día.

A mi tutor del proyecto, por sus orientaciones y correcciones.

PRESENTACIÓN DEL AUTOR DEL TRABAJO FIN DE CARRERA

Este TFC ha sido realizado por D. **David Olmedo Díaz**.

Tras su paso por la Universidad de Málaga, donde comenzó la titulación de Ingeniería Técnica de Telecomunicación en la especialidad de Sistemas de Telecomunicación en la Escuela Técnica de Ingenieros de Telecomunicación, cursa en el momento de la redacción de este documento la titulación de Ingeniería Técnica de Telecomunicación en la especialidad de Sistemas Telemáticos en la Universitat Oberta de Catalunya (UOC).

En la actualidad compagina los estudios universitarios con el desempeño laboral como Director de Siniestros y Director de Calidad en una compañía de Asistencia Sanitaria que opera a nivel nacional.

Además junto con dos socios ha emprendido, a comienzos del año 2012, un nuevo e ilusionante reto empresarial en el área del marketing hospitalario (*Digital Signage*, nuevas tecnologías, etc.)

El siguiente objetivo académico, tras la consecución del título oficial de Ingeniero Técnico de Telecomunicaciones, es la realización de un MBA a través de alguna de las escuelas de negocios que ofrecen esta titulación orientada a mejorar los conocimientos adquiridos y aplicarlos directamente al mundo empresarial.

ÍNDICE

Índice de contenidos

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1. Presentación del TFC

Este TFC se basa en el diseño de un adaptador de interfaz para leer y/o escribir señales analógicas mediante ModBus RTU sobre RS-485 (***extensor de entradas y salidas analógicas por ModBus RTU sobre RS-485***)

A nivel de bloques debemos diseñar un sistema como el que se muestra en la siguiente figura:

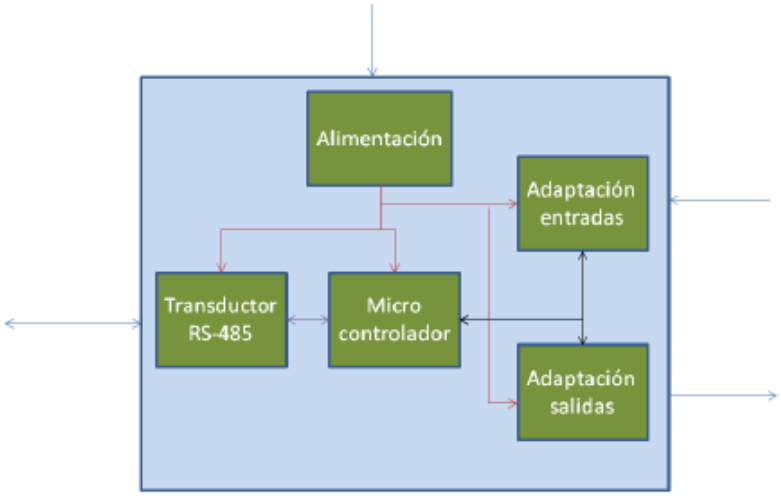


Figura 1. Diagrama de bloques

Descripción de bloques:

- **Transductor RS-485** : Normalmente los microcontroladores no incorporan salida RS-485 y por lo tanto hace falta algún adaptador que convierta la señal del microcontrolador a RS-485.
- **Microcontrolador**: Componente programable en el cual se debe implementar el control del sistema.
- **Adaptación entradas**: Conjunto de circuitos que adaptan la señal a medir para que el microcontrolador pueda adquirir un valor correcto y adecuado.
- **Adaptación salidas**: Conjunto de circuitos que adaptan la señal que se obtiene del microcontrolador para generar la salida deseada.
- **Alimentación**: Conjunto de elementos que adaptan la tensión de entrada a las tensiones que necesitan cada uno de los bloques.

Especificaciones del adaptador:

- **Tensión de alimentación**: Podrá alimentarse con cualquier tensión continua que esté comprendida entre 12 y 48 Voltios.
- **Interfaz RS-485**:
  - Implementa el protocolo ModBus.
  - Velocidad de transmisión, paridad e identificador de dirección de dispositivo se tiene que poder configurar también por escritura a registros ModBus.
  - Se debe definir el mapa de memoria para leer/escribir las entradas/salidas.
  - Se debe definir el mapa de memoria para modificar parámetros del ModBus.

- Se valorará que sea aislada eléctricamente.

- **Entradas analógicas**:
  - Tiene que tener dos entradas analógicas que permitan medir tensiones entre 0 y 10 V, con un mínimo de 10 bits de precisión.
  - Tiene que tener dos entradas analógicas para medir bucle de corriente 4-20 mA pasivo.
- **Salidas analógicas**:
  - Tiene que tener dos salidas analógicas que permitan generar tensiones entre 0 y 10 V, con un mínimo de 10 bits de precisión y dando un máximo de 200 mA de corriente cada una.

1.2. Objetivos

1.2.1. Objetivos de la asignatura

El objetivo principal de la asignatura es la realización de un trabajo teórico-práctico de síntesis en el que aplicar los conocimientos adquiridos al cursar asignaturas relacionadas con las aplicaciones electromagnéticas y electrónicas de la Ingeniería Técnica de Telecomunicación y resto de materias de carácter más genérico. Entre otros, los objetivos generales de esta asignatura son:

- Identificar los objetivos generales asociados a un proyecto en el ámbito de las TIC así como los objetivos técnicos específicos que tendrán que ser planteados para la resolución de una problemática concreta.
- Integrar diferentes disciplinas asociadas a una Ingeniería de Telecomunicación de forma óptima y nueva para la resolución de unos objetivos generales y específicos planteados en un proyecto.
- Analizar los resultados y pruebas realizadas sobre el proyecto para emitir unas conclusiones que recojan tanto la aportación concreta del desarrollo realizado como las líneas futuras que surgen del mismo.
- Conocer la estructura y el proceso de diseño de un sistema de adaptación de interfaz y aplicarlo al caso de la expansión de entradas y salidas analógicas.
- Diseñar circuitos analógicos que realicen funciones de amplificación, comparación y generación de señal.
- Diseñar fuentes de alimentación óptimas con un ancho margen de tensión de entrada.
- Programar procesadores digitales de la señal, en su lenguaje de programación específico, para la manipulación y procesamiento de las señales.

En definitiva, esta asignatura trata de poner en práctica y preparar al alumno para enfrentarse al reto que muchos de los profesionales, tanto en el ámbito de la ingeniería como en otras disciplinas, tienen que enfrentar a la hora de que un cliente solicite el estudio, prototipado o simple generación de un presupuesto económico que implique un estudio, planificación, estimación de tiempos, recursos humanos y/o técnicos, costes, etc. y que necesariamente conlleve la elaboración de un proyecto. Podemos resumir las fases principales y genéricas que, normalmente, se llevan a cabo durante el estudio, planificación y desarrollo de un proyecto singular:

- Análisis de un problema práctico o necesidad del cliente para transformarlo en un proyecto sólido tangible que pueda implementar una posible solución al problema. Como ingenieros, lo ideal sería proponer la mejor solución con los costes más ajustados posibles.

- Planificación y estructuración de tareas, hitos, etc. por las que debe pasar el proyecto hasta conseguir su completa implementación.
- Elaboración y redacción de la memoria técnica final y presentación (exposición al cliente) del mismo.

1.2.2. Objetivos del TFC

Los objetivos del TFC propuesto relacionan directamente la electrónica analógica/digital, con la programación de un microprocesador, para poder lograr la intercomunicación de equipos diversos mediante la interfaz RS-485 y usando para ello el protocolo de comunicación ModBus RTU. Es decir:

- Diseñar un sistema extensor de entradas y salidas analógicas por ModBus RTU sobre RS-485.
  - Conocer y estudiar el protocolo de comunicaciones ModBus RTU.
  - Conocer y estudiar el estándar de comunicaciones por RS-485.
- Diseñar la fuente de alimentación.
- Diseñar los circuitos de adaptación de las entradas y salidas a niveles del microcontrolador.
- Diseñar el programa de control que se cargará en el microcontrolador.
- Diseñar el layout del circuito resultante, como primer paso para poder proceder a su implementación física.

1.3. Calendario de trabajo

Teniendo en cuenta la evaluación continua de las últimas 2 asignaturas que estoy cursando junto con el TFC, el trabajo profesional y la conciliación de mi vida personal, tengo previsto dedicar al proyecto el tiempo siguiente:

- Lunes a viernes: Un total de 3 horas.
- Sábados: 4 horas
- Domingos: 4 horas
- Festivos: 4 horas
- Día de vacaciones: 4 horas

Figura 2. Calendario TFC

El proyecto se ha llevado a cabo a lo largo 16 semanas y ha implicado un total de 229 horas de trabajo real. Esto ha supuesto una desviación del 12,25% con respecto a las 204 horas estimadas en el "plan de trabajo".

1.4. Planificación de tareas

A continuación podemos ver, de manera detallada, las tareas y subtareas que se han programado y llevado a cabo para la realización de este TFC:

Id.	Tarea	Horas	Fecha Inicio	Fecha Fin
1	**DEFINICIÓN DEL TFC**			
1.1	Recopilación y primera lectura de la documentación disponible en el aula y en el tablón	3	19/09/12	21/09/12
1.2	Encuentro presencial por videoconferencia múltiple con el consultor y resto de alumnos del aula	2	22/09/12	22/09/12
2	**PLAN DE TRABAJO**			
2.1	**Documento Plan de Trabajo**			
2.1.1	Lectura de documentación. Estructuración y esquematización de la documentación leída (otras memorias, otros planes de proyecto, documentos de estilo, documentos aportados por el consultor)	2	24/09/12	27/09/12
2.1.2	Búsqueda de bibliografía útil para el proyecto a medio plazo: sitios web, libros, documentos	2	24/09/12	27/09/12
2.1.3	Redacción del índice del borrador.	1	24/09/12	27/09/12
2.1.4	Redacción del borrador. Descripción, objetivos, actividades, tareas, hitos, estructura del PFC, posibles incidencias y soluciones	2	28/09/12	29/09/12
2.1.5	Planificación de tareas. Establecimiento los hitos del proyecto así como los contenidos de PEC2 y PEC3. Establecimiento del calendario de trabajo, teniendo en cuenta los posibles riesgos, temporización de las tareas en proporción al esfuerzo que se calcula que requerirán	3	28/09/12	29/09/12
2.1.6	Finalización del borrador y entrega al consultor	1,5	28/09/12	29/09/12
2.1.7	Corrección de borrador con las anotaciones y sugerencias del consultor	1	29/09/12	29/09/12
2.1.8	Finalización y completado del Plan de Trabajo	10	29/09/12	02/10/12
2.1.9	Entrega del Plan de Proyecto. Éste es el Plan de proyecto definitivo, que coincide con la PEC1	0,5	02/10/12	02/10/12
3	**PEC2**			
3.1	**MODBUS**			
3.1.1	Búsqueda y recopilación de información	4	02/10/12	05/10/12
3.1.2	Estudio del protocolo MODBUS	3	06/10/12	06/10/12
3.1.3	Redacción de la documentación	3	07/10/12	07/10/12
3.2	**RS-485**			
3.2.1	Búsqueda y recopilación de información	1	06/10/12	06/10/12
3.2.2	Estudio del sistema de transmisión RS-485	1	07/10/12	07/10/12
3.2.3	Redacción de la documentación	3	08/10/12	11/10/12
3.3	**Adaptación E/S**			
3.3.1	Búsqueda y recopilación de información	3	12/10/12	12/10/12
3.3.2	Estudio de los requisitos e implementación de la solución	4	13/10/12	13/10/12
3.3.3	Simulación de circuitos	6	14/10/12	15/10/12
3.3.4	Redacción de la documentación	2	15/10/12	18/10/12

3.4	**Transductor RS-485**			
3.4.1	Búsqueda y recopilación de información	1	19/10/12	19/10/12
3.4.2	Estudio de los requisitos e implementación de la solución	2	20/10/12	20/10/12
3.4.4	Redacción de la documentación	2	20/10/12	20/10/12
3.5	**Procesador de control**			
3.5.1	Búsqueda y recopilación de información	3	22/10/12	25/10/12
3.5.2	Estudio de los requisitos e implementación de la solución	6	26/10/12	27/10/12
3.5.3	Redacción de la documentación	2	27/10/12	27/10/12
3.6	**Documento PEC2**			
3.6.1	Redacción del Borrador	8	29/10/12	04/11/12
3.6.2	Envío del borrador al consultor	0,5	04/11/12	04/11/12
3.6.3	Corrección de borrador con las anotaciones y sugerencias del consultor	4	05/11/12	08/11/12
3.6.4	Incorporación de apartados no incluidos en el borrador	4	09/11/12	09/11/12
3.6.5	Simulación de circuitos	4	10/11/12	10/11/12
3.6.6	Finalización y completado de la PEC2	4	11/11/12	11/11/12
3.6.7	Entrega de la PEC2	0,5	12/11/12	12/11/12
4	**PEC3**			
4.1	**MODBUS**			
4.1.1	Mapa de Memoria	2	12/11/12	12/11/12
4.1.2	Redacción de la documentación	0,5	12/11/12	12/11/12
4.2	**Fuente de Alimentación**			
4.2.1	Búsqueda y recopilación de información	3	13/11/12	14/11/12
4.2.2	Estudio de los requisitos e implementación de la solución	10	15/11/12	02/12/12
4.2.3	Simulación del circuito	2	02/12/12	02/12/12
4.2.4	Redacción de la documentación	2	03/12/12	03/12/12
4.3	**Procesador de control**			
4.3.1	Búsqueda y recopilación de información	3	19/11/12	22/11/12
4.3.2	Estudio de los requisitos e implementación de la solución	3	23/11/12	23/11/12
4.3.3	Programación del código de control del microprocesador	18,5	24/11/12	16/12/12
4.3.4	Simulación y depuración	3	06/12/12	16/12/12
4.3.5	Redacción de la documentación	3	16/12/12	16/12/12
4.4	**Circuito completo**			
4.4.1	Simulación del circuito completo	6	08/12/12	09/12/12
4.4.2	Mejoras y optimización	2	08/12/12	09/12/12
4.4.2	Redacción de la documentación	2	12/12/12	12/12/12
4.5	**Layout PCB**			
4.5.1	Diseño del Layout PCB	6	14/12/12	15/12/12
4.5.2	Redacción de la documentación	2	15/12/12	16/12/12
4.6	**Documento PEC3**			
4.6.1	Redacción de la Borrador	4	09/12/12	11/12/12
4.6.2	Enviar el borrador al consultor	0,5	11/12/12	11/12/12
4.6.3	Corrección de borrador con las anotaciones y sugerencias del consultor	8	12/12/12	16/12/12
4.6.4	Finalización y completado de la PEC3	4	15/12/12	16/12/12
4.6.5	Entrega de la PEC3	0,5	16/12/12	16/12/12

5	MEMORIA TFC			
5.1	**Documento Memoria Técnica TFC**			
5.1.1	Recopilación de documentación generada	1	17/12/12	17/12/12
5.1.2	Integración de la documentación en la Memoria Técnica	3	18/12/12	20/12/12
5.1.3	Redacción de la conclusiones	2	21/12/12	21/12/02
5.1.4	Redacción del Borrador	8	22/12/12	23/12/12
5.1.5	Envío del borrador al consultor	0,5	23/12/12	23/12/12
5.1.6	Corrección de borrador con las anotaciones y sugerencias del consultor	3	24/12/12	28/12/12
5.1.7	Finalización y completado de la Memoria Técnica	6	29/12/12	30/12/12
5.1.8	Revisión del documento definitivo (ortografía, gramática, sintaxis, estilo, etc.)	2	30/12/12	31/12/12
5.1.9	Entrega de la Memoria Técnica	1	07/01/13	07/01/13
5.2	**Vídeo Presentación**			
5.2.1	Confección de la presentación en Power Point	3,5	01/01/13	04/01/13
5.2.2	Grabación de la locución de la exposición	4	05/01/13	05/01/13
5.2.3	Maquetado, edición y montaje del vídeo de presentación	4	05/01/13	06/01/13
5.2.4	Envío del vídeo a la plataforma Present@	0,5	07/01/13	07/01/13
5.3	**Ficheros**			
5.3.1	Recopilación de archivos generados	1	05/01/13	05/01/13
5.3.2	Compresión de los archivos	0,25	07/01/13	07/01/13
5.3.3	Entrega de los archivos comprimidos	0,25	07/01/13	07/01/13
5.4	**Debate Virtual**			
5.4.1	Recopilación de las cuestiones y dudas del tribunal	1	23/01/13	25/01/13
5.4.2	Contestar a las cuestiones formuladas	4,5	24/01/13	26/01/13
5.4.3	Enviar documento con las respuestas	0,5	24/01/13	26/01/13
	TOTAL HORAS TFC	**229**	**12/11/12**	**16/11/12**

Tabla 1. Tareas TFC

1.5. Hitos

Se establecen los hitos del proyecto, que serán directamente las entregas parciales

Hito TFC	Fecha límite
Borrador Plan de Trabajo	29/09/2012
Corrección Borrador Plan de Trabajo	02/10/2012
Entrega Plan de Trabajo	02/10/2012
Borrador PEC2	06/11/2012
Corrección Borrador PEC2	12/11/2012
Entrega PEC2	12/11/2012
Borrador PEC3	11/12/2012
Corrección Borrador PEC3	16/12/2012
Entrega PEC3	16/12/2012
Borrador Memoria	23/12/2012
Corrección Borrador Memoria	28/12/2012
Entrega Memoria y Presentación	07/01/2013

Tabla 2. Hitos TFC

1.6. Diagramas de Gantt

A continuación se detallan los diagramas de Gantt de las siguientes fases programadas para la *Definición del TFC* y el *Plan de trabajo*:

	Nombre de tarea	Comienzo	Fin	Predecesora	% completa
0	**dolmedod_Planificacion_TCF_Completo**	**mié 19/09/12**	**sáb 26/01/13**		**95%**
1	**1 Definición del TFC**	**mié 19/09/12**	**sáb 22/09/12**		**100%**
2	1.1 Recopilar y primera lectura de la documentación disponible en el aula y en el tablón	mié 19/09/12	vie 21/09/12		100%
3	1.2 Encuentro presencial por videoconferencia con el consultor	sáb 22/09/12	sáb 22/09/12	2	100%
4	**2 PLAN DE TRABAJO**	**lun 24/09/12**	**mar 02/10/12**		**100%**
5	**2.1 Documento Plan de Trabajo**	**lun 24/09/12**	**mar 02/10/12**		**100%**
6	2.1.1 Leer documentación. Estructurar y esquematizar la documentación	lun 24/09/12	jue 27/09/12		100%
7	2.1.2 Buscar bibliografía. Búsqueda de bibliografía útil para el proyecto a medio plazo	lun 24/09/12	jue 27/09/12		100%
8	2.1.3 Redactar índice del borrador	lun 24/09/12	jue 27/09/12	6;7CC	100%
9	2.1.4 Redactar borrador	vie 28/09/12	sáb 29/09/12	8	100%
10	2.1.5 Planificar tareas. Establecer los hitos del proyecto así como los contenidos de PEC2 y PEC3	vie 28/09/12	sáb 29/09/12	9	100%
11	2.1.6 Acabar el borrador y entregarlo al consultor	vie 28/09/12	sáb 29/09/12	10	100%
12	2.1.7 Corregir borrador con las correcciones y sugerencias del consultor	sáb 29/09/12	sáb 29/09/12	11	100%
13	2.1.8 Finalizar y completar el Plan de Trabajo	sáb 29/09/12	mar 02/10/12	12	100%
14	2.1.9 Entregar Plan de Proyecto	mar 02/10/12	mar 02/10/12	13	100%

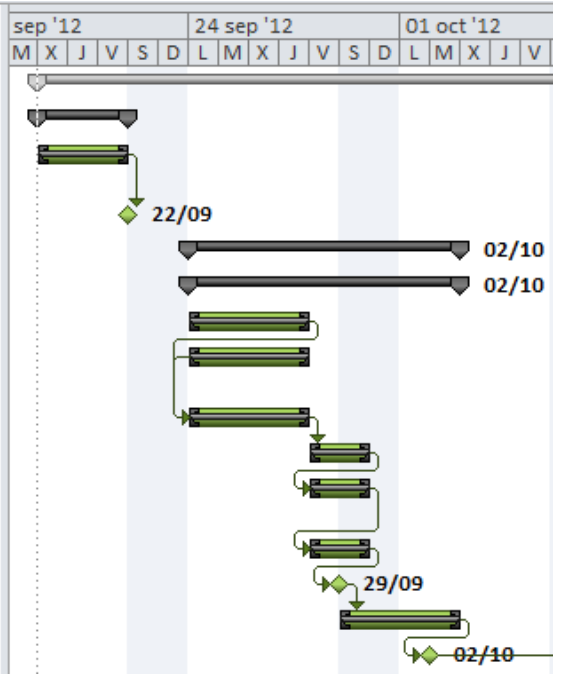


Figura 3. Diagrama de Gantt Definición y Plan de Trabajo TFC

En las siguientes figuras podemos ver los diagramas de Gantt con la distribución y la planificación de tareas de la *PEC2* y la *PEC3*

	Nombre de tarea	Comienzo	Fin	Predecesora	% completa
15	**3 PEC2**	**mar 02/10/12**	**lun 12/11/12**		**100%**
16	**3.1 ModBus**	**mar 02/10/12**	**dom 07/10/12**		**100%**
17	3.1.1 Buscar y recopilar información	mar 02/10/12	jue 04/10/12		100%
18	3.1.2 Estudio del protocolo ModBus	sáb 06/10/12	sáb 06/10/12	17	100%
19	3.1.3 Redactar documentación	dom 07/10/12	dom 07/10/12	18	100%
20	**3.2 RS-485**	**sáb 06/10/12**	**jue 11/10/12**		**100%**
21	3.2.1 Buscar y recopilar información	sáb 06/10/12	sáb 06/10/12	19	100%
22	3.2.2 Estudio del sistema de transmisión RS-485	dom 07/10/12	dom 07/10/12	21	100%
23	3.2.3 Redactar documentación	lun 08/10/12	jue 11/10/12	22	100%
24	**3.3 Adaptación E/S**	**vie 12/10/12**	**jue 18/10/12**		**100%**
25	3.3.1 Buscar y recopilar información	vie 12/10/12	vie 12/10/12	23	100%
26	3.3.2 Estudio requisitos e implementación solución	sáb 13/10/12	sáb 13/10/12	25	100%
27	3.3.3 Simulación circuito	dom 14/10/12	lun 15/10/12	26	100%
28	3.3.4 Redactar documentación	lun 15/10/12	jue 18/10/12	27	100%
29	**3.4 Transductor RS-485**	**vie 19/10/12**	**sáb 20/10/12**		**100%**
30	3.4.1 Buscar y recopilar información	vie 19/10/12	vie 19/10/12	23	100%
31	3.4.2 Estudio requisitos e implementación solución	sáb 20/10/12	sáb 20/10/12	30	100%
32	3.4.3 Redactar documentación	sáb 20/10/12	sáb 20/10/12	31	100%
33	**3.5 Procesador de control**	**lun 22/10/12**	**sáb 27/10/12**		**100%**
34	3.5.1 Buscar y recopilar información	lun 22/10/12	jue 25/10/12	28	100%
35	3.5.2 Estudio requisitos e implementación solución	vie 26/10/12	sáb 27/10/12	34	100%
36	3.5.3 Redactar documentación	sáb 27/10/12	sáb 27/10/12	35	100%
37	**3.6 Documento PEC2**	**lun 29/10/12**	**lun 12/11/12**		**100%**
38	3.6.1 Redactar Borrador	lun 29/10/12	dom 04/11/12	19;23;28;32;36	100%
39	3.6.2 Enviar el borrador al consultor	dom 04/11/12	dom 04/11/12	38	100%
40	3.6.3 Corregir borrador con las correcciones y sugerencias del consultor	lun 05/11/12	jue 08/11/12	39	100%
41	3.6.4 Añadir apartados no incluidos en el borrador	vie 09/11/12	vie 09/11/12	40	100%
42	3.6.5 Simular circuitos	sáb 10/11/12	sáb 10/11/12	41	100%
43	3.6.6 Finalizar y completar la PEC2	dom 11/11/12	dom 11/11/12	42	100%
44	3.6.7 Entregar la PEC2	lun 12/11/12	lun 12/11/12	43	100%

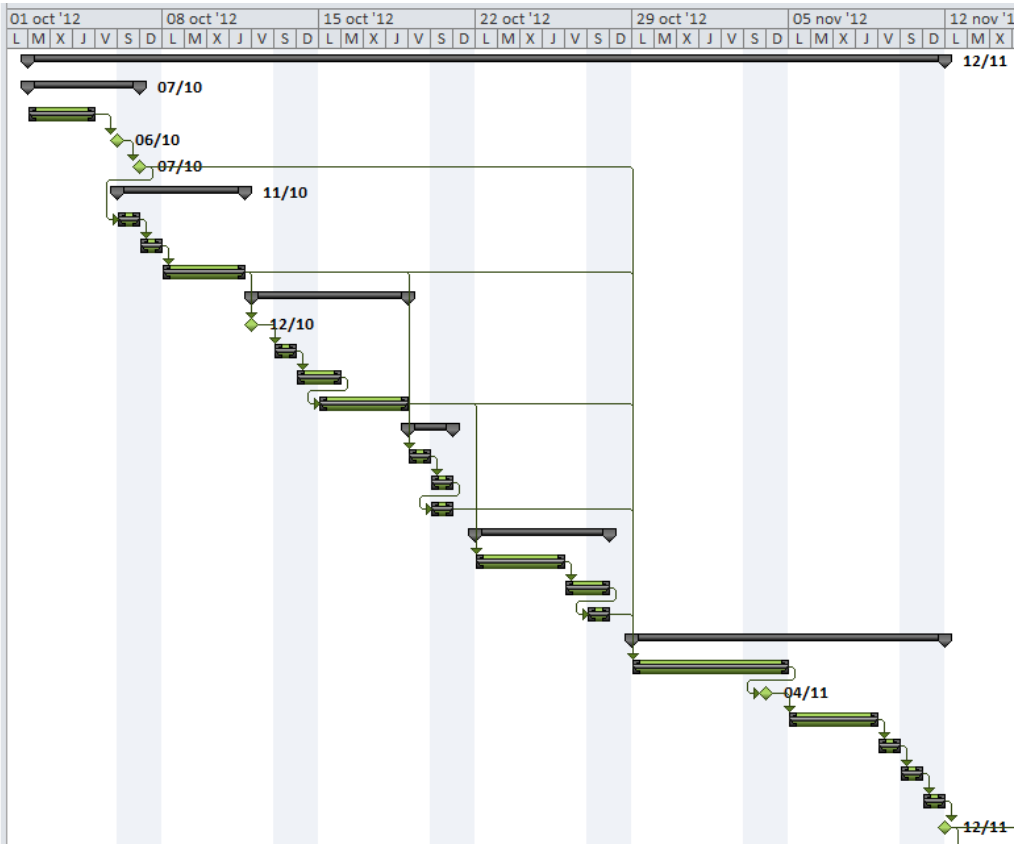


Figura 4. Diagrama de Gantt PEC2

	Nombre de tarea	Comienzo	Fin	Predecesora	% completa
45	**4 PEC3**	**lun 12/11/12**	**dom 16/12/12**		**100%**
46	**4.1 Modbus**	**lun 12/11/12**	**lun 12/11/12**		**100%**
47	4.1.1 Mapa de memoria	lun 12/11/12	lun 12/11/12		100%
48	4.1.2 Redactar documentación	lun 12/11/12	lun 12/11/12	47	100%
49	**4.2 Fuente de Alimentación**	**lun 12/11/12**	**jue 13/12/12**		**100%**
50	4.2.1 Buscar y recopilar información	lun 12/11/12	jue 13/12/12	44	100%
51	4.2.2 Estudio requisitos e implementación solución	jue 15/11/12	dom 02/12/12	50	100%
52	4.2.3 Simulación circuito	dom 02/12/12	dom 02/12/12	51	100%
53	4.2.4 Redactar documentación	lun 03/12/12	lun 03/12/12	52	100%
54	**4.3 Procesador de control**	**lun 19/11/12**	**dom 16/12/12**		**100%**
55	4.3.1 Buscar y recopilar información	lun 19/11/12	jue 22/11/12	48	100%
56	4.3.2 Estudio requisitos e implementación solución	vie 23/11/12	vie 23/11/12	55	100%
57	4.3.3 Programar el código de control del microprocesador	sáb 24/11/12	dom 16/12/12	56	100%
58	4.3.4 Simular PIC con el programa	sáb 24/11/12	dom 16/12/12	57	100%
59	4.3.5 Redactar documentación	dom 16/12/12	dom 16/12/12	58	100%
60	**4.4 Circuito completo**	**sáb 08/12/12**	**mié 12/12/12**		**100%**
61	4.4.1 Simulación circuito completo	sáb 08/12/12	dom 09/12/12	44	100%
62	4.4.2 Mejoras y optimización	sáb 08/12/12	dom 09/12/12	61	100%
63	4.4.3 Redactar documentación	mié 12/12/12	mié 12/12/12	62	100%
64	**4.5 Layout PCB**	**vie 14/12/12**	**dom 16/12/12**		**100%**
65	4.5.1 Recopilar listado de componentes	vie 14/12/12	vie 14/12/12	63	100%
66	4.5.2 Diseñarl Layout PCB	vie 14/12/12	sáb 15/12/12	65	100%
67	4.5.3 Redactar documentación	sáb 15/12/12	dom 16/12/12	66	100%
68	**4.6 Documento PEC3**	**dom 09/12/12**	**dom 16/12/12**		**100%**
69	4.6.1 Redactar Borrador	dom 09/12/12	mar 11/12/12	59;63;67;48;53	100%
70	4.6.2 Enviar el borrador al consultor	mar 11/12/12	mar 11/12/12	69	100%
71	4.6.3 Corregir borrador con las correcciones y sugerencias del consultor	mié 12/12/12	dom 16/12/12	70	100%
72	4.6.4 Finalizar y completar la PEC3	sáb 15/12/12	dom 16/12/12	71	100%
73	4.6.5 Entregar la PEC3	dom 16/12/12	dom 16/12/12	72	100%

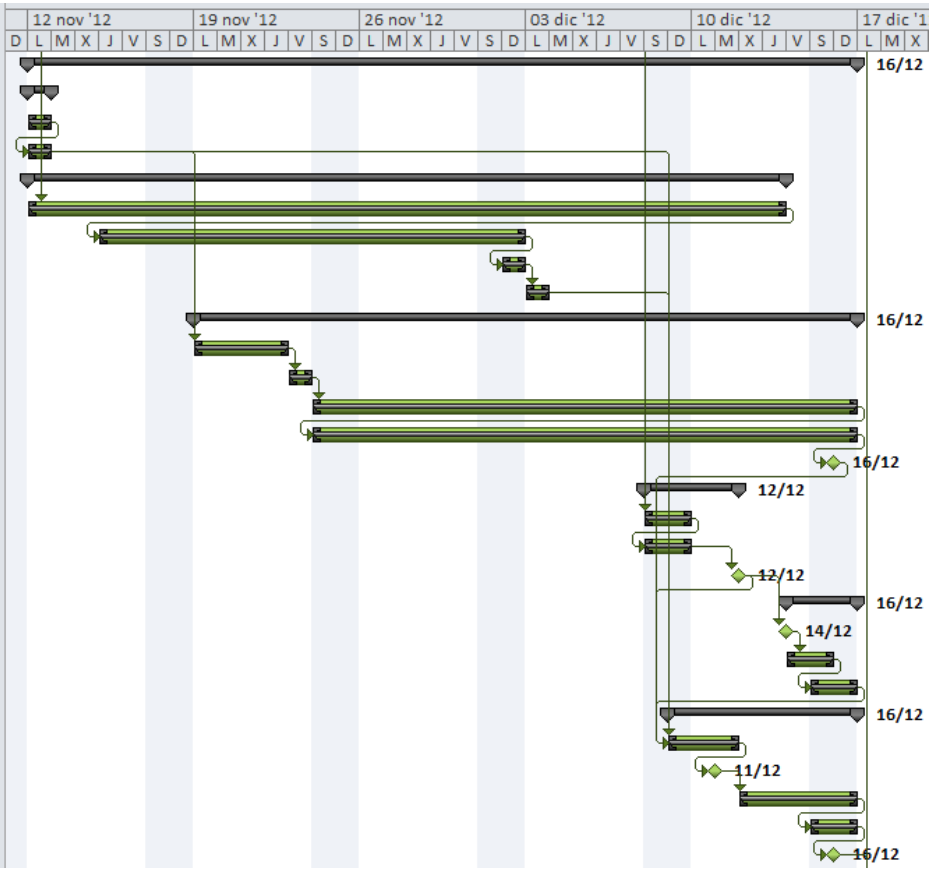


Figura 5. Diagrama de Gantt PEC3

Por último se muestra el diagrama de Gantt con las tareas programadas para la *Memoria Técnica* y el *Vídeo de Presentación Virtual*:

	Nombre de tarea	Comienzo	Fin	Predecesora	% completa
74	**5 MEMORIA TFC**	**lun 17/12/12**	**sáb 26/01/13**		**70%**
75	**5.1 Documento Memoria Técnica TFC**	**lun 17/12/12**	**lun 07/01/13**		**100%**
76	5.1.1 Recopilar documentación generada	lun 17/12/12	lun 17/12/12	14;44;73	100%
77	5.1.2 Integrar documentación en el documento	mar 18/12/12	jue 20/12/12	76	100%
78	5.1.3 Redactar conclusiones	vie 21/12/12	vie 21/12/12	77	100%
79	5.1.4 Redactar Borrador	sáb 22/12/12	dom 23/12/12	78	100%
80	5.1.5 Enviar el borrador al consultor	dom 23/12/12	dom 23/12/12	79	100%
81	5.1.6 Corregir borrador con las correcciones y sugerencias del consultor	lun 24/12/12	vie 28/12/12	80	100%
82	5.1.7 Finalizar y completar la Memoria Técnica	sáb 29/12/12	dom 30/12/12	81	100%
83	5.1.8 Revisión del documento definitvo (ortografía, gramática, sintaxis, estilo, etc.)	dom 30/12/12	lun 31/12/12	82	100%
84	5.1.9 Entregar la Memoria Técnica	lun 07/01/13	lun 07/01/13	83	100%
85	**5.2 Video Presentación**	**mar 01/01/13**	**dom 06/01/13**		**100%**
86	5.2.1 Confeccionar presentación Power Point	mar 01/01/13	vie 04/01/13	83	100%
87	5.2.2 Grabar locución de la exposición	sáb 05/01/13	sáb 05/01/13	86	100%
88	5.2.3 Maquetar y montar del video presentación	sáb 05/01/13	dom 06/01/13	87	100%
89	**5.3 Ficheros**	**sáb 05/01/13**	**sáb 05/01/13**		**0%**
90	5.3.1 Recopilar ficheros generados	sáb 05/01/13	sáb 05/01/13	83	75%
91	5.3.2 Comprimir los archivos	sáb 05/01/13	sáb 05/01/13	90	0%
92	**5.4 Debate Virtual**	**mié 23/01/13**	**sáb 26/01/13**		**0%**
93	5.4.1 Recopilar las cuestiones y dudas del tribunal	mié 23/01/13	vie 25/01/13	84	0%
94	5.4.2 Contestar a las cuestiones formuladas	jue 24/01/13	sáb 26/01/13	93	0%
95	5.4.3 Enviar documento con las respuestas	jue 24/01/13	sáb 26/01/13	94	0%

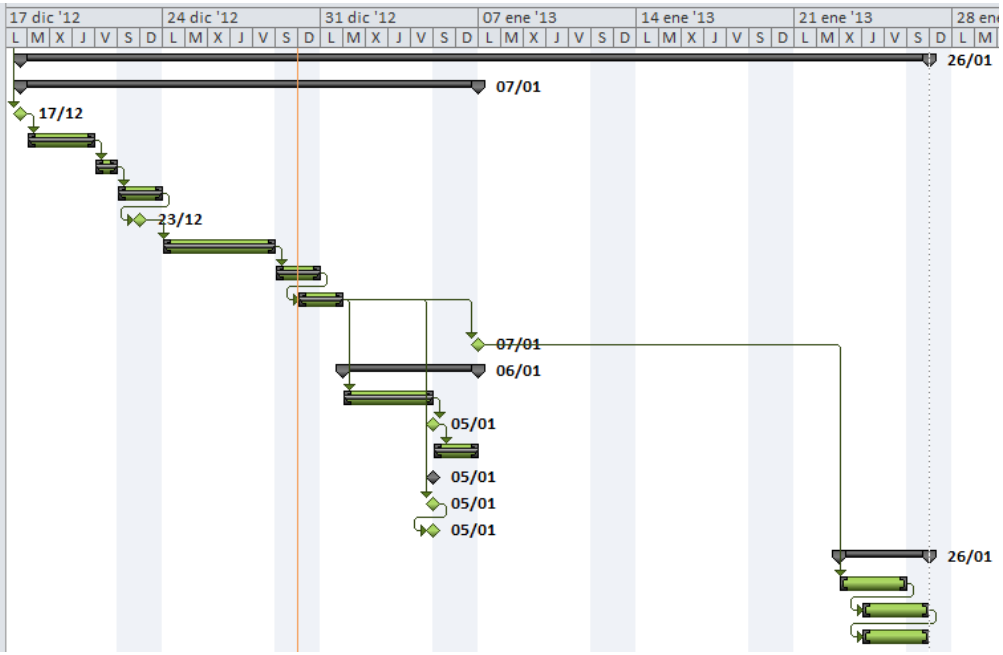


Figura 6. Diagrama de Gantt Memoria TFC

2. PROTOCOLO MODBUS

2.1. Introducción

MODBUS es un protocolo de comunicaciones que se sitúa en el nivel 7 del Modelo OSI. Éste proporciona comunicación tipo cliente/servidor o maestro/esclavo entre distintos dispositivos que pueden estar conectados a diferentes buses o redes.

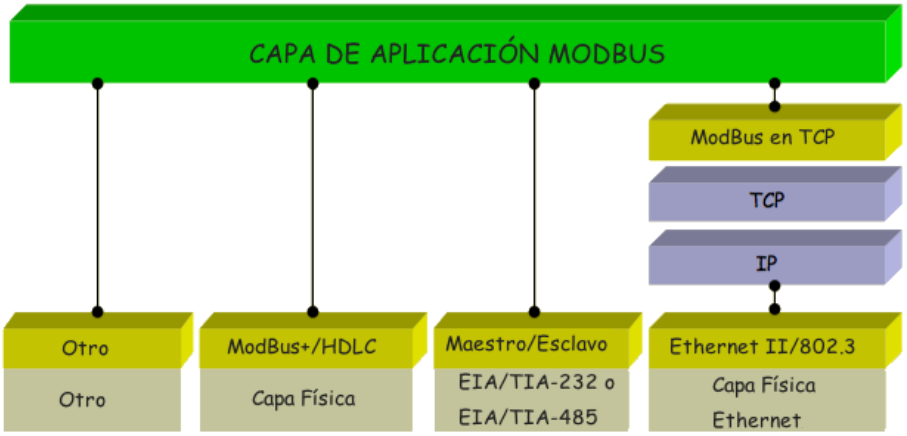


Figura 7. Pila de comunicación MODBUS

Fue diseñado en 1979 por Modicon para controlar sus PCLs. Actualmente se utiliza como estándar de comunicaciones principalmente en el sector industrial, ya que es público, su implementación es sencilla, requiere poco desarrollo y maneja bloques de datos sin restricción alguna. Incluso MODBUS tiene reservado el puerto 502 en la pila TCP/IP.

Para nuestro TFC vamos a centrarnos sobre la comunicación serie. Por lo tanto en la capa física se suelen usar diferentes interfaces (RS-485 o RS-232). Normalmente la opción de un par de cables trenzados es la más común para trabajar en modo *half-duplex*, aunque se puede trabajar con dos pares trenzados obteniendo una comunicación *full-duplex*.

2.2. Capa Enlace de Datos

Vamos a estudiar con más detenimiento la capa de enlace Maestro/Esclavo que se apoya sobre la capa física RS-485.

2.2.1. Modos de funcionamiento Maestro/Esclavo en MODBUS

MODBUS se compone de un dispositivo que actúa como *maestro* y de 1 a 247 dispositivos que actúan como *esclavos*. La comunicación se basa en el método petición-respuesta.

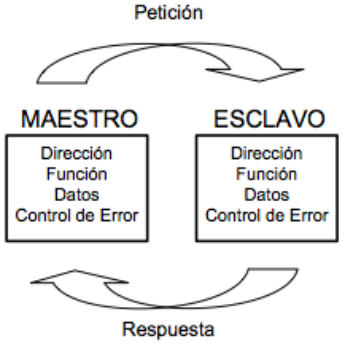


Figura 8. Funcionamiento maestro / esclavo

Ahora bien, el maestro puede realizar una petición a un dispositivo en concreto (*unicast*) que debe enviar la correspondiente respuesta o bien puede lanzar una petición de difusión (*broadcast*) que todos los dispositivos deben aceptar sin devolver respuesta alguna.

2.2.2. Direccionamiento

En MODBUS el direccionamiento se compone de 256 direcciones diferentes que deben identificar de manera unívoca a cada uno de los dispositivos de la red.

0	1 a 247	248 a 255
Dirección de difusión	Dirección individual de cada dispositivo esclavo	Reservadas

Tabla 3. Direccionamiento MODBUS

Nota: El dispositivo maestro no tiene una dirección específica.

2.2.3. Formato de las tramas

El protocolo MODBUS en su capa de aplicación define un único PDU (Protocol Data Unit) independientemente de las capas inferiores sobre las que se apoye.

MODBUS PDU	
Código de función	Datos

Tabla 4. PDU MODBUS

El mapeo del protocolo MODBUS sobre buses específicos o redes se realiza añadiendo campos adicionales al PDU básico. El cliente o maestro que inicia la comunicación añade los campos necesarios para construir el PDU apropiado para cada caso.

	MODBUS PDU		
Dirección	Código de función	Datos	CRC (o LRC)
PDU MODBUS Serie			

Tabla 5. PDU MODBUS serie

- El campo *dirección* contiene la dirección del esclavo al que va dirigida la petición.
- El campo *código de función* indica qué tipo de acción hay que realizar.
- El campo *datos* contiene la información necesaria para lleva a cabo la acción especificada
- El campo *CRC* (o LRC) es un campo de comprobación de errores. En nuestro caso, en el modo RTU, que veremos más adelante, usaremos CRC.

2.2.4. Diagramas de estado

Vamos a mencionar sin entrar en detalles, ya que no es objeto de nuestro TFC, los diagramas de estado en los cuales se puede encontrar un dispositivo maestro:

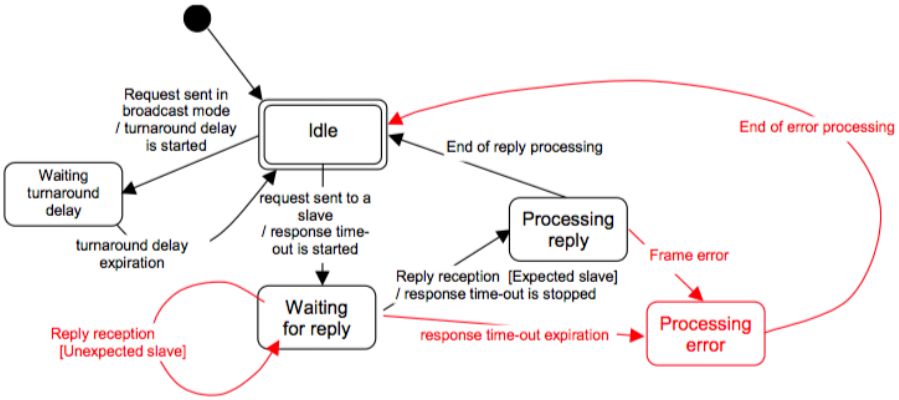


Figura 9. Diagrama de estados dispositivo maestro

- Estado "***Idle***" = Ninguna petición pendiente. Éste es el estado inicial tras iniciar el sistema. Una petición sólo puede ser enviada cuando el dispositivo se encuentra en este estado. Una vez que se envía una petición el maestro abandona este estado y no puede enviar una nueva petición al mismo tiempo.
- Cuando se envía una petición *unicast* a un esclavo, el maestro pasa al estado "***Waiting for reply***", y el tiempo de respuesta máximo ("*Response Time-out turnaroud*") se inicia. Esto previene al Maestro de estar indefinidamente esperando una respuesta "***Waiting for reply***".
- Cuando una respuesta es recibida, el Maestro la verifica antes de comenzar el procesado de datos.
- Si no se recibe una respuesta, el tiempo de respuesta máximo expira y se genera un error. Es entonces cuando el Maestro pasa de nuevo al estado inicial "***Idle***".
- Cuando una petición *multicast* es enviada por el bus, no se espera respuesta de los esclavos. No obstante el Maestro respeta el tiempo de respuesta ("*Turnaround delay*") antes de enviar una nueva petición.
- En modo *unicast* el tiempo de respuesta máximo debe durar lo suficiente para que el esclavo pueda procesar la petición, los datos y devolver la respuesta.
- ***Frame error*** consiste en: 1) Comprobación de paridad a cada caracter; 2) CRC aplicado a la trama completa.

Y un esclavo:

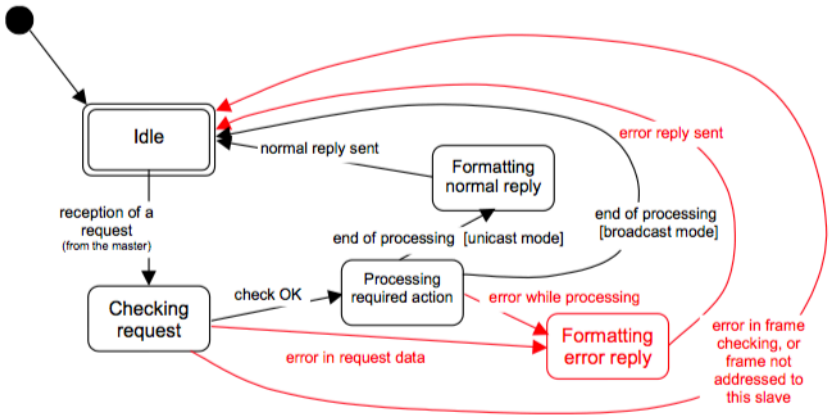


Figura 10. Diagrama de estados dispositivo esclavo

- Estado "***Idle***" = Ninguna petición pendiente. Éste es el estado inicial tras iniciar el sistema.
- Cuando una petición es recibida, el Esclavo comprueba el paquete antes de realizar la acción requerida por el Maestro.
- Una vez que la acción solicitada se ha completado, un mensaje *unicast* requiere que una respuesta con el formato correcto debe ser enviada al Maestro.
- Si el Esclavo detecta un error en la trama recibida, no enviará una respuesta al Maestro.

2.2.5. Modos de transmisión serie.

MODBUS define dos modos de transmisión serie: Modo RTU (que es objeto de este TFC) y modo ASCII. Es importante destacar que si se selecciona un modo de transmisión, todos los dispositivos conectados a ese bus serie o red deben usar el mismo método. No obstante, el método de transmisión por defecto es el RTU.

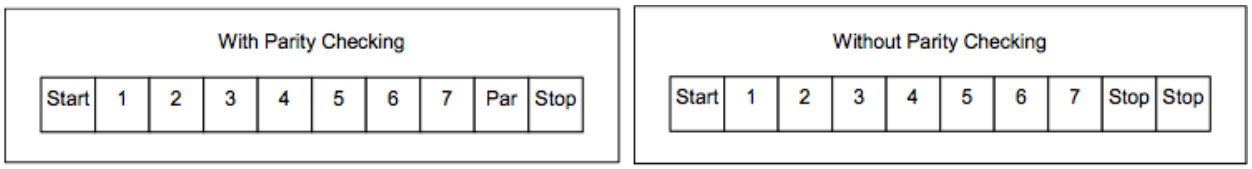
MODBUS utiliza representación "*big-Endian*" para direcciones y datos. Esto significa que cuando se transmiten más de un byte, el byte más significativo se envía en primer lugar. Por ejemplo, si tenemos un registro de 16 bits (2 bytes) con el valor 0x1234, el primer byte que se enviaría es el 0x12 y después el 0x34.

Modo ASCII

En la transmisión mediante el modo ASCII (*American Standard Code for Information Interchange*), cada byte -8 bits- en un mensaje se envía como dos caracteres ASCII. La principal ventaja de este modo es que permite intervalos de tiempo de hasta un segundo entre caracteres sin dar lugar a error.

El formato para cada byte en modo ASCII es:

- Sistema de codificación: Hexadecimal, caracteres ASCII 0-9, A-F. Un caracter hexadecimal contenido en cada carácter ASCII del mensaje.
- Bits por byte: 1 bit de arranque. 7 bits de datos, el menos significativo se envía primero. 1 bit para paridad Par o Impar; ningún bit para No paridad. 1 bit de paro si se usa paridad; 2 bits si no se usa paridad.



- Campo de comprobación de error: *Logitudinal Redundancy Checking* (LRC).

El tamaño máximo de una trama MODBUS ASCII es de 513 caracteres:

Comienzo	Dirección esclavo	Código de función	Datos	LRC	Fin
1 char	2 chars	2 chars	0 a 2x252 chars	2 chars	2 chars

Tabla 6. Trama MODBUS ASCII

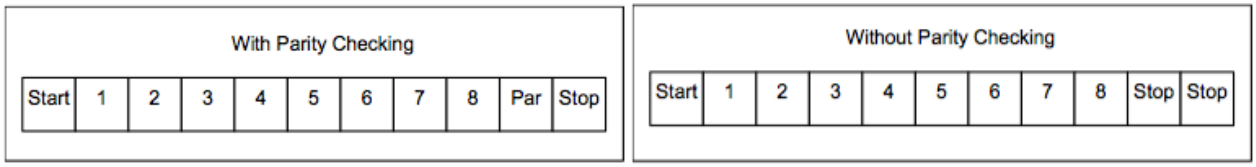
Modo RTU

En la transmisión mediante el modo RTU (*Remote Terminal Unit*), cada byte en un mensaje contiene dos dígitos hexadecimales de 4 bits. La principal ventaja de este modo es que su mayor densidad de caracteres permite mejor rendimiento que el modo ASCII a una misma velocidad. Cada mensaje debe ser transmitido en un flujo continuo.

El formato para cada byte en modo RTU es:

- Sistema de codificación: Binario 8-bits, hexadecimal 0-9,A-F. Dos dígitos hexadecimales contenidos en cada campo de 8 bits del mensaje.

- Bits por byte: 1 bit de arranque. 8 bits de datos, el menos significativo se envía primero. 1 bit para paridad Par o Impar; ningún bit para No paridad. 1 bit de paro si se usa paridad; 2 bits si no se usa paridad.



- Campo de comprobación de error: *Cyclical Redundancy Checking* (CRC).

El tamaño máximo de una trama MODBUS RTU es de 256 bytes:

Dirección esclavo	Código de función	Datos	CRC
1 byte	1 byte	0 a 252 bytes	2 bytes

Tabla 7. Trama MODBUS RTU

En el siguiente esquema podemos ver los estados maestro/esclavo de una transmisión MODBUS RTU

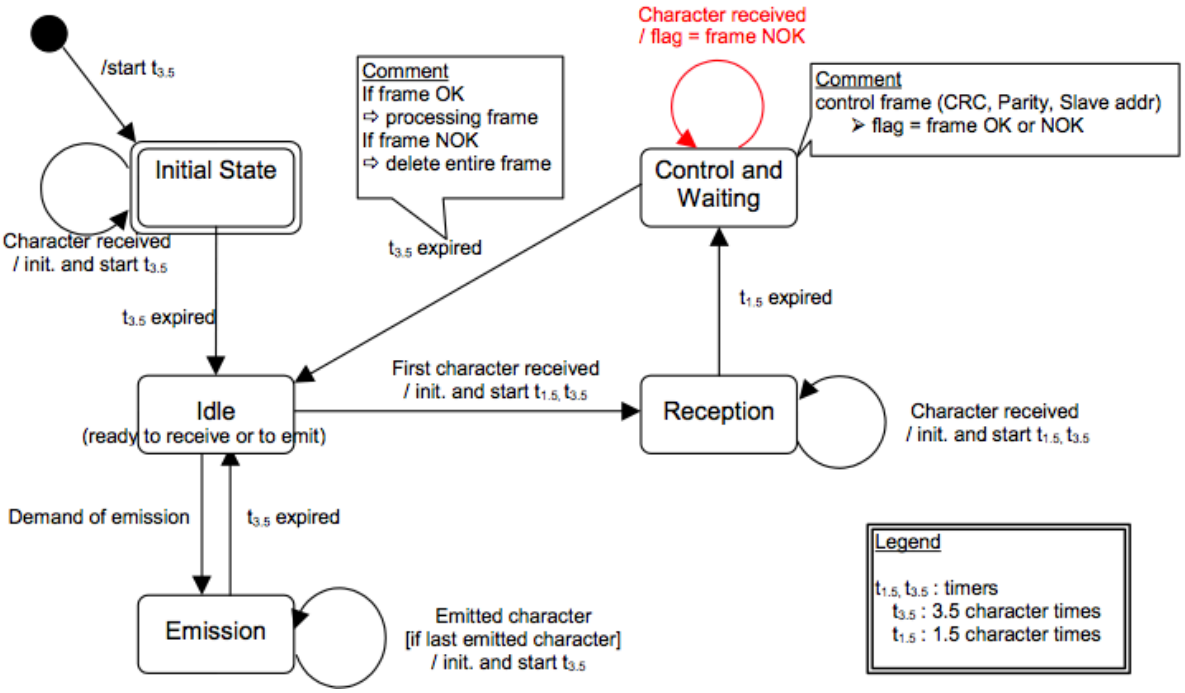


Figura 11. Diagrama de estados de una transmisión MODBUS RTU

- La transición desde el estado inicial "***Initial State***" al estado "***Idle***" necesita que transcurran los 3.5t para asegurar el retraso inter-trama.
- El estado "***Idle***" es el estado natural cuando ni se envía una petición ni se espera una respuesta.
- Cuando se detecta un inicio de trama (*start of frame*), se pasa al estado "activo".
- Cuando no se detectan más caracteres, tras el intervalo 3.5t, se identifica como un fin de trama (*end of frame*).
- Después de detectar un fin de trama, el cálculo y la comprobación CRC es completado.

2.2.6. Tipos de datos

MODBUS diferencia cuatro tipos distintos de datos y tiene funciones específicos para ellos, con un direccionamiento independiente. Se diferencian, básicamente, entre lectura o escritura y si son de tamaño bit o Word (16 bits). Así pues tenemos:

Tipo de dato	Lectura / Escritura	Tamaño	Observaciones
Entrada digital *(discrete input)*	Lectura	1 bit	Variable de un bit que el maestro puede leer
Salida digital *(coil)*	Lectura / Escritura	1 bit	Variable de un bit que el maestro puede leer y escribir
Registro de entrada *(input register)*	Lectura	16 bits (1 Word)	Variable de 16 bits que el maestro puede leer
Registro de almacenamiento *(holding register)*	Lectura / Escritura	16 bits (1 Word)	Variable de 16 bits que el maestro puede leer y escribir

Tabla 8. Tipos de datos MODBUS

Aunque el direccionamiento de cada tipo de dato es independiente, el esclavo o el maestro puede tener estas áreas solapadas de manera que podamos acceder al mismo dato de diferente forma.

2.2.7. Mapa de Memoria

Por norma general, en MODBUS cada tipo de dato se mapea en un rango de memoria concreto:

- **@1-10000 (DOs - *digital outputs*):** 1 bit por dirección para indicar el estado de una salida, mando o relé (0 desactivado, 1 activado). A las direcciones de este rango se suelen acceder mediante las funciones 1 (lectura), 5 (escritura), 15 (escritura múltiple).
- **@10001-20000 (DIs - *digital inputs*):** 1 bit por dirección para leer el estado de una entrada digital (0 desactivada, 1 activada) también denominadas DIs (*Digital Inputs*). A las direcciones de este rango se suelen acceder con la función 2 (lectura) y llevan implícita la dirección 10001 como dirección base (para acceder a una dirección bastará con especificar la distancia entre ésta y la dirección base).
- **@20001-30000:** el protocolo MODBUS estándar no hace uso de este rango de direcciones.
- **@30001-40000 (AIs - *analog inputs*):** 16 bits por dirección con el estado de las medidas o entradas analógicas también denominadas AIs (*Analog Inputs*). Dependiendo del dispositivo, éste puede hacer uso de más de un registro para almacenar la información de la medida, así con 2 registros consecutivos podríamos almacenar medidas de 32 bits. A las direcciones de este rango se accede mediante la función 4 (lectura) y llevan implícita la dirección 30001 como dirección base (para acceder a una dirección bastará con especificar la distancia entre ésta y la dirección base).
- **@40001-49990 (AOs - *analog outputs*):** 16 bits con los registros de salidas analógicas o de propósito general (*Output Registers – Holding Registers*). Se accede con las funciones 3 (lectura), 6 (escritura) o 16 (escritura múltiple) y llevan implícita la dirección 40001 como dirección base (para acceder a una dirección bastará con especificar la distancia entre ésta y la dirección base).
- **@49991-50000 (Modbus configuration parameters):** 10 Registros donde se guarda la configuración de los parámetros Modbus. Actualmente usaremos sólo 3.

Por lo tanto, el mapa de memoria de nuestro dispositivo será el siguiente:

Dirección @	Dirección I/O	Id.	Descripción	Tipo	Acceso R/W	Rango Valores
Salidas Digitales – Rango de direcciones @00001-10000						
--	--	--	--	--	--	--
Entradas Digitales – Rango de direcciones @10001-20000						
--	--	--	--	--	--	--
Reservado – Rango de direcciones @20001-30000						
--	--	--	--	--	--	--
Entradas Analógicas– Rango de direcciones @30001-40000						
@30001	0x0000	IN_01	Entrada Analógica Tensión	Word	R	0-10 V
@30002	0x0001	IN_02	Entrada Analógica Tensión	Word	R	0-10 V
@30003	0x0002	IN_03	Entrada Analógica Corriente	Word	R	4-20 mA
@30004	0x0003	IN_04	Entrada Analógica Corriente	Word	R	4-20 mA
Salidas Analógicas– Rango de direcciones @40001-49990						
@40001	0x0004	OUT_01	Salida Analógica Tensión	Word	W	0-10 V
@40002	0x0005	OUT_02	Salida Analógica Tensión	Word	W	0-10 V
Registros de configuración Modbus - Rango de direcciones @49991-50000						
@49991	0x1000	V_Tx	Velocidad de Transmisión Modbus	Word	R/W	*Ver nota1
@49992	0x1001	PAR	Paridad Modbus RTU	Word	R/W	*Ver nota2
@49993	0x1002	S_ID	Identificador Dispositivo	Word	R/W	1-247

Tabla 9. Mapa de memoria MODBUS

Nota1: *1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bauds*
Nota2: 0 = Ninguna, 1 = impar, 2 = par

2.2.8. Principales funciones

En MODBUS existen tres categorías de códigos de funciones:

- Códigos de funciones públicos.
- Códigos de funciones definidos por el usuario.
- Códigos de funciones reservados.

				Function Codes			
				code	Sub code	(hex)	Section
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02	6.2
		Internal Bits Or Physical coils	Read Coils	01		01	6.1
			Write Single Coil	05		05	6.5
			Write Multiple Coils	15		0F	6.11
	16 bits access	Physical Input Registers	Read Input Register	04		04	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03	6.3
			Write Single Register	06		06	6.6
			Write Multiple Registers	16		10	6.12
			Read/Write Multiple Registers	23		17	6.17
			Mask Write Register	22		16	6.16
			Read FIFO queue	24		18	6.18
	File record access		Read File record	20		14	6.14
			Write File record	21		15	6.15
Diagnostics			Read Exception status	07		07	6.7
			Diagnostic	08	00-18,20	08	6.8
			Get Com event counter	11		OB	6.9
			Get Com Event Log	12		0C	6.10
			Report Slave ID	17		11	6.13
			Read device Identification	43	14	2B	6.21
Other			Encapsulated Interface Transport	43	13,14	2B	6.19
			CANopen General Reference	43	13	2B	6.20

Tabla 10. Principales funciones MODBUS

Vamos a detallar algunas de las funciones básicas de escritura y lectura sobre los tipos de datos en MODBUS y vamos a indicar si nuestro dispositivo la implementará para cumplir con los requisitos exigidos:

Código	Función	Descripción	Requisito
01	*Read Coil Status*	Obtiene el estado actual ON/OFF de un grupo de bobinas lógicas. Lee n salidas digitales.	No
02	*Read Input Status*	Obtiene el estado actual ON/OFF de un grupo de entradas lógicas. Lee n entradas digitales.	No
03	*Read Holding Register*	Obtiene el valor binario de uno o más registros de almacenamiento. Lee n registros de salida (*holding registers*).	No
04	*Read Input Register*	Obtiene el valor binario de uno o más registros de entrada (*input registers*).	Sí
05	*Force Single Coil*	Fuerza el estado de una bobina (coil).	No
06	*Presets Single Register*	Escribe el valor binario de un registro de almacenamiento (*holding register).*	Sí
15	*Force Multiple Coils*	Fuerza el estado de un grupo de bobinas (*coils*).	No
16	*Presets Multiple Register*	Escribe el valor binario de un grupo de registros de almacenamiento.	Sí

Tabla 11. Funciones básicas MODBUS

EJEMPLO: CONFIGURACION DE PARÁMETROS DE COMUNICACIÓN MODBUS (función 0x10)

Usaremos esta función para escribir en los tres registros habilitados para ello (velocidad de transmisión, paridad e identificación del esclavo).

Petición															
Id del esclavo	**Código de función**	**Datos**												**CRC**	
		Dirección 1er registro		**Nº de registros a escribir (máx. 123)**		**Contador de Bytes**		**Valor del registro 1**		**Valor del registro …**		**Valor del registro N**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	…	…	MSB	LSB	MSB	LSB

Respuesta							
Dirección del dispositivo	**Código de función**	**Datos**				**CRC**	
		Dirección 1er registro		**Nº Bytes escritos**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Ejemplo de petición para configurar los valores 9600 (0x2580) bauds, paridad =0 e id=1.

Petición															
Dirección esclavo	Código de función	Datos												CRC	
01	10	10	00	00	03	00	06	25	80	00	00	00	01	E1	74

Respuesta							
Dirección esclavo	Código de función	Datos				CRC	
01	10	10	00	00	03	84	C8

Por software debemos detectar si se manda a la dirección de *multicast* (0x00) y no actualizar la id de los dispositivos esclavos (por ejemplo obligando al dispositivo maestro a colocar en el registro id=0x1002 el valor 0x0000), lanzando un mensaje de error a través de una excepción si no fuera así. Debemos indicar esta instrucción de configuración en el ***manual de uso*** que se debería entregar junto con el dispositivo real a un posible cliente.

También se deberá incluir en el ***manual de uso*** la condición de que la configuración de los parámetros Modbus debe hacerse con los tres parámetros (velocidad de transmisión, paridad e identificación del dispositivo) y no se contempla la posibilidad de cambiarlos uno a uno.

EJEMPLO: LECTURA DE DOS ENTRADAS ANALÓGICAS (función 0x04)

Usaremos esta función para leer una o varias entradas analógicas, ya sean de tensión o de corriente.

Petición							
Dirección del dispositivo	**Código de función**	**Datos**				**CRC**	
		Dirección 1er registro		**Nº de registros a leer (máx. 51)**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Respuesta									
Dirección del dispositivo	**Código de función**	**Nº bytes leídos**	**Datos**					**CRC**	
			Valor 1er registro		**…**	**Valor último registro**			
1 byte	1 byte	1 byte	MSB	LSB	…	MSB	LSB	MSB	LSB

Ejemplo de petición para leer las dos entradas de tensión de nuestro dispositivo. Vamos a suponer que la entrada 1 tiene un valor de 5 V (0x0200 – 512d) y la entrada 2 tiene un valor de 3 V (0x0133 – 307d):

Petición							
Dirección esclavo	Código de función	Datos				CRC	
01	04	00	00	00	02	71	CB

Respuesta								
Dirección esclavo	Código de función	Nº bytes leídos	Datos				CRC	
01	04	04	02	00	01	33	BB	9B

Si se produjera un error, por ejemplo, que el dispositivo estuviera ocupado:

Respuesta Error				
Dirección esclavo	Código de función	Código de Error	CRC	
01	84	06	C3	02

EJEMPLO: ESCRITURA DE UNA SALIDA ANALÓGICA (función 0x06)

Usaremos esta función para escribir en un registro el valor que queremos transmitir a una de las salidas analógicas.

Petición							
Dirección del dispositivo	**Código de función**	**Datos**				**CRC**	
		Dirección del registro		**Valor del registro**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Respuesta							
Dirección del dispositivo	**Código de función**	**Datos**				**CRC**	
		Dirección del registro		**Valor del registro**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Ejemplo de petición para escribir en la salida analógica 1 el valor 7.5V (0x0300 – 768d)

Petición							
Dirección esclavo	Código de función	Datos				CRC	
01	06	00	04	03	00	C8	FB

Respuesta							
Dirección esclavo	Código de función	Datos				CRC	
01	06	00	04	03	00	C8	FB

Si se produjera un error, por ejemplo, que el dispositivo no respondiera:

Respuesta Error				
Dirección esclavo	Código de función	Código de Error	CRC	
01	86	04	43	A3

EJEMPLO: ESCRITURA DE DOS SALIDAS ANALÓGICAS (función 0x10)

Usaremos esta función para escribir en dos registros los valores que queremos transmitir a las dos salidas analógicas.

Petición															
Id del esclavo	**Código de función**	**Datos**												**CRC**	
		Dirección 1er registro		**Nº de registros a escribir (máx. 123)**		**Contador de Bytes**		**Valor del registro 1**		**Valor del registro …**		**Valor del registro N**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	…	…	MSB	LSB	MSB	LSB

Respuesta							
Dirección del dispositivo	**Código de función**	**Datos**				**CRC**	
		Dirección 1er registro		**Nº Bytes escritos**			
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Ejemplo de petición para escribir en las salidas analógicas los valores 5 V (0x0200 – 512d) y 3 V (0x0133 – 307d) respectivamente.

Petición													
Dirección esclavo	Código de función	Datos										CRC	
01	10	00	04	00	02	00	04	02	00	01	33	6D	FC

Respuesta							
Dirección esclavo	Código de función	Datos				CRC	
01	10	00	04	00	02	00	09

Si se produjera un error, por ejemplo, que la función no pudiera realizarse en ese momento:

Respuesta Error				
Dirección esclavo	Código de función	Código de Error	CRC	
01	90	0F	0C	04

2.2.9. Códigos de error

Si un esclavo no puede realizar la función requerida por el maestro debe devolver una trama de error.

Del mismo modo que para cada petición existe una respuesta, para cada función existe una trama para la respuesta de error. No es objeto de este TFC estudiarlas, pero vamos a enumerar los distintos códigos de error que pueden producirse:

Código	Tipo de error	Descripción
01	Función ilegal	La función recibida no está permitida en el esclavo.
02	Dirección ilegal	La dirección está fuera del rango permitido.
03	Dato ilegal	El dato contiene un valor no válido.
04	Fallo en el dispositivo	El controlador no responde o ha ocurrido un error.
05	Reconocimiento (ACK)	Se ha aceptado la función y se está procesando.
06	Ocupado	El mensaje ha sido recibido sin error, pero el dispositivo no puede procesarlo en este momento.
15	Reconocimiento negativo (NAK)	La función solicitada no puede realizarse en este momento.

Tabla 12. Códigos de error MODBUS

***Nota**: Cuando se produce un error se pone a 1 el bit de mayor peso del código de la función solicitada.*

2.2.10. Solución adoptada para el TFC

A nivel protocolo de comunicaciones MODBUS, por las especificaciones del enunciado del TFC, vamos a necesitar la siguiente funcionalidad:

- Leer una o varias entradas analógicas de las que dispone el dispositivo extensor (ya sean de tensión y/o corriente).
- Escribir en una o las dos salidas de tensión analógicas que tiene disponibles el dispositivo.
- Escribir los parámetros de configuración Modbus mediante petición del dispositivo maestro.

Tenemos que considerar nuestro dispositivo como un esclavo que debe volcar al bus RS-485, previa petición por parte del maestro, el valor de una de las entradas cuando el código de función que haya solicitado el maestro sea el 0x04 (leer de uno o más registros el/los valor/es de una o varias entradas) o tome del bus RS-485 y escriba en una o en las dos salidas el/los valor/es que el maestro haya volcado en el bus junto con el código de función 0x06 (escribir un valor en un registro) o 0x10 (escribir en varios registros).

Así pues, vamos a suponer que este dispositivo extensor esclavo tiene una dirección válida, como un esclavo más del bus (entre 0 y 247). Por norma general se suele asignar la dirección 01d – 01h a nuestro dispositivo extensor, pero también se podrá cambiar por petición del dispositivo maestro.

Por lo tanto, vamos a ver las funciones y tramas de las peticiones y las respuestas (correctas y de error) que se van a generar en cada uno de los casos.

ENTRADAS: CONVERSIÓN A/D

Las entradas que tenemos son analógicas, pero para poder tratarlas con el microcontrolador debemos muestrearlas y realizar la conversión A/D. Para ello, usaremos el conversor A/D que trae incorporado el microcontrolador.

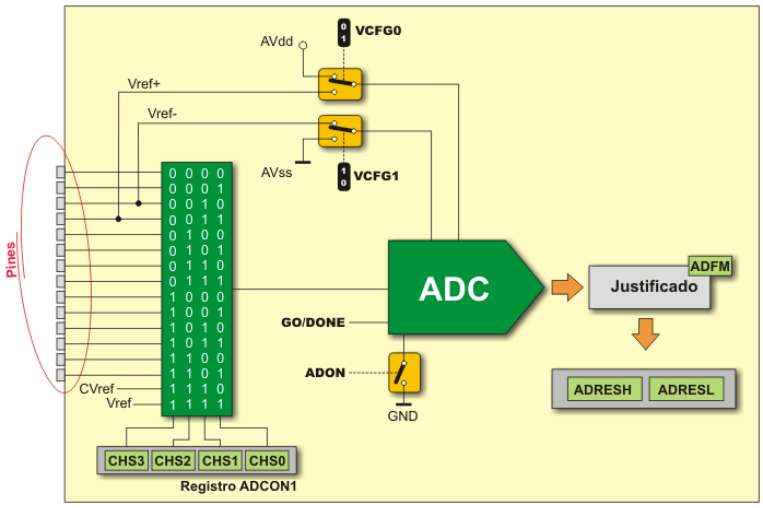


Figura 12. Conversor A/D

Por software, procederemos a realizar la conversión A/D y a guardar el resultado de cualquiera de los cuatro canales que usaremos en varios registros o variables de modo que podamos acceder, si fuera necesario, al valor de estas entradas en cualquier momento.

SALIDAS: CONVERSIÓN D/A

Al contrario que ocurre con las entradas, el microprocesador nos entrega en sus puertos de salida solamente señales digitales, por lo tanto como nuestros requisitos marcan que debemos tener salidas analógicas, tendremos que realizar la conversión D/A.

Existen multitud de soluciones para realizar una conversión D/A, tal y como utilizar dispositivos específicos que realizan esta operación. En nuestro caso, vamos a usar una técnica que consiste en generar una señal PWM que pasaremos a través de un filtro RC de modo que obtengamos una señal analógica.

Nota: Se recomienda para obtener una señal analógica con poca fluctuación que la frecuencia de la señal PWM sea varios órdenes de magnitud superior a la frecuencia de corte del filtro RC)

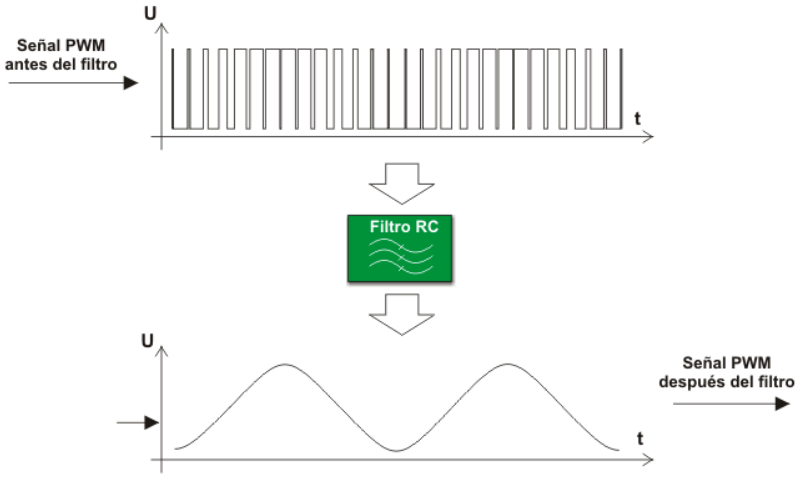


Figura 13. Conversor D/A

También por software, activaremos los módulos CCP para que trabajen en modo PWM, indicando que el ciclo de trabajo de éstos sea el valor de la salida que queremos obtener.

2.3. Capa física (RS-485)

2.3.1. Introducción

También conocido como EIA/TIA-485 es un estándar de comunicaciones en bus de la capa física del modelo OSI. Es un sistema en bus de transmisión que soporta altas velocidades sobre largas distancias (35 Mbit/s hasta 10 metros y 100 Kbits/s en 1200 metros) y a través de canales ruidosos al caracterizarse por ser un bus multipunto diferencial.

2.3.2. Topología

Los estándares RS-485 sugieren que sus nodos se dispongan siguiendo la topología *daisy-chain*, también conocida como *party line* o topología de bus. En esta topología, los *drivers*, *receivers* y *transceivers* se conectan al bus mediante pequeños *stubs* de red.

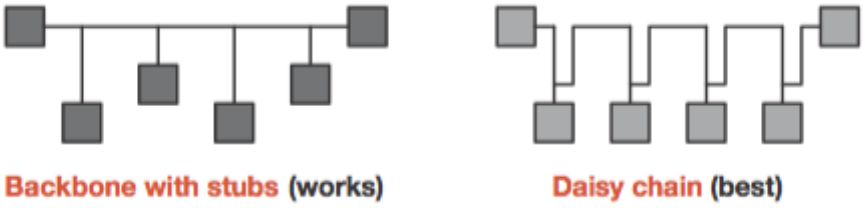


Figura 14. Topologías recomendadas para trabajar con RS-485

Las topologías que no están permitidas son en estrella, anillo o *backbone* con estrellas o *clusters*.

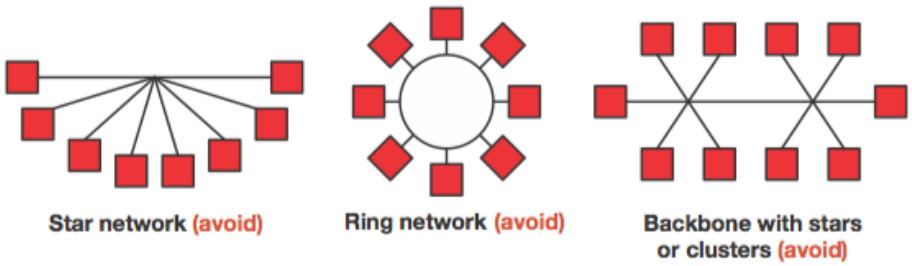


Figura 15. Topologías prohibidas para trabajar con RS-485

Este bus de transmisión permite tanto la implementación *full-duplex* con 2 pares trenzados de cables o *half-duplex* usando solamente un par trenzado.

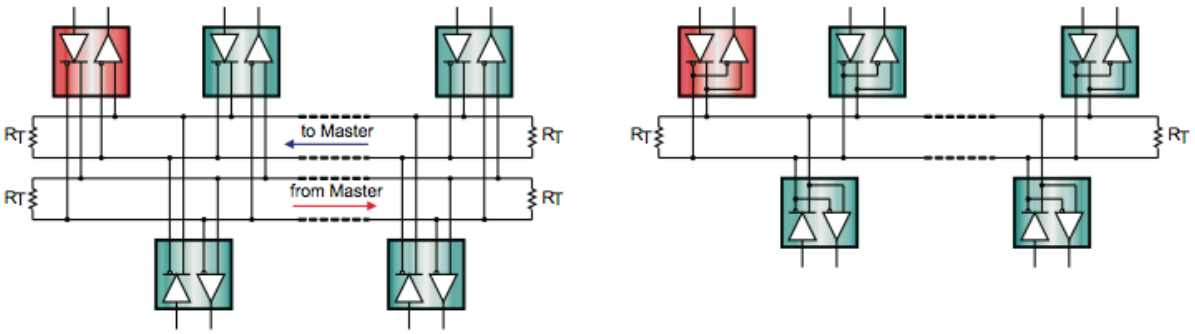


Figura 16. Esquema de conexionado full-duplex y half-duplex

En nuestro proyecto, nos decantaremos por usar la opción *half-duplex*.

2.3.3. Niveles de señal

El estándar RS-485 señala que los *drivers* deben proporcionan una salida diferencial de un mínimo de 1,5 V a través de una carga de 54 Ω, y que los *receivers* deben tener un nivel mínimo de 200 mV de tensión diferencial a la entrada.

Estos valores proporcionan un margen suficiente para asegurar una transmisión de datos fiable, incluso aunque se produzca una severa degradación, de la señal a través del cable y/o de los conectores. Esta robustez es la razón principal por la que RS-485 es muy adecuado para la creación de redes de larga distancia en entornos ruidosos.

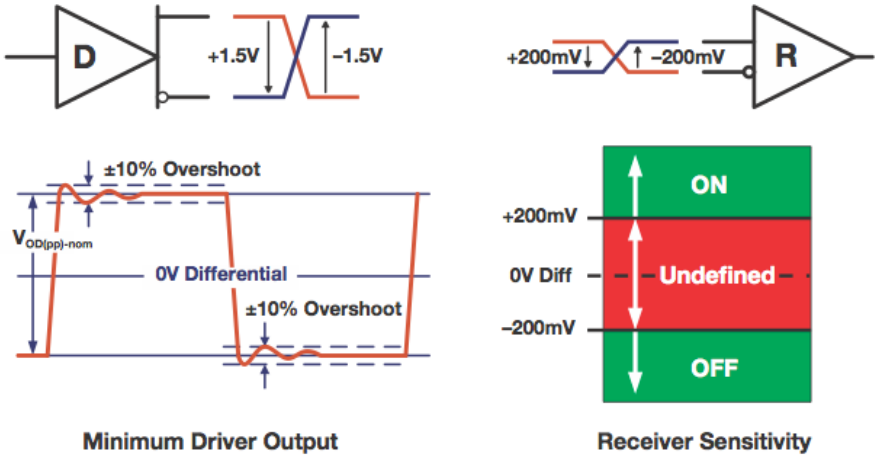


Figura 17. Niveles de tensión RS-485

2.3.4. Cableado y conectores

Las aplicaciones bajo RS-485 se benefician de la señal diferencial sobre pares de cable trenzado porque el ruido externo que se puede introducir se anula por la corriente diferencial que circula por cada uno de los cables trenzados. Existen multitud de cables industriales en el mercado: con cubierta aislados (FTP), sin aislamiento (UTP) con impedancias características de 120 Ω y 22-24 AWG.

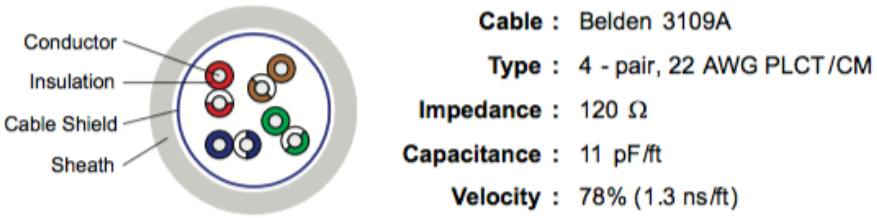


Figura 18. Características del cable de bus RS-485

Las líneas de transmisión de datos deben tener siempre dispositivos terminadores y los *stubs* deben ser lo más cortos posibles para evitar reflexiones en la línea. La terminación adecuada requiere que coincida la impedancia del terminador Z_T con la impedancia característica de la línea de transmisión Z_0. Como el estándar recomienda cables con $Z_0 = 120\ \Omega$ lo habitual es colocar resistencias de 120 Ω en cada extremo del cableado.

El uso en aplicaciones con entornos ruidosos normalmente reemplazan las resistencias de 120 Ω por dos resistencias de 60 Ω, formando un filtro paso bajo que elimine o atenúe esas interferencias.

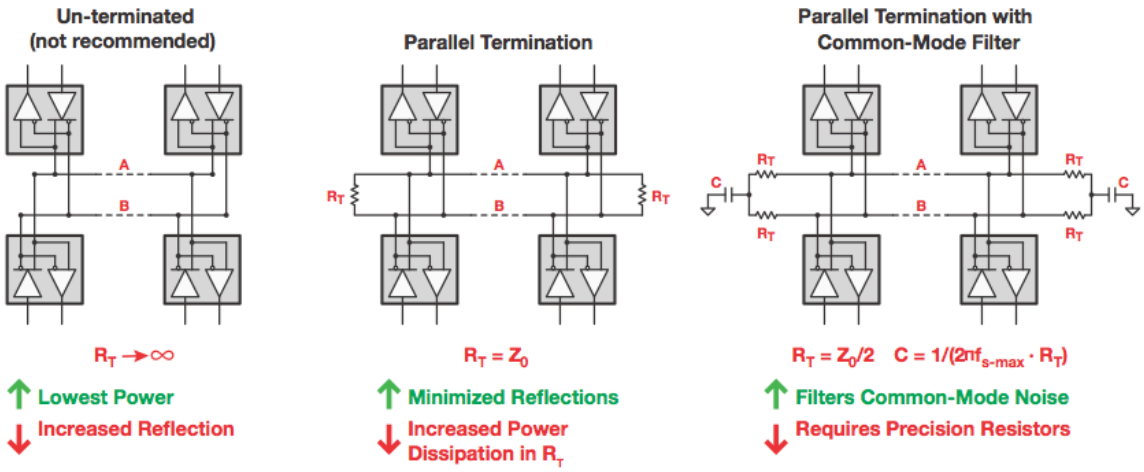


Figura 19. Terminaciones de bus RS-485

Podemos ver un ejemplo de conexionado al bus RS-485 en cable de par trenzado con topología *daisy-chain*

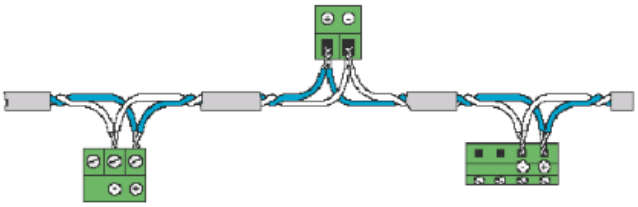


Figura 20. Conexionado físico RS-485

En la opción que hemos seleccionado para nuestro TFC, la *half-duplex*, en realidad se usan tres conductores (los dos del bus más uno "*común*") para interconectar todos los dispositivos de la red.

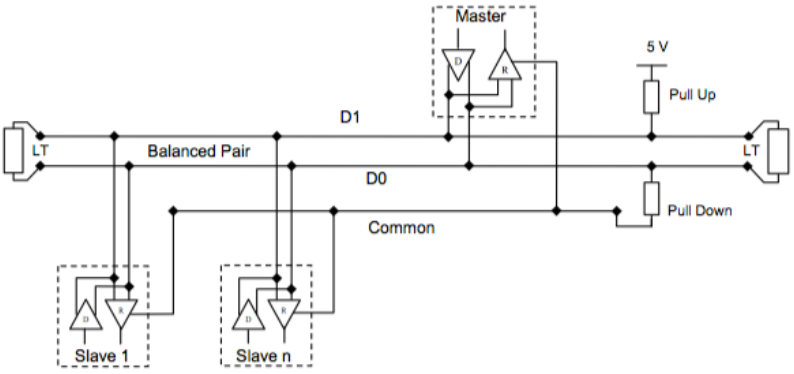


Figura 21. Esquema de conexionado solución TFC

Para minimizar los errores en el cableado, se recomienda el siguiente código de colores es para los cables RS485-MODBUS:

	Señal	*Color recomendado*
	D1 – *TXD1*	**Amarillo**
	D0 – *TXD0*	**Marrón**
	Common	**Gris**
4 cables (opcional)	*RXD0*	*Blanco*
4 cables (opcional)	*RXD1*	*Azul*

Tabla 13. Código de colores del cable RS-485

3. ADAPTACIÓN DE ENTRADAS / SALIDAS ANALÓGICAS

3.1. Entradas analógicas

3.1.1. Requisitos

Según indican las especificaciones del enunciado, el dispositivo debe contar con:

- 2 entradas analógicas para medir tensiones entre 0 y 10 V, con 10 bits de precisión.
- 2 entradas analógicas para medir bucle de corriente 4-20 mA pasivo.

3.1.2. Adaptación de entradas analógicas

El microcontrolador solamente trabaja con señales digitales por lo tanto hay que realizar una conversión A/D previa de las entradas analógicas que nos permita tratar y manejarlas de manera correcta.

ENTRADAS DE TENSIÓN:

La conversión A/D de las entradas analógicas de tensión 0-10V las haremos con el conversor A/D que trae integrado el microcontrolador. Por lo tanto, uno de los requisitos que debe tener el microcontrolador que seleccionemos es que disponga de al menos un conversor A/D de, al menos 4 canales, con una precisión mínima de 10 bits. Aunque más adelante detallaremos la elección del microcontrolador, vamos a decantarnos por uno de la marca *Microchip* de 8 bits de la familia PIC16F8XX.

La tensión de alimentación de esta familia es 4.0 a 5.5 V. La mayoría de los microcontroladores (al menos los de uso generalista) no permiten que la tensión de referencia para el muestreo A/D sea superior a la tensión de alimentación. Por lo tanto debemos reducir la tensión de la entrada para adaptarla los niveles de tensión adecuados para poder hacer un correcto uso del ADC.

Como principio de buen diseño, debemos tratar de independizar las impedancias de las entradas analógicas (por ejemplo de un sensor analógico) de las del ADC. Así pues, usaremos un seguidor de tensión a la entrada (etapa 1), un divisor de tensión (etapa 2) con ganancia 0,5 y otro seguidor de tensión a la salida del adaptador (etapa3) tal y como se muestra en el siguiente circuito:

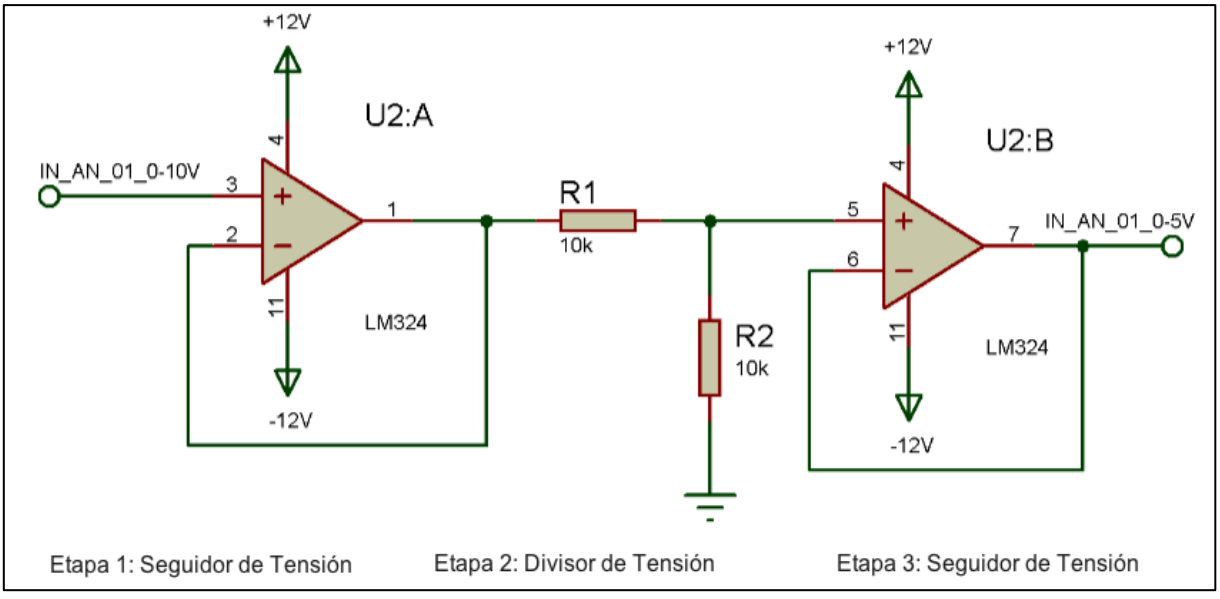


Figura 22. Adaptación de entradas analógicas de tensión 0-10 V

Vamos a realizar los cálculos para las distintas etapas adaptadoras.

Etapa 1: Seguidor de tensión

Podemos comprobar que la ganancia de esta etapa es 1, o lo que es lo mismo:

$$V_{o1} = V_i$$

Etapa 2: Divisor de tensión

Donde $R_1 = R_2$. Podemos comprobar como la ganancia es $V_{o1} = V_i \cdot \frac{R}{2R}$, o lo que es lo mismo:

$$V_{o2} = \frac{V_{o1}}{2}$$

Etapa 3: Seguidor de tensión

Podemos comprobar que la ganancia de esta etapa es 1, o lo que es lo mismo:

$$V_{o3} = V_{o2}$$

Por lo tanto:

$$\boxed{V_{o3} = \frac{V_i}{2}}$$

Simulación

Para realizar la simulación hemos usado una fuente de tensión de 10 VDC y un potenciómetro que actuará como un divisor de tensión de modo que podamos simular la tensión de la entrada entre 0 y 10 VDC.

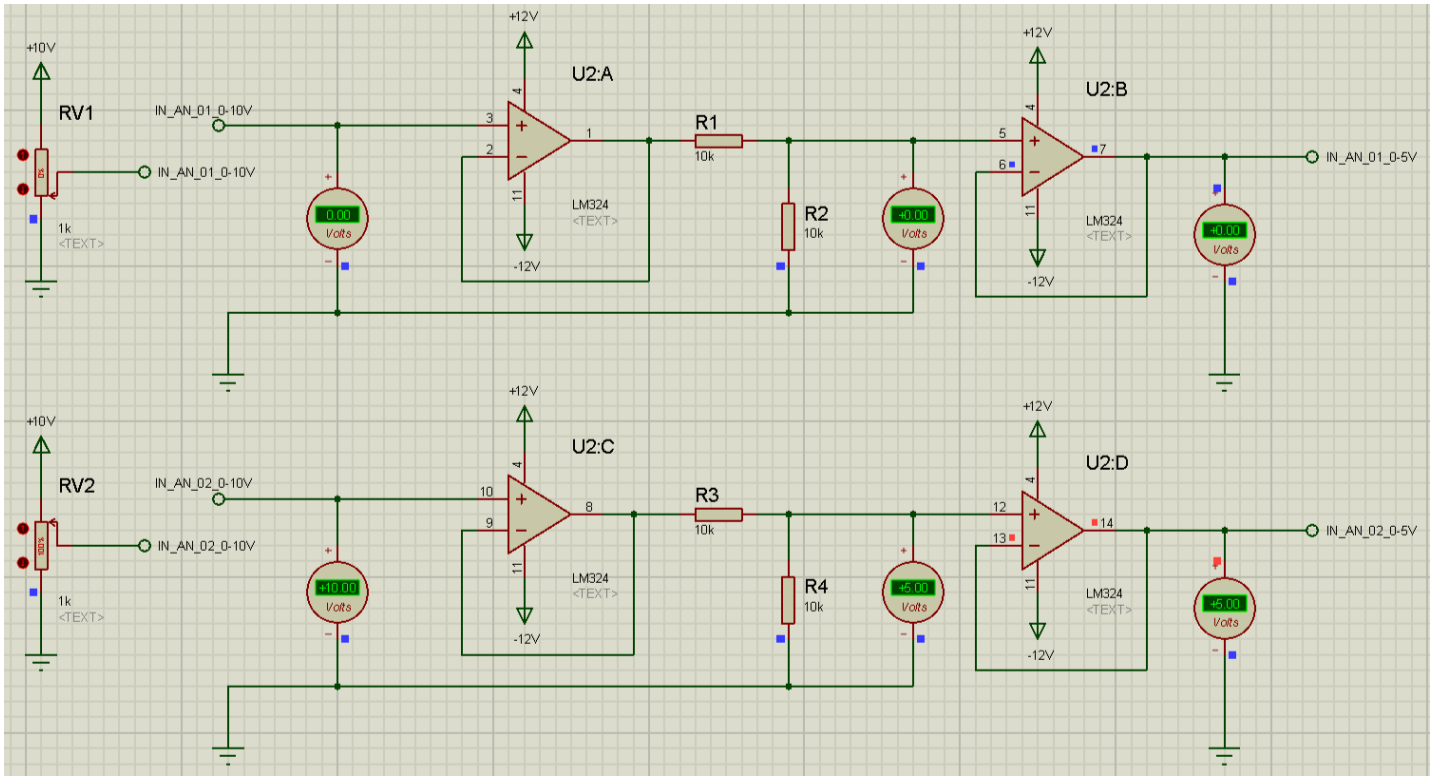


Figura 23. Simulación de entradas analógicas de tensión 0-10 V

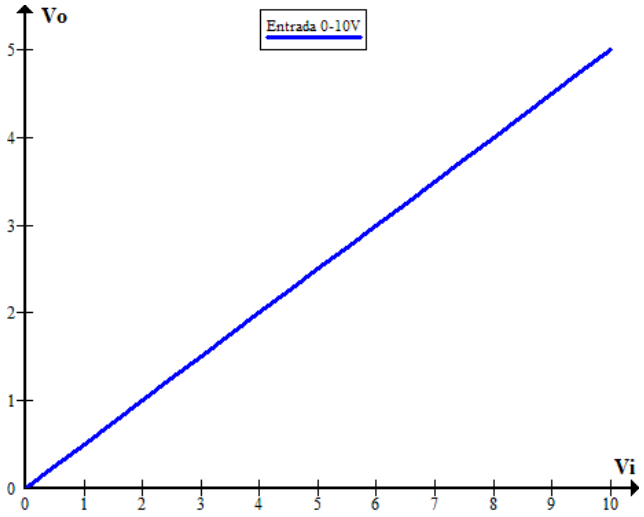


Figura 24. Relación Vo/Vi entradas analógicas de 0-10 V

ENTRADAS DE CORRIENTE:

Una comunicación entre dispositivos a 4-20 mA puede considerarse como una comunicación analógica en banda base. La frecuencia no suele superar los 10 Hz. Este sistema de comunicación a 4-20 mA punto a punto es uno de los más utilizados en plantas industriales para comunicar los transmisores y actuadores con los módulos E/S.

En el siguiente esquema puede apreciarse el funcionamiento de un lazo de corriente 4-20 mA.

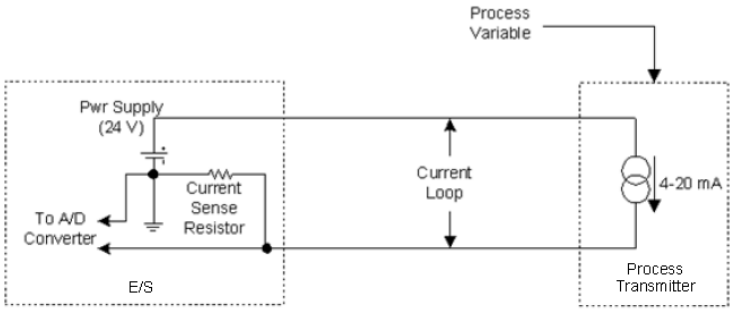


Figura 25. Esquema lazo de corriente 4-20 mA

La ventaja de los 4-20 mA frente a los 0-20 mA o los 0-5 V es que detecta el circuito abierto por rotura del lazo al leer los 0 mA, que es un valor no válido en condiciones normales.

En el mercado existen circuitos integrados que realizan la conversión directa I-V 4-20 mA a 0-5 V, como el RCV420 de Texas Instruments, pero vamos a optar por diseñar nuestro propio adaptador de transimpedancia.

Así pues, como el enunciado no lo especifica, vamos a perder precisión en la representación de estas señales de entrada (un 20%) frente a la seguridad de la detección de un posible fallo del lazo. Para adaptar el lazo de corriente que tenemos a la entrada, en primer lugar vamos a convertir esa corriente en lazo abierto en una tensión colocando una resistencia de $250\ \Omega$ y después usaremos un seguidor de tensión para aislar las impedancias.

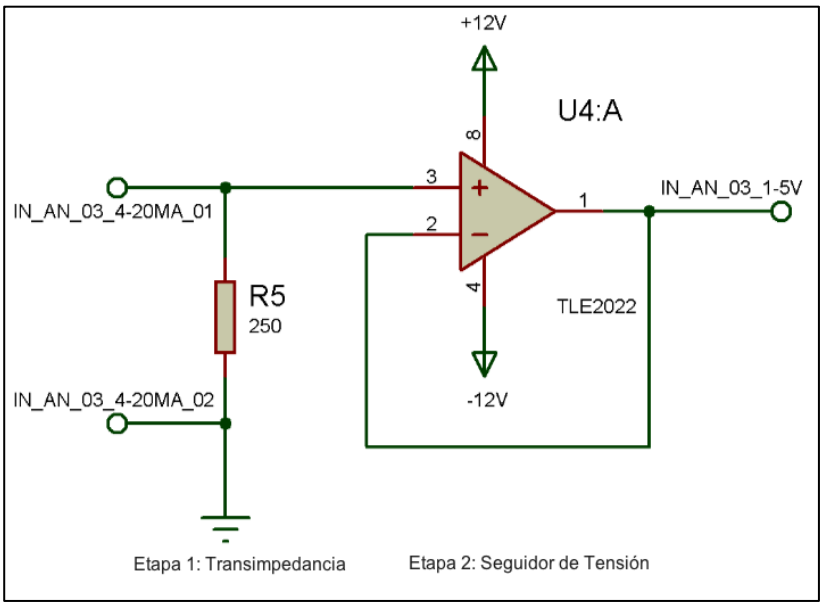


Figura 26. Adaptación de entradas analógicas de corriente 4-20 mA

Etapa 1: Transimpedancia (Conversor I-V)

Utilizando la ley de Ohm, obtenemos la relación de la tensión de salida:

$$V_{o1} = I_i \cdot R$$

Etapa 2: Seguidor de tensión

Podemos comprobar que la ganancia de esta etapa es 1, o lo que es lo mismo:

$$\boxed{V_o = I_i \cdot R}$$

Simulación

Para realizar la simulación hemos usado una fuente de corriente con un potenciómetro que actuará como un divisor de corriente de modo que podamos simular una corriente de entrada entre 4 y 20 mA.

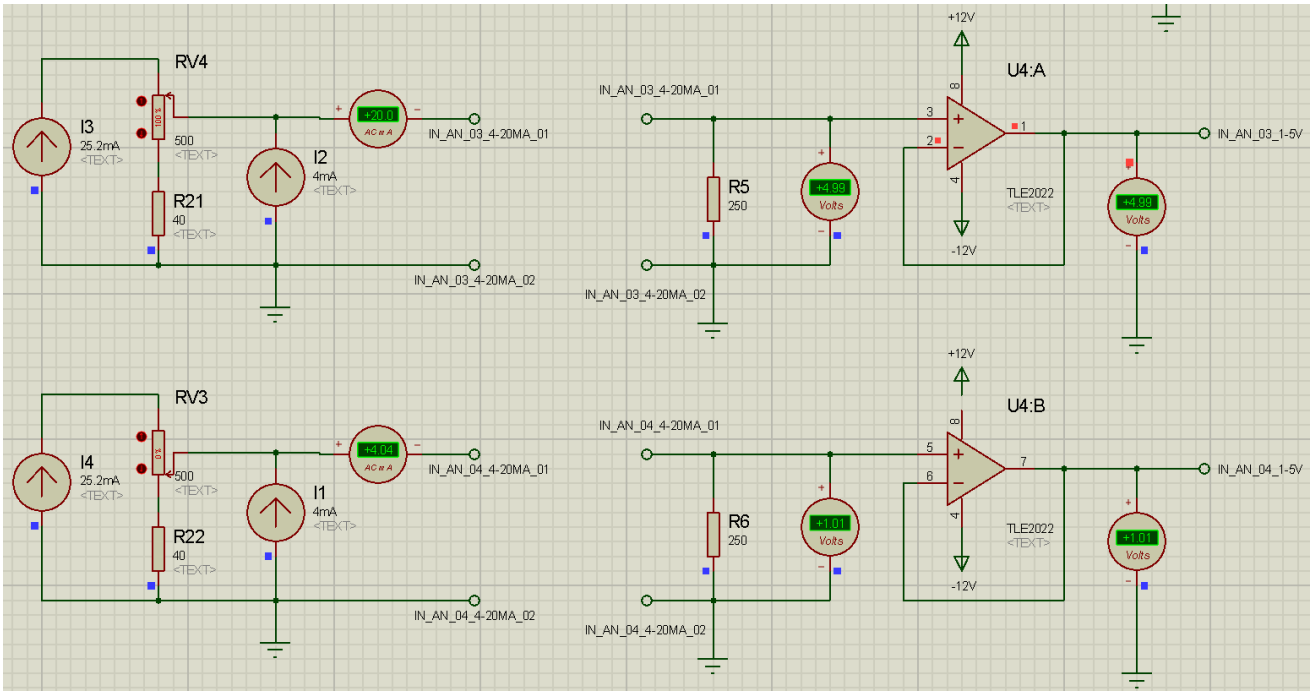


Figura 27. Simulación de entradas analógicas de corriente 4-20 mA

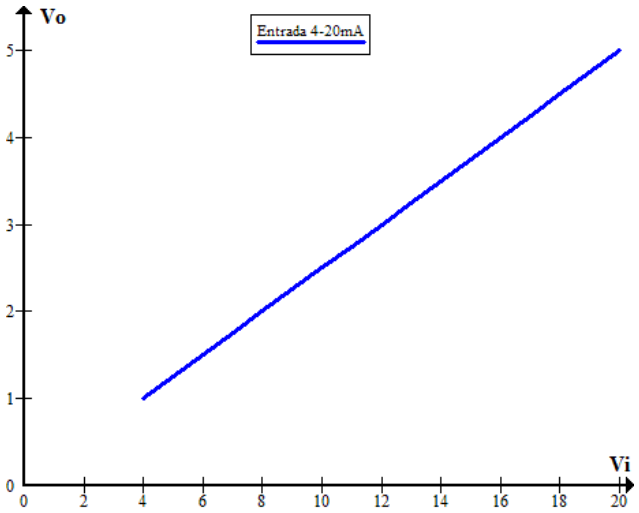


Figura 28. Relación Vo/Vi entradas analógicas 4-20 mA

3.2. Salidas analógicas

3.2.1. Requisitos

Según indican las especificaciones del enunciado, el dispositivo debe contar con:

- 2 salidas analógicas que permitan generar tensiones entre 0 y 10 V, con un mínimo de 10 bits de precisión, y con una corriente máxima de 200 mA.

3.2.2. Adaptación de salidas analógicas

El microcontrolador dispone de un conversor A/D, pero no de un D/A. El modelo seleccionado PIC16F877A, en su pin RA2 tiene conectado un generador interno de voltaje CV_{RSRC} (que se utiliza para el comparador) que podría ser usado como salida analógica (limitada a las especificaciones del generador de voltaje interno, tal y como se especifica en el capítulo 13 de la hoja de características del microcontrolador).

Sin embargo nos vamos a decantar por aplicar un filtro RC para la conversión A/D de la señal PWM y que amplificaremos posteriormente con un amplificador operacional (en configuración no inversora y ganancia 2) cuya salida sea como mínimo de 200 mA (tal y como marcan los requisitos).

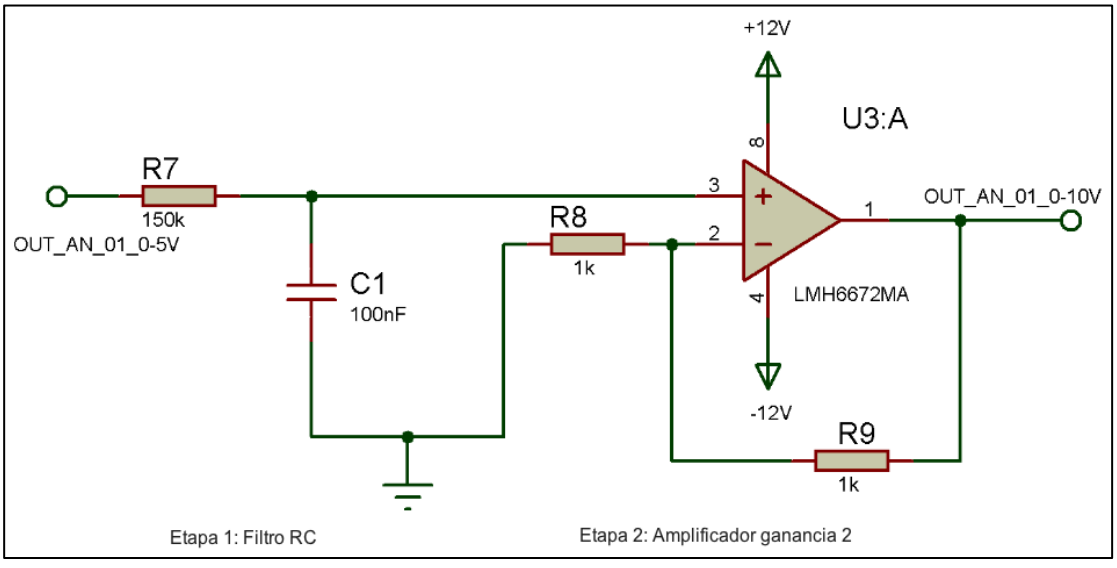


Figura 29. Adaptación de salidas analógicas de tensión 0-10 V

Mediante software activaremos cada uno de los módulos CCP en modo PWM para generar una señal para cada una de las salidas. Éstas dependerán de cuál sea el ciclo de trabajo de la señal. Vamos a configurar ambos PWMs para que trabajen a una frecuencia de 10 KHz (muy por encima de la frecuencia de corte del filtro para evitar grandes fluctuaciones en la señal generada, de modo que obtenemos una variación en este voltaje de 0,0083 V para un ciclo de trabajo del 50%)

Etapa 1: Filtro RC

Según los valores de la resistencia y el condensador del filtro RC, tendremos una frecuencia de corte:

$$f_c = \frac{1}{2\pi RC} = 10{,}61\ Hz$$

Por lo tanto dependiendo del ciclo de trabajo de la señal PWM obtendremos un voltaje a las salida del microcontrolador:

$$0 < V_{o1} < 5$$

Etapa 2: Amplificador de ganancia 2

La etapa amplificadora la vamos a realizar con el LMH6672 de *National Semiconductor*, ya que cumple con el requisito de que la corriente de salida alcanza al menos los 200 mA (tal y como puede apreciarse en la Figura 30).

Electrical Characteristics (Continued)

T_J = 25°C, G = +2, V_S = ±2.5 to ±6V, R_F = R_{IN} = 470Ω, R_L = 100Ω; Unless otherwise specified.

Symbol	Parameter	Conditions	Min (Note 6)	Typ (Note 5)	Max (Note 6)	Units
I_{SC}	Output Current (Note 3)	V_O = 0, V_S = ±6V	350	525		mA
		V_O = 0, V_S = ±6V, T_J = –40°C to 125°C	260	600		mA
Power Supply						
I_S	Supply Current/Amp	V_S = ±6V			8	mA
		V_S = ±6V, T_J = –40°C to 125°C		7.2	9	
PSRR	Power Supply Rejection Ratio	V_S = ±2.5V to ±6V, T_J = –40°C to 125°C	72	88.5		dB

Figura 30. Corriente de salida del A. O. LMH6672

La configuración no inversora será la siguiente:

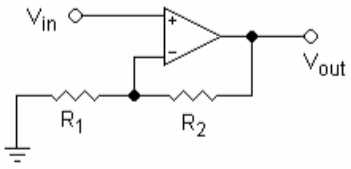


Figura 31. Configuración no inversora del A.O.

Cuya tensión de salida viene dada por: $V_o = V_i\left(1 + \frac{R_2}{R_1}\right)$. Para obtener una ganancia de 2, entonces se debe cumplir que $R_1 = R_2$

Como contamos con dos módulos PWM (CCP1 y CCP2), usaremos cada uno de ellos para las distintas señales analógicas que debemos proporcionar a las salidas.

Simulación

Para realizar la simulación hemos usado el generador de señales PWM del microcontrolador (al cual hemos cargado el programa .HEX), fijando los valores de R = 150 KΩ, C = 100 nF y la frecuencia de la señal PWM = 10 KHz.

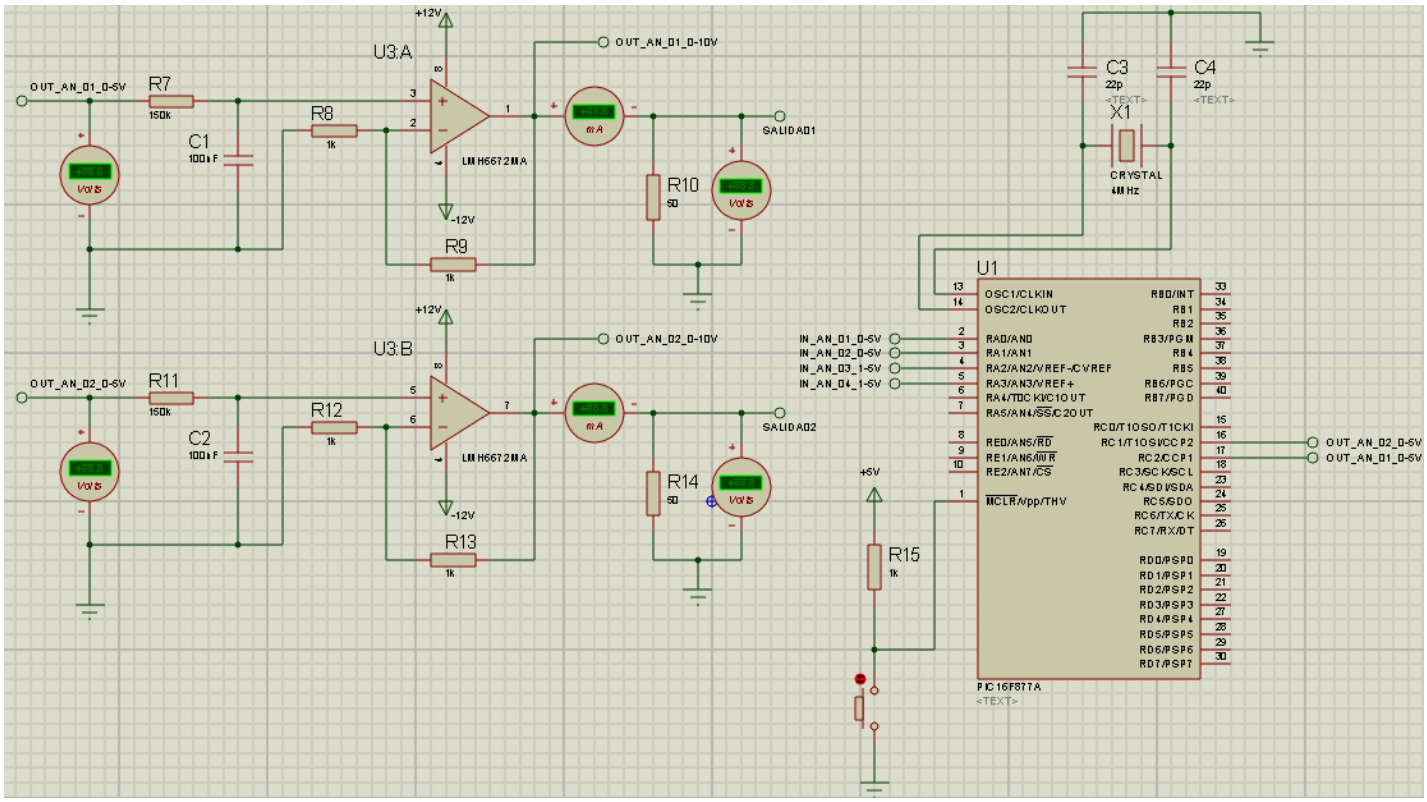


Figura 32. Simulación de salidas analógicas de tensión 0-10 V (máx. 200 mA)

Podemos ver el resultado de varias simulaciones con distinto ciclo de trabajo:

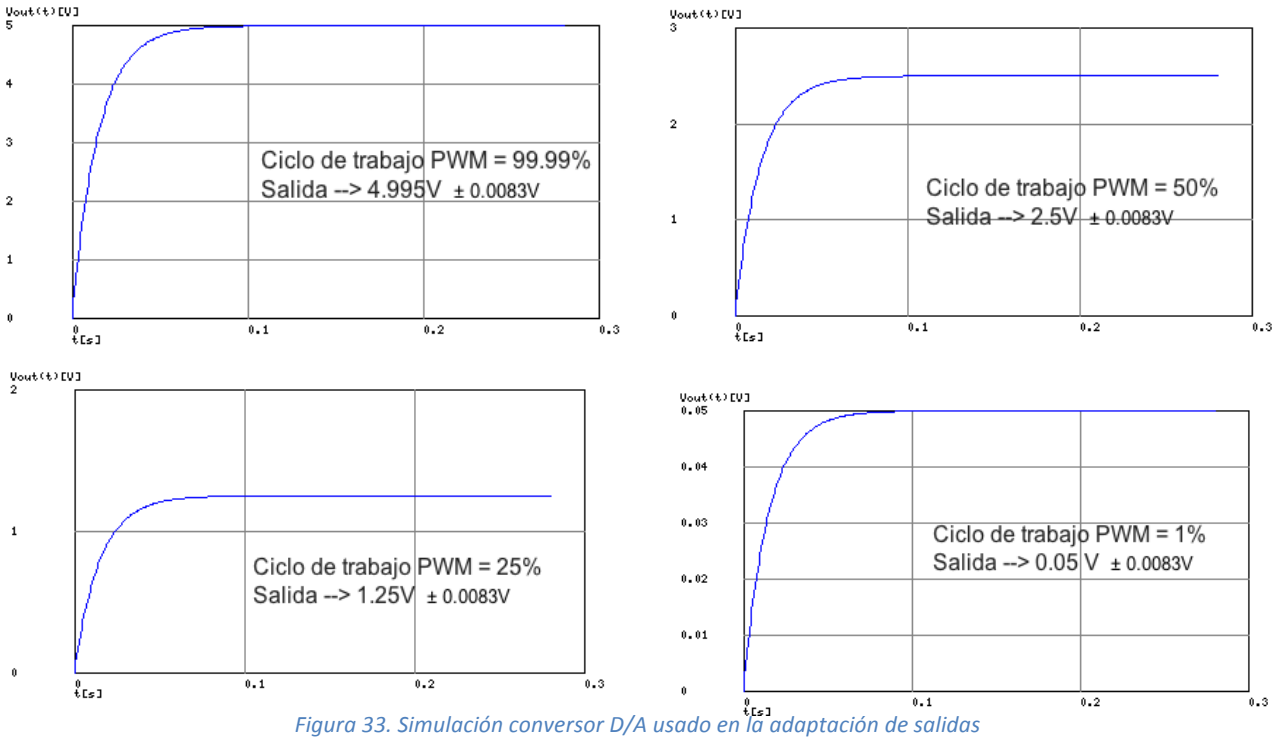


Figura 33. Simulación conversor D/A usado en la adaptación de salidas

Tras esta etapa de conversión D/A, tenemos un amplificador de ganancia 2 (con la particularidad de entregar 200 mA a la salida).

4. TRANSDUCTOR RS-485

4.1. Requisitos

Las diferentes lecturas que hagamos de las entradas analógicas o los valores que queramos colocar en las distintas salidas analógicas deben pasar por el bus de comunicaciones RS-485 y por lo tanto debemos seleccionar un transductor que adapte eléctricamente las señales procesadas por el microcontrolador al bus RS-485.

4.2. Solución

En el mercado existen multitud de transductores que pueden cumplir esta función, pero nos decantaremos por el MAX487 de *Maxim Integrated*. Este transductor es *half-duplex* y supera las especificaciones mínimas marcadas por la EIA/TIA para este estándar.

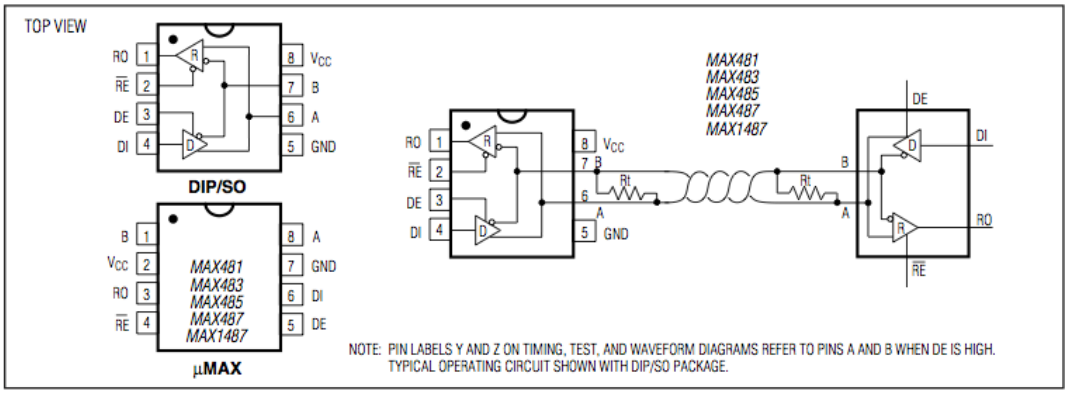


Figura 34. Transductor TTL a RS-485

Entre otros motivos, la razón de seleccionar este dispositivo es porque su tensión de alimentación es de 12 V y por lo tanto eso simplificará el futuro diseño de la fuente de alimentación del dispositivo, ya que trataremos de escoger dispositivos que tengan una misma tensión de alimentación (+5 V o ±12 V) y evitar así que diseñar una fuente conmutada con muchas salidas de tensión diferentes.

Como podemos observar en el diagrama del circuito integrado, DE y $\overline{RE}$ son señales complementarias que controlaremos mediante dos pines de salida del microcontrolador. Lo habitual es que el dispositivo siempre esté en modo recepción, activando el modo transmisión cuando proceda.

Este circuito integrado adaptará eléctricamente la salida TTL del módulo USART del PIC para una correcta interconexión con el bus RS-485.

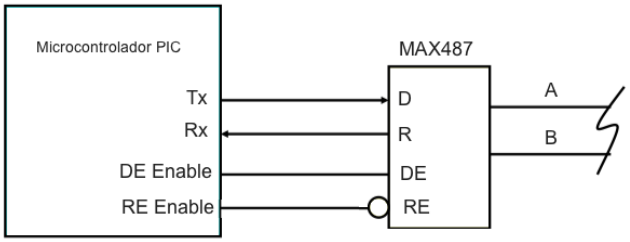


Figura 35. Conexionado Transductor RS-485 y PIC16F877A

5. PROCESADOR DE CONTROL

5.1. Requisitos

El enunciado no especifica de manera explícita ningún requisito en cuanto a la selección del microcontrolador, pero por las diferentes decisiones de diseño que hemos ido tomando necesitamos un PIC que, al menos, cuente con:

- Conversor A/D de al menos 4 canales y 10 bits de precisión.
- Puertos E/S (4 entradas analógicas y 2 salidas analógicas).
- Módulo PWM de al menos 2 canales.
- Módulo USART (para comunicación RS-485).
- Memoria ROM suficiente para cargar el programa.
- Memoria EEPROM para guardar datos de E/S en registros.
- Tensión de alimentación de 5 V (para simplificar la fuente de alimentación).

5.2. Solución adoptada

En el mercado hay infinidad de modelos de microcontroladores que pueden servir para el propósito de este TFC. Nosotros vamos a decantarnos por el modelo PIC16F887A de Microchip por diversos motivos:

- Cumple con los requisitos que necesitamos porque, aunque perdemos precisión en la tensión de salida usando el módulo PWM, es suficiente para nuestro propósito, el diseño se simplifica y el PIC es más económico.
- Existe una amplia documentación en la red acerca de este PIC que nos ayudará a conseguir el objetivo de diseño e implementación del extensor.

5.3. Características técnicas PIC16F887A

A continuación, vamos a enumerar algunas de las características básicas del PI16F887A:

- Arquitectura RISC
    - El microcontrolador cuenta con solo 35 instrucciones diferentes
    - Todas las instrucciones son uniciclo excepto las de ramificación
- Frecuencia de operación 0-20 MHz
- Oscilador interno de alta precisión
    - Calibrado de fábrica
    - Rango de frecuencia de 8MHz a 31KHz seleccionado por software
- **Voltaje de la fuente de alimentación de 2.0V a 5.5V**
    - Consumo: 220uA (2.0V, 4MHz), 11uA (2.0 V, 32 KHz) 50nA (en modo de espera)
- Ahorro de energía en el Modo de suspensión
- Brown-out Reset (BOR) con opción para controlar por software
- **35 pines de entrada/salida**
    - Alta corriente de fuente y de drenador para manejo de LED
    - Resistencias pull-up programables individualmente por software
    - Interrupción al cambiar el estado del pin
- **Memoria ROM de 8K con tecnología FLASH**
    - El chip se puede re-programar hasta 100.000 veces
- Opción de programación serial en el circuito
    - El chip se puede programar incluso incorporado en el dispositivo destino
- **256 bytes de memoria EEPROM**
    - Los datos se pueden grabar más de 1.000.000 veces

- 368 bytes de memoria RAM
- **Conversor A/D**:
  - **14 canales**
  - **resolución de 10 bits**
- 3 temporizadores/contadores independientes
- Temporizador perro guardián
- Módulo comparador analógico con
  - Dos comparadores analógicos
  - Referencia de voltaje fija (0.6V)
  - Referencia de voltaje programable en el chip
- Módulo PWM incorporado
  - 2 Canales
  - Hasta 10 bits de resolución
- **Módulo USART mejorado**
  - **Soporta las comunicaciones seriales RS-485**, RS-232 y LIN2.0
  - Auto detección de baudios
- Puerto Serie Síncrono Maestro (MSSP)
  - Soporta los modos SPI e I2C

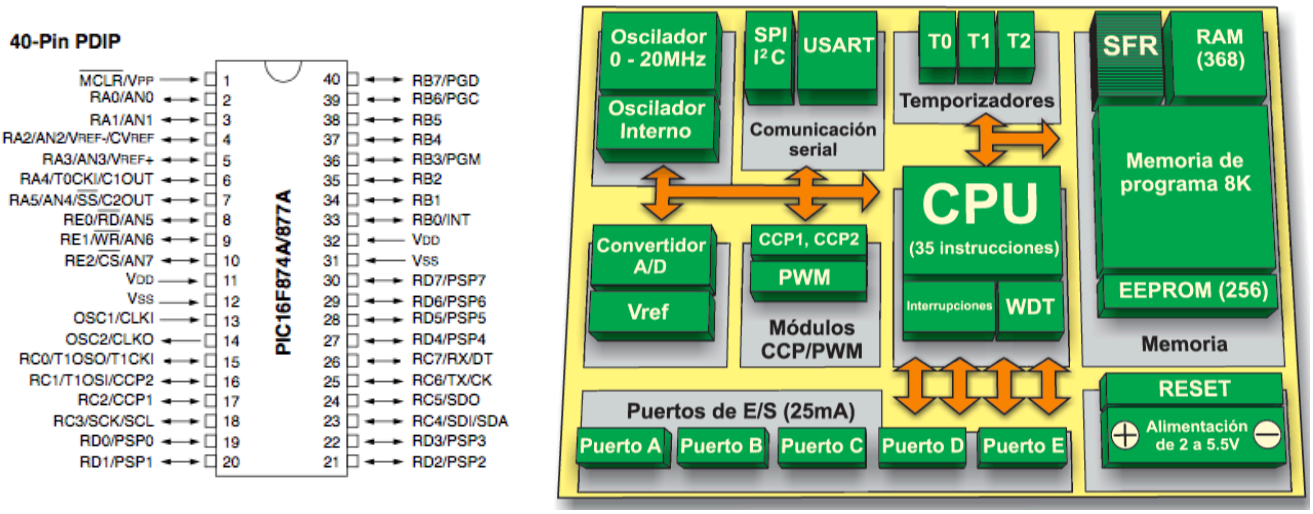


Figura 36. Encapsulado, pines y diagrama de bloques del PIC 16F877A

***Nota**: Podemos ver como la mayoría de los pines del microcontrolador son multipropósito.*

5.4. Configuración módulo ADC

En primer lugar vamos a comenzar por configurar las entradas analógicas (A) que vamos a necesitar. Según los requisitos, 4 entradas analógicas. Como ya hemos indicado el PIC trabaja con señales digitales y es por ello que vamos a aprovechar su ADC de 10 bits de precisión. Por lo tanto, en el Módulo Conversor A/D debemos configurar los siguientes registros para habilitar, al menos, 4 entradas analógicas en el puerto:

ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Por lo tanto, debemos indicar, según se indica en la figura siguiente que nuestro ADC va a tener 5 entradas analógicas y que como tensiones de referencia V_{REF+} y V_{REF-} usará V_{DD} y V_{SS} respectivamente.

PCFG3:PCFG0: A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

Tabla 14. Configuración de puertos de E/S en el ADC

***Nota**: El resto de parámetros de estos registros se configurarán a alto nivel mediante programación en C con CSS. Toda la información acerca de este módulo se encuentra disponible en el capítulo 11.0 de la hoja de características del PIC16F877A.*

5.5. Configuración del módulo PWM

El diagrama de bloques de este módulo es el siguiente:

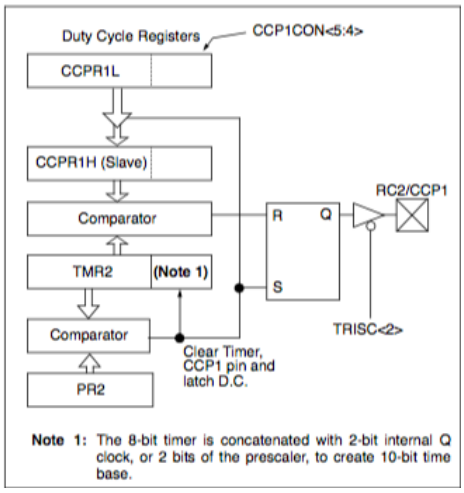


Figura 37. Diagrama de bloques del módulo PWM

Para habilitar el módulo CCPX para trabajar en modo PWM debemos configurar:

- El periodo PWM en el registro PR2.
- El ciclo de trabajo PWM.
- El pin CCPX como salida mediante el pin2 del registro TRISC.
- El temporizador TMR2 en el registro T2CON.
- Configurar el pin CCPX para trabajar en modo PWM.

***Nota**: Toda la información acerca de este módulo se encuentra disponible en el capítulo 8.3 de la hoja de características del PIC16F877A.*

5.6. Configuración del módulo USART

Vamos a usar el módulo USART para comunicación serie en modo asíncrono. Para ello, debemos configurar el módulo USART como:

Transmisor

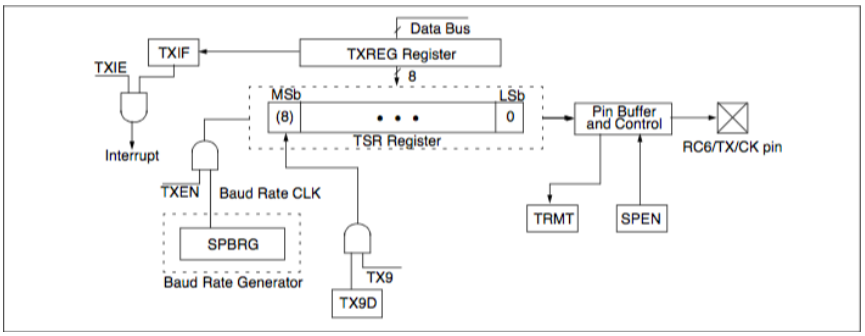


Figura 38. Diagrama de bloques del transmisor USART

Para habilitar el modo transmisor debemos configurar:

- Inicializar el registro SPBRG con los baudios apropiados para la transmisión.
- Habilitar el puerto serie asíncrono deshabilitando el bit SYNC y habilitando el bit SPEN.
- Habilitar la transmisión habilitando los bits TXEN y TXIF.
- Cargar los datos en el registro TXREG (comienza la transmisión).

Receptor

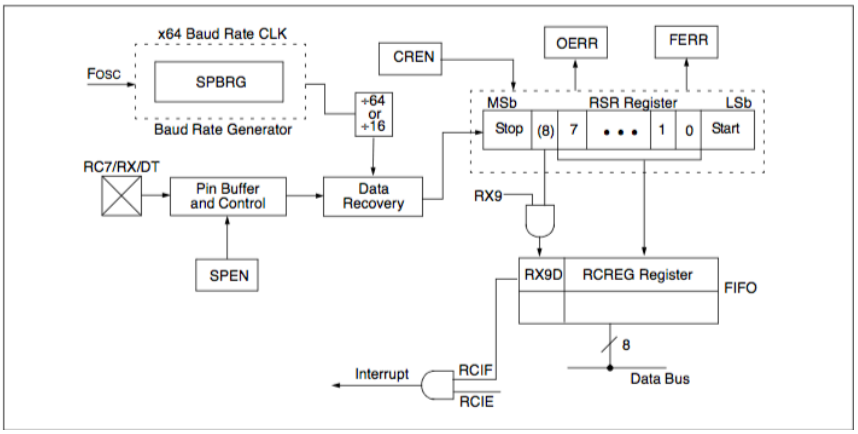


Figura 39. Diagrama de bloques del receptor USART

- Inicializar el registro SPBRG con los baudios apropiados para la recepción.
- Habilitar el puerto serie asíncrono deshabilitando el bit SYNC y habilitando el bit SPEN.
- Habilitar la recepción habilitando el bit CREN.
- El *flag* RCIF debe habilitarse cuando la recepción se complete.
- Leer el dato de 8 bits recibido en el registro RCREG.

5.7. Programa de control

Para poder programar el microcontrolador y poder implementar todas las especificaciones solicitadas en el enunciado del TFC, vamos a usar un lenguaje de alto nivel en lugar de usar ensamblador. Vamos a optar por sacrificar el tamaño que ocupa el programa frente a la sencillez que tiene el uso del lenguaje C.

El entorno de programación que he usado es el CCS de *Custom Computer Services Inc.*, ya que dispone un entorno sencillo y además cuenta con un "*wizard*" de proyectos que ayuda mucho al usuario menos experimentado, como es mi caso particular.

Además del programa principal, he incluido la librería modbus.c, que también incluye este software, pero adaptándola a nuestras especificaciones y arreglando varios errores que hacían que directamente no compilara el programa.

Se adjunta el código fuente del proyecto (.pjt, .c y .h), así como los ficheros del programa que cargaremos en el microcontrolador PIC (.hex y .cof) para realizar la simulación. No obstante, vamos a comentar algunas de las partes más importantes del código:

```
#include <Modbus-RS485_David_Olmedo.h>

// Definimos los parámetros Modbus por defecto
#define MODBUS_TYPE MODBUS_TYPE_SLAVE  // Modbus type: Slave
#define MODBUS_SERIAL_TYPE MODBUS_RTU     // Modbus serial type: RTU
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64  // Buffer size
#define MODBUS_SERIAL_BAUD 9600  // Modbus serial bauds
#define MODBUS_PARITY 0  // Parity: None
#define MODBUS_ADDRESS 0x0001  // Slave id address

#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA  // RDA Int.

#include <Modbus_Protocol.c>

// Mapa de Memoria Modbus
#define ENTRADA_ANALOGICA01 0x0000
#define ENTRADA_ANALOGICA02 0x0001
#define ENTRADA_ANALOGICA03 0x0002
#define ENTRADA_ANALOGICA04 0x0003

#define SALIDA_ANALOGICA01 0x0004
#define SALIDA_ANALOGICA02 0x0005

#define CONF_VELOCIDAD 0x1000
#define CONF_PARIDAD 0x1001
#define CONF_ID_ADDRESS 0x1002
```

Podemos ver que en esta parte del código, incluimos la librería .h correspondiente al archivo .c principal y la librería modificada que contiene el protocolo Modbus adaptado a nuestras necesidades. A continuación se definen los *parámetros* por defecto y el *mapa de memoria* Modbus.

Posteriormente se implementa en programa principal del microcontrolador, donde se configuran los puertos analógicos de entrada que vamos a utilizar, con el ACD, y donde se declaran e inicializan, si fuera necesario, las variables que necesitaremos.

```
void main()
{
   setup_adc(adc_clock_internal);  // Habilitamos el reloj interno para el ADC
   setup_adc_ports(AN0_AN1_AN2_AN3_AN4); // Configuramos los puertos ADC
   setup_comparator(NC_NC_NC_NC);
   setup_ccp1(CCP_PWM); // Configuramos el CCP1 como PWM
   setup_ccp2(CCP_PWM); // Configuramos el CCP2 como PWM

   int8 coils = 0b00000101;  // Inicializamos las bobinas (en previsión)
   int8 inputs = 0b00001001; // Inicializamos las entradas digitales - bobinas (en previsión)

//Registros de parámetros de transmisión MODBUS
   int16 modbus_params[]={9600,0,1};

//Registros de entradas y salidas
   int16 input_regs[]={0x0000,0x0000,0x0000,0x0000};
   int16 hold_regs[]={0x0000,0x0000};

   int16 event_count = 0;
   int8 cont, salida;
   int16 direccion_inicial, direccion_config;
   int16 entrada = 0, num_regs;

   modbus_init();

   while(TRUE)
   {
      while(!modbus_kbhit());

      delay_us(50);

      //check address against our address, 0 is broadcast
      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)
      {
         switch(modbus_rx.func)
```

Con la instrucción modbus_init(), ponemos en marcha el protocolo, se llama a RCV_ON(), que lo que hace es habilitar la recepción de datos. En nuestro caso, a través de una interrupción de la USART (INT_RDA) para capturar los datos que estén en el bus, tal y como puede verse en el código siguiente:

```
// Purpose:    Initialize RS485 communication. Call this before
//             using any other RS485 functions.
// Inputs:     None
// Outputs:    None
void modbus_init()
{
   output_low(MODBUS_SERIAL_ENABLE_PIN);

   RCV_ON();

   setup_timer_2(T2_DIV_BY_16,249,5);    //~4ms interrupts
   enable_interrupts(GLOBAL);
}
```

```
// Purpose:    Enable data reception
// Inputs:     None
// Outputs:    None
void RCV_ON(void)
{
   #if (MODBUS_SERIAL_INT_SOURCE!=MODBUS_INT_EXT)
      while(kbhit(MODBUS_SERIAL)) {getc();}  //Clear RX buffer. Clear RDA interrupt flag. Clear overrun error flag.
      #if (MODBUS_SERIAL_INT_SOURCE==MODBUS_INT_RDA)
        clear_interrupt(INT_RDA);
      #else
        clear_interrupt(INT_RDA2);
      #endif
```

Posteriormente, de nuevo en la función main(), se comprueba si los datos que hay en el bus tienen la dirección de *broadcast* o la de nuestro dispositivo esclavo para actuar. Si la petición no fuera dirigida a nuestro dispositivo, no se atendería. Así pues, en caso de tener que atender una petición que solicite el dispositivo maestro, comprobaríamos qué código de función está solicitando para actuar y enviar la respuesta correspondiente.

Vamos a estudiar el código de las funciones básicas que debemos implementar bajo los requisitos del enunciado del TFC:

Función 0x04 – Leer registros de entrada

```
      //check address against our address, 0 is broadcast
      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)
      {
         switch(modbus_rx.func)
         {
            case FUNC_READ_INPUT_REGISTERS:  //read input registers - function 0x04
               if(modbus_rx.data[0] || modbus_rx.data[2] ||
                  modbus_rx.data[1] >= 4 || modbus_rx.data[3]+modbus_rx.data[1] > 4)
                  modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
               else
               {
                  for (cont=0; cont<4; cont++) // Se leen todas las entradas
                  {
                     set_adc_channel(cont);
                     delay_ms(6);
                     entrada = read_adc();
                     int8 entradaMSB = make8(entrada,1);  //Bits más altos
                     int8 entradaLSB = make8(entrada,0);  //Bits más bajos
                     entrada=make16(entradaLSB,entradaMSB); // Se intercambian los Bytes
                     input_regs[cont]= entrada;
                  }
                  //sólo se mandan al maestro las que ha solicitado
                  modbus_read_input_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2),input_regs+modbus_rx.data[1]);

                  event_count++;
               }
               break;
```

Comprobamos que la dirección es correcta, si no lo fuera se lanza una respuesta con el error ILEGAL_DATA_ADDRESS. Si es una dirección válida, activamos en ACD de manera secuencial para las cuatro entradas de modo que vamos guardando el valor en un array, input_regs[ ], que almacena datos de tipo Word. Tras realizar varias simulaciones, hemos detectado que antes de lanzar la respuesta, los datos debían intercambiar sus bytes alto y bajo para que el sistema funcione correctamente.

Una vez que está todo listo para lanzar la respuesta, se llama a la función modbus_read_input_registers_rsp(), que lo único que hace es preparar la trama para lanzarla byte a byte por el bus:

```
read_input_registers_rsp
Input:     int8       address           Slave Address
           int8       byte_count        Number of bytes being sent
           int8*      input_data        Pointer to an array of data to send
Output:    void
*/
void modbus_read_input_registers_rsp(int8 address, int8 byte_count,
                                     int8 *input_data)
{
   int8 i;

   modbus_serial_send_start(address, FUNC_READ_INPUT_REGISTERS);

   modbus_serial_putc(byte_count);

   for(i=0; i < byte_count; ++i)
   {
      modbus_serial_putc(*input_data);
      input_data++;
   }

   modbus_serial_send_stop();
}
```

Función 0x06 – Escribir en un registro de salida

```
   case FUNC_WRITE_SINGLE_REGISTER:  //write single register - function 0x06

      if(modbus_rx.data[0] || modbus_rx.data[1] >= 8)
         modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
      else
      {
         //the registers are stored in big endian format
         if (make16(modbus_rx.data[0],modbus_rx.data[1])== SALIDA_ANALOGICA01)
         {
            hold_regs[0] = make16(modbus_rx.data[2],modbus_rx.data[3]);
            salida = adapta_PWM_timer2(hold_regs[0]); // porque el timer2 tiene definidos 256 niveles
            set_pwm1_duty(salida);  // Salida 01
            delay_ms(10);
            modbus_write_single_register_rsp(MODBUS_ADDRESS,
                      make16(modbus_rx.data[0],modbus_rx.data[1]),
                      make16(modbus_rx.data[2],modbus_rx.data[3]));

            event_count++;

         }else if (make16(modbus_rx.data[0],modbus_rx.data[1])== SALIDA_ANALOGICA02)
         {
            hold_regs[1] = make16(modbus_rx.data[2],modbus_rx.data[3]);
            salida = adapta_PWM_timer2(hold_regs[1]); // porque el timer2 tiene definidos 256 niveles
            set_pwm2_duty(salida);  // Salida 02
            delay_ms(10);
            modbus_write_single_register_rsp(MODBUS_ADDRESS,
                      make16(modbus_rx.data[0],modbus_rx.data[1]),
                      make16(modbus_rx.data[2],modbus_rx.data[3]));

            event_count++;
         }else
         {
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
         }

      }
      break;
```

Al igual que en la función anterior, si se produce una petición a una dirección ilegal, se lanza una respuesta con el correspondiente error. Si la petición es correcta:

1. se guarda en un registro temporal el valor que debe contener la salida
2. se adapta la salida con ciclo de trabajo del timer2 (256 niveles) llamando a la función adapta_PWM_timer2()

```
/*This function adapts "salida" 1024 to 256 levels according to timer2 configuration */
int8 adapta_PWM_timer2(int16 salida)
{
   int8 salidaPWM;
   return salidaPWM=(salida/4);
}
```

3. Dependiendo de la dirección de la salida:
   a. Se activa el módulo PWM1 o PWM2
   b. Si la salida no existe (no es la 1 o la 2), se genera una respuesta de error ILLEGAL_DATA_VALUE.
4. Si todo es correcto, se hace una llamada a la función modbus_write_single_register_rsp() que va montando la trama de respuesta para lanzarla byte a byte al bus, devolviendo un error si se produjera.

```
1241  write_single_register_rsp
1242  Input:     int8       address            Slave Address
1243             int16      reg_address        Echo of register address received
1244             int16      reg_value          Echo of register value received
1245  Output:    void
1246  */
1247  void modbus_write_single_register_rsp(int8 address, int16 reg_address,
1248                                        int16 reg_value)
1249  {
1250     modbus_serial_send_start(address, FUNC_WRITE_SINGLE_REGISTER);
1251
1252     modbus_serial_putc(make8(reg_address,1));
1253     modbus_serial_putc(make8(reg_address,0));
1254
1255     modbus_serial_putc(make8(reg_value,1));
1256     modbus_serial_putc(make8(reg_value,0));
1257
1258     modbus_serial_send_stop();
1259  }
```

Función 0x10 – Escribir en varios registros

```
239  case FUNC_WRITE_MULTIPLE_REGISTERS:   // write multiple registers - function 0x10
240       num_regs=make16(modbus_rx.data[2],modbus_rx.data[3]);
241       switch(num_regs)
242       {
243          case 2: // Escribir en las 2 salidas analógicas
244          {
245             direccion_inicial=make16(modbus_rx.data[0],modbus_rx.data[1]);
246             if(direccion_inicial==SALIDA_ANALOGICA01)  // Escribimos en las 2 salidas
247             {
248                 hold_regs[0] = make16(modbus_rx.data[6],modbus_rx.data[7]);
249                 salida = adapta_PWM_timer2(hold_regs[0]); // porque el timer2 tiene definidos 256 niveles
250                 set_pwm1_duty(salida);  // Salida 01
251                 delay_ms(10);
252
253                 hold_regs[1] = make16(modbus_rx.data[8],modbus_rx.data[9]);
254                 salida = adapta_PWM_timer2(hold_regs[1]); // porque el timer2 tiene definidos 256 niveles
255                 set_pwm2_duty(salida);  // Salida 02
256                 delay_ms(10);
257
258                 modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
259                           make16(modbus_rx.data[0],modbus_rx.data[1]),
260                           make16(modbus_rx.data[2],modbus_rx.data[3]));
261                 event_count++;
262
263             }else
264             {
265                 modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
266             }
267          }
268          break;
```

Aquí asumiremos que si la petición del dispositivo maestro quiere escribir 2 registros será para escribir sobre las 2 salidas que tenemos, activamos el módulo PWM1 y PWM2 con el valor correspondiente de manera secuencial.

También asumiremos que si la petición del dispositivo maestro quiere escribir 3registros será para escribir sobre los registros de configuración Modbus, tal y como se pude observar a continuación:

```
270          case 3: //Escribimos los registros de configuración modbus
271          {
272             direccion_config=make16(modbus_rx.data[0],modbus_rx.data[1]);
273             if(direccion_config==CONF_VELOCIDAD)  // Comprobamos que es la dirección correcta
274             {
275                if (modbus_rx.address == 0) // Multicast --> No hay respuesta
276                {
277                   modbus_params[0]=make16(modbus_rx.data[6],modbus_rx.data[7]);
278                   modbus_params[1]=make16(modbus_rx.data[8],modbus_rx.data[9]);
279                }else
280                {
281                   modbus_params[0]=make16(modbus_rx.data[6],modbus_rx.data[7]);
282                   modbus_params[1]=make16(modbus_rx.data[8],modbus_rx.data[9]);
283                   modbus_params[2]=make16(modbus_rx.data[10],modbus_rx.data[11]);
284
285                   // respuesta funcion 0x10
286                   modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
287                            make16(modbus_rx.data[0],modbus_rx.data[1]),
288                            make16(modbus_rx.data[2],modbus_rx.data[3]));
289                   event_count++;
290                }
291             }else if (make16(modbus_rx.data[0],modbus_rx.data[1])<0x1000)
292             {
293                int i,j;
294
295                for(i=0,j=5; i < modbus_rx.data[4]/2; ++i,j+=2)
296                   hold_regs[i] = make16(modbus_rx.data[j+1],modbus_rx.data[j]);
297
298                modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
299                               make16(modbus_rx.data[0],modbus_rx.data[1]),
300                               make16(modbus_rx.data[2],modbus_rx.data[3]));
301                event_count++;
302
303             }else
304             {
305                modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
306             }
307          }
308          break;
```

Podemos observar que si es una petición *multicast*, no se envía ninguna respuesta por nuestro dispositivo esclavo.

Por último, hemos definido el resto de los casos, en los cuales se hace una comprobación de que los datos y las direcciones son correctas y se actúa en consecuencia.

En todas los casos, cuando sea necesario emitir una respuesta, se llama a la función modbus_write_multiple_registers_rsp() que va montando la trama de respuesta para lanzarla byte a byte al bus:

```
1373 write_multiple_registers_rsp
1374 Input:     int8       address            Slave Address
1375            int16      start_address      Echo of address to start at
1376            int16      quantity           Echo of amount of registers written to
1377 Output:    void
1378 */
1379 void modbus_write_multiple_registers_rsp(int8 address, int16 start_address,
1380                                          int16 quantity)
1381 {
1382    modbus_serial_send_start(address, FUNC_WRITE_MULTIPLE_REGISTERS);
1383
1384    modbus_serial_putc(make8(start_address,1));
1385    modbus_serial_putc(make8(start_address,0));
1386
1387    modbus_serial_putc(make8(quantity,1));
1388    modbus_serial_putc(make8(quantity,0));
1389
1390    modbus_serial_send_stop();
1391 }
```

5.8. Simulación peticiones / respuestas

Para la simulación del programa y comprobar que el protocolo MODBUS está correctamente implementado usaremos la aplicación *Real Pic Simulator* de *Digital ElectroSoft*. En la simulación podemos ver que RC6 corresponde al pin de recepción de la UART y RB4 al pin RX Enable.

Función 0x04 – Leer registros de entrada

Petición: 01 04 0000 0004 F1C9
Respuesta: 01 04 08 0000 0200 0300 03FF 651B

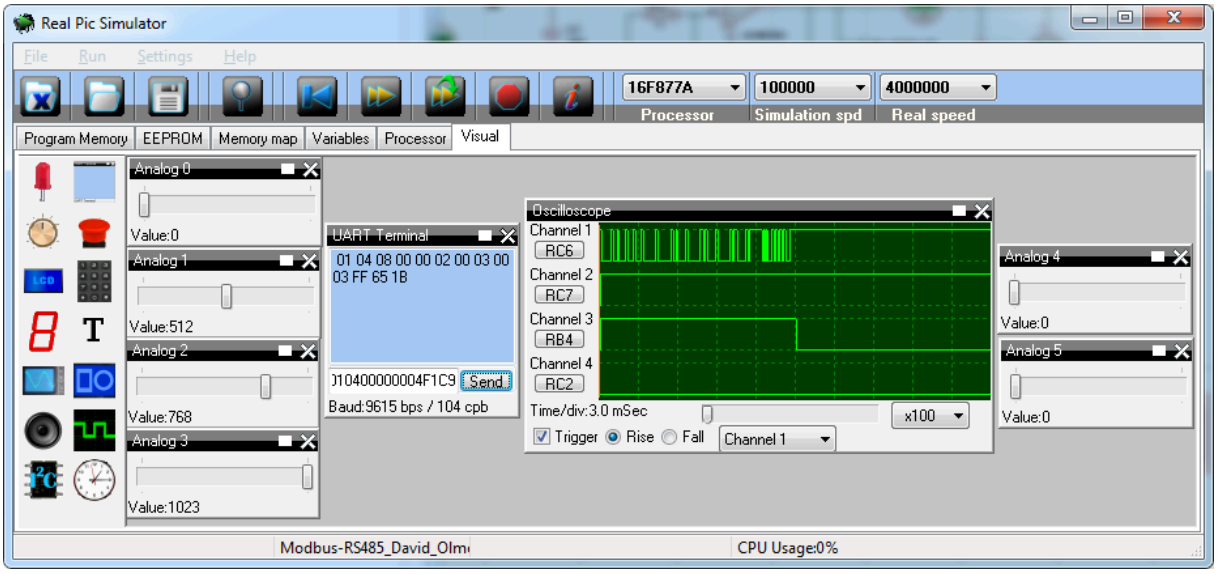


Figura 40. Simulación función 04 MODBUS

Tal y como puede observarse en la figura, la entrada 01 vale 0x0000 – 0d, la entrada 02 vale 0x0200 – 512d, la entrada 03 vale 0x0300 – 768d y la entrada 04 vale 0x03FF – 1024d, tal y como se indica en la respuesta que obtenemos a través de la UART.

Función 0x06 – Escribir en un registro de salida

Petición: 01 06 0004 0300 C8FB
Respuesta: 01 06 0004 0300 C8FB (La salida debe valer 0x0300 – 768d)

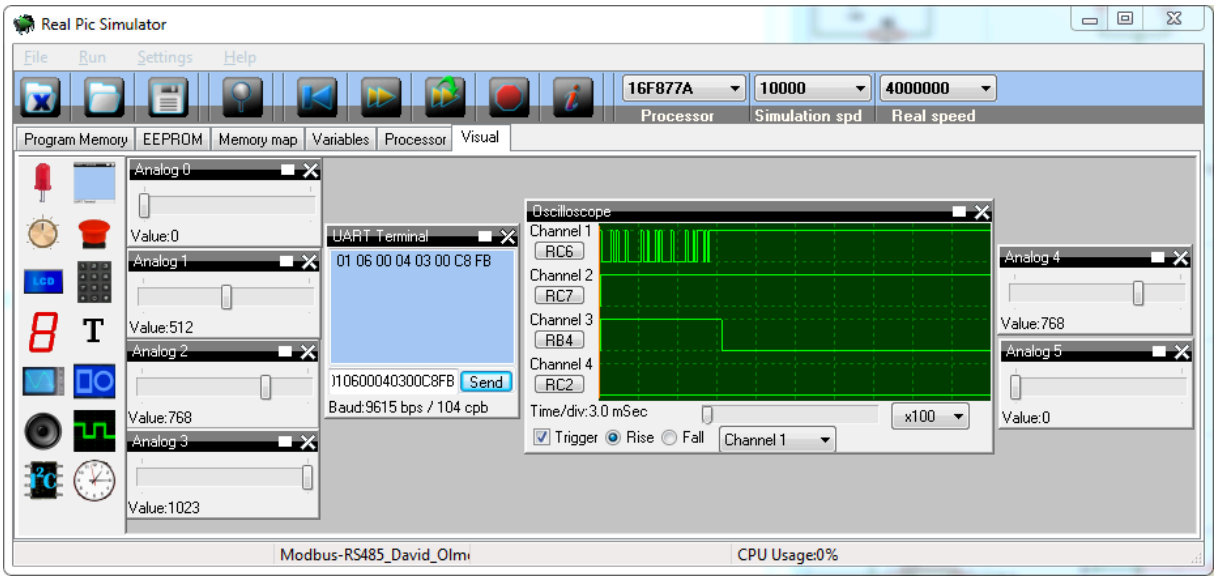


Figura 41. Simulación función 06 MODBUS

Función 0x10 – Escribir en varios registros de salida

Petición: 01 10 0004 0002 0004 0200 0133 6DFC
Respuesta: 01 10 0004 0002 0009 (Las salidas deben valer 0x0200 y 0x0133)

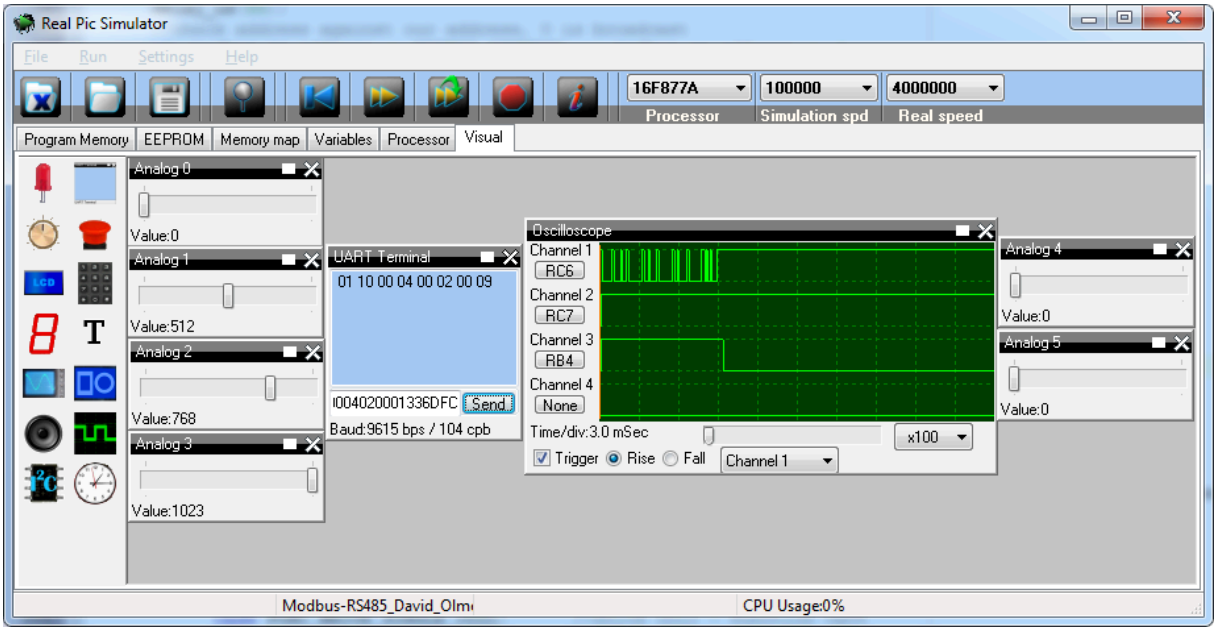


Figura 42. Simulación función 16 MODBUS

También hemos simulado 3 peticiones y sus respuestas, una tras otra:

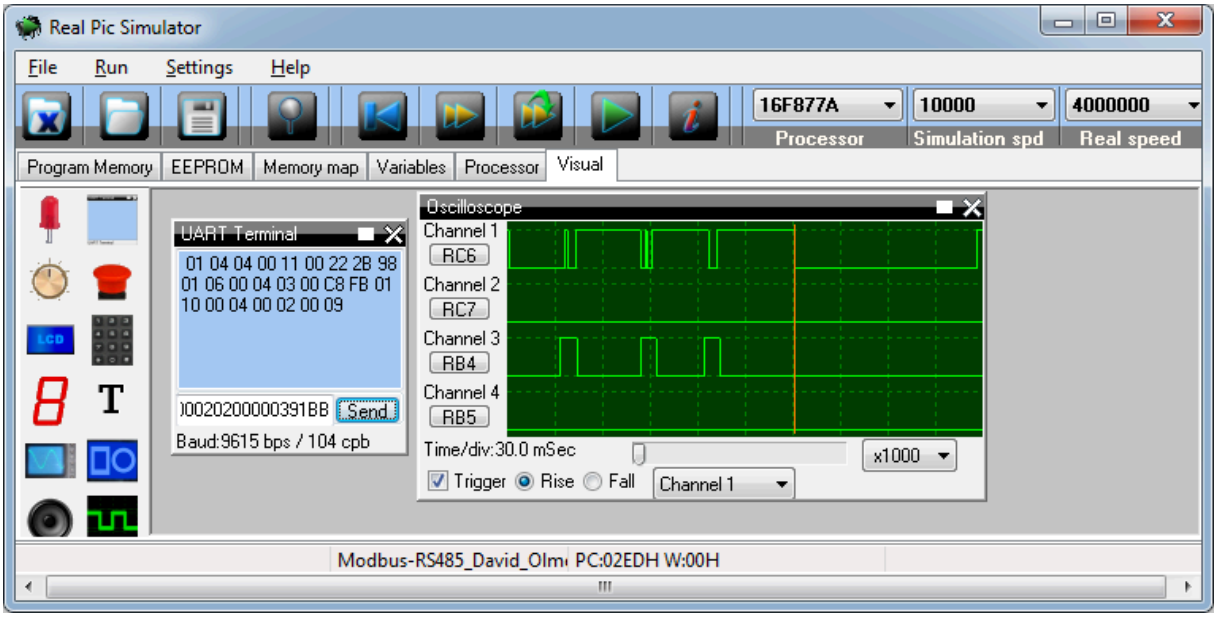


Figura 43. Simulación 3 peticiones y 3 respuestas MODBUS

Podemos ver que las tres recepciones y transmisiones se realizan de manera correcta, obteniendo por la UART terminal las respuestas que esperábamos para las peticiones realizadas a través de la misma terminal UART.

6. FUENTE DE ALIMENTACIÓN

6.1. Requisitos

Se debe diseñar una fuente de alimentación conmutada que soporte tensiones de entrada en corriente continua (DC) entre 12 y 48 V.

La tensión de entrada se usará para alimentar los distintos componentes de los bloques funcionales que hemos diseñado. Nuestras necesidades serán:

- +5 V DC para alimentar el microcontrolador PIC16F877A
- ±12 V DC para alimentar los amplificadores operacionales de los distintos bloques.

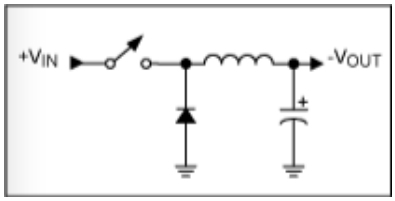
6.2. Solución

Para el diseño de la fuente, usaremos un conversor DC-DC. Este tipo de fuentes, basados en estos conversores DC-DC son más eficientes que las fuentes lineales tradicionales. Aunque pueden generar y propagar ruido a través de la línea de entrada al resto del sistema.

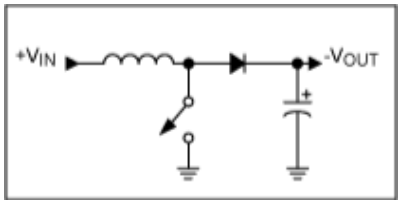
Un conversor DC-DC es un circuito que cuenta con un interruptor de potencia, un inductor y un diodo para transferir la energía de la entrada a la salida. Opcionalmente se pueden usar también condensadores.

Existen diferentes configuraciones:

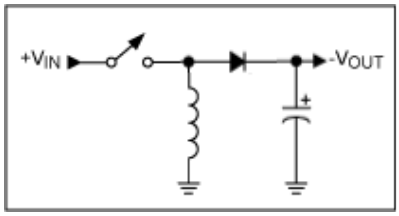
- Reductores
  - Convertidor Buck



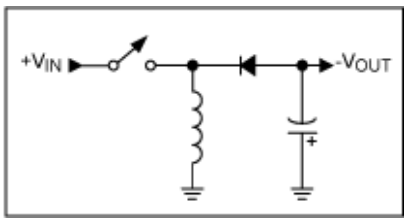
- Elevadores
  - Convertidor Boost



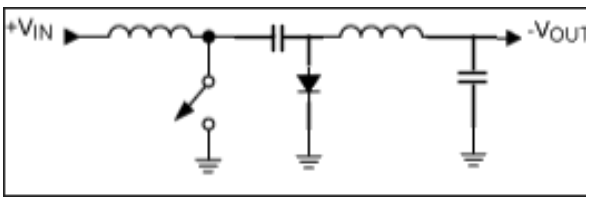
- Reductores-Elevadores
  - Convertidor Buck-Boost



Convertidor Flyback



Convertidor Cuk



En nuestro caso vamos a usar un **convertidor reductor** (*buck*), ya que la tensión de salida siempre será menor o igual que la tensión de entrada (independientemente de cual sea ésta).

Para diseñar la fuente, usaremos la herramienta que proporciona Texas Instruments, WEBENCH Power Architect:

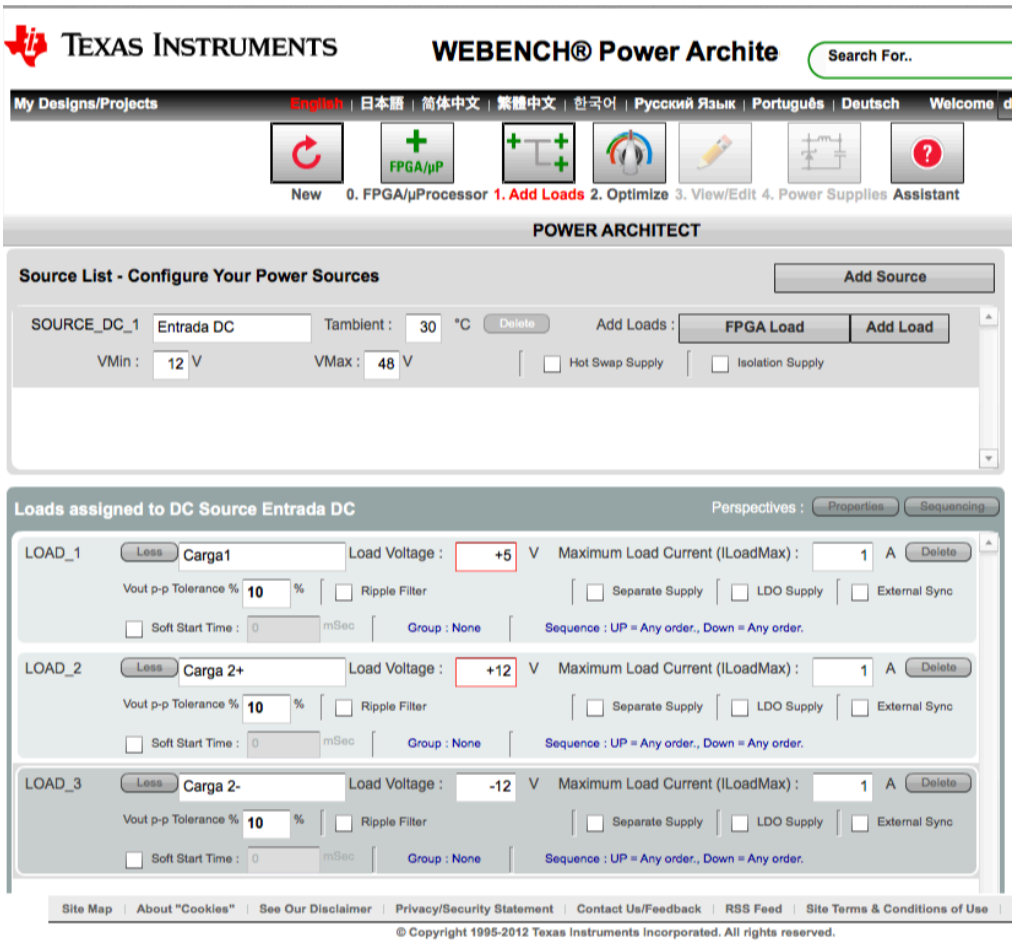


Figura 44. Entrada de datos WEBENCH

Hemos decidido que la fuente debe entregar como máximo 1A* por cada una de las cargas, ya que en el caso peor, los dispositivos pueden llegar a consumir**:

- Fuente de +5V: La usaremos para alimentar el microntrolador PIC16F877A
  - Corriente máxima de salida por Vss: 300 mA
  - Corriente máxima por cada pin E/S: 25 mA
  - Corriente máxima de los puertos combinados: 200 mA
- Fuentes de ±12V: Las usaremos para alimentar simétricamente los amplificadores operacionales de las distintas etapas de E/S y el transductor RS-485.
  - Corriente máxima que puede consumir el LM324: 243 mA
  - Corriente máxima que puede consumir el TLE2022: ±140 mA
  - Corriente máxima que puede consumir el LMH6672: 409 mA
  - Corriente máxima que puede consumir el MAX487: 251 mA

Nota 1: *Como puede observarse, la fuente está sobredimensionada en previsión a futuras ampliaciones o mejoras en el dispositivo.*

***Nota 2**: *Se ha calculado el consumo mediante la suma de corrientes (salida, alimentación, etc).*

La herramienta nos ofrece varias opciones, pero nos vamos a decantar por aquélla que es más eficiente:

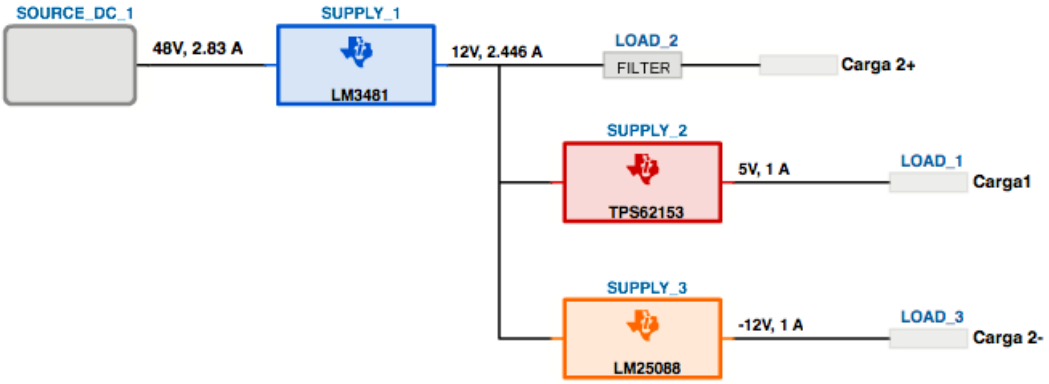


Figura 45. Diagrama de bloques fuente de alimentación

Podemos obtener los esquemas eléctricos y la posibilidad de exportarlos a varios programas CAD, como puede ser Cadence OrCAD Capture CIS, así como el listado de componentes, y la simulación de cada una de las etapas o cargas.

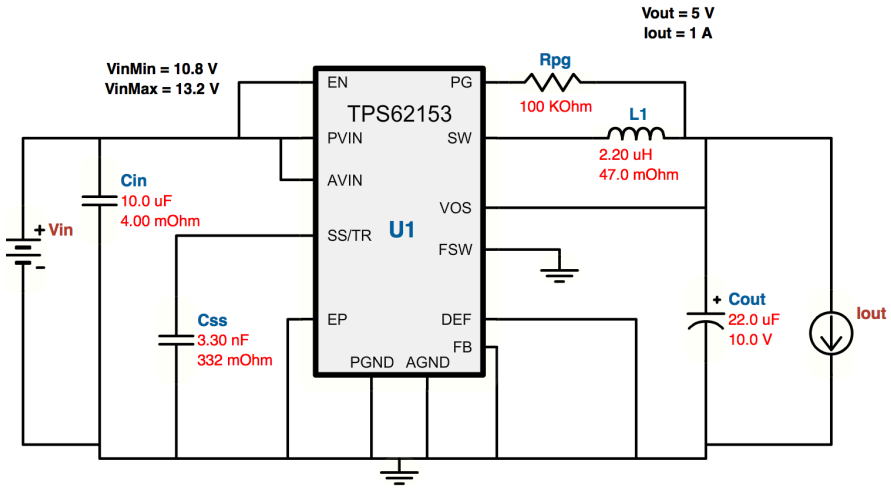


Figura 46. Esquema eléctrico fuente +5 V

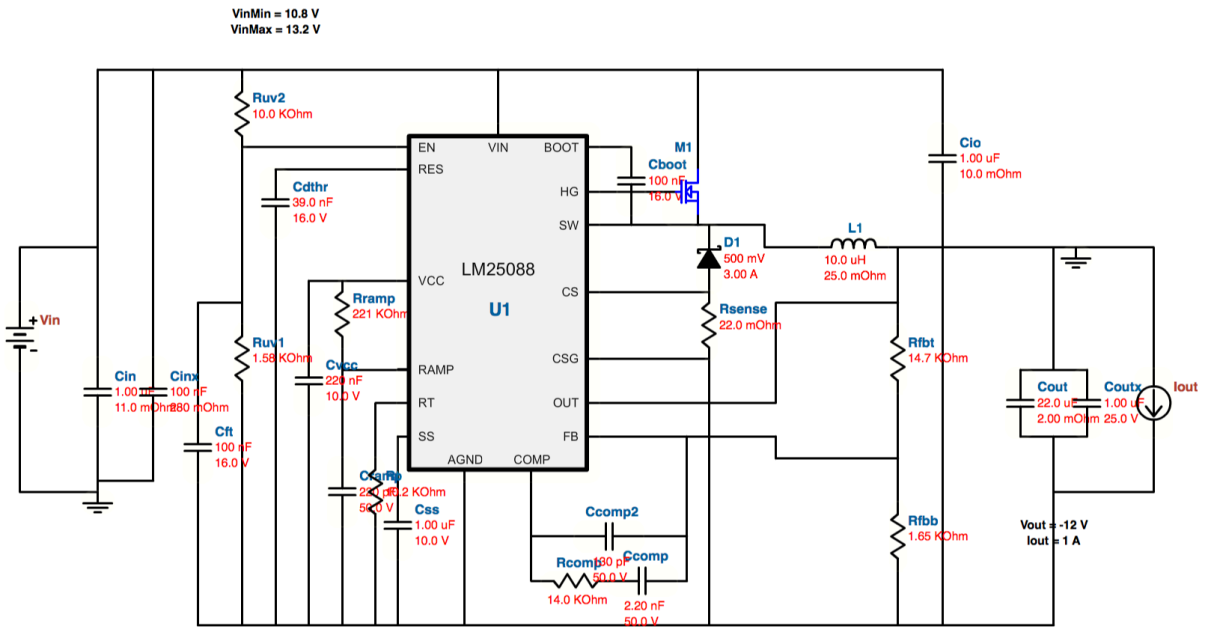


Figura 47. Esquema eléctrico fuente -12 V

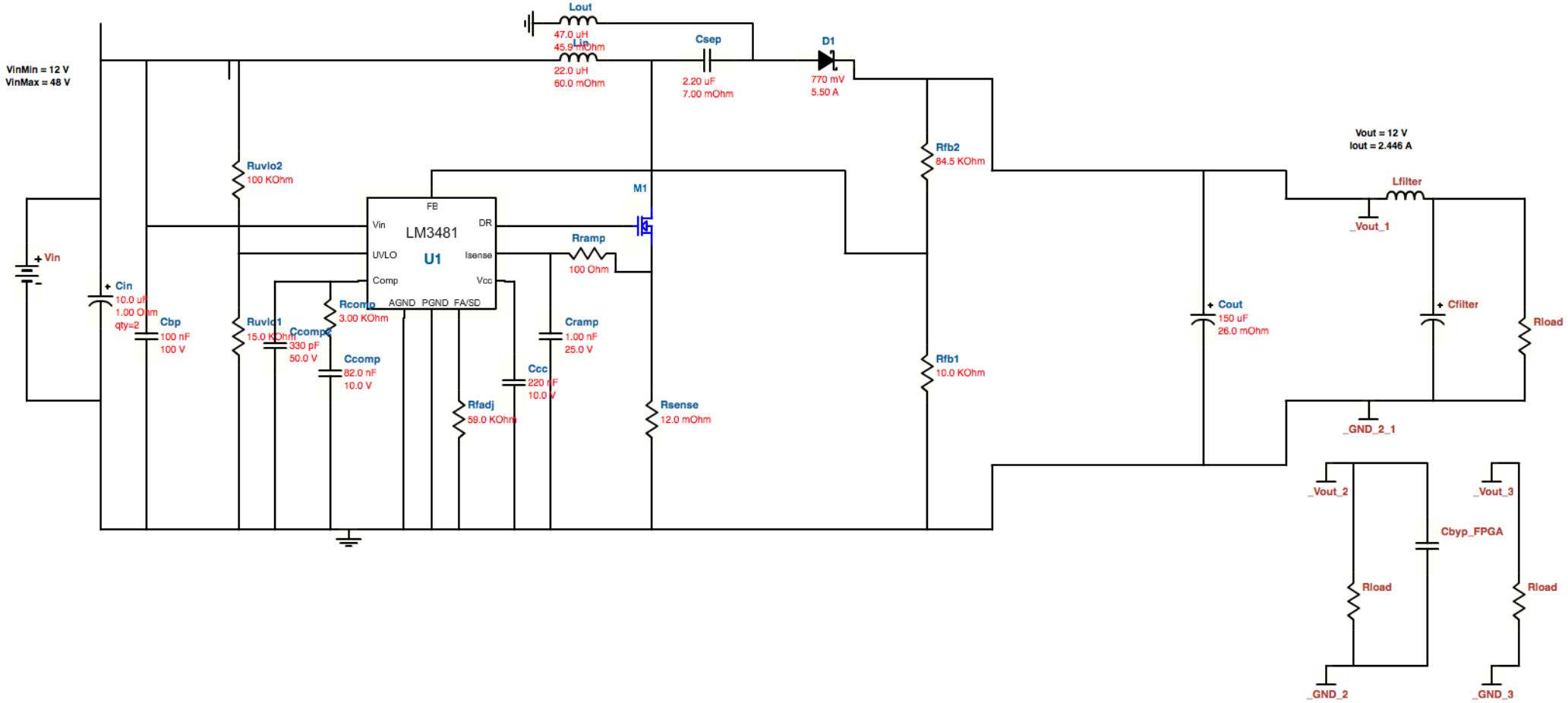


Figura 48. Esquema eléctrico fuente +12 V

7. CIRCUITO ELÉCTRICO

7.1. Esquema eléctrico

El circuito eléctrico del dispositivo, sin incluir la fuente es el que se adjunta a continuación:

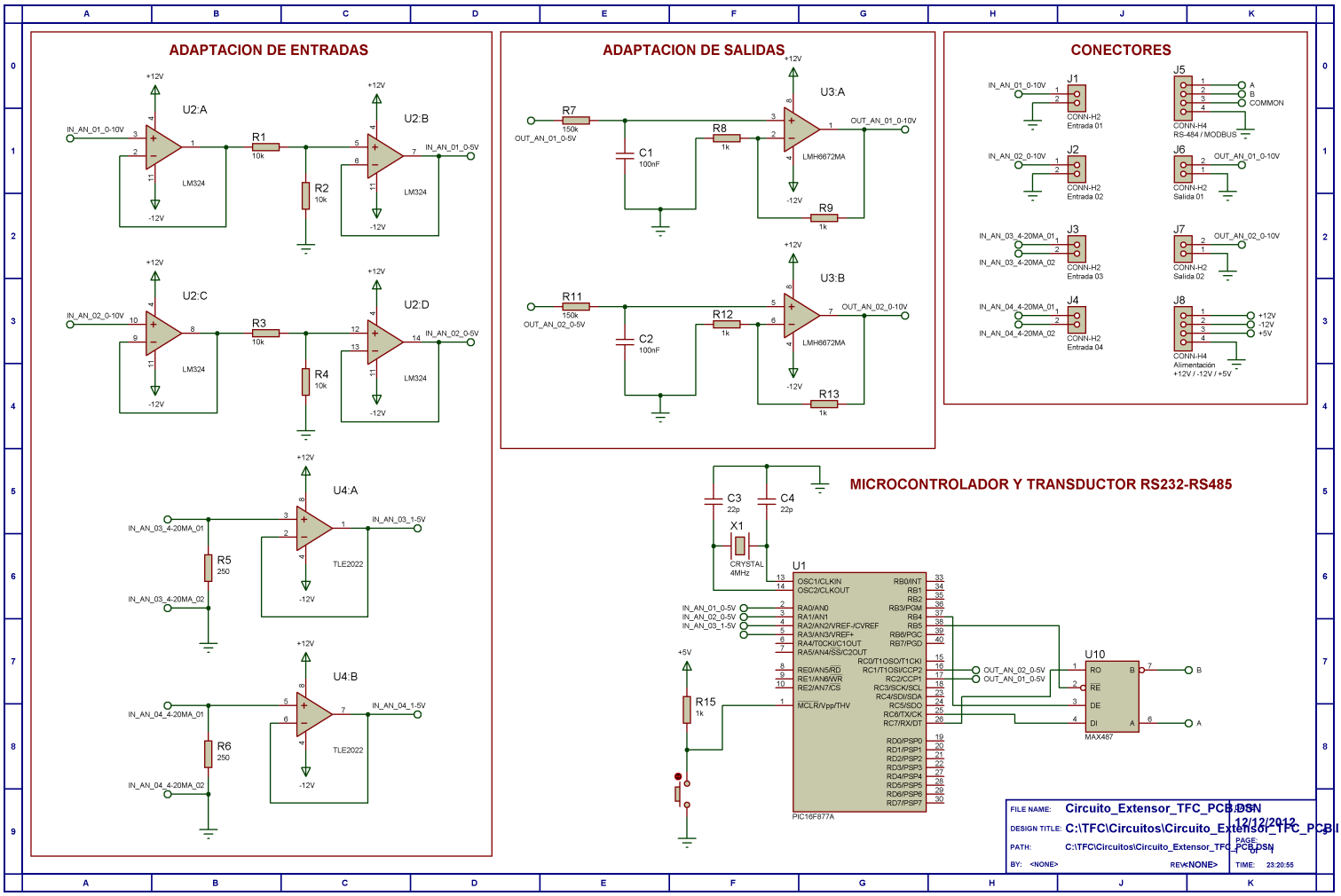


Figura 49. Esquema eléctrico Extensor entradas y salidas analógicas MODBUS RTU / RS-485

7.2. Optimización y Mejoras

Con el objetivo de abaratar costes, en lugar de usar un chip para cada amplificador operacional, optaremos por seleccionar chips con 2 o 4 A.O. integrados para la adaptación de entradas y uno con 2 A.O. para la adaptación de salidas. De este modo conseguimos además reducir la superficie de la PCB, la caja donde se alojará si se implementa, etc.

También hemos optado por independizar la fuente del resto del circuito de modo que podamos modularizarlo y poder ampliar o recortar el rango de alimentación de la fuente, siempre respetando las tensiones de salida de la misma (+5 V, +12 y -12V), simplemente realizando un reemplazo de ésta.

8. LAYOUT

Para simplificar el diseño y dar versatilidad al mismo, vamos a crear una PCB con el circuito de la fuente y otro con el resto de bloques. De ese modo, si se decide cambiar tensión de alimentación, simplemente habría que sustituir esta placa y no todo el circuito completo.

Como no tenemos requisitos ni limitaciones de espacio, ni por exceso ni por defecto, hemos optado por diseñar la PCB mediante una placa de doble cara mediante el uso del software ARES PCB Profesional.

8.1. Diseño de la PCB de la fuente de alimentación

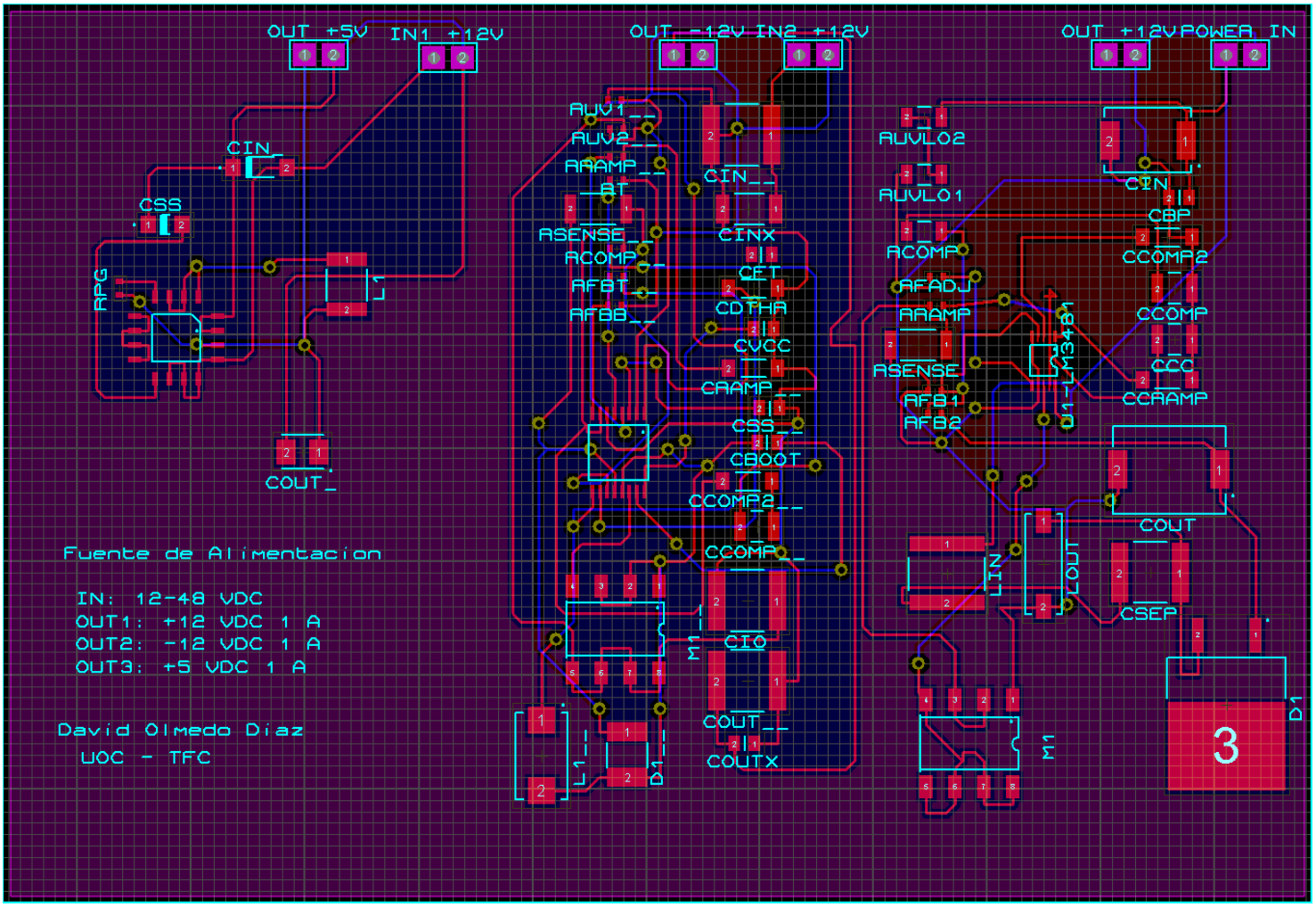


Figura 50. Layout PCB fuente de alimentación

***Nota**: Como norma de buen diseño y para evitar interferencias EMG, hemos distribuido la señal GND ocupando aquellas partes de la PCB en las cuales no existe cableado.*

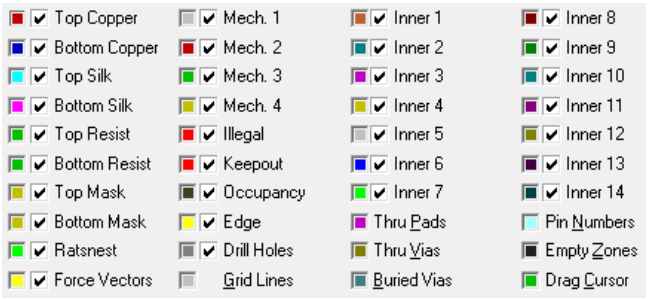


Figura 51. Leyenda de colores de las capas del layout

Para poder insolar la PCB es necesario imprimir los fotolitos de cada una de las caras de la PCB, teniendo presente el imprimir una de ellas en modo reflejado, para que coincidan al superponerlas.

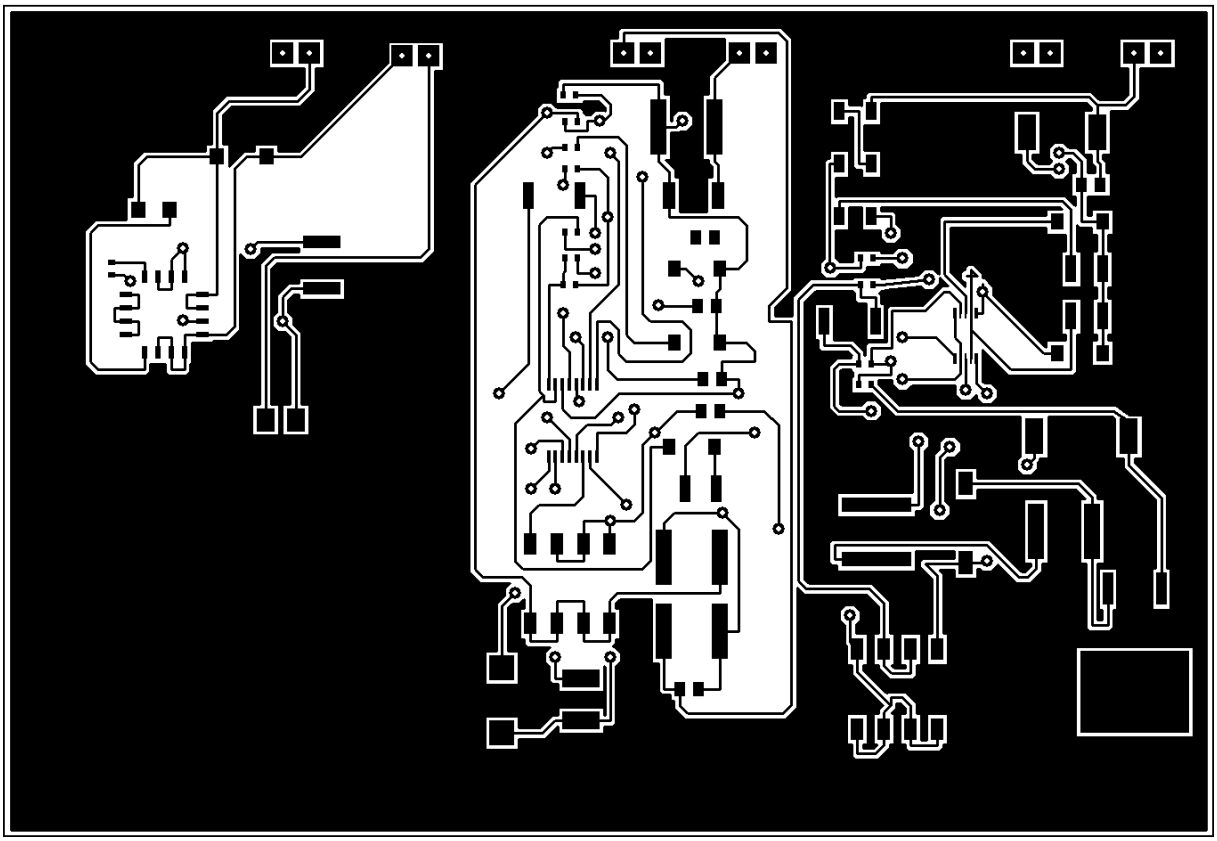


Figura 52. Fotolito PCB fuente de alimentación. Cara superior

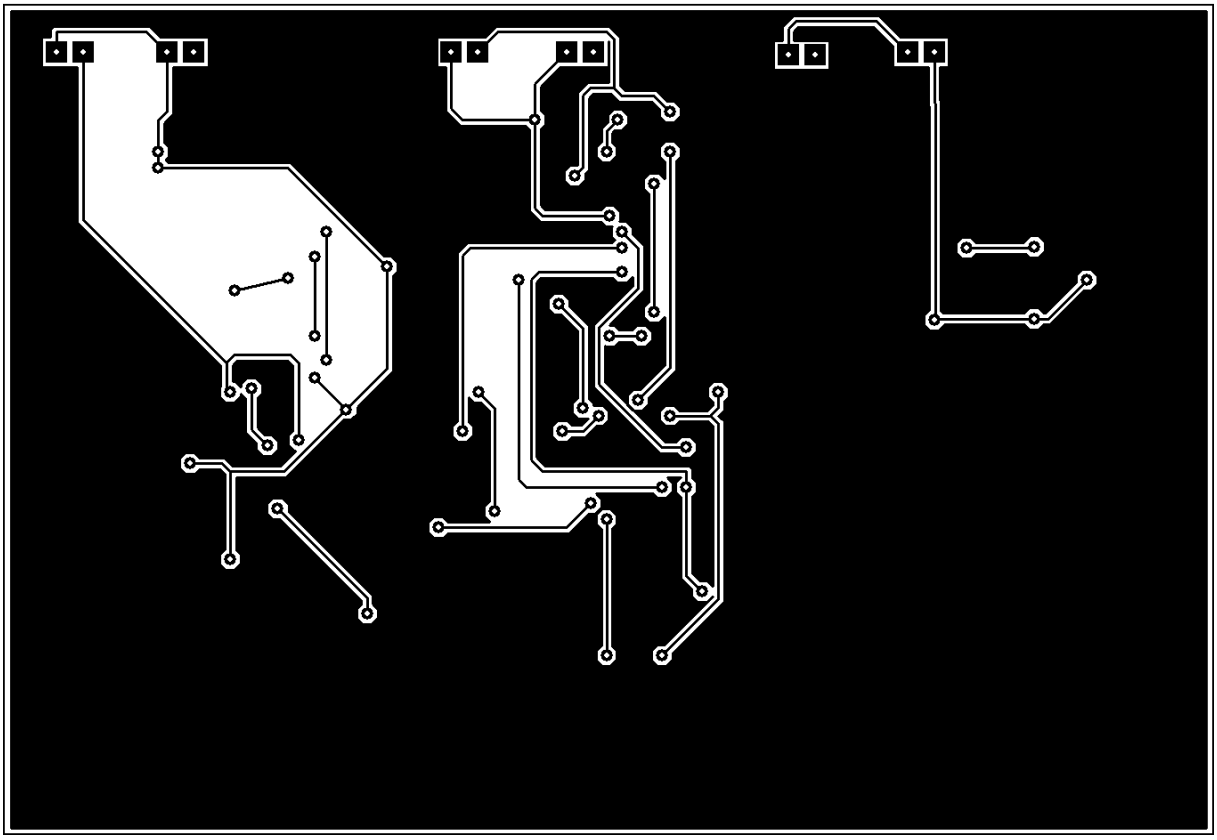


Figura 53. Fotolito PCB fuente de alimentación. Cara inferior

8.2. Simulación virtual de la fuente de alimentación

La misma aplicación con la que hemos diseñado la PCB nos permite hacer una simulación virtual en 2D y 3D de cómo sería el resultado final de ésta:

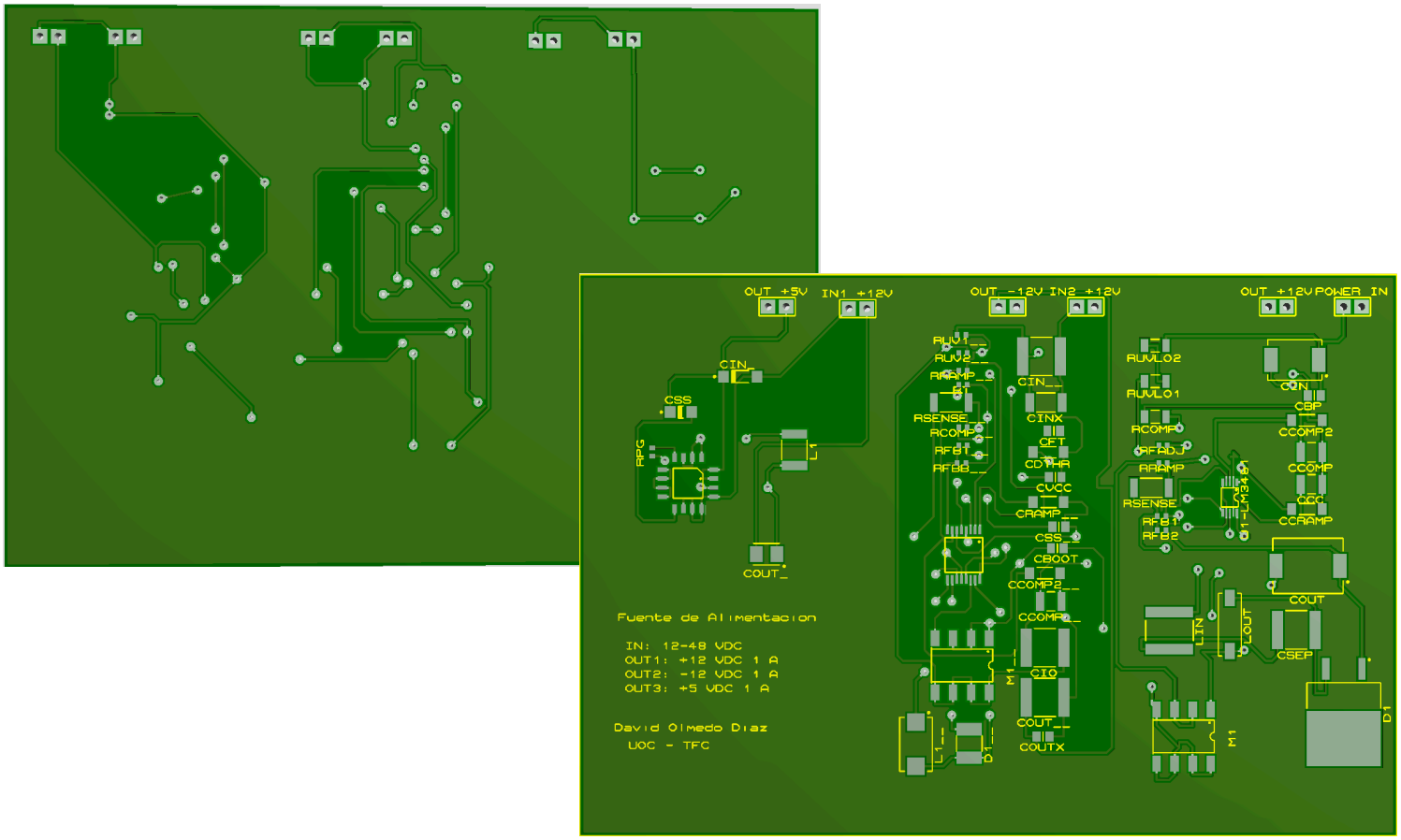


Figura 54. Simulación 2D virtual PCB de la fuente de alimentación

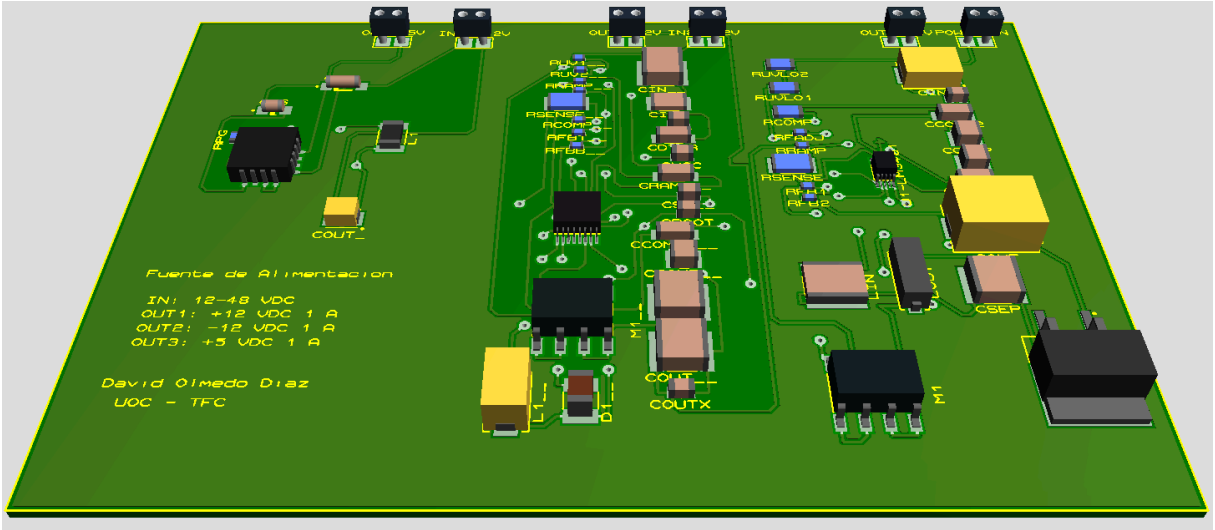


Figura 55. Simulación 3D virtual de la fuente con componentes

8.3. Diseño de la PCB del circuito extensor

El software CAD que hemos usado permite la opción de *auto-route* de modo que una vez que hemos colocado los distintos componentes es capaz de trazar de manera automática, haciendo uso de la teoría de grafos, la mejor ruta para la conexión de los distintos pines.

Una vez que la aplicación te realiza la conexión física mediante las pistas, se pueden modificar, cambiando la capa y variando el trazado hasta conseguir el resultado deseado.

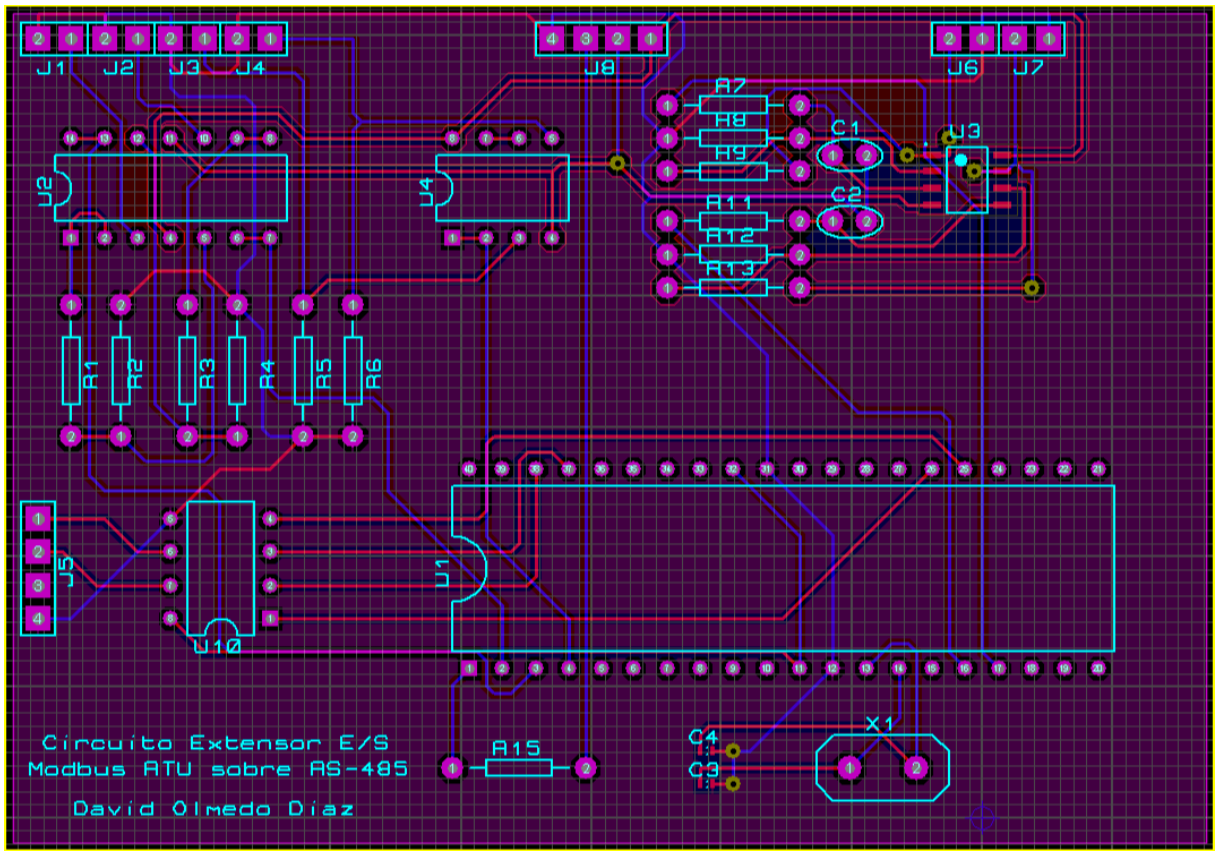


Figura 56. Layout PCB Extensor de entradas y salidas analógicas MODBUS RTU / RS-485

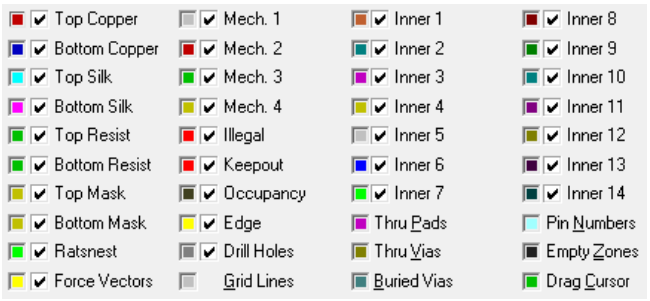


Figura 57. Leyenda de colores de las capas del layout

***Nota**: Como norma de buen diseño y para evitar interferencias EMG, hemos distribuido la señal GND ocupando aquellas partes de la PCB en las cuales no existe cableado.*

Para poder insolar la PCB es necesario imprimir los fotolitos de cada una de las caras de la PCB, teniendo presente el imprimir una de ellas en modo reflejado, para que coincidan al superponerlas.

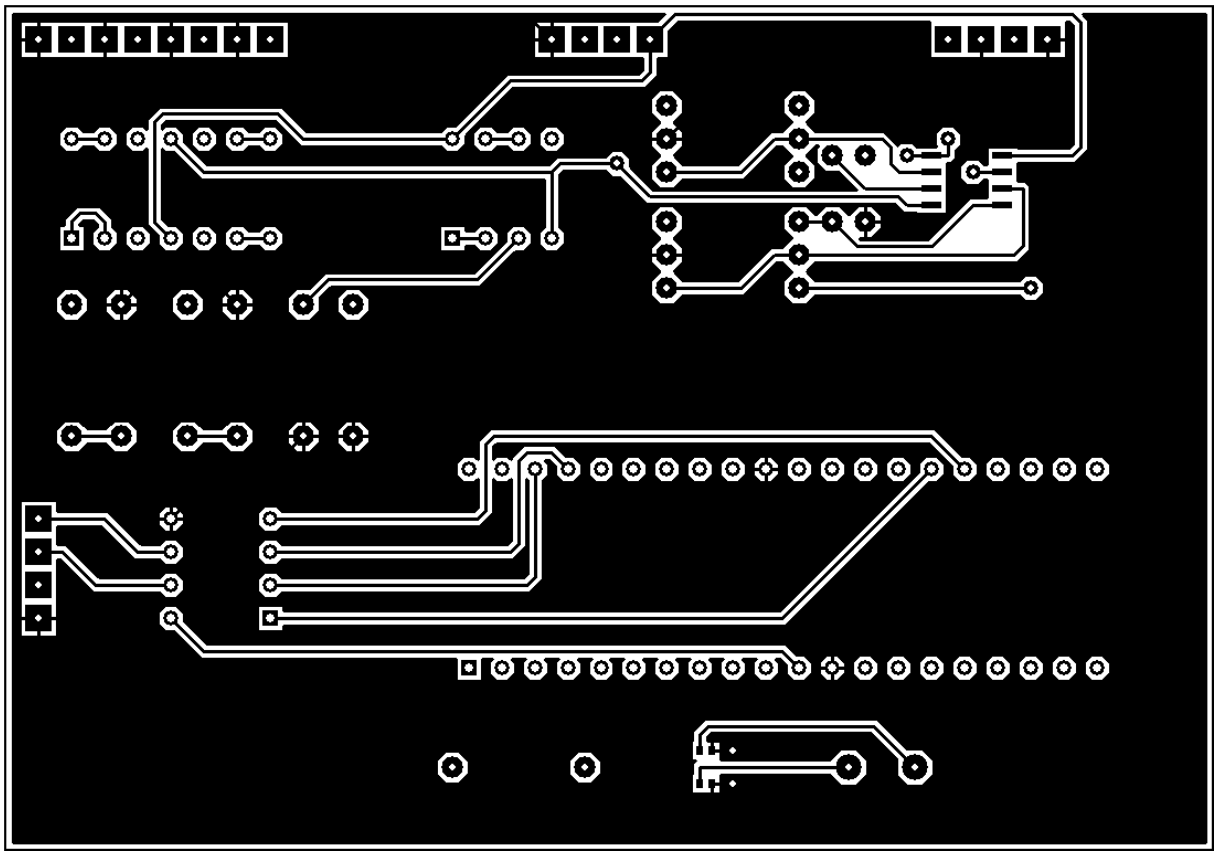


Figura 58. Fotolito PCB extensor. Cara superior

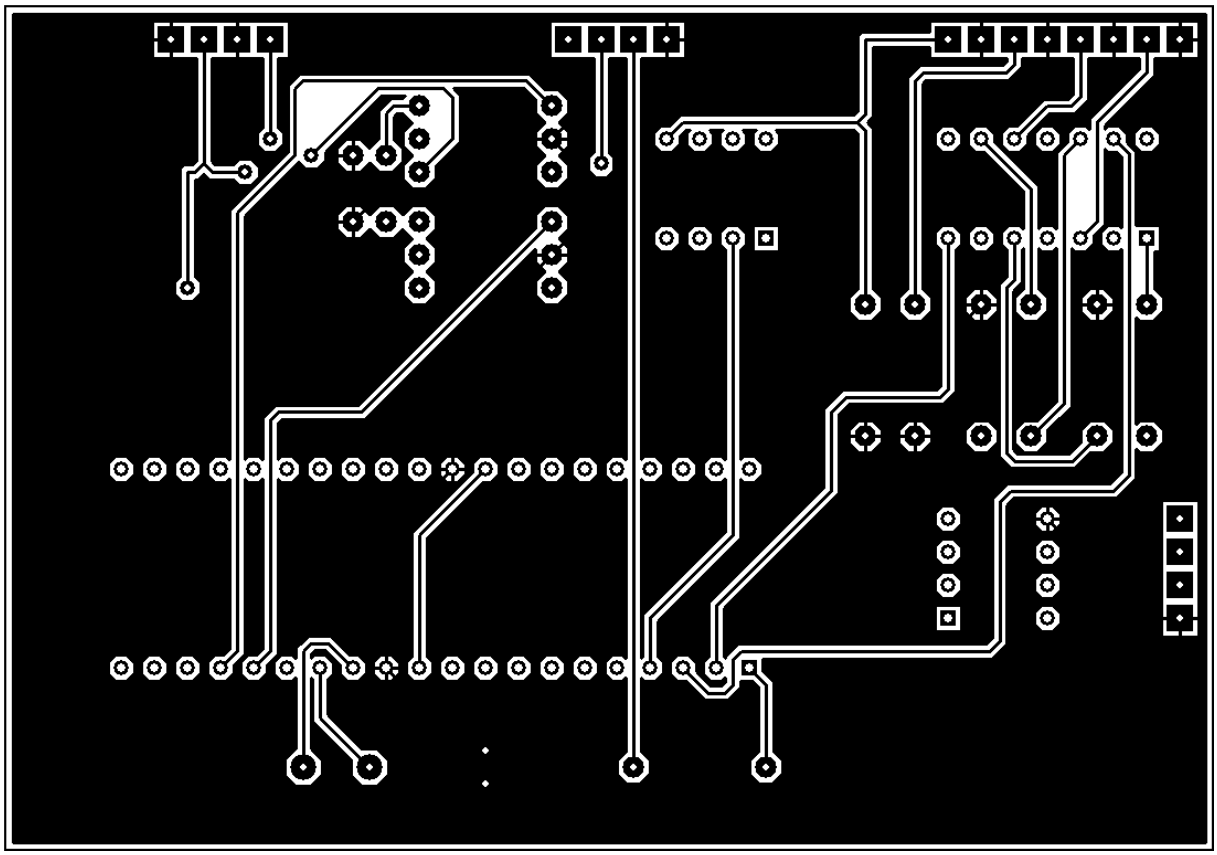


Figura 59. Fotolito PCB fuente de alimentación. Cara inferior

8.4. Simulación virtual del circuito extensor

La misma aplicación con la que hemos diseñado la PCB nos permite hacer una simulación virtual en 2D y 3D de cómo sería el resultado final de ésta:

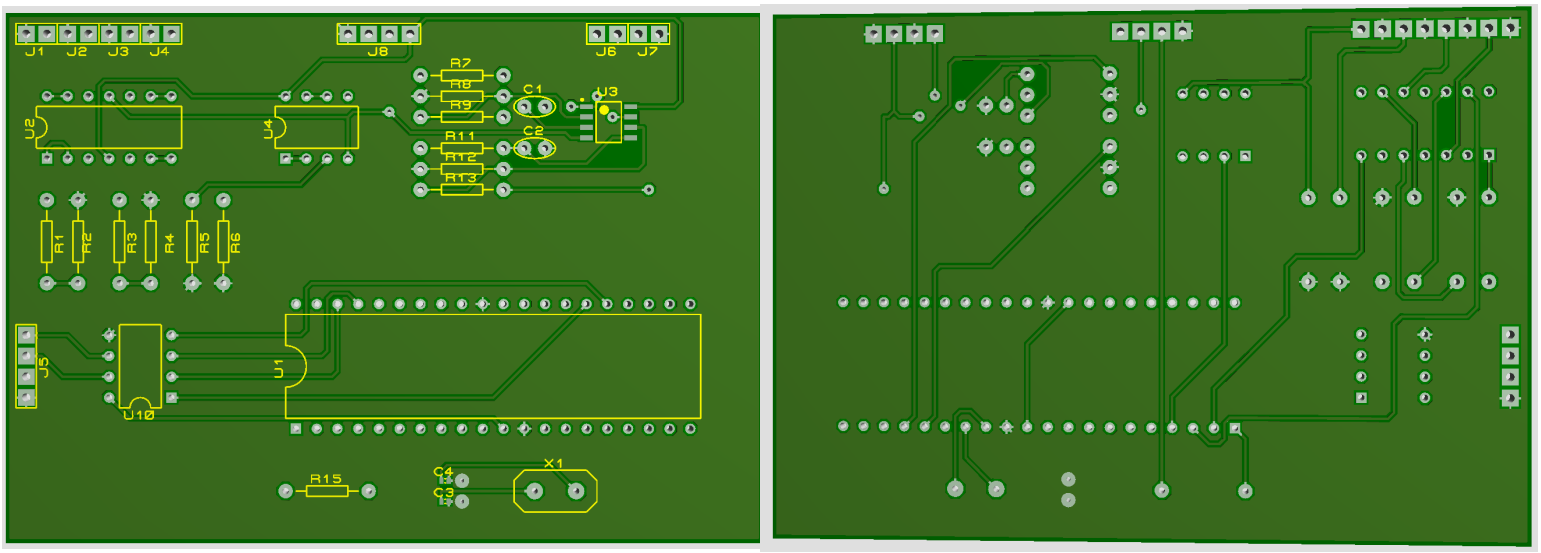


Figura 60. Simulación 2D virtual PCB del extensor

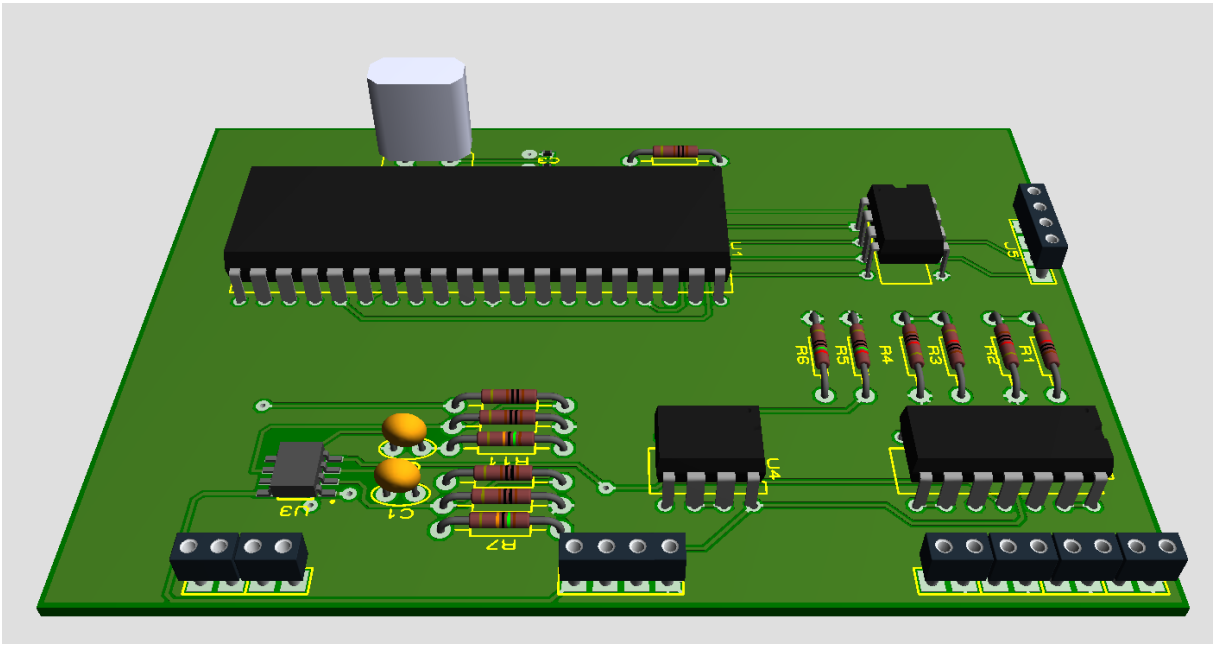


Figura 61. Simulación 3D virtual del extensor con componentes

9. LISTADO DE COMPONENTES

A continuación se listan los componentes que hemos usado para el diseño tanto de la fuente de alimentación como para el resto del circuito extensor, así como sus valores.

9.1. Listado componentes del circuito extensor

Componente	Cant.	Referencias	Valor	Observaciones	Precio Total
Resistencias	4	R1-R4	10k Ω		0,064 €
Resistencias	2	R5, R6	250 Ω		0,032 €
Resistencias	2	R7, R11	150k Ω		0,032 €
Resistencias	4	R8, R9, R12, R13	1k Ω		0,064 €
Resistencias	1	R15	1k Ω		0,016 €
Condensadores	2	C1, C2	100n F		0,199 €
Condensadores	2	C3, C4	22p F		0,077 €
Circuito Integrado	1	U1	PIC16F877A	Microcontrolador	12,970 €
Circuito Integrado	1	U2	LM324	A. Operacional x 4	0,530 €
Circuito Integrado	1	U3	LMH6672MA	A. Operacional x 2	3,447 €
Circuito Integrado	1	U4	TLE2022	A. Operacional x 2	1,532 €
Circuito Integrado	1	U10	MAX487	Transductor RS-485	1,210 €
Varios	6	J1-J4, J6, J7	CONN-H2	Conector x 2	2,573 €
Varios	2	J5, J8	CONN-H4	Conector x 4	1,149 €
Varios	1	X1	CRYSTAL	Cristal 4 MHz	2,970 €

Tabla 15. Listado de componentes del circuito extensor

9.2. Listado componentes de la fuente de alimentación

Componente	Cant.	Referencia	Valor	Part Number	Fabricante	Precio Total
Fuente +5 V DC						
Condensador	1	Cin	10u F	GRM31CR61E106KA12L	MuRata	0,054 €
Condensador	1	Cout	22u F	EMVA100ADA220MD55G	Nippon Chemi-Con	0,054 €
Condensador	1	Css	3.3n F	C0805C332K5RACTU	Kemet	0,008 €
Inductor	1	L1	2.2u H	SDR0403-2R2ML	Bourns	0,153 €
Resistencia	1	Rpg	100K Ω	CRCW0402100KFKED	Vishay-Dale	0,008 €
C. Integrado	1	U1	+5V	TPS62153RGTR	Texas Instruments	0,766 €
Fuente -12 V DC						
Condensador	1	Ccomp	2.2n F	CC0805KRX7R9BB222	Yageo America	0,008 €
Resistencia	1	Rcomp	14K Ω	CRCW040214K0FKED	Vishay-Dale	0,008 €
Condensador	1	Ccomp2	130p F	GRM1555C1H131JA01D	MuRata	0,008 €
Condensador	1	Cboot	100n F	C0603C104Z4VACTU	Kemet	0,008 €
Condensador	1	Cdthr	39n F	GRM155R71C393KA01D	MuRata	0,008 €
Condensador	1	Cft	100n F	C0603C104Z4VACTU	Kemet	0,008 €
Condensador	1	Cin	1u F	C1206C105K3RACTU	Kemet	0,031 €
Condensador	1	Cinx	100n F	08053C104KAT2A	AVX	0,008 €
Condensador	1	Cio	1u F	C3216X7R1H105K	TDK	0,031 €
Condensador	1	Cout	22u F	GRM32ER61C226KE20L	MuRata	0,229 €
Condensador	1	Coutx	1u F	GRM188R61E105KA12D	MuRata	0,015 €
Condensador	1	Cramp	220p F	CC0805KRX7R9BB221	Yageo America	0,008 €

Componente	Cant.	Referencia	Valor	Part Number	Fabricante	
Condensador	1	Css	1u F	GRM155R61A105KE15D	MuRata	0,008 €
Condensador	1	Cvcc	220n F	GRM155R61A224KE19D	MuRata	0,008 €
VFatIo	1	D1	0.5 V	B340A-13-F	Diodes Inc.	0,084 €
Inductor	1	L1	10u H	SRU1038-100Y	Bourns	0,276 €
VdsMax	1	M1	60 V	AON7246	AOS	0,253 €
Resistencia	1	Rfbb	1.65K Ω	CRCW04021K65FKED	Vishay-Dale	0,008 €
Resistencia	1	Rfbt	14.7K Ω	CRCW040214K7FKED	Vishay-Dale	0,008 €
Resistencia	1	Rramp	221K Ω	CRCW0402221KFKED	Vishay-Dale	0,008 €
Resistencia	1	Rsense	0.022 Ω	ERJ-L14KF22MU	Panasonic	0,084 €
Resistencia	1	Rt	10.2K Ω	CRCW040210K2FKED	Vishay-Dale	0,008 €
Resistencia	1	Ruv1	1.58K Ω	CRCW04021K58FKED	Vishay-Dale	0,008 €
Resistencia	1	Ruv2	10K Ω	CRCW040210K0FKED	Vishay-Dale	0,008 €
C. Integrado	1	U1	-12V	LM25088MH-1/NOPB	Texas Instruments	0,957 €
Fuente +12 V DC						
Condensador	1	Cout	150u F	APXA160ARA151MJ80G	Nippon Chemi-Con	0,291 €
Condensador	1	Ccomp2	330p F	CC0805KRX7R9BB331	Yageo America	0,008 €
Inductor	1	Lin	22u H	XAL6060-223MEB	Coilcraft	0,582 €
Inductor	1	Lout	47u H	7447709470	Wurth Elektronik eiSos	1,547 €
Condensador	1	Ccomp	82n F	GRM155R61A823KA01D	MuRata	0,008 €
Resistencia	1	Rcomp	3K Ω	RR1220P-302-D	Susumu Co Ltd	0,008 €
Condensador	1	Cbp	100n F	GRM188R72A104KA35D	MuRata	0,023 €
Condensador	1	Ccc	220n F	GRM155R61A224KE19D	MuRata	0,008 €
Condensador	2	Cin	10u F	293D106X9063E2TE3	Vishay-Sprague	4,029 €
Condensador	1	Cramp	1n F	GRM216R71E102KA01D	MuRata	0,008 €
Condensador	1	Csep	2.2u F	GRM32ER72A225KA35L	MuRata	0,291 €
VFatIo	1	D1	0.77 V	50WQ10FNPBF	Vishay-Semiconductor	0,315 €
VdsMax	1	M1	80 V	BSC340N08NS3 G	Infineon Technologies	0,230 €
Resistencia	1	Rfadj	59K Ω	CRCW040259K0FKED	Vishay-Dale	0,008 €
Resistencia	1	Rfb1	10K Ω	CRCW040210K0FKED	Vishay-Dale	0,008 €
Resistencia	1	Rfb2	84.5K Ω	CRCW040284K5FKED	Vishay-Dale	0,008 €
Resistencia	1	Rramp	100 Ω	CRCW0402100RFKED	Vishay-Dale	0,008 €
Resistencia	1	Rsense	0.012 Ω	CSR1206FK12L0	Stackpole Electronics	0,084 €
Resistencia	1	Ruvlo1	15K Ω	CRCW080515K0FKEA	Vishay-Dale	0,008 €
Resistencia	1	Ruvlo2	100K Ω	CRCW0402100KFKED	Vishay-Dale	0,008 €
C. Integrado	1	U1	+12V	LM3481MM/NOPB	Texas Instruments	0,689 €

Tabla 16. Listado de componentes de la fuente de alimentación

El coste aproximado de los componentes del extensor es de 26,87 € y los de la fuente 11,29€. Por lo tanto el coste total asciende a es **38,16 €** *

***Nota**: A los que habrá que sumar el coste de las PCBs de doble cara, materiales de insolado, revelado y materiales de soldado de componentes (sin incluir la mano de obra).*

10. CONCLUSIONES

Durante el proceso de estudio y diseño de las distintas fases que componen el TFC, he llegado a la conclusión de que para poder realizar un correcto diseño de la fuente de alimentación debemos tener muy bien definidas todas las alimentaciones de los distintos bloques funcionales del dispositivo y es por ello, que este bloque de alimentación ha sido el último que he diseñado.

Otro de los problemas con los que me he encontrado durante el proceso de diseño ha sido la selección de componentes electrónicos para los distintos bloques funcionales, ya que no sólo hay que elegir entre la gran oferta disponible en el mercado, sino que tienen que cumplir los requisitos y por si fuera poco, además, hay que seleccionar aquellos que tengan un la librería o modelo funcional en el software de simulación de circuitos. Hasta que he aprendido este detalle, he tenido que re-hacer dos o tres veces el texto redactado y capturas adaptándolas posteriormente al modelo con el que se podría realizar la correspondiente simulación.

Este problema de la selección de componentes que cumplieran con los requisitos y que, además, se encontraran en la librería de la aplicación CAD de simulación de circuitos ha sido uno de los motivos por los que me decantado por usar directamente el WEBENCH para el diseño de la fuente de alimentación. Este aplicación proporciona en su mayoría componentes específicos que no se encuentran en PROTEUS y aunque existe la posibilidad de exportar el diseño de la fuente en OrCAD Capture CIS, uno de los bloques lanza un error y por eso tampoco he podido simularlo con este software.

En general, el mayor de los problemas ha sido la depuración de errores , ya que no se puede simular y depurar todo en un una misma aplicación. Un cambio en el circuito eléctrico implica un cambio en el código del microprocesador y una vez que se compila, hay que verificar el correcto funcionamiento en el simulador de peticiones/respuestas Modbus (REAL PIC SIMULATOR). Del mismo modo, un cambio en el código, implica, en su mayoría de ocasiones, probar que funciona la simulación eléctrica en PROTEUS y su posterior comprobación en el simulador de peticiones/respuestas Modbus.

A pesar de los muchos problemas con los que me ido encontrando a lo largo del TFC, creo que los he solventado todos, con mayor o menor acierto. He aprendido mucho acerca de la planificación de tareas, el estudio de protocolos y el manejo de programas CAD eléctricos y PCBs.

Si con la experiencia adquirida en estos meses de trabajo tuviera que empezar de cero con otro proyecto similar, estoy seguro que lo afrontaría de una manera más profesional. Evitaría caer en algunos errores de neófito y seguro que optimizaría mucho mejor el tiempo dedicado.

Me hubiera gustado poder trasladar a la realidad el prototipo realizado, pero la falta de tiempo lo ha hecho totalmente inviable. De haberlo implementado, me habría encontrado, probablemente, con otra serie de problemas (soldado de componentes de encapsulado superficial, funcionamiento diferente al teórico y/o ideal, etc.), pero seguro que serviría de nuevo para adquirir experiencia y poder aplicarla en el futuro.

La gran cantidad de tiempo y esfuerzo dedicados a este TFC se compensa, con creces, con la satisfacción personal del resultado obtenido.

11. BIBLIOGRAFÍA Y REFERENCIAS

[1] **Roselló Canal, Mar**, (2010). "Aplicaciones electromagnéticas y electrónicas" <http://cvapp.uoc.edu/autors/MostraPDFMaterialAction.do?id=163586>

[2] **Bataller Díaz, Alfons**, (2007). "Gestión y desarrollo de proyectos. Conceptos y sugerencias" <http://cv.uoc.es/cdocent/V20T7OHXRRBG6LOUUE48.pdf>

[3] **Sáenz Higueras, Nita; Vidal Oltra, Rut,** (2007). "Redacción de textos científico-técnicos" <http://cv.uoc.es/cdocent/_D_CBBU62JZTHQ9CQQGJ.pdf>

[4] **Pérez i Navarro, Antoni; Martinez Carrascal, Juan Antonio; Muñoz Medina, Olga**,(2006). "Teoría de circuitos". <http://cvapp.uoc.edu/autors/MostraPDFMaterialAction.do?id=153326>

[5] **Pérez i Navarro, Antoni; Bara Iniesta, Marc; Martínez Carrascal, Juan Antonio; Roselló Canal, Mar**. (2006)."Sistemas Electrónicos Digitales". <http://materials.cv.uoc.edu/cdocent/JRAFKCLMZLU7EPFRZT5S.pdf>

[6] **Alan V. Oppenheim, Willsky y otros**, (2002). "Señales y Sistemas". Editorial Prentice Hall. ISBN: 0138147574.

[7] **Keithley Instruments, Inc.** (2004). "Low Level Measurements Handbook. Precision DC Current, Voltage And Resistance Measurements". <http://staff.washington.edu/wbeaty/Keithley_Low_Level_Measurements_Handbook.pdf>

[8] **Wilson, Jon S.**, (2004). "Sensor Technology Handbook". <http://een.iust.ac.ir/profs/Esmaeilzadeh/Instrumentation/Sensor%20Technology%20Handbook.pdf>

[9] **Luecke, Jerry** (2005). "Analog And Digital Circuits For Electronic Control System Applications". <http://www.sciencedirect.com/science/book/9780750678100>

[10] **Fischer-Cripps, Tony,** (2002). "Interfacing Companion: Computers, Transducers, Instrumentation And Signal Processing". <http://www.amazon.es/Newnes-Interfacing-Companion-Transducers-Instrumentation/dp/0750657200>

[11] **Microchip,** (10/31/2003). "PIC16F87XA Datasheet". < http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf>

[12] **Microchip,** (1997). "Using PWM to Generate Analog Output". <http://ww1.microchip.com/downloads/en/AppNotes/00538c.pdf>

[13] **Texas Instruments**, (12/2000). "Using PWM Timer_B as a DAC" <http://www.ebn2.gaw.ru/pdf/TI/app/msp430/slaa116.pdf>

[14] **Texas Instruments**, (19/10/2004). "LM741 Operational Amplifier (Rev. B)" <http://www.ti.com/lit/gpn/lm741>

[15] **Texas Instruments**, (30/03/2010). "LM324 Quadruple operational amplifiers (Rev. T)" <http://www.ti.com/lit/gpn/lm324>

[16] **Texas Instruments**, (01/05/2004). "LMH6672 Dual, High Output Current, High Speed Op Amp (Rev. F)" <http://www.ti.com/lit/gpn/lmh6672>

[17] **Texas Instruments**, (24/10/2010). "High-Speed Low-Power Precision Operational Amplifiers. (Rev. D) " <http://www.ti.com/lit/gpn/tle2022>

[18] **Texas Instruments**, (27/09/2000). "Precision 4mA to 20mA Current Loop Receiver" <http://www.ti.com/lit/gpn/rcv420>

[19] **Texas Instruments.** Olmedo Díaz, David, (01/12/2012). "Fuente de Alimentación TFC" <http://webench4.ti.com/appinfo/webench/scripts/SP1.cgi?ID=5484::powerarchitect::davidolmedo@gmail.com>

[20] **Texas Instruments**, (03/04/2012). "LM3481 High Eff Low-Side N-Chan Cntrl for Switch Regs (Rev. E)" <http://www.ti.com/lit/gpn/lm3481>

[21] **Texas Instruments**, (14/03/2011). "LM25088 Wide Input Range Non-Synchr. Buck Controller (Rev. G)" <http://www.ti.com/lit/gpn/lm25088>

[22] **Texas Instruments**, (17/10/2012). "3-17V 1A 3MHz Step-Down Converter in 3x3 QFN Package (Rev. A) " <http://www.ti.com/lit/gpn/tps62153>

[23] **Maxim Integrated**, (09/2009). "Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers (Rev.9)"

<http://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf>

[25] **Maxim Integrated**, (29/11/2001). "DC-DC Converter Tutorial"
<http://pdfserv.maximintegrated.com/en/an/AN2031.pdf>

[26] **Maxim Integrated**, (22/09/2010). "Building a DC-DC Power Supply that Works"
<http://pdfserv.maximintegrated.com/en/an/AN1897.pdf>

[27] **Linear.** "LT1886 – Dual 700 MHz, 200 mA Operational Amplifier"
<http://www.datasheetcatalog.net/es/catalogo/p519920.shtml>

[28] **MikroElektronika**, (2010). "Libro de Programación de los microcontroladores PIC en Basic"
< http://www.mikroe.com/products/view/476/pic-microcontrollers-programming-in-basic/>

Enlaces de interés:

www.uoc.es
www.wikipedia.es
www.modbus.com
www.modbus.org
www.ti.com
www.microchip.com
www.linear.com
www.maximintegrated.com
www.digitalelectrosoft.com/pic-simulator
www.tolaemon.com/site/protocolo_modbus
www.lammertbies.nl