UNIVERSITAT OBERTA DE CATALUNYA

Enginyeria Tècnica de Telecomunicació,

especialitat Telemàtica

DISSENY D'UN EXPANSOR D'ENTRADES I SORTIDES ANALÒGIQUES PER MODBUS RTU SOBRE RS-485

Alumne: Oscar Piris Agustí

Dirigit per: Carlos Alberto Pacheco Navas

CURS 2012-13 (1er Semestre)

Agraïments

En la realització d'aquest treball final de carrera volia agrair a tots els consultors que he tingut durant aquests set semestres a la UOC, ja que sense ells no hagués estat possible la realització d'aquest treball final de carrera.

També cal donar les gràcies a totes aquelles persones que em coneixen, ja siguin familiars o amistats, i què m'han donat tot el suport i els ànims necessaris per a dur a terme la finalització d'aquesta enginyeria, ja que no és gens fàcil compaginar la vida laboral, familiar, social i estudiar al mateix temps.

El que està clar és que tot aquest sacrifici, realitzat durant aquests set semestres, al menys ha valgut la pena i s'ha aconseguit assolir l'objectiu final, què no és més que la de graduar-se en *Enginyeria Tècnica de Telecomunicació, especialitat en Telemàtica* per la UOC.

Resum

Aquest treball final de carrera ha estat desenvolupat per Oscar Piris Agustí, estudiant *d'Enginyeria Tècnica de Telecomunicació, especialitat en Telemàtica* per la UOC. Aquest TFC té com a objectiu el disseny d'un adaptador d'interfície per a llegir i/o escriure senyals analògics mitjançant *ModBus RTU* sobre *RS-485*.

Tal i com comprovarem al llarg de tots els apartats, aquest TFC és bastant complert ja que abasta varies temàtiques. Primerament s'ha fet la planificació de totes les tasques i fites què conformen aquest TFC. Seguidament es realitza el disseny dels circuits electrònics d'adaptació de les entrades i sortides analògiques, i de la font d'alimentació capaç d'admetre a la seva entrada entre 12 i 48 volts, i treure unes sortides de +15V, -15V, +5V, -1V i massa. Amb aquests dissenys dels circuits electrònics farem els dissenys dels *layouts* de les dues *PCB* a realitzar.

Explicarem el funcionament del protocol *ModBus* i el bus *RS-485*. *ModBus* defineix els datagrames a enviar/rebre, així com les funcions que s'envien, i els codis d'error o excepcions. En canvi el bus *RS-485* defineix les característiques elèctriques de funcionament del bus.

Per a controlar les entrades i sortides analògiques i la comunicació, el nostre expansor utilitza un microcontrolador *PIC16F876A*. Explicarem les seves principals característiques i funcionament. També s'explicarà el programa de control de l'expansor d'entrades i sortides analògiques per *ModBus RTU* sobre *RS-485*.

Finalment farem les conclusions finals, així com una petita valoració econòmica del disseny final d'aquest expansor d'entrades i sortides analògiques. Juntament amb aquesta memòria es fa entrega d'un vídeo de presentació del TFC, així com dels arxius del codi font, del programa de control, del PIC.

Índex

Índex de figures

Índex de taules

1. Introducció

Aquest treball final de carrera (TFC) ha estat desenvolupat per **Oscar Piris Agustí**, amb DNI [redacted], natural de Ciutadella de Menorca, Illes Balears, estudiant *d'Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica* per la Universitat Oberta de Catalunya (UOC). El treball final de carrera té com a objectiu el disseny d'un adaptador d'interfície per a llegir i/o escriure senyals analògics mitjançant *ModBus RTU* sobre *RS-485*.

Aquesta memòria és el resultat de les tasques que es planificaren en el pla de treball entregat el dia 2 d'octubre de 2012. A l'apartat 1 tenim aquesta introducció, on també es farà un resum del pla de treball, tot indicant la justificació i objectius del TFC, així com el mètode seguit. També veurem la planificació de les tasques, i finalment els productes obtinguts. A l'apartat 2 tenim l'estudi del protocol *ModBus RTU* i del bus *RS-485*, s'indicaran les característiques de les diferents capes del protocol, així com les seves característiques elèctriques. A l'apartat 3 tenim el disseny d'adaptació de les entrades i sortides de l'expansor. A l'apartat 4 farem el disseny de la font d'alimentació per admetre una entrada de 12 a $48\ V$, per obtenir els diferents voltatges a utilitzar per l'expansor. A l'apartat 5 tenim la definició del mapa de memòria, i a l'apartat 6 tenim l'elecció del microcontrolador així com la seva programació per tal d'implementar l'expansor. Finalment, a l'apartat 7 tenim el disseny de les plaques *PCB*, i a l'apartat 8 les conclusions finals del TFC així com una valoració econòmica i les seves futures ampliacions. Per completar aquesta memòria tenim un glossari a l'apartat 9, una bibliografia a l'apartat 10, i un annex a l'apartat 11 amb els esquemes dels circuits electrònics finals d'aquest disseny.

1.1. Justificació TFC

El TFC és una assignatura, dintre del pla d'estudis *d'Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica*, on s'ha de fer una síntesi dels coneixements adquirits en les altres assignatures del pla d'estudis i on s'han de posar en pràctica en un treball final concret. Aquest TFC s'inclou dintre de l'àrea *d'Aplicacions Electromagnètiques i Electròniques*, amb codi 19.038. Durant la carrera s'han estudiat assignatures relacionades amb l'electromagnetisme i l'electrònica (tant analògica com digital), i per tant, en aquest TFC s'han de posar en pràctica els coneixements adquirits sobre aquestes matèries[1,2,3].

També es tindran en compte els conceptes apresos en l'assignatura de *Gestió de Projectes*, en quant a la planificació del projecte a realitzar[4,5,6]. Concretament, en aquest TFC es proposa dissenyar un adaptador d'interfície per a llegir i/o escriure senyals analògics mitjançant *ModBus RTU* sobre *RS-485*. Aquest objectiu i altres són descrits al següent punt 1.2.

1.2. Objectius TFC

Els objectius generals d'aquest TFC són demostrar i aplicar els coneixements adquirits en les diverses assignatures cursades durant la realització d'aquesta *Enginyeria Tècnica de Telecomunicacions, especialitat Telemàtica*. Concretament les assignatures relacionades amb l'electrònica analògica, l'electrònica digital i la gestió de projectes.

La finalitat d'aquesta assignatura és el desenvolupament d'un treball final de carrera, on s'han de planificar les diferents tasques tenint en compte el temps disponible i de complir amb les

diferents fites marcades, què corresponen amb els lliuraments de les diferents *PAC* de l'avaluació continuada. Aquesta planificació del TFC correspon amb l'aprés a l'assignatura de *Gestió de Projectes*[4]. Els objectius específics del TFC a desenvolupar són:

- El disseny d'un sistema expansor d'entrades i sortides analògiques per *ModBus RTU* sobre *RS-485*.
- El disseny de la font d'alimentació i dels circuits d'adaptació de les entrades i sortides analògiques.
- La programació de les instruccions a implementar pel microcontrolador.
- El disseny de les plaques *PCB* que contindran tots els elements de l'expansor.

1.3. Enfocament i mètode seguit

Al llarg del semestre s'aniran fent varies entregues del que s'ha dut a terme fins aquell moment. Les entregues són el pla de treball, PAC2, PAC3 i finalment una memòria de fins a 60 pàgines, un vídeo de presentació de fins a 20 minuts, i dels arxius del programa de control.

Per tal d'aconseguir els objectius del punt 1.2, el TFC es pot dividir en una sèrie de tasques: estudi del protocol *ModBus* i del bus *RS-485*, disseny dels circuits d'adaptació d'entrades i sortides analògiques, disseny de la font d'alimentació, definició del mapa de memòria *ModBus*, elecció del microcontrolador i la implementació del programa de control, i finalment el disseny dels *layout* de les plaques *PCB*. Tal i com es pot veure al següent diagrama de blocs:

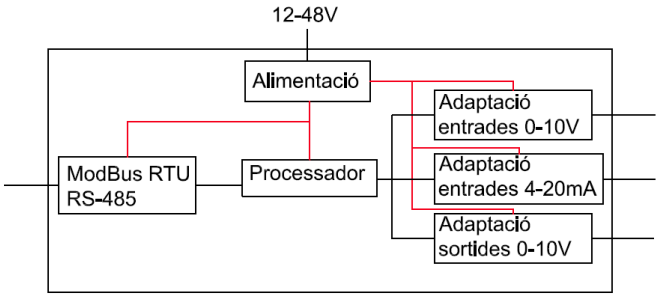


Figura 1 Diagrama de Blocs

1.4. Planificació TFC

Tal i com es va detallar al pla de treball, aquest TFC està dividit en una sèrie de tasques i activitats per a dur-lo a terme, tot complint unes fites marcades. Primer farem una valoració del temps disponible per a dedicar a la realització d'aquest TFC.

Al treballar només pels matins disposo de temps lliure per les tardes, així com els caps de setmana. S'ha arribat a la conclusió que els dies festius nacionals i locals no es treballarà, per tant aquests dies no es tindran en compte en la planificació del TFC, els dies són el 12 d'octubre, 1 de novembre, 6, 24, 25 i 31 de desembre, i 1 de gener.

Al estar cursant tres assignatures més, a part del TFC, hi ha que deixar temps per a aquestes. Tots els caps de setmana dels mesos de setembre, octubre, novembre, desembre i gener disposo de fins a 5h/dia per a dedicar-l'hi. El mes de setembre, al ser final de temporada

turística a Menorca, no disposo de gaire temps, i per tant, només disposo de 2h/dia els dies laborals. Les dues primeres setmanes d'octubre faig vacances, i per tant puc dedicar-hi fins a 4h/dia. Des del dia 15 d'octubre de 2012 fins el 7 de gener de 2013 puc dedicar fins a 3h/dia els dies laborals. Finalment, els dies 24 i 25 de gener de 2013 disposaré de fins a 4h/dia per a dedicar a respondre a les preguntes del tribunal d'avaluació.

Els dies d'avaluació, als quals acudiré presencialment a fer els exàmens i proves de validació, són el 12, 19 i 23 de gener de 2013. Tenint en compte que la darrera entrega és dia 7 de gener, i que el debat virtual del tribunal d'avaluació són els dies 23, 24 i 25 de gener, només m'afectarà al dia 23, el qual no podré respondre a les possibles preguntes del tribunal, i què seran contestades entre els dies 24 i 25 de gener.

La durada del TFC és de 111 dies, començant el dia 19 de setembre de 2012 i finalitzant el 25 de gener de 2013, la qualificació del qual s'obtindrà el dia 5 de febrer de 2013. El número total d'hores impartides per a realitzar aquestes tasques - activitats és de 329,8 hores. El calendari on es mostren els dies festius, els dies d'avaluació presencial, i els dies de les fites del TFC és:

Setembre 2012

L	M	X	J	V	S	D
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Octubre 2012

L	M	X	J	V	S	D
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Novembre 2012

L	M	X	J	V	S	D
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Desembre 2012

L	M	X	J	V	S	D
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Gener 2013

L	M	X	J	V	S	D
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Dies festius nacionals/locals
Dies d'avaluació presencial
Límits de les diferents fites del TFC

Taula 1 Calendari Festius i Fites

Tenint en compte totes aquestes consideracions, tot seguit es presenta una taula, resum del pla de treball, on s'especifiquen les diferents tasques i la seva durada, en fons gris les tasques principals i en fons groc les fites:

Codi	*Nom*	*Descripció*	*Dates*	*Hores*	*Precedents*
1	Pla de Treball	Document on es reflectirà tota la planificació del semestre per a la realització del treball final de carrera.	19-09 → 2-10	35,7	
1.1	Descarregar enunciat TFC	Descarregar del campus de la UOC l'enunciat del TFC.	19-09-2012	0,1	
1.2	Llegir mòduls i enunciat TFC	Llegir l'enunciat del TFC i els mòduls "Gestió i desenvolupament de projectes", "Redacció de textos cientificotècnics", i "Presentació de documents i elaboració de presentacions".	19-09 → 25-09	13,9	
1.3	Videoconferència trobada d'inici	Realització d'una videoconferència de trobada d'inici entre el consultor i els diferents alumnes de les aules catalana i castellana.	22-09-2012	1,5	
1.4	Lectura resum trobada d'inici	Descarregar i llegir document preparat pel consultor del resum de la videoconferència de la trobada d'inici.	26-09-2012	0,2	1.3

1.5	Definició de fites i tasques	Planificació de les fites i tasques de les quals estarà compost el TFC a realitzar.	26-09-2012	1,8	
1.6	Temporització de les tasques	Gestionar les hores necessàries per a dur a terme les activitats definides durant l'activitat 1.5.	27-09-2012	1,9	1.5
1.7	Instal·lació de MS Project	Reinstal·lació de MS Project per poder fer el diagrama de *Gantt*.	27-09-2012	0,1	
1.8	Elaboració del Diagrama de Gantt	Realització del Diagrama de *Gantt* mitjançant *MS Project*.	28-09-2012	1h	1.5 1.6 1.7
1.9	Elaboració del pla de treball	Redacció del document Pla de treball tenint en compte la planificació feta durant totes les activitats anteriors.	28-09 → 29-09	10h	1.8
1.10	Entrega esborrany Pla de treball	Entrega del document Pla de treball al consultor per tal que en faci una lectura i faci indicacions dels punts a corregir.	29-09-2012	0,1	1.9
1.11	Correcció Pla de treball	Correcció del document Pla de treball segons les indicacions del consultor.	30-09 → 2-10	4h	
1.12	Lliurament Pla de treball	Lliurament a la bústia de l'aula del document Pla de treball.	02-10-2012	0,1	1.11
2	Estudi del protocol *ModBus* i del bus *RS-485*	Estudi del protocol *ModBus* a implementar en aquest expansor d'entrades i sortides, així com l'estudi del bus *RS-485*, el qual ens donarà les especificacions físiques.	02-10 → 07-10	22,9	
2.1	Recerca d'informació protocol *ModBus*	Recerca d'informació i característiques del protocol *ModBus*.	02-10-2012	3,9	
2.2	Recerca d'informació bus *RS-485*	Recerca d'informació i característiques del bus *RS-485*.	03-10-2012	4h	
2.3	Documentació PAC2	Redactar la informació cercada a les activitats 2.1 i 2.2 a la memòria del TFC.	04-10 → 07-10	15h	2.1 2.2
3	Disseny dels circuits d'adaptació d'entrades i sortides	Disseny dels circuits d'adaptació d'entrades i sortides per adequar les senyals a les tensions que utilitzarà el microcontrolador.	08-10 → 23-10	55h	
3.1	Adaptació tensions entrades analògiques 0-10V	Adaptar les tensions de les entrades analògiques 0-10V per tal de poder implementar-les al microcontrolador.	08-10 → 09-10	6	
3.2	Adaptació tensions entrades analògiques 4-20mA	Adaptar les tensions de les entrades analògiques de 4-20mA per tal de poder implementar-les al microcontrolador.	10-10-2012	4	
3.3	Adaptació tensions sortides analògiques 0-10V	Adaptar les tensions de les sortides *PWM* del microcontrolador en sortides analògiques de 0-10V.	11-10-2012	4	
3.4	Altres alternatives	Alternatives a les activitats 3.1, 3.2 i 3.3 que s'han descartat	13-10-2012	5	3.1 3.2 3.3
3.5	Instal·lació *software* disseny i simulació de circuits	Instal·lació *software* disseny i simulació de circuits, ja sigui *TINA-TI* o *Schematics* de *Orcad*.	14-10-2012	2	
3.6	Implementar circuits i simulació	Dissenyar i implementar en *software* els circuits d'adaptació d'entrades i sortides analògiques. També simular-lo.	14-10 → 18-10	15	3.1 3.2 3.3 3.5
3.7	Documentació PAC2	Redactar la informació de les activitats 3.1 a 3.6 a la memòria del TFC.	19-10 → 23-10	19	3.6
4	Disseny de la font d'alimentació	Dissenyar la font d'alimentació per què admeti com a entrada de 12 a 48V, així com les diferents tensions de sortida que necessitem a les *PCB* a dissenyar.	24-10 → 02-11	30,9	
4.1	Estudi alimentacions dels circuits de les *PCB*	Estudiar les diferents tensions que han d'alimentar els elements de les *PCB*, així com els seus consums.	24-10-2012	3	
4.2	Adaptar tensions d'entrada d'alimentació	Adaptar el sistema d'entrada d'alimentació per admetre tensions de 12 a 48 V.	25-10-2012	3	4.1
4.3	Adaptar tensions de sortida d'alimentació	Segons l'estudi de l'activitat 4.1, adaptar les diferents tensions de sortida d'alimentació.	26-10-2012	3	4.1
4.4	Instal·lar *software* de disseny i simulació de circuits	Instal·lació *software* disseny i simulació de circuits, ja sigui *TINA-TI* o *Schematics* de *Orcad*.	27-10-2012	1	
4.5	Implementar circuits i simulació	Dissenyar i implementar en software els circuits d'adaptació d'alimentació, així com simular-los.	27-10 → 29-10	12	4.1 4.2 4.3 4.4
4.6	Documentació PAC2	Redactar la informació de les activitats 4.1 a 4.5 a la memòria del TFC.	30-10 → 02-11	8,9	4.5

5	Entrega esborrany PAC2	Entrega del document PAC2 al consultor per tal que en faci una lectura i faci indicacions dels punts a corregir.	02-11-2012	0,1	
6	Correcció PAC2	Correcció del document PAC2 segons les indicacions del consultor.	05-11-2012	3	5
7	Lliurament PAC2	Lliurament a la bústia de l'aula del document PAC2.	06-11-2012	0,1	6
8	Definició del mapa de memòria *ModBus*	Definir les diferents instruccions *ModBus* que implementarà el nostre expansor d'entrades i sortides.	06-11 → 16-11	33,9	
8.1	Instruccions *ModBus* que implementarem	Definir les diferents instruccions *ModBus* que implementarà el nostre expansor d'entrades i sortides.	06-11 → 07-11	5,9	
8.2	Realitzar mapa de memòria	Realitzar el mapa de memòria al igual que els documents comercials existents.	08-11 → 12-11	3	8.1
8.3	Documentació PAC3	Redactar la informació de les activitats 8.1 i 8.2 a la memòria del TFC.	13-11 → 16-11	3	8.2
9	Elecció del microcontrolador i implementació del programa de control	Elecció del microcontrolador de l'empresa *Microchip* i implementació del programa de control.	17-11 → 26-11	32	
9.1	Elecció del microcontrolador	Elecció del microcontrolador de l'empresa *Microchip*, segons les característiques que ha de tenir l'expansor.	17-11-2012	5	
9.2	Instal·lar software de programació *PIC*	Instal·lar software de programació *PIC*, el *MPLAB* o similar.	18-11-2012	2	
9.3	Implementar programa	Implementar el programa de control de l'expansor.	18-11 → 22-11	15	9.2
9.4	Simular execució	Simular l'execució del programa de control de l'expansor.	23-11 → 24-11	8	9.3
9.5	Documentació PAC3	Redactar la informació de les activitats 9.1 a 9.4 a la memòria del TFC.	25-11 → 26-11	8	9.1 9.2 9.4
10	Disseny dels *layout* de les plaques *PCB*	Disseny dels *layout* de les plaques *PCB* que contindran tots els elements descrits en les altres tasques.	27-11 → 06-12	27	
10.1	Llistar diferents elements de les *PCB*	Llistar els diferents elements que contindran les *PCB*	27-11-2012	2	
10.2	Instal·lar software de disseny *PCB*	Instal·lar *software* de disseny *PCB*, *Eagle* 6.3.0 o similar.	28-11-2012	1	
10.3	Dissenyar *PCB*	Dissenyar plaques *PCB* amb *software Eagle*.	28-11 → 03-12	18	10.2
10.4	Documentació PAC3	Redactar la informació de les activitats 10.1 i 10.3 a la memòria del TFC.	04-12 → 05-12	6	10.3
11	Entrega esborrany PAC3	Entrega del document PAC3 al consultor per tal que en faci una lectura i faci indicacions dels punts a corregir.	07-12-2012	0,1	
12	Correcció PAC3	Correcció del document PAC3 segons les indicacions del consultor.	10-12-2012	3	11
13	Lliurament PAC3	Lliurament a la bústia de l'aula del document PAC3.	11-12-2012	0,1	12
14	Conclusions i ampliacions	Redactar les conclusions del TFC, les seves futures ampliacions i la seva valoració econòmica.	11-12 → 14-12	11,9	
14.1	Conclusions del TFC	Redactar les conclusions del TFC a les que arribem.	11-12 → 12-12	5,9	
14.2	Futures ampliacions	Redactar les futures ampliacions de l'expansor dissenyat.	13-12-2012	3	
14.3	Valoració econòmica	Fer una valoració econòmica del TFC realitzat i de l'expansor dissenyat.	14-12-2012	3	
15	Revisió dissenys circuits electrònics	Revisió dels dissenys dels circuits electrònics realitzats.	15-12-2012	3	
16	Revisió programació *PIC*	Revisió programació del microcontrolador *PIC*.	16-12-2012	4	
17	Revisió disseny *PCB*	Revisió dels dissenys de les plaques *PCB* realitzades.	17-12-2012	3	
18	Revisió documentació	Revisió de la redacció de la memòria del treball final de carrera.	18-12-2012	2,9	
19	Entrega esborrany memòria final	Entrega del document Memòria final al consultor per tal que en faci una lectura i faci indicacions dels punts a corregir.	18-12-2012	0,1	18
20	Presentació	Realització de la presentació del TFC mitjançant un arxiu de vídeo *mp4*.	19-12 → 02-01	38,1	
20.1	Instal·lació *software* edició vídeo	Instal·lació *software* edició vídeo *Adobe Premiere CS4*.	19-12-2012	1,5	
20.2	Instal·lació *software* gravació d'escriptori	Instal·lació d'un *software* de gravació de l'escriptori de l'ordinador, encara per a definir.	19-12-2012	1,5	
20.3	Preparació guió de la presentació	Redactar un guió per tal de gravar les diferents escenes que formaran part de la presentació, així com del text que les acompanyarà.	20-12 → 21-12	6	
20.4	Gravar les diferents escenes de la presentació	Gravar les diferents escenes de la presentació, mitjançant el *software* instal·lat en les activitats 20.1 i 20.2, segons el guió de l'activitat 20.3.	22-12 → 23-12	10	20.1 20.2 20.3

20.5	Primera edició general de la presentació	Primera edició general de la presentació mitjançant els vídeos de l'activitat 20.4.	26-12 → 30-12	18	20.4
20.6	Penjar vídeo a la xarxa	Penjar el vídeo de la presentació a *Youtube* o similar per poder passar el *link* al consultor perquè en faci una valoració.	30-12-2012	1	20.5
20.7	Passar *link* al consultor per a la seva revisió	Passar el *link* al consultor perquè en faci una valoració.	02-01-2013	0,1	20.6
21	Correcció memòria final	Correcció del document memòria final segons les indicacions del consultor.	03-01 → 04-01	3	
22	Correcció vídeo de presentació	Correcció del vídeo de presentació segons les indicacions del consultor.	05-01-2013	8,9	
23	Lliurament de la memòria i la presentació	Lliurament de la Memòria final del TFC i el vídeo de la presentació.	07-01-2013	0,1	21 22
24	Debat virtual	Debat virtual del Tribunal d'avaluació del TFC	23-01 → 25-01	4h	
24.1	Contestar a les possibles preguntes del Tribunal d'avaluació	Contestar a les possibles preguntes del Tribunal d'avaluació, termini de 24 hores per fer-ho.	25-01-2013	4h	

Taula 2 Temporització Tasques

1.4.1. Fites

Aquest treball final de carrera disposa d'una sèrie de fites, les quals es corresponen amb el lliurament de les diferents PAC, així com el lliurament de la memòria final i la seva presentació. També s'han definit com a fites del TFC l'entrega dels esborranys dels documents anteriors, així com el debat inicial de trobada i el debat virtual del tribunal d'avaluació. Per tant, les fites del TFC i les seves dates límit són:

Fita	***Data***
Videoconferència trobada d'inici	22-09-2012
Entrega esborrany Pla de treball	29-09-2012
Lliurament Pla de treball	02-10-2012
Entrega esborrany PAC2	02-11-2012
Lliurament PAC2	06-11-2012
Entrega esborrany PAC3	07-12-2012
Lliurament PAC3	11-12-2012
Entrega esborrany memòria final	18-12-2012
Entrega *link* presentació memòria	02-01-2013
Lliurament de la memòria i presentació	07-01-2013
Debat virtual del tribunal d'avaluació	25-01-2013

Taula 3 Fites TFC

1.4.2. Diagrama de Gantt

Segons la planificació de les tasques, activitats i fites, dels apartats anteriors, hem obtingut el diagrama de *Gantt* del TFC. Amb la intenció de donar el màxim detall es mostra en quatre fragments, corresponents a les diferents entregues de les PAC:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**1 Pla de Treball**	**12 días**	**mié 19/09/12**	**mar 02/10/12**
2	1.1 Descarregar enunciat	1 día	mié 19/09/12	mié 19/09/12
3	1.2 Llegir mòduls i enunciat TFC	5 días	jue 20/09/12	mar 25/09/12
4	1.3 Videoconferència trobada inicial	1 día	sáb 22/09/12	sáb 22/09/12
5	1.4 Lectura resum trobada d'inici	1 día	mié 26/09/12	mié 26/09/12
6	1.5 Definició de fites i tasques	1 día	mié 26/09/12	mié 26/09/12
7	1.6 Temporització de les tasques	1 día	jue 27/09/12	jue 27/09/12
8	1.7 Instal·lació de MS Project	0,1 horas	jue 27/09/12	jue 27/09/12
9	1.8 Elaboració del Diagrama de Gantt	1 día	vie 28/09/12	vie 28/09/12
10	1.9 Elaboració del pla de treball	1 día	sáb 29/09/12	sáb 29/09/12
11	1.10 Entrega esborrany pla de treball	0 días	sáb 29/09/12	sáb 29/09/12
12	1.11 Correcció pla de treball	2 días	lun 01/10/12	mar 02/10/12
13	1.12 Lliurament pla de treball	0 días	mar 02/10/12	mar 02/10/12

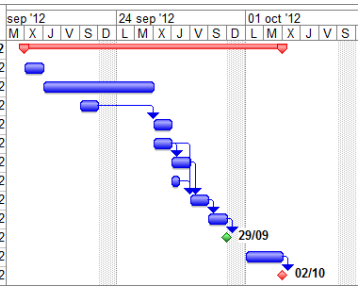


Figura 2 Diagrama de Gantt 1

Id	Nombre de tarea	Duración	Comienzo	Fin
14	**PAC2**	**29 días**	**mar 02/10/12**	**mar 06/11/12**
15	**2 Estudi del protocol ModBus i del bus RS-485**	**5 días**	**mar 02/10/12**	**dom 07/10/12**
16	2.1 Recerca d'informació protocol ModBus	1 día	mar 02/10/12	mar 02/10/12
17	2.2 Recerca d'informació bus RS-485	1 día	mié 03/10/12	mié 03/10/12
18	2.3 Documentació PAC2	3 días	jue 04/10/12	dom 07/10/12
19	**3 Disseny dels circuits d'adaptació d'entrades i sortides**	**14 días**	**lun 08/10/12**	**mar 23/10/12**
20	3.1 Adaptació tensions entrades analògiques 0-10V	2 días	lun 08/10/12	mar 09/10/12
21	3.2 Adaptació tensions entrades analògiques 4-20mA	1 día	mié 10/10/12	mié 10/10/12
22	3.3 Adaptació tensions sortides analògiques 0-10V	1 día	jue 11/10/12	jue 11/10/12
23	3.4 Altres alternatives descartades	1 día	sáb 13/10/12	sáb 13/10/12
24	3.5 Instal·lació software disseny i simulació de circuits	1 día	dom 14/10/12	dom 14/10/12
25	3.6 Implementar circuits i simulació	4 días	lun 15/10/12	jue 18/10/12
26	3.7 Documentació PAC2	3 días	vie 19/10/12	mar 23/10/12
27	**4 Disseny de la font d'alimentació**	**9 días**	**mié 24/10/12**	**vie 02/11/12**
28	4.1 Estudi alimentacions dels circuits de la PCB	1 día	mié 24/10/12	mié 24/10/12
29	4.2 Adaptar tensions d'entrada d'alimentació	1 día	jue 25/10/12	jue 25/10/12
30	4.3 Adaptar tensions de sortida d'alimentació	1 día	vie 26/10/12	vie 26/10/12
31	4.4 Instal·lar software de disseny i simulaicó de circuits	0 días	sáb 27/10/12	sáb 27/10/12
32	4.5 Implementar circuits i simulació	2 días	sáb 27/10/12	lun 29/10/12
33	4.6 Documentació PAC2	4 días	mar 30/10/12	vie 02/11/12
34	5 Entrega esborrany PAC2	0 días	vie 02/11/12	vie 02/11/12
35	6 Correcció PAC2	1 día	lun 05/11/12	lun 05/11/12
36	7 Lliurament PAC2	0 días	mar 06/11/12	mar 06/11/12

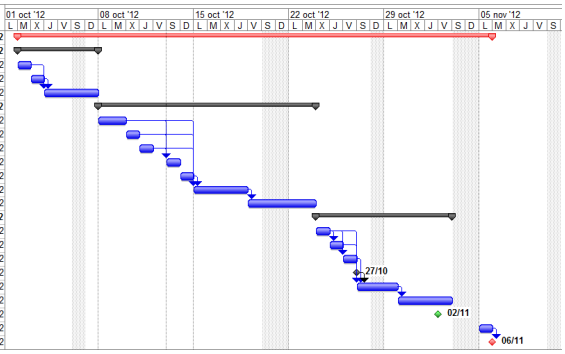


Figura 3 Diagrama de Gantt 2

Id	Nombre de tarea	Duración	Comienzo	Fin
37	**PAC3**	**29 días**	**mar 06/11/12**	**mar 11/12/12**
38	**8 Definició del mapa de memòria ModBus**	**9 días**	**mar 06/11/12**	**vie 16/11/12**
39	8.1 Instruccions ModBus que implementarem	2 días	mar 06/11/12	mié 07/11/12
40	8.2 Realitzar mapa de memòria	3 días	jue 08/11/12	lun 12/11/12
41	8.3 Documentació PAC3	4 días	mar 13/11/12	vie 16/11/12
42	**9 Elecció del processador i implementació del programa de control**	**10 días**	**sáb 17/11/12**	**lun 26/11/12**
43	9.1 Elecció del processador	1 día	sáb 17/11/12	sáb 17/11/12
44	9.2 Instal·lar software de programació PIC	0 días	dom 18/11/12	dom 18/11/12
45	9.3 Implementar programa	5 días	dom 18/11/12	jue 22/11/12
46	9.4 Simular execució	1 día	sáb 24/11/12	sáb 24/11/12
47	9.5 Documentació PAC3	2 días	dom 25/11/12	lun 26/11/12
48	**10 Disseny del layout de la placa PCB**	**7 días**	**mar 27/11/12**	**mié 05/12/12**
49	10.1 Llistar diferents elements del PCB	1 día	mar 27/11/12	mar 27/11/12
50	10.2 Instal·lar software de disseny PCB	0 días	mié 28/11/12	mié 28/11/12
51	10.3 Dissenyar PCB	4 días	mié 28/11/12	lun 03/12/12
52	10.4 Documentació PAC3	2 días	mar 04/12/12	mié 05/12/12
53	11 Entrega esborrany PAC3	0 días	vie 07/12/12	vie 07/12/12
54	12 Correcció PAC3	1 día	lun 10/12/12	lun 10/12/12
55	13 Lliurament PAC3	0 días	mar 11/12/12	mar 11/12/12

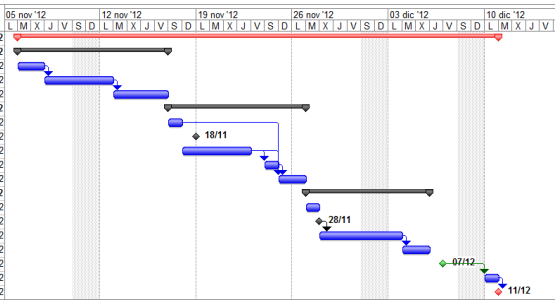


Figura 4 Diagrama de Gantt 3

Id	Nombre de tarea	Duración	Comienzo	Fin
56	**Memoria i presentació**	**25 días**	**mar 11/12/12**	**lun 07/01/13**
57	**14 Conclusions i ampliacions**	**4 días**	**mar 11/12/12**	**vie 14/12/12**
58	14.1 Conclusions TFC	2 días	mar 11/12/12	mié 12/12/12
59	14.2 Futures ampliacions	1 día	jue 13/12/12	jue 13/12/12
60	14.3 Valoració econòmica	1 día	vie 14/12/12	vie 14/12/12
61	15 Revisió dissenys circuits electrònics	1 día	sáb 15/12/12	sáb 15/12/12
62	16 Revisió programació PIC	1 día	dom 16/12/12	dom 16/12/12
63	17 Revisió disseny PCB	1 día	lun 17/12/12	lun 17/12/12
64	18 Revisió documentació	1 día	mar 18/12/12	mar 18/12/12
65	19 Entrega esborrany memòria final	0 días	mar 18/12/12	mar 18/12/12
66	**20 Presentació**	**13 días**	**mié 19/12/12**	**mié 02/01/13**
67	20.1 Instal·lació software edició vídeo	1 día	mié 19/12/12	mié 19/12/12
68	20.2 Instal·lació software gravació d'escriptori	1 día	mié 19/12/12	mié 19/12/12
69	20.3 Preparar guió de la presentació	2 días	jue 20/12/12	vie 21/12/12
70	20.4 Gravar les diferents escenes de la presentació	2 días	sáb 22/12/12	dom 23/12/12
71	20.5 Primera edició general de la presentació	4 días	mié 26/12/12	dom 30/12/12
72	20.6 Penjar vídeo a la xarxa	0 días	dom 30/12/12	dom 30/12/12
73	20.7 Passar link al consultor per a la seva revisió	0 días	mié 02/01/13	mié 02/01/13
74	21 Correcció memòria final	2 días	jue 03/01/13	vie 04/01/13
75	22 Correcció vídeo de presentació	1 día	sáb 05/01/13	sáb 05/01/13
76	23 Lliurament de la memòria i la presentació	0 días	lun 07/01/13	lun 07/01/13
77	**24 Debat Virtual**	**3 días**	**mié 23/01/13**	**vie 25/01/13**
78	24.1 Contestar a les possibles preguntes del Tribunal d'avaluació	3 días	mié 23/01/13	vie 25/01/13

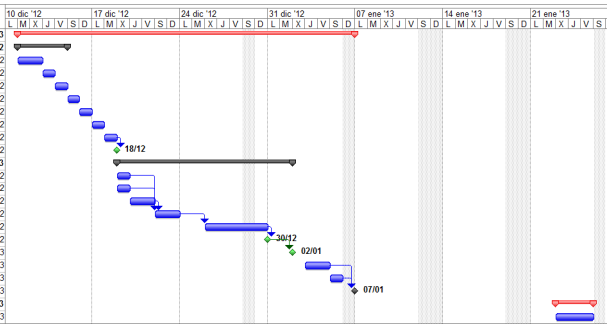


Figura 5 Diagrama de Gantt 4

1.4.3. Material a utilitzar

El material que necessitarem per a realitzar aquest treball final de carrera consisteix bàsicament de programari i un ordinador personal. Concretament utilitzaré un equip PC de sobre taula amb processador *Intel Core i3* de 2,93GHz amb 8GB de memòria *RAM* amb el sistema operatiu *Windows 7 Ultimate* de 64 bits.

La redacció dels diferents documents es realitzarà amb la suite *Microsoft Office 2007*. Per a la realització del vídeo s'utilitzarà la suite de *Adobe Master Collection CS4*. Concretament s'utilitzaran els programes *Premiere Pro* i *Media Encoder*.

Per al disseny i simulació de circuits analògics s'utilitzarà el programa *TINA-TI* de *Texas Instruments*, el qual és gratuït. Pel disseny de les plaques *PCB* s'utilitzarà el programa *Eagle 6.3.0*, el qual és *freeware*. Finalment per a la programació del *PIC* s'utilitzarà el programa *CCS C Compiler* de la companyia *Custom Computer Service, Inc.*, la *demo* del qual dura 45 dies. Per simular el PIC s'utilitzarà el programa *Real Pic Simulator* de *Digital Electro Soft*, la *demo* del qual dura 30 dies.

1.4.4. Incidències i Riscos

Les possibles incidències i riscos d'aquest TFC són:

- ***Indisposició***: Al treballar a Aena a vegades em desplacen de província per a realitzar cursos, la mitjana d'aquests cursos és de 3 a 4 dies. Aquests dies són difícils de planificar, i la solució és intentar treure temps els dies abans del viatge per tal d'avançar les tasques.
- ***Fallada PC***: El PC que utilitzo ara ja té alguns anys. Per tal d'evitar la pèrdua de dades es treballarà amb un disc dur extern, i també s'aniran fent copies de seguretat. Un lloc per ubicar aquestes copies és el servei de *Google*, el *Google Drive*, el qual admet fins a 5GB d'emmagatzematge *online* gratuïts.
- ***Fallada de connexió xarxa***: Al viure en una illa hi ha certs dies a l'any que es perd la comunicació d'Internet. Si la línea ADSL falla es disposa d'un mòdem *USB 3G*.

1.5. Productes Obtinguts

Els productes obtinguts en la realització d'aquest TFC són:

- Memòria treball final de carrera:
  `piris_memoria.pdf`
- Vídeo de presentació del treball final de carrera:
  `piris_presentacio.mp4`
- Arxius del programa de control del *PIC*, realitzat amb el compilador *CCS C*:
  `PIC16F786A.h`, `piris_modbus.c`, `piris_modbus.hex`

2. Estudi protocol ModBus i bus RS-485

En aquest apartat definirem el protocol *ModBus*[7] i el bus *RS-485*[1,7]. L'estàndard *ModBus* defineix els missatges de les aplicacions que seran transmesos per un medi físic. La comunicació entre dos dispositius serà del tipus mestre - esclau, i la forma de connectar-se el definirà la capa 1 del model *OSI*, la capa física. A la capa física es poden utilitzar diferents busos o interfícies, però el més normal és utilitzar un bus sèrie, concretament el *RS-485* o el *RS-232* (només utilitzat per a molt curtes distàncies).

Concretament, aquest protocol defineix les característiques de les capes 2 i 7 del model *OSI*. A la capa 7, o capa d'aplicació, es defineixen les funcions, les quals seran utilitzades per enviar i rebre dades entre els dispositius, tal i com veurem a l'apartat 5. A la capa 2, capa d'enllaç, es defineixen els paquets o datagrames que seran enviats o rebuts, tal i com veurem en aquest apartat. A la següent figura es compara el model *OSI* amb el model *ModBus*:

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)

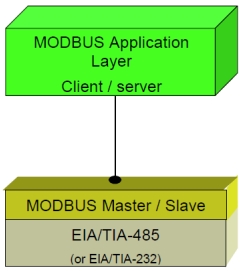


Figura 6 Capes ModBus - OSI

2.1. Protocol ModBus

La capa d'enllaç del *ModBus* només permet que hi hagi un mestre i fins a 247 esclaus connectats al mateix bus. La comunicació és sempre inicialitzada pel mestre, el què implica que cap esclau pot començar una transmissió. Els esclaus tampoc es poden comunicar entre ells. Per tant, l'esclau només respondrà a la petició realitzada pel mestre.

El mestre es pot comunicar en mode *unicast*, és a dir, es comunica amb un únic esclau, el qual es diferencia dels altres esclaus mitjançant una única adreça. L'altre mode és *broadcast*, és a dir, el mestre es comunica amb tots els esclaus a l'hora, ho fa mitjançant l'adreça 0. En el mode *unicast* hi ha la transferència de dos missatges, una petició de mestre - esclau i una resposta d'esclau - mestre. En canvi en el mode *broadcast* només s'envia un missatge de mestre - esclaus, el mestre no espera cap resposta:

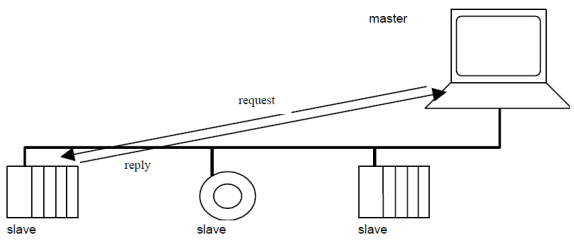


Figura 7 Mode unicast

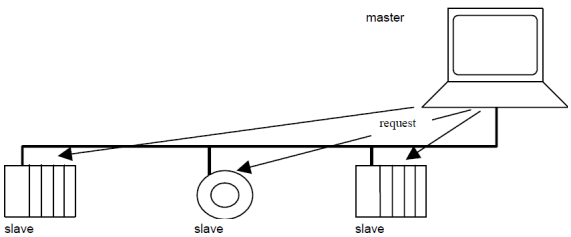


Figura 8 Mode broadcast

ModBus accepta fins a 256 adreces diferents, com ja hem dit, l'adreça 0 s'utilitza en el mode *broadcast*, les adreces de la 1 a la 247 s'utilitzen per a enumerar els esclaus, i les adreces de la 248 fins la 255 estan reservades.

La capa d'aplicació de *ModBus* defineix el *Protocol Data Unit* (*PDU*), què és independent de les altres capes. Aquest *PDU* està compost per 1 byte que defineix la funció a tractar, i de 0 a 252 bytes que contenen les dades dels paràmetres transmesos.

La capa d'enllaç de *ModBus* construeix el datagrama a transferir afegint 1 byte al principi del *PDU*, el qual conté l'adreça de l'esclau a qui anirà destinat aquest missatge. També afegeix al final del *PDU* 2 bytes que contenen el *Cyclical Redundancy Checking* (*CRC*) per tal de poder trobar errors en la recepció del missatge (el seu càlcul s'explica al punt 5.1). El datagrama tindrà un màxim de 256 bytes de longitud.

Existeixen dos modes de transmissió, el *Remot Terminal Unit* (*RTU*) i el mode *ASCII*. El primer és el més comú de l'estàndard, el qual realitza la transmissió de bits o bytes. En canvi, el segon realitza la transmissió de caràcters en lloc de bytes, per cada byte envia dos caràcters *ASCII*.

En el mode *RTU*, què serà el que utilitzarem en aquest TFC, els bytes es transmeten bit per bit, afegint un bit de començament, un bit de paritat (opcional), i un bit de finalització. Primer es transmet el bit menys significatiu, i finalitza amb el més significatiu:

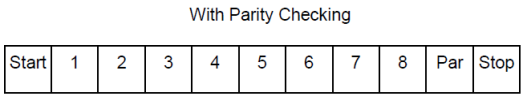


Figura 9 Seqüència de bits en el mode RTU

La distribució dels bytes dintre d'un datagrama és:

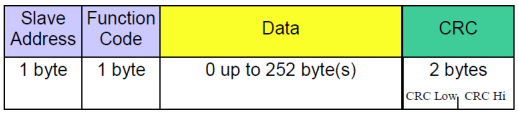


Figura 10 Datagrama RTU ModBus

En la transmissió de datagrames hi ha d'haver un espai de silenci entre elles, aproximadament un mínim igual al temps que es tarda en enviar 3,5 caràcters (bytes). Depèn de la velocitat a la qual funcioni el bus *RS-485*.

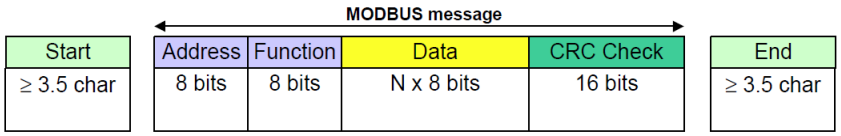


Figura 11 Datagrama RTU ModBus en el temps

2.2. Bus RS-485

El bus *RS-485* és un estàndard creat el 1983 pel comitè *EIA-485*, aquest estàndard és un bus de comunicacions que s'ubica a la capa física del model *OSI*. Aquest bus es pot utilitzar per a transmetre dades punt a punt, o punt a multipunt. Al utilitzar tensions diferencials en la seva transmissió fa que sigui un bus fort front a les interferències o sorolls captats pel cablejat.

Aquest bus funciona amb un rang de voltatge de $-7V$ fins a $12V$, per tant, el diferencial entre els dos cables és de $5V$. Aquest fet fa que quan un bus capti una interferència, aquesta afectarà als dos cables per igual, fent que el seu diferencial sigui sempre de $5V$.

En un sistema *ModBus*, el dispositiu mestre i un o més dispositius esclaus es poden comunicar mitjançant aquest bus. Cada dispositiu pot ser connectat amb un repartidor passiu o actiu, o directament entrant i sortint el bus del dispositiu (mode *Daisy-Chain*). Un repartidor passiu és aquell que connecta el bus amb el dispositiu, però és el dispositiu el qui intercanvia els senyals transmesos. Un repartidor actiu és aquell on ell mateix intercanvia els senyals transmesos i els adequa pel dispositiu al què està connectat. Un exemple de les connexions seria:

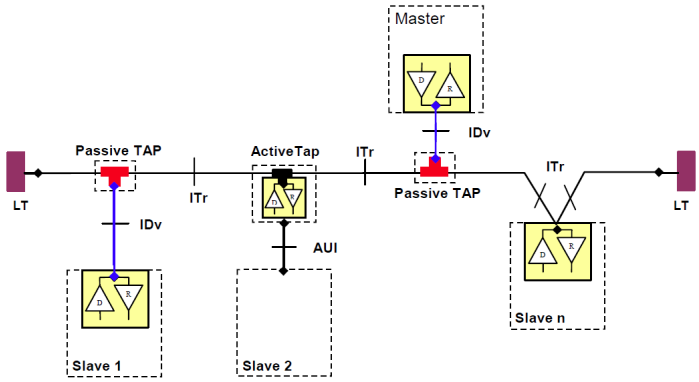


Figura 12 Tipus connexió bus RS-485

En aquest TFC utilitzarem el mode de connexió *Daisy-Chain*, així que en el mateix expansor haurem d'adequar els senyals pel bus *RS-485*. La topologia que utilitzarem en aquest TFC és la de dos cables:

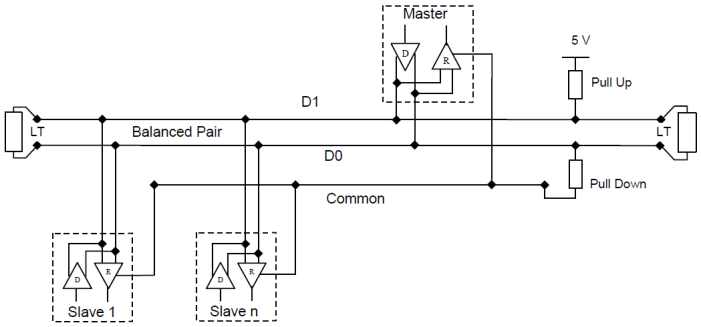


Figura 13 Topologia 2 cables Half-Duplex

Finalment, en un sistema multipunt només es pot instal·lar fins a 32 dispositius sense haver d'utilitzar cap repetidor. Al bus se l'hi ha de connectar uns finals de línea, és a dir, se l'hi ha de connectar una càrrega entre els extrems dels dos cables. Aquesta càrrega està formada per una resistència de $120\ \Omega$ en sèrie a un condensador de $1nF$ mínim, depenen de la impedància del

cable. Quan el bus no té activitat és propens a capturar interferències exteriors, és per això que s'utilitzen dues resistències com a *pull-up* i *pull-down*, les quals donen respectivament una referència a $5V$ i massa.

Tenint en compte totes aquestes consideracions, en el nostre TFC necessitem comunicar el nostre expansor al bus *ModBus RTU* sobre *RS-485*. No tindrem en compte ni el mestre ni l'estat del bus. Per tant, el nostre dispositiu actuarà com a esclau. L'expansor utilitzarà un transceptor *RS-485* aïllat elèctricament, d'aquesta manera ens assegurarem que no introduirem pertorbacions elèctriques en el nostre dispositiu. El transceptor escollit és el ***ISO3088***[8] de *Texas Instruments*, el qual s'alimenta a $5V$, i transforma els senyals del bus *RS-485* a uns senyals lògics (de 0 a $5V$) que poden comunicar-se mitjançant la *USART* dels microcontroladors *PIC*. El consum màxim és de $\pm 60mA$. La seva velocitat màxima és de $20Mbps$, és *Half-duplex* i el diagrama de pins és:

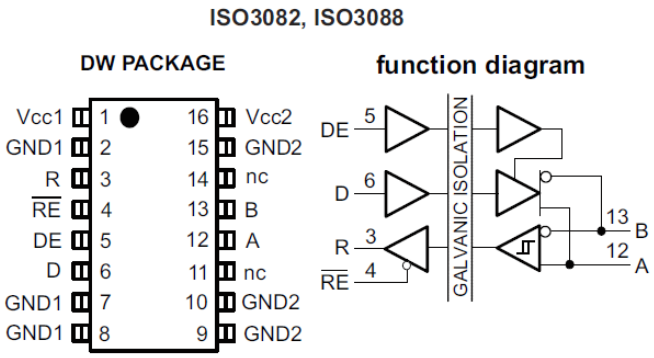


Figura 14 Transceptor RS-485 ISO3088

Els pins 12 i 13 aniran connectats als dos cables del bus *RS-485*, anomenats A i B respectivament, a la figura 3 s'anomenen $D0$ i $D1$. Els pins 3 i 4, anomenats R i $\overline{RE}$ són els pins de recepció, el pin 3 anirà connectat a l'entrada Rx del microcontrolador. Els pins 5 i 6, anomenats DE i D són els pins de transmissió, el pin 6 anirà connectat a la sortida Tx del microcontrolador. Els pins 4 i 5 aniran connectats a la mateixa sortida digital del microcontrolador, i quan el dispositiu vulgui rebre dades, desactivarà la tensió al pin DE, fent que el pin de recepció $\overline{RE}$ valgui 1. Si vol enviar dades activarà la tensió al pin DE, fent que el pin $\overline{RE}$ valgui 0. D'aquesta manera ja tenim connectada la *USART* del microcontrolador al bus *RS-485*, i aïllada elèctricament. Per a les seves connexions veure l'apartat 11 d'annex.

Tal i com hem dit anteriorment, els voltatges del bus *RS-485* són de $-7V$ i $12V$. Per a transmetre un 1 lògic s'ha de complir que $A - B < -0{,}3V$, i per a transmetre un 0 lògic s'ha de complir que A$-B > 0{,}3V$.

3. Disseny circuits adaptació entrades i sortides analògiques

En aquest apartat descriurem com adaptar els senyals analògics d'entrada i de sortida del nostre expansor amb els voltatges que admeten alguns microcontroladors *PIC*, concretament de 0 a $5V$.

Primer hi ha que tenir en compte l'adaptació de impedàncies d'entrada de les senyals analògiques de l'expansor. Per aconseguir-ho utilitzarem amplificadors operacionals[9] (*AO*), els quals disposen d'una impedància d'entrada molt elevada (entorn dels $10\ T\Omega$) i una impedància de sortida mínima (entorn dels $100\ \Omega$). D'aquesta manera no distorsionarem els senyals d'entrada.

L'expansor ha de donar solució a dos tipus diferents d'entrades analògiques. Un primer tipus són aquelles on els sensors connectats proporcionen un valor entre dues tensions. Un segon tipus són aquelles on els sensors connectats proporcionen un valor entre dues intensitats, com que els *PIC* només accepten tensions com a entrada, haurem de convertir aquesta intensitat en una tensió. La intensitat serà de $4mA$ fins a $20mA$, al convertir en tensió obtindrem valors de 0 a $5V$.

Les tensions d'entrada seran de 0 a $10V$, i per tant hi haurà que reduir aquestes tensions mitjançant *AO*. Al utilitzar amplificadors inversors també hi haurà que canviar la polaritat de la tensió, mitjançant un altre amplificador operacional. Al haver d'utilitzar tres *AO* per cada entrada, utilitzarem amplificadors de la família ***OPA4277***[10] de *Texas Instruments*, que disposen de quatre amplificadors operacionals dintre del mateix encapsulat. Aquesta família presenta un voltatge de *offset* molt baix, de l'ordre dels $\pm 10\ \mu V$, amb un màxim de $\pm 50\ \mu V$. Es poden alimentar amb una tensió de $\pm 15\ V$. Els voltatges màxims a la seva sortida seran de $-14{,}5V$ fins a $13{,}8V$. El seu consum màxim és d'uns $\pm 3{,}6mA$. Per tant, per a una entrada de $0V$ obtindrem, més o menys, $\pm 10\ \mu V$ a la sortida de l'etapa d'adaptació. I com que en els nostres circuits les tensions màximes a tractar seran de $\pm 10\ V$, aquests amplificadors podran proporcionar la tensió adequada a la seva sortida.

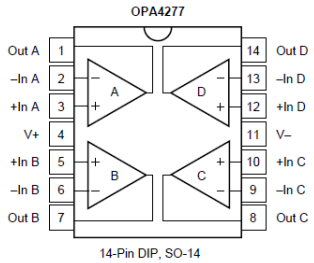


Figura 15 Diagrama amplificador operacional OPA4277

Si mirem les característiques dels microcontroladors *PIC*, podem veure com aquests tenen una certa tolerància a les seves entrades, concretament per a la família *PIC16* tenim que la tensió mínima és de $-0{,}3V$ i la seva tensió màxima és de $5{,}3V$. Per tant, si a la sortida de l'etapa d'adaptació d'entrades obtenim un voltatge de $-10\ \mu V$, el *PIC* pot suportar aquesta tensió negativa.

Les entrades analògiques han de tenir una precisió de 10 bits mínima. Els *PIC* de la família 16 disposen de convertidors analògics digitals (*CAD*) de 10 bits, amb un rang de $0\ a\ 5V$. Per tant, la sensibilitat del convertidor serà:

$$Sensibilitat = \frac{5V}{2^{10}} = 4{,}88mV \approx 5mV$$

Les tensions de $\pm 50\ \mu V$ seran codificades com a $0V$, ja que el següent pas/salt és als $4{,}88mV$.

Per a les sortides analògiques utilitzarem les sortides *PWM* del microcontrolador, a les quals incorporarem dos filtres passabaix per a estabilitzar la sortida i reduir la component alterna del senyal. Utilitzarem un amplificador operacional per a duplicar la tensió de sortida, en mode amplificador no inversor, i així aconseguir els $0 - 10V$. Aquestes sortides han de suportar una intensitat màxima de $200mA$, per tant l'amplificador escollit és el ***OPA551***[11] el qual pot proporcionar aquesta intensitat. El seu consum és de $\pm 10mA$.

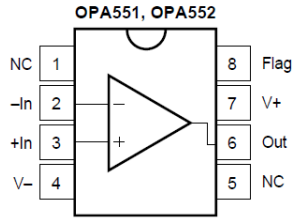


Figura 16 Diagrama amplificador operacional OPA551

Utilitzarem resistències de la sèrie *E24* de la norma *IEC60063*, què tenen una tolerància de $\pm 5\ \%$. Finalment, degut a què la pràctica és diferent que la teoria, simularem els circuits d'entrada i sortida, i podem veure com els valors obtinguts difereixen dels teòrics.

3.1. Adaptació entrades analògiques 0-10V

Aquest tipus d'entrada ens proporciona una tensió entre 0 i $10V$. Aquestes tensions hauran de ser reduïdes, ja que els *PIC* només accepten voltatges de 0 a $5V$ a les seves entrades analògiques. Per aconseguir-ho ho farem a partir de tres etapes utilitzant amplificadors operacionals.

Un primer *AO* actuarà com a seguidor de tensió, el qual proporciona la mateixa tensió d'entrada a la seva sortida. Aquest amplificador és utilitzat com a adaptador d'impedàncies:

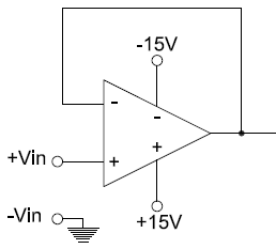


Figura 17 Seguidor de tensió

Un segon *AO* actuarà com a amplificador inversor. Hem de convertir els $10V$ en $-5V$. La sortida d'aquest *AO* és $V_{out2} = -V_{in} \cdot \frac{R_2}{R_1}$, per tant necessitem calcular aquest guany:

$$V_{out2} = -V_{in} \cdot \frac{R_2}{R_1}, \quad \frac{V_{out2}}{-V_{in}} = \frac{R_2}{R_1}, \quad \frac{-5V}{-10V} = \frac{R_2}{R_1}, \quad \frac{R_2}{R_1} = \frac{5}{10} = \frac{1}{2}$$

Uns possibles valors teòrics són $R_1 = 2\ k\Omega$ i $R_2 = 1\ k\Omega$. Utilitzant aquesta relació obtindrem els voltatges de sortida finals de:

$$V_{out2} = -0V \cdot \frac{1\ k\Omega}{2\ k\Omega} = 0V, \quad \text{o} \quad V_{out2} = -10V \cdot \frac{1\ k\Omega}{2\ k\Omega} = -5V$$

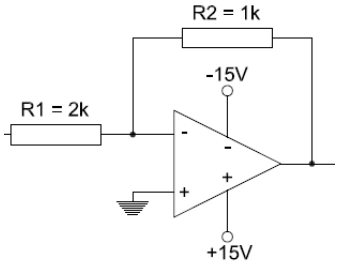


Figura 18 Amplificador inversor G = -1/2

Utilitzant un tercer *AO* com a amplificador inversor de guany unitat (totes les resistències són del mateix valor), convertirem aquestes tensions negatives en positives:

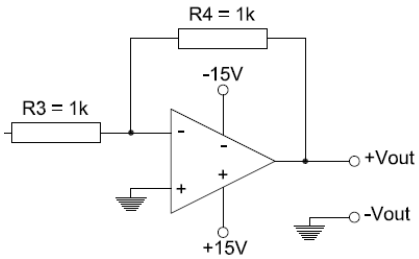


Figura 19 Amplificador inversor G = -1/1

$$\boxed{V_{out} = 0V \cdot \frac{1\ k\Omega}{1\ k\Omega} = 0V} \quad \text{o} \quad \boxed{V_{out} = -\left(-5V \cdot \frac{1\ k\Omega}{1\ k\Omega}\right) = 5V}$$

El circuit d'adaptació teòric és:

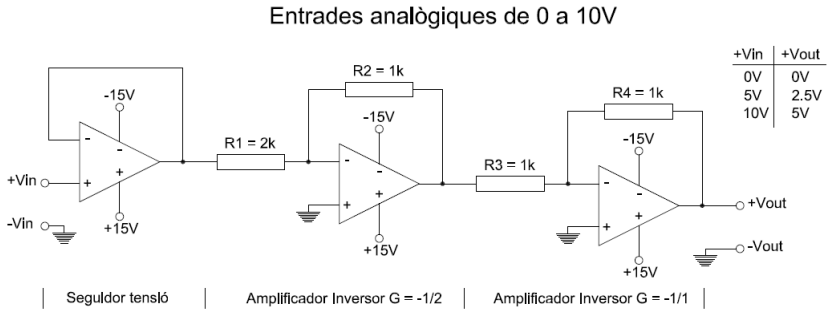


Figura 20 Entrades analògiques de tensió teòriques

Com podem veure, per una entrada de $0V$ s'obtindran $0V$ a la sortida, per una entrada de $5V$ tindrem $2{,}5V$, i finalment per a una entrada de $10V$ obtindrem una sortida de $5V$. Per tant, estarem entre els límits del *PIC*, de 0 a $5V$.

Al utilitzar components reals tenim certes limitacions i toleràncies. A la següent simulació, utilitzant el programa *TINA-TI* de *Texas Instruments*, veurem l'efecte de *Offset* dels amplificadors operacionals quan la tensió d'entrada és de $0V$. Tenint en compte que l'entrada d'alimentació de l'expansor és de 12 a $48V$, alimentarem els amplificadors a la tensió de $\pm 15\ V$. El circuit utilitzant components reals amb *TINA-TI* és:

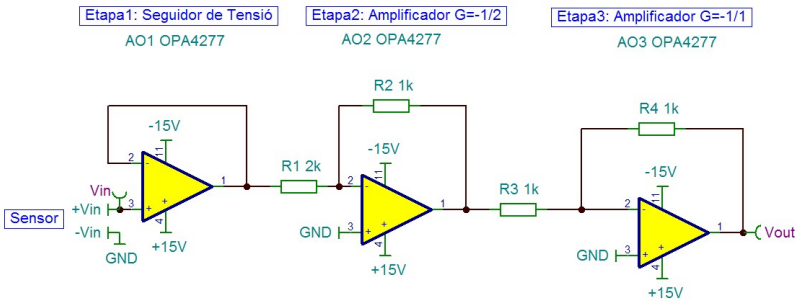


Figura 21 Entrades analògiques de tensió reals

Simulant el circuit per unes tensions d'entrada de 0 a $10V$ obtenim:

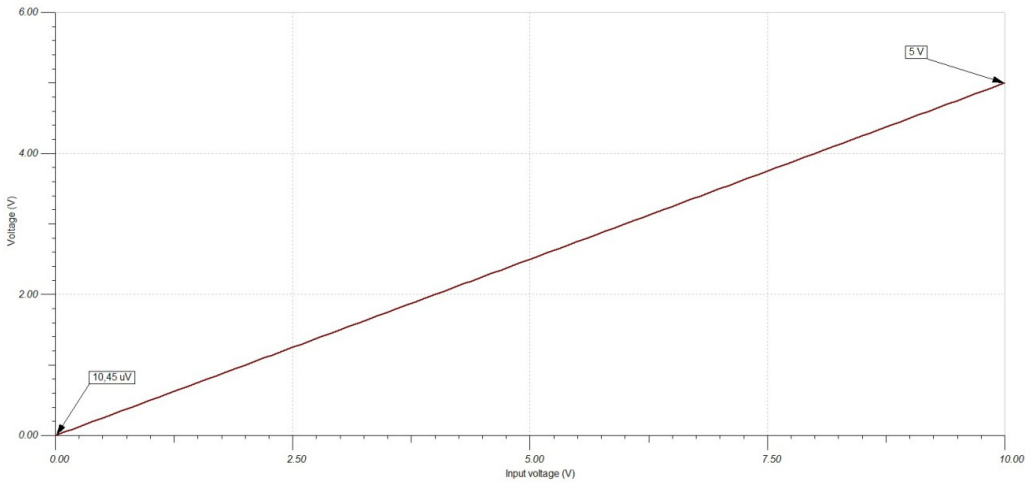


Figura 22 Simulació entrades analògiques de 0-10V

Com podem observar per a $10V$ obtenim $5V$, i en el cas de $0V$ la sortida és precisament la tensió *d'offset* dels *AO* utilitzats, què ronda els $\pm 10\ \mu V$, que en aquest cas és de $+10{,}45\ \mu V$, el *CAD* del microcontrolador ho interpretarà com a $0V$.

3.2. Adaptació entrades analògiques 4-20mA

Aquest tipus d'entrada ens proporcionen unes intensitats de $4mA$ fins a $20mA$, el qual l'haurem de convertir en tensió, ja que els *PIC* només accepten voltatges de 0 a $5V$ a les seves entrades analògiques. Per aconseguir-ho, ho farem a partir de tres etapes utilitzant amplificadors operacionals.

Un primer *AO* actuarà com a convertidor de corrent a tensió. Aquests convertidors també inverteixen la tensió de sortida, ja que es compleix que $V_{out} = -R_1 \cdot I_{in}$. El valor mínim el

tindrem per a $4mA$ i el valor màxim per a $20mA$. Si utilitzem una resistència de 250Ω, tindrem:

$$V = -250\Omega \cdot 4mA = -1V, \quad \text{o} \quad V = -250\Omega \cdot 20mA = -5V$$

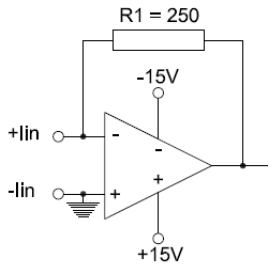


Figura 23 Convertidor corrent - tensió

Al obtenir una tensió de $-1V$ en lloc de $0V$ per a una entrada de $4mA$, veiem que aquest senyal té un *offset* de 1. Utilitzarem un segon *AO* diferencial, què restarà aquest volt demés del senyal. Si utilitzem totes les resistències que intervenen del mateix valor es complirà que $V_{out} = V_- - V_+$, és a dir, la sortida serà igual a l'entrada inversora menys l'entrada no inversora. Per tant, necessitem una font de $-1V$ per aconseguir-ho. A la sortida d'aquest segon *AO* tindrem:

$$V_{out} = (-1V) - (-1V) = 0V, \quad \text{o} \quad V_{out} = (-5V) - (-1V) = -4V$$

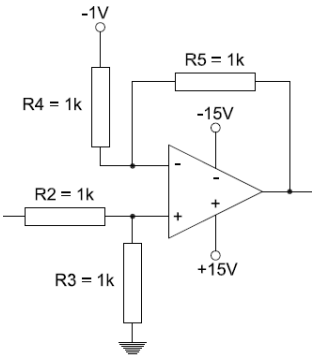


Figura 24 Amplificador diferencial 1V

Finalment necessitem un tercer *AO* que actuarà com a amplificador inversor. Hem de convertir els $-4V$ en $5V$. La sortida d'aquest *AO* és $V_{out} = -V_{in} \cdot \frac{R_7}{R_6}$, per tant necessitem calcular aquest guany:

$$V_{out} = -V_{in} \cdot \frac{R_7}{R_6}, \quad \frac{V_{out}}{-V_{in}} = \frac{R_7}{R_6}, \quad \frac{5V}{-(-4V)} = \frac{R_7}{R_6}, \quad \frac{R_7}{R_6} = \frac{5}{4}$$

Uns possibles valors teòrics són $R_6 = 4\ k\Omega$ i $R_7 = 5\ k\Omega$. Utilitzant aquesta relació obtindrem els voltatges de sortida finals de:

$$\boxed{V_{out} = -0V \cdot \frac{5\ k\Omega}{4\ k\Omega} = 0V} \quad \text{o} \quad \boxed{V_{out} = -(-4V) \cdot \frac{5\ k\Omega}{4\ k\Omega} = 5V}$$

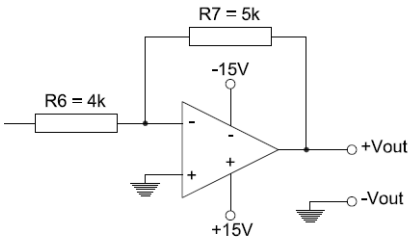


Figura 25 Amplificador inversor G = -5/4

El circuit d'adaptació teòric és:

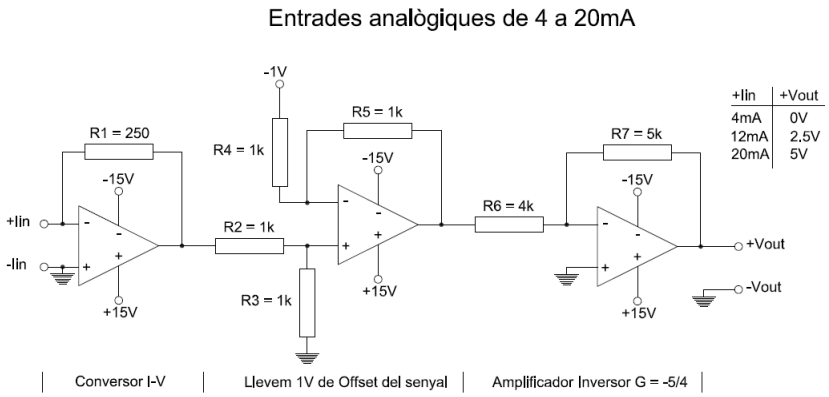


Figura 26 Entrades analògiques de mA teòriques

Com podem veure, per una entrada de $4mA$ s'obtindran $0V$ a la sortida, per una entrada de $12mA$ tindrem $2{,}5V$, i finalment per a una entrada de $20mA$ obtindrem una sortida de $5V$. Per tant, estarem entre els límits del *PIC*, de 0 a $5V$.

Al igual que abans, al utilitzar components reals tenim certes limitacions i toleràncies. A la següent simulació veurem l'efecte de *offset* dels amplificadors operacionals quan la intensitat d'entrada és de $4mA$. Els amplificadors s'alimentaran a $\pm 15\ V$. La resistència de $4k$ serà substituïda per dues resistències en sèrie de $2k$ cadascuna, al igual que la resistència de $5k$ què serà substituïda per una de $2k$ i una altre de $3k$. El circuit utilitzant components reals amb *TINA-TI* és:

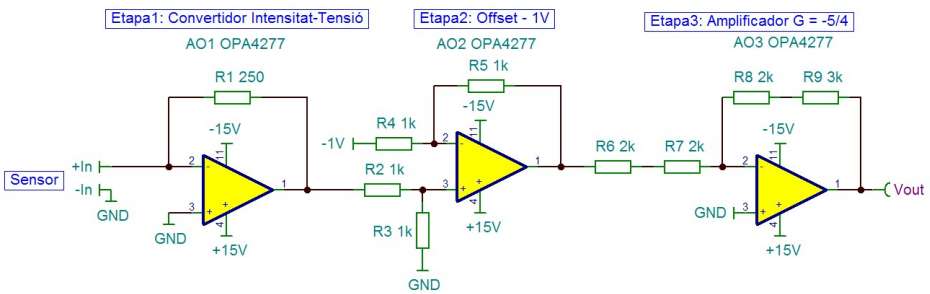


Figura 27 Entrades analògiques de mA reals

Simulant el circuit per unes intensitats d'entrada de $4mA$ a $20mA$ obtenim:

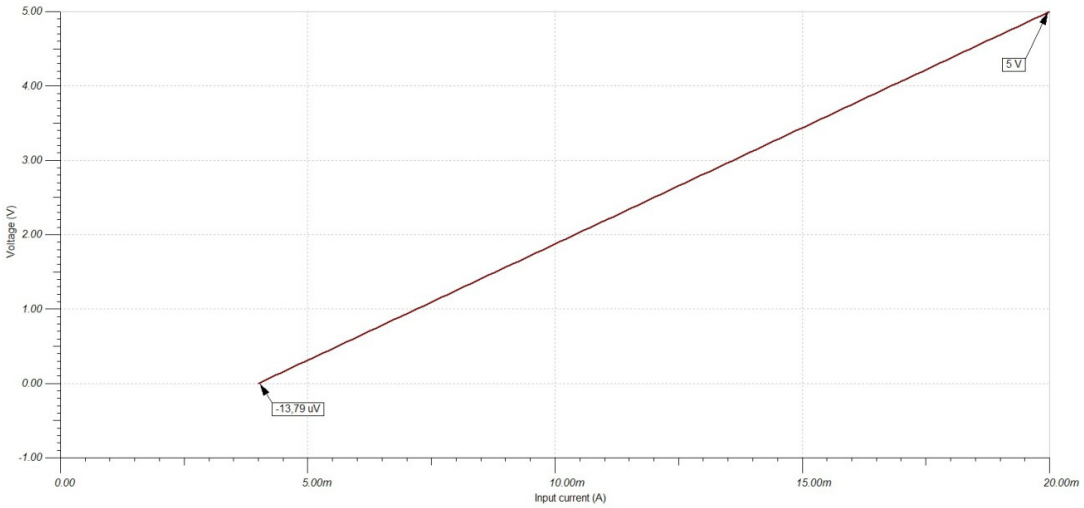


Figura 28 Simulació entrades 4-20mA

Com podem observar en el cas de $20mA$ obtenim $5V$, i en el cas de $4mA$ la sortida és precisament la tensió de *offset* dels *AO* utilitzats, què ronda els $\pm 10\ \mu V$, en aquest cas és de $-13{,}79\ \mu V$, el *CAD* del microcontrolador ho interpretarà com a $0V$.

3.3. Adaptació sortides analògiques 0-10V

Els microcontroladors solen incorporar sortides *PWM* (*Pulse Width Modulation*), aquestes sortides aporten un senyal modulat per ampla de pols, és a dir, es pot modificar l'ampla de banda del senyal. Segons la nota d'aplicació AN538[12] de *Microchip* i la *Application Report* SPRAA88A[13] de *Texas Instruments*, aquest senyal de polsos es pot utilitzar per a generar una sortida analògica. Per tant, els senyals *PWM* poden ser utilitzats per a crear un convertidor digital - analògic juntament amb resistències i condensadors formant un filtre passabaix, eliminant les freqüències no desitjades, fent que la sortida sigui el més estable possible.

Una característica[14] dels senyals *PWM* és que es pot modificar el cicle de treball (*duty cycle*) sense modificar la freqüència del *PWM*. A mesura que el *duty cycle* augmenta, la potencia entregada pel *PWM* també augmenta. El *duty cycle* és igual al temps que el senyal està activat dividit pel període del *PWM*:

$$duty\ cycle = \frac{T_{on}}{T_{PWM}}$$

A la següent figura podem veure com afecta el *duty cycle* al valor mig del senyal (V_{RMS}):

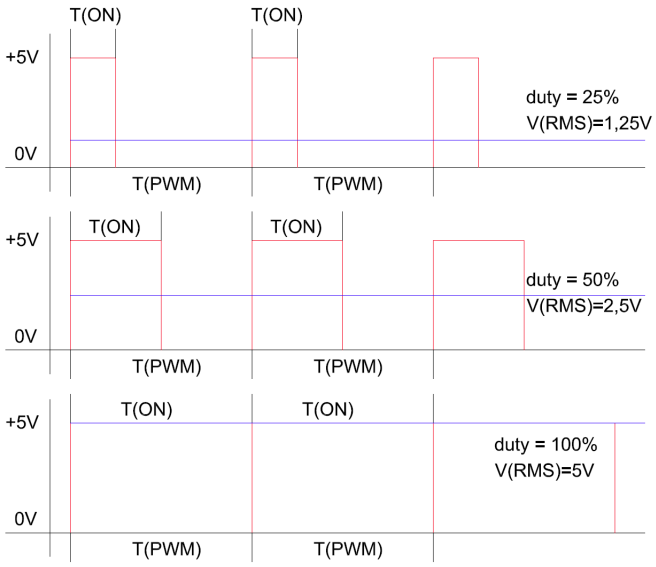


Figura 29 Diferents cicles de treball de PWM

El valor mig del senyal és igual al valor màxim del pols pel cicle de treball, o *duty cycle*:

$$V_{RMS} = V_{max} \cdot duty\ cycle$$

Per tant, controlant el temps T_{on} podem variar la tensió de sortida del nostre convertidor digital - analògic mitjançant *PWM*. Això és el que podrem veure a l'apartat 6 de la programació del *PIC*.

Aquest senyal modulat per ampla de pols presenta harmònics a les freqüències $f = K/T_{PWM}$, on K és un nombre enter. Aquestes freqüències són el soroll no desitjat que apliquen un valor de corrent altern al senyal i que han de ser eliminats per tenir un valor de corrent continu estable. Per aconseguir-ho necessitem utilitzar un filtre passabaix, el qual eliminarà les freqüències no desitjades, és a dir, les majors que f_{PWM}.

Mirant les especificacions dels microcontroladors de la família *PIC16*, per obtenir una resolució de 10 bits a la sortida analògica, o *PWM*, utilitzant un oscil·lador extern de $8\ MHz$, hem d'utilitzar una freqüència de $f_{PWM} = 7812{,}5\ Hz$.

Si agafem una freqüència de tall de $f_{BW} = 4kHz$, podrem calcular el valor d'aquesta resistència i del condensador per construir el filtre passabaix:

$$RC = \frac{1}{2\pi f_{BW}} = \frac{1}{2\pi \cdot 4000} = 3{,}97 \cdot 10^{-5}$$

Si agafem un condensador de $10nF$, la resistència serà de $3{,}9k$, la freqüència de tall real serà:

$$f_{BW} = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 3{,}9k \cdot 10nF} = 4{,}08kHz$$

La sortida d'aquest filtre passabaix presentarà un arrissat:

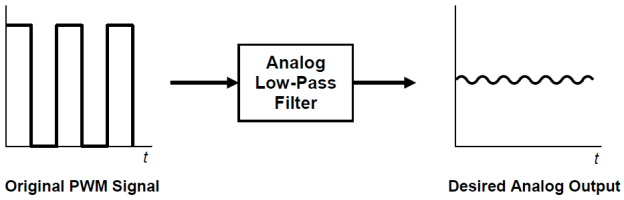


Figura 30 Diagrama PWM-Filtre-Sortida

De les notes de *Texas Instruments* i de *Microchip*[12,13] podem concloure que construir un filtre perfecte és impossible o com a mínim molt car de construir. Ens presenten alternatives què són utilitzar més d'un filtre passabaix de primer ordre en cascada. Per tant, nosaltres utilitzarem dos filtres passabaix en cascada.

Ara que ja tenim els valors dels filtres passabaix, per evitar problemes d'adaptació d'impedàncies utilitzarem l'amplificador operacional OPA551, de *Texas Instruments*, el qual pot proporcionar una sortida de $200mA$ a la tensió màxima d'alimentació menys 3 volts, és a dir, si l'alimentem a $\pm 15V$ pot treure fins a $\pm 12V$.

Com que el valor mig de la senyal *PWM* estarà entre 0 i $5V$, necessitarem duplicar aquests valors per obtenir una sortida entre 0 i $10V$, per tant, l'amplificador operacional OPA551 l'instal·larem com a amplificador no inversor de guany 2, ja que totes les seves resistències són iguals. El circuit d'una sortida analògica teòrica és:

Sortides analògiques de 0 a 10V

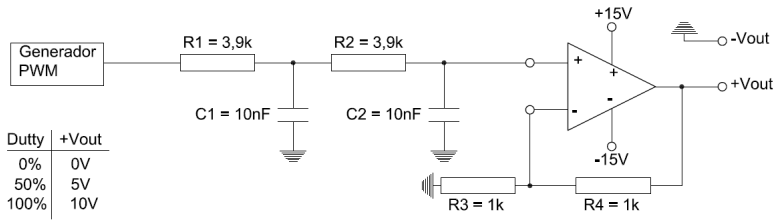


Figura 31 Sortides analògiques teòriques

Com podem veure, per a un cicle de treball del 100% la sortida serà de $10V$, per a un cicle de treball del 50% serà de $5V$ i per a un cicle de 0% serà de $0V$.

El circuit d'una sortida analògica utilitzant components reals amb *TINA-TI* és:

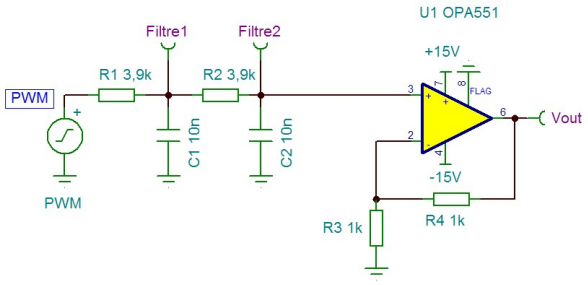


Figura 32 Sortida analògica real

Simulant el circuit per un cicle de treball del 98% obtenim:

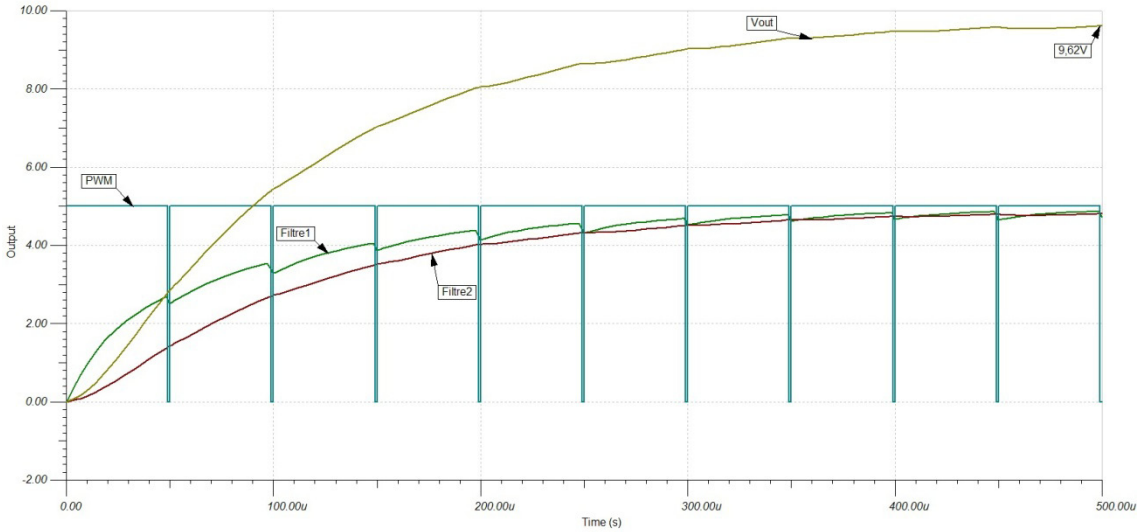


Figura 33 Sortida analògica amb cicle de treball de 98%

Com podem observar la sortida comença a estabilitzar-se a partir dels $500\mu s$. Simulant el circuit per un cicle de treball del 50% obtenim:

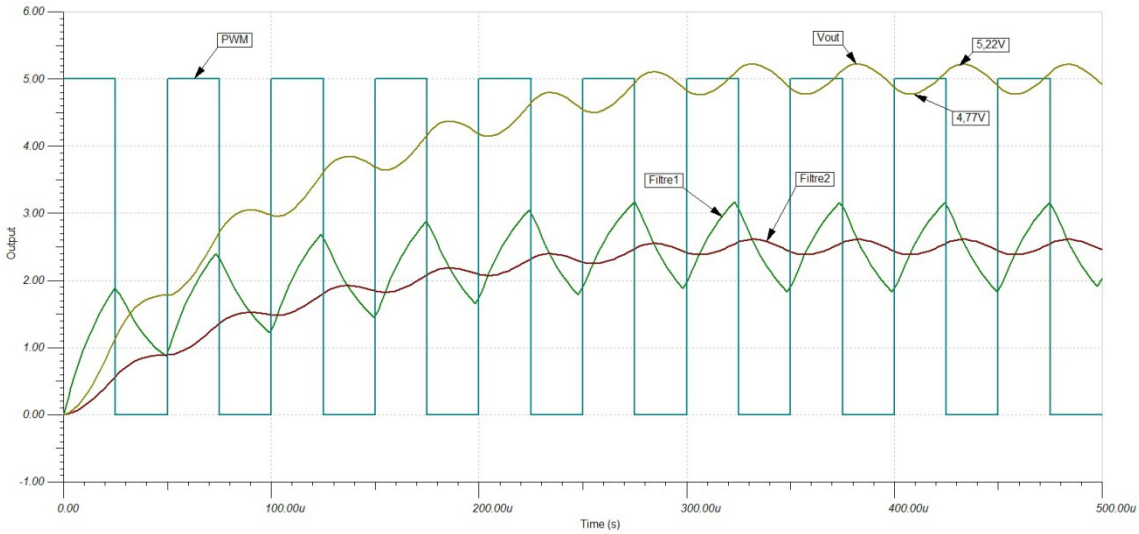


Figura 34 Sortida analògica amb cicle de treball de 50%

En aquest cas podem veure que també a partir dels $500\mu s$ la sortida comença a estabilitzar-se, encara que el arrissat de la tensió va de $4{,}77V$ a $5{,}22V$ i la variació és de $\pm 4\%$, la seva mitjana és de $4{,}995V \approx 5V$. Al següent punt 3.4 analitzarem una altra opció i justificarem l'escollida.

3.4. Alternatives descartades

Per a les entrades analògiques de intensitat també es va mirar d'utilitzar el integrat ***RCV420***[1] de *Texas Instruments*, el qual ja està dissenyat per a rebre una entrada de $4mA$ a $20mA$ i transformar-les en una tensió de fins a $12V$. Com que els microcontroladors només accepten una tensió d'entrada entre 0 i $5V$, havíem d'utilitzar amplificadors operacionals per a disminuir aquesta tensió. Per tant, es va decidir fer el disseny de l'adaptació d'entrada analògica per intensitat mitjançant només amplificadors operacionals, ja que en un mateix encapsulat hi ha fins 4 amplificadors.

Per a les sortides analògiques s'havia pensat en utilitzar un convertidor digital a analògic del tipus ***DAC101C0XX***[2], però es va descartar per dos motius. Un primer motiu perquè utilitza la *USART* del *PIC* per comunicar-se entre ells, i nosaltres ja utilitzem aquesta *USART* per al *RS-485*. El segon motiu és perquè la seva sortida és com a molt de $5V$, per tant hauríem d'utilitzar amplificadors operacionals per augmentar aquesta tensió fins als $10V$. Per això utilitzem la opció del *PWM*.

[1] Descàrrega del datasheet: http://www.ti.com/lit/ds/symlink/rcv420.pdf
[2] Descàrrega del datasheet: http://www.ti.com/lit/ds/symlink/dac101c081.pdf

4. Disseny de la font d'alimentació

En aquest apartat farem el disseny de la font d'alimentació del nostre expansor d'entrades i sortides. Primer farem un càlcul dels consums que tindran els diferents dispositius que componen aquest expansor, així com les tensions necessàries per alimentar-los.

Un cop tinguem clars els consums i les tensions necessàries explicarem el funcionament del convertidor de tensió contínua en tensió contínua ***ADP1621***[15] de *Analog Devices*. També utilitzarem el document *FCDC01101*[16] de *Analog Devices* on s'explica com fer una font dual utilitzant només un dispositiu *ADP1621*. Podrem veure la complexitat dels càlculs per a configurar el dispositiu, i per això veurem com *Analog Devices* ens proporciona uns fulls de càlcul on introduint els requisits que volem de la nostra font d'alimentació, ells ens llistaran els elements a utilitzar, el seu valor, i a més ens proporciona una sèrie de dades com són la eficiència i els consums. Nosaltres utilitzarem les gràfiques d'eficiència per poder veure com la nostra font d'alimentació tindrà una eficiència aproximada del 90% a $1A$ de consum.

Finalment, per aconseguir el voltatge de $-1V$ utilitzarem un amplificador operacional ***OPA277***[10] en mode amplificador inversor. I per aconseguir el voltatge de $5V$ utilitzarem el regulador de tensió ***LM7805***[17] de *Fairchild Semiconductor*.

4.1. Estudi alimentacions circuits

El nostre expansor d'entrades i sortides està compost per 1 transceptor *RS-485 ISO3088* el qual s'alimenta a una tensió de $5V$ i el seu consum màxim és de $\pm 60mA$.

Per a les entrades analògiques, tant les de tensió com les de corrent, utilitzarem 4 encapsulats *OPA4277* (què contenen 4 amplificadors operacionals) els quals s'alimenten a $\pm 15V$, i cada amplificador pot suportar fins a $\pm 0{,}9mA$, per tant cada encapsulat consumirà com a màxim $\pm 3{,}6mA$. El total dels amplificadors operacionals és de $\pm 14{,}4mA$.

Per a cada sortida analògica s'utilitzarà un amplificador operacional *OPA551*, el qual s'alimenta a $\pm 15V$ i pot proporcionar una corrent de sortida de fins a $200mA$, el seu consum és de $\pm 10mA$. En total les dues sortides tindran un consum màxim de $420mA$. Els microcontroladors de la família *PIC16* s'alimenten a $5V$ i tenen un consum de $300mA$.

Finalment el regulador de tensió *LM7805* proporciona una tensió de sortida de $5V$, i s'alimentarà a $15V$, el seu consum en repòs és de $8mA$ i el corrent màxim que pot proporcionar és de $1A$. En el nostre cas haurà d'alimentar el transceptor *RS-485* i el *PIC*, per tant el corrent màxim que haurà de proporcionar a $5V$ és de $300 + 60 = 360mA$. Al ser un component lineal i alimentar-se a $15V$, haurem de dissipar el calor produït mitjançant un dissipador. La potència a dissipar és:

$$P = VI = (15V - 5V) \cdot 360mA = 10V \cdot 360mA = 3{,}6W$$

Si la resistència tèrmica[17] entre la unió i l'aire del *LM7805* és de $\theta_{jA} = 65ºC/W$, haurem de dissipar una temperatura de:

$$T = \theta_{jA} \cdot P = 65\frac{ºC}{W} \cdot 3{,}6W = 234ºC$$

Com podem veure, el dissipador que s'ha d'utilitzar ha de reduir molt aquesta temperatura, ja que sinó correm el risc què se'ns cremi el regulador de tensió.

Els consums finals per a cada tensió serà de:

Tensió	*Intensitats*	*Consum total*
15V	14,4+10+10+200+200+8+360 =	802,4mA
-15V	14,4+10+10 =	34,4mA
5V	300+60 =	360mA

Taula 4 Consums expansor

Amb aquests consums dissenyarem una font d'alimentació dual de $\pm 15V$ proporcionant un màxim de $1A$ per a cada sortida. Pel disseny d'aquesta font d'alimentació utilitzarem un convertidor de tensió contínua en tensió contínua, concretament el *ADP1621* de *Analog Devices*, tal i com podem veure al punt 4.2, també s'utilitzarà un amplificador operacional *OPA277* de *Texas Instruments* per tal d'obtenir una font de $-1V$, com veurem al punt 4.3.

4.2. Font d'alimentació dual de $\pm 15V$

El *ADP1621* de *Analog Devices* és un convertidor de tensió contínua en tensió contínua, i és capaç d'aconseguir fins a un 92% d'eficiència. S'alimenta a una tensió entre $2{,}9V$ i $5{,}5V$. Tal i com comenten en el seu *datasheet*[15], per a tensions superiors a $5{,}5V$ podem utilitzar una resistència i un transistor *NPN* per delimitar la tensió màxima d'entrada al *ADP1621*. Aquesta resistència absorbirà part de la tensió d'entrada, i a la vegada polaritzarà el transistor *NPN*, el qual deixarà passar una tensió màxima de $5{,}5V$.

L'estructura del *ADP1621* és:

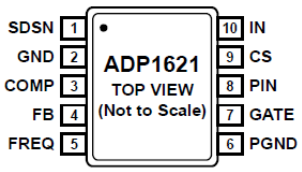


Figura 35 Convertidor DC-DC ADP1621

El pin *SDSN* és l'entrada de sincronització i l'entrada d'activació o desactivació. Quan l'entrada està a nivell alt el *ADP1621* està operatiu, en cas contrari s'apaga. El pin *GND* és la connexió a massa. El pin *COMP* és el control de regulació, connectant una resistència i un condensador en sèrie a massa es pot compensar l'error de l'amplificador interior.

El pin *FB* és l'entrada de *feedback*, aquesta entrada està connectada a un divisor de tensió format per dues resistències connectades a la sortida de la font. Els valors d'aquestes resistències condiciona el valor del voltatge de sortida. El pin *FREQ* és l'entrada del control de freqüència de l'oscil·lador. Aquesta entrada està connectada a una resistència i a massa. Segons el valor d'aquesta resistència la freqüència pot variar de $100kHz$ fins a $1{,}5MHz$.

El pin *PGND* va connectat a massa. El pin *GATE* és la sortida que anirà connectada a la porta del *MOSFET* de canal *N*, el qual és el qui controlarà el valor de la sortida de la font a partir d'una corrent polsant. El pin *PIN* és l'entrada per a alimentar la porta, on hi ha que connectar un condensador en paral·lel a massa major o igual a $0{,}1\mu F$, pel tema de filtratge.

El pin *CS* és l'entrada que controla la sensibilitat de l'amplificador, és recomanable connectar-li una resistència per a controlar la pendent de la compensació. El pin *IN* és l'entrada de voltatge per alimentar els circuits interiors del *ADP1621*.

Per a configurar el *ADP1621* hem de tenir en compte una sèrie de fórmules. Les més destacades són les de la tensió de sortida $V_{OUT} = 1{,}125V \cdot \frac{RF_1}{RF_2}$; el arrissat màxim de la corrent que proporcionarà l'inductor $\Delta I_L = 0{,}3 \cdot \frac{I_{carga\ màx.}}{1-D}$, on D és el cicle de treball; i el valor de l'inductor a utilitzar és $L = \frac{V_{IN} \cdot D \cdot (1-D)}{0{,}3 \cdot f_{SW} \cdot I_{carga\ màx.}}$, on f_{SW} és la freqüència de commutació.

Com podem comprovar, existeix una varietat important de fórmules per a poder determinar els elements exteriors necessaris per a configurar el comportament del *ADP1621*. A part dels càlculs, s'han de tenir en compte també els valors estàndards de les resistències, condensadors, bobines, díodes i transistors, el que complica encara més els càlculs dels elements. *Analog Devices* proporciona una sèrie d'eines amb les quals poder calcular de forma senzilla i automàtica tots aquest elements.

Per a calcular els elements per a una sortida de $15V$ positius he utilitzat el full de càlcul ADP1621_SEPICDesigner, i per a una sortida de $15V$ negatius el full de càlcul ADP1621_CukDesigner. Al utilitzar aquests fulls de càlcul, hem d'introduir una tensió d'entrada mínima de $12V$, una tensió d'entrada màxima de $48V$, una tensió de sortida de 15 o $-15V$ (segons el cas) i un consum de $1A$, una temperatura ambient de $55ºC$ (s'ha pensat en un ambient industrial) i finalment s'ha escollit un disseny tenint en compte el més eficient.

Aquests càlculs ens proporcionen els mateixos resultats en ambdós casos. Si ens fixem amb els circuits dels dos fulls de càlcul podem veure que només hi ha tres diferències. Una primera diferència és la connexió dels condensadors de l'entrada $COMP$. Una segona diferència és la connexió de la sortida, en el cas positiu la sortida està connectada a la bobina de manera que la intensitat entre les bobines sigui sortint, en canvi en el cas negatiu és entrant, el que fa que el voltatge de sortida sigui negatiu. I la tercera diferència és en el canal del *feedback*, en el cas positiu l'entrada va connectada al divisor de tensió, però en el cas negatiu hi ha intercalat un amplificador inversor, ja que l'entrada només accepta tensions positives.

La conclusió a la que arribem amb tot això és que utilitzant un mateix *ADP1621* i combinant els circuits de les figures anteriors podem crear una font d'alimentació dual. Aquesta idea també s'ha extret del document *FCDC01101*[16] de *Analog Devices* on dissenyen una font d'alimentació dual amb una tensió d'entrada de $10V$ a $30V$ i una tensió de sortida de $\pm 15V$ amb una intensitat de $1A$. L'esquema general d'aquesta font dual és:

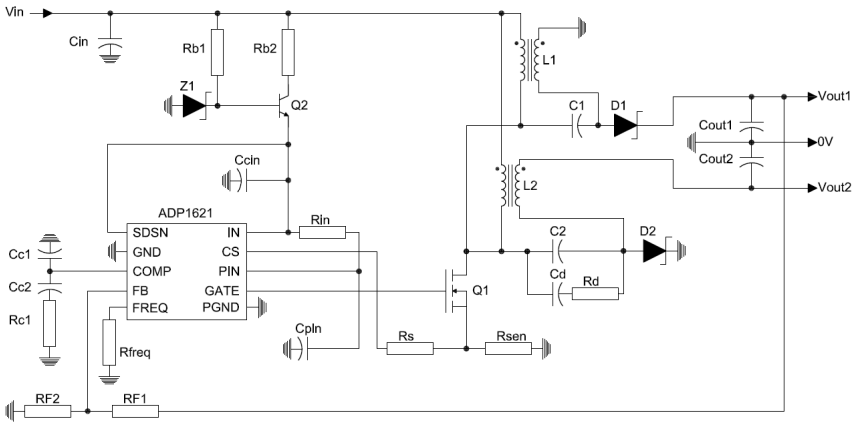


Figura 36 Esquema Font Dual $\pm 15V$

Els condensadors C_{in}, C_{out1} i C_{out2} són per a filtrar l'entrada i les sortides de la font. El conjunt format per les resistències R_{b1} i R_{b2}, el diode Zener Z_1, el transistor *NPN* Q_2, els condensadors C_{cin}, C_{pin} i la resistència R_{in} són per regular la tensió d'entrada als pins *IN* i *PIN*. Els condensadors C_{c1}, C_{c2} i la resistència R_{c1} són per a compensar l'error d'amplificació interior del *ADP1621*. La resistència R_{freq} és per determinar la freqüència de funcionament del corrent polsant. Les resistències R_s i R_{sen} són per a determinar la sensibilitat de l'amplificació i controlar la pendent de compensació. El divisor de tensió format per les resistències R_{F1} i R_{F2} és per determinar el voltatge de sortida escollit. El conjunt format per l'inductor L_1, el condensador C_1 i el diode D_1són la sortida positiva de la font d'alimentació. El conjunt format per l'inductor L_2, els condensadors C_2, C_d, la resistència R_d i el diode D_2 són la sortida negativa de la font d'alimentació. I finalment el *MOSFET* Q_1 és el qui controla les dues sortides.

El funcionament, tant pel cas positiu com pel cas negatiu, és el següent: L'inductor L_1 està format per dues bobines amb una relació de 1:1, quan el MOSFET Q_1 s'activa i el diode D_1 es desactiva, el voltatge d'entrada carrega la primera bobina de l'inductor i el condensador C_1carrega la segona bobina. El condensador de sortida subministra el corrent de càrrega durant aquest temps. Quan el MOSFET Q_1 es desactiva i el diode D_1 s'activa, l'energia de l'inductor s'allibera per a carregar els condensadors C_1 i C_{out1}, i també per subministrar corrent a la sortida. D'aquesta manera, amb la càrrega i descàrrega de les bobines i condensadors s'obté una sortida del voltatge desitjat i una corrent màxima. Els valors d'aquests elements els podem veure al punt 7.1 on llistem els elements que formen la *PCB*.

Segons els fulls de càlcul, les gràfiques d'eficiència són:

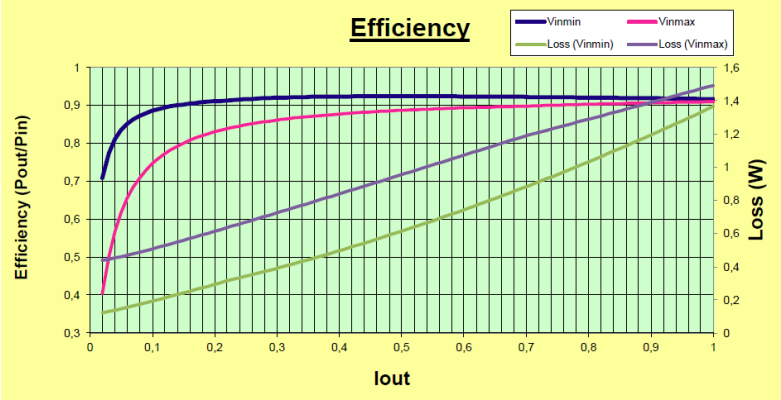


Figura 37 Eficiència tensió positiva

Figura 38 Eficiència tensió negativa

Com podem veure tant pel cas positiu com pel cas negatiu podem aconseguir una eficiència entre el 80% i el 90% a $1A$ de consum.

4.3. Disseny final de la font d'alimentació

Ara que a les sortides V_{out1} i V_{out2} tenim $15V$ i $-15V$ respectivament, necessitem aconseguir una fot d'alimentació de $5V$ i una de $-1V$. Per aconseguir el volt negatiu utilitzarem un amplificador operacional *OPA277* de *Texas Instruments* en mode amplificador inversor. Per tant, necessitem calcular el valor de les resistències per aconseguir passar dels $15V$ a $-1V$. La sortida d'aquest *AO* és $V_{out} = -V_{in} \cdot \frac{R_2}{R_1}$,

$$V_{out} = -V_{in} \cdot \frac{R_2}{R_1}$$

$$\frac{V_{out}}{-V_{in}} = \frac{R_2}{R_1}$$

$$\frac{-1V}{-15V} = \frac{R_2}{R_1}$$

$$\frac{R_2}{R_1} = \frac{1}{15}$$

Uns possibles valors teòrics són $R_1 = 15\ k\Omega$ i $R_2 = 1\ k\Omega$.

Per aconseguir els $5V$ s'utilitzarà un regulador de tensió *LM7805*, el qual s'alimentarà a $15V$ i pot proporcionar una sortida de $5V$ amb un corrent màxim de $1A$. Els esquemes bàsics d'aquests dos últims són:

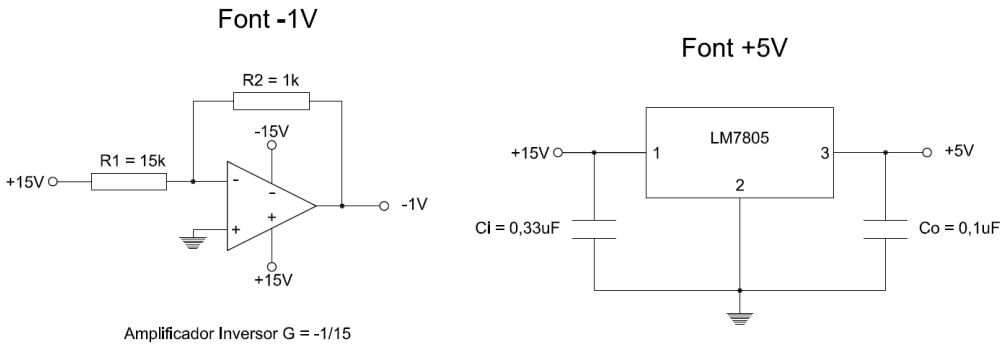


Figura 39 Esquema font 5V i -1V

Finalment, l'esquema del disseny final de la font d'alimentació el podem veure a la figura 47 de l'apartat 11 d'annex d'aquesta memòria.

5. Definició del mapa de memòria ModBus

Tal i com hem vist a l'apartat 2, *ModBus*[7,18] és un protocol de mestre - esclau. Durant la transmissió d'aquests datagrames s'intercanvien codis de funció i les dades necessàries. En aquest apartat explicarem la capa d'aplicació del protocol *ModBus*. Els codis de funció s'explicaran en el punt 5.1, i el mapa de memòria en el punt 5.2.

Un esclau no pot iniciar una conversació, només pot contestar a les peticions que van dirigides a ell. Si l'esclau rep un datagrama amb la direcció 0x00, indica que el mestre ha enviat un *broadcast* i simplement ha d'executar l'ordre rebuda però sense enviar cap resposta al mestre.

En els datagrames es pot adreçar a les dades desitjades (adreça d'inici dintre del camp *data*) amb les direccions de 0x0000 fins a 0xFFFF. Aquestes direccions es classifiquen en cinc categories de 10.000 adreces cadascuna.

Les direccions entre 0x0001 i 0x2710 (1-10.000) són utilitzades per a escriure sortides digitals, cada adreça identifica a un sol bit de sortida. Les direccions entre 0x2711 i 0x4E20 (10.001-20.000) són utilitzades per a llegir entrades digitals, aquí també cada adreça identifica a un sol bit d'entrada. Les direccions entre 0x4E21 i 0x7530 (20.001-30.000) no són utilitzades i estan reservades.

Ara, les direccions entre 0x7531 i 0x9C40 (30.001-40.000) són utilitzades per a llegir entrades analògiques, aquí cada adreça identifica a un registre de 16 bits (2 bytes). Les direccions entre 0x9C41 i 0xC350 (40.001-50.000) són utilitzades per a escriure sortides analògiques o registres de propòsit general, també identifiquen a registres de 16 bits.

El protocol *ModBus* ens diu que per fer referència a una adreça d'alguna categoria anterior, hem de començar a redireccionar-les a partir de l'adreça 0x0000. Per tant, per saber a quina adreça del mapa de memòria ens referim, hem de tenir en compte la funció a tractar. Per exemple, per a llegir una entrada analògica amb direcció 0x7531 dintre del mapa de memòria, enviarem realment el codi de la funció 0x03 (veure punt 5.1 per a més detalls) més l'adreça a llegir com a 0x0000.

5.1. Instruccions ModBus

El protocol *ModBus* a part de definir els datagrames a enviar durant les peticions i respostes, també defineix uns codis de funció. Aquests codis es codifiquen amb un byte, és a dir, podem tenir fins a 256 funcions diferents. La funció 0x00 no existeix, les funcions 0x80 fins a 0xFF estan reservades per a les respostes d'errors o excepcions. Com veurem més endavant, per enviar un error hem de sumar 0x80 al codi de la funció que estàvem executant.

Finalment, les funcions 0x01 fins a 0x7F són les que s'utilitzen. D'aquestes funcions existeixen dos grups diferenciats, un grup que conté les funcions públiques i un grup que conté les funcions privades, o les que un usuari pot definir i que no estan implementades en les funcions públiques. Els codis de les funcions privades estan entre 0x41 i 0x48, i entre 0x64 i 0x6E. La resta de codis formen el grup de funcions públiques definides per *ModBus*.

A mode de resum, exposarem a la següent taula algunes funcions públiques, encara que no les implementarem totes en el nostre expansor d'entrades i sortides analògiques:

Tipus	*Paràmetre*	*Codi*	*Descripció*
Accés a dades	Bit	0x01	Llegir valor digital
		0x02	Llegir entrada digital
		0x05	Escriure valor digital
		0x0F	Escriure N valors digitals
	16 bits o 2 Bytes	0x03	Llegir registres
		0x04	Llegir entrades
		0x06	Escriure un registre
		0x10	Escriure N registres
		0x17	Llegir/Escriure N registres
Diagnòstics		0x07 i/o 0x08	Llegir estat d'excepció. Tipus de diagnòstic.

Taula 5 Resum codis de funció principals ModBus

El nostre expansor d'entrades i sortides només disposa d'entrades i sortides d'analògiques, per això tots els codis de funcions que tracten amb valors digitals són descartats. Els codis de les funcions 0x07 i 0x08 s'utilitzen per a fer diagnòstics dels dispositius, com pot ser el número d'execucions d'una funció realitzada, els motius d'algunes excepcions, etc. Aquestes dues funcions tampoc seran implementades en el nostre dispositiu.

Per tant, ens quedarem amb les funcions que tracten valors analògics, o registres. La funció amb codi 0x04 s'utilitza per a llegir entrades analògiques, però com que el nostre microcontrolador converteix els valors analògics en digitals, haurem d'accedir als registres que contenen aquests valors digitals de les entrades analògiques. Per això aquesta funció tampoc la implementarem en el nostre dispositiu, en el seu lloc utilitzarem el codi de funció 0x03.

Les funcions que sí implementarem en el nostre expansor d'entrades i sortides analògiques són les funcions amb codi 0x03, 0x06, 0x10 i 0x17. Aquestes funcions són per a llegir una o varies entrades analògiques, escriure una sortida analògica, escriure una o varies sortides analògiques, i per a llegir i escriure una o varies entrades i sortides analògiques.

A part d'aquestes quatre funcions, també definirem una funció privada amb codi 0x65 per tal de modificar l'adreça del dispositiu o la velocitat de transmissió del bus *RS-485*. Aquesta funció es podria implementar perfectament amb el codi de funció 0x06, però utilitzarem el codi de funció 0x65 per tal de mostrar com implementar una funció privada, a part de les públiques.

Tal i com hem comentat anteriorment, el protocol *ModBus* defineix uns codis d'excepció que són utilitzats com a resposta a un error, durant l'execució d'una funció, tant per part del mestre com de l'esclau. Hi ha 9 codis d'excepció, alguns d'ells només s'utilitzen en conjunt a uns codis de funció que no hem explicat, però el nostre expansor utilitzarà els 3 codis principals, què són:

Codi	*Nom*	*Descripció*
0x01	ILLEGAL FUNCTION	Quan el dispositiu no reconeix la funció que li demanen.
0x02	ILLEGAL DATA ADDRESS	Quan l'adreça de les dades a accedir no és correcte.
0x03	ILLEGAL DATA VALUE	Quan les dades rebudes no són correctes.

Taula 6 Codis excepció ModBus

Durant la transmissió dels datagrames, primer s'envia el byte de l'adreça de l'esclau, després el byte amb el codi de la funció a executar, tot seguit s'envien les dades del datagrama, que poden anar de 0 a 252 bytes.

Dintre d'aquestes dades, es poden enviar els bytes que fan referència a les adreces inicials per accedir als registres, els bytes que fan referència a la quantitat de registres a accedir, els bytes amb els valors dels registres, etc. La peculiaritat d'aquests bytes és que primer s'envia el byte

de major ordre seguit del byte de menor ordre en el cas de les dades que ocupen més d'un byte. La majoria de dades que tractarem ocupen 2 bytes (realment només 10 bits).

Finalment, el datagrama el finalitza la transmissió de dos bytes que formen el *CRC*, amb la peculiaritat que aquests bytes s'envien primer el de menor ordre i després el de major ordre.

El codi de funció 0x03 (*Read Holding Registers*) és per a llegir una o més entrades analògiques. Es pot accedir fins a un màxim de 125 registres, ja que per a cada registre es necessiten 2 bytes, què juntament amb el byte del número de registres fan que el datagrama arribi als 256 bytes màxims. El format del datagrama en les peticions i en les respostes per a aquesta funció és:

0x03: Read Holding Registers		
Petició:		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Funció	1 Byte	0x03
Adreça registre inicial	2 Bytes	De 0x0000 a 0xFFFF
N registres a llegir	2 Bytes	De 0x0001 a 0x007D
CRC	2 Bytes	Segons bytes anteriors
Resposta:		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Funció	1 Byte	0x03
Nº bytes llegits	1 Byte	2 * *N* registres a llegir
Valors llegits	(2 * *N*) Bytes	De 0x00 a 0xFF cada byte
CRC	2 Bytes	Segons bytes anteriors

Taula 7 Estructura funció 0x03

El codi de funció 0x06 (*Write Single Register*) és per a escriure una sortida analògica. El format del datagrama en les peticions i en les respostes per a aquesta funció és:

0x06: Write Single Register		
Petició:		
Adreça esclau	1 Byte	Entre 0x00 i 0xF7
Codi Funció	1 Byte	0x06
Adreça registre	2 Bytes	De 0x0000 a 0xFFFF
Valor registre	2 Bytes	De 0x0000 a 0xFFFF
CRC	2 Bytes	Segons bytes anteriors
Resposta:		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Funció	1 Byte	0x06
Adreça registre	2 Bytes	De 0x0000 a 0xFFFF
Valor registre	2 Bytes	De 0x0000 a 0xFFFF
CRC	2 Bytes	Segons bytes anteriors

Taula 8 Estructura funció 0x06

El codi de funció 0x10 (*Write Multiple Registers*) és per a escriure a una o varies sortides analògiques. Es pot accedir fins a un màxim de 123 registres. El format del datagrama en les peticions i en les respostes per a aquesta funció és:

0x10: Write Multiple Registers		
Petició:		
Adreça esclau	1 Byte	Entre 0x00 i 0xF7
Codi Funció	1 Byte	0x10
Adreça registre inicial	2 Bytes	De 0x0000 a 0xFFFF
N registres a escriure	2 Bytes	De 0x0001 a 0x007B
Nº bytes a escriure	1 Byte	2 * *N* registres a escriure
Valors a escriure	(2 * *N*) Bytes	De 0x00 a 0xFF cada byte
CRC	2 Bytes	Segons bytes anteriors
Resposta:		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Funció	1 Byte	0x10
Adreça registre inicial	2 Bytes	De 0x0000 a 0xFFFF
N registres a escriure	2 Bytes	De 0x0001 a 0x007B
CRC	2 Bytes	Segons bytes anteriors

Taula 9 Estructura funció 0x10

El codi de funció 0x17 (Read/*Write Multiple Registers*) és per a llegir una o varies entrades analògiques, i per a escriure una o varies sortides analògiques. És una combinació de les funcions 0x03 i 0x10. Es pot accedir fins a un màxim de 125 registres per a llegir, i a un màxim de 121 registres per a escriure. El format del datagrama en les peticions i en les respostes per a aquesta funció és:

0x17: Read/Write Multiple Registers		
Petició:		
Adreça esclau	1 Byte	Entre 0x00 i 0xF7
Codi Funció	1 Byte	0x17
Adreça registre inicial llegir	2 Bytes	De 0x0000 a 0xFFFF
N registres a llegir	2 Bytes	De 0x0001 a 0x007D
Adreça registre inicial escriure	2 Bytes	De 0x0000 a 0xFFFF
N de registres a escriure	2 Bytes	De 0x0001 a 0x0079
Nº bytes a escriure	1 Byte	2 * *N* registres a escriure
Valors a escriure	(2 * *N*) Bytes	De 0x00 a 0xFF cada byte
CRC	2 Bytes	Segons bytes anteriors
Resposta:		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Funció	1 Byte	0x17
Nº bytes llegits	1 Byte	2 * *N* registres a llegir
Valors llegits	(2 * *N*) Bytes	De 0x00 a 0xFF cada byte
CRC	2 Bytes	Segons bytes anteriors

Taula 10 Estructura funció 0x17

Finalment, el codi de funció privada 0x65 (*Canviar Paràmetres*) és per a canviar l'adreça de l'esclau o per a canviar la velocitat del *bus RS-485*. La nova adreça només pot valer entre *1 i 247*. Els codis de les velocitats són: *0x00=1200bauds*; *0x01=2400 bauds; 0x02=9600 bauds, 0x03=19200 bauds, 0x04=28800 bauds;* i *0x05=33600 bauds* (utilitzant un oscil·lador de $8MHz$ no podem utilitzar una velocitat de *57600bauds*). El format del datagrama en les peticions i en les respostes per a aquesta funció és:

0x65: Canviar Paràmetres		
Petició:		
Adreça esclau	1 Byte	Entre 0x00 i 0xF7
Codi Funció	1 Byte	0x65
Adreça registre	1 Byte	0x00 o 0x01
Valor paràmetre	1 Byte	De 0x01 a 0xF7, o de 0x00 a 0x05
CRC	2 Bytes	Segons bytes anteriors
Resposta:		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Funció	1 Byte	0x65
Adreça registre	1 Byte	0x00 o 0x01
Valor paràmetre	1 Byte	De 0x01 a 0xF7, o de 0x00 a 0x05
CRC	2 Bytes	Segons bytes anteriors

Taula 11 Estructura funció 0x65

Com ja hem dit anteriorment, quan es produeix una excepció durant l'execució d'una funció, el protocol *ModBus* estableix una resposta d'error. El format del datagrama de resposta d'error està format per un byte de l'adreça de l'esclau, un byte amb el codi d'error de funció, un byte amb el codi de l'excepció i finalment els dos bytes de *CRC*. El codi d'error de funció és igual al codi de la funció que ha provocat l'error més 0x80, d'aquesta manera els codis d'error sempre estaran per damunt del codis de funció 0x80.

Resposta error - excepció		
Adreça esclau	1 Byte	Entre 0x01 i 0xF7
Codi Error	1 Byte	Codi funció + 0x80
Codi excepció	1 Byte	De 0x01 a 0x04 (en el nostre cas)
CRC	2 Bytes	Segons bytes anteriors

Taula 12 Estructura resposta excepció

Per tant, la funció 0x03 genera un codi d'error igual a 0x83, la funció 0x06 igual a 0x86, la funció 0x10 igual a 0x90, la funció 0x17 igual a 0x97 i finalment la funció 0x65 igual a 0xE5. Com que el nostre expansor només disposa de 4 entrades, la quantitat màxima de registres a llegir serà de 0x04. I com que només disposa de 2 sortides, la quantitat màxima de registres a escriure serà de 0x02.

Per finalitzar aquest punt explicarem en què consisteix el *CRC* (*Cyclic Redundancy Check*). El *CRC* està compost per dos bytes. El mestre calcula el *CRC* dels bytes a enviar i adjunta aquests dos bytes al final del datagrama. L'esclau ha de calcular el *CRC* amb les dades rebudes per després comparar-lo amb el rebut. Si aquests valors no coincideixen s'envia una excepció de dades incorrectes.

Per a realitzar el càlcul del *CRC*, el protocol *ModBus* defineix dues taules[7], una pel byte de major ordre i una altre pel byte de menor ordre. També defineix una funció, el procediment de la qual és:

1. Carreguem una variable *CRC* de 16 bits a 0xFFFF.
2. Agafem com a índex la *XOR* entre el byte a enviar i els 8 bits de menor ordre de la variable *CRC*.
3. Dintre dels 8 bits de menor ordre de la variable *CRC* carreguem el resultat de la *XOR* entre els 8 bits de major ordre i el contingut de la taula de major ordre en la posició de l'índex calculat en el pas anterior.
4. Dintre dels 8 bits de major ordre de la variable *CRC* carreguem el contingut de la taula de menor ordre en la posició de l'índex calculat en el pas 2.
5. Repetim els passos 3 i 4 per a cada byte que formarà el datagrama a enviar.

5.2. Mapa memòria

Al tenir quatre entrades analògiques les adreçarem com a 0x7531, 0x7532, 0x7533 i 0x7534 respectivament. Per a saber a quina d'elles ens referim en una petició, enviarem com a adreces de dades 0x0000, 0x0001, 0x0002 i 0x0003 respectivament.

Al tenir dues sortides analògiques les adreçarem com a 0x9C51 i 0x9C52 respectivament. També per a saber a quina d'elles ens referim en una petició, enviarem les adreces de dades 0x0010 i 0x0011 respectivament.

També hem definit un parell de registres per a modificar l'adreça de l'esclau o la velocitat de la *USART*. Aquests registres els adreçarem com a 0x9C41 i 0x9C42 respectivament. Per saber a quin d'ells ens referim en una petició, enviarem les adreces de dades 0x0000 i 0x0001 respectivament.

A la següent taula podem veure el resum del mapa de memòria *ModBus* que utilitzarà el nostre dispositiu:

Adreça ModBus	***Adreça Registre***		***Tipus Paràmetre***	***Codi Funció Acceptada***	***Tipus Accés***	***Descripció***	***Rang Ajustable***
	Hi	*Lo*					
0x7531	00	00	2 Bytes	0x03 0x17	Lectura	Entrada analògica A	De 0 a 1023
0x7532	00	01	2 Bytes			Entrada analògica B	
0x7533	00	02	2 Bytes			Entrada analògica C	
0x7534	00	03	2 Bytes			Entrada analògica D	
0x9C51	00	10	2 Bytes	0x06 0x10 0x17	Escriptura	Sortida analògica A	De 0 a 1023
0x9C52	00	11	2 Bytes			Sortida analògica B	
0x9C41	00	00	2 Bytes	0x65	Escriptura	Direcció esclau	De 1 a 247
0x9C42	00	01	2 Bytes			Velocitat *RS-485*	(en *bauds*) 0 = 1200 1 = 2400 2 = 9600 3 = 19200 4 = 28800 5 = 33600

Taula 13 Mapa memòria ModBus

Tant per a les entrades com per a les sortides analògiques al utilitzar una resolució de 10 bits, els seus valors només poden estar entre 0x0000 i 0x03FF (0 i 1023).

Quan el dispositiu rebi com a direcció d'esclau l'adreça 0x00 significa que el mestre estableix una transmissió de *broadcast*, i en aquest cas no enviarem cap resposta. Les funcions 0x06, 0x10, 0x17 i 0x65 modificaran les sortides o els paràmetres sense enviar cap resposta. La funció 0x03 no realitzarà cap tasca i tampoc respondrà al mestre.

Finalment, el nostre dispositiu està configurat amb una adreça d'esclau inicial, i uns paràmetres de *USART* inicials. L'adreça d'esclau és 0x05 (totalment a l'atzar, el més comú és utilitzar la 0x01), i la configuració de la *USART* és de 9600 bauds de velocitat de transmissió, sense paritat i 2 bits de stop.

Al punt 6.2 podrem veure quins registres del *PIC* i quines funcions utilitzarem per a tractar amb totes aquestes funcions i dades.

6. Elecció µC i implementació programa de control

A l'apartat 3 hem dissenyat els circuits d'adaptació d'entrades i sortides per a adequar-les a les tensions estàndards dels microcontroladors *PIC*. Concretament, les entrades tenen un rang de 0 a $5V$, i les sortides del *PIC* de 0 a $5V$ són adaptades a un rang de sortida de 0 a $10V$. Per a la lectura de les entrades s'utilitzarà un convertidor analògic a digital de 10 bits de resolució. Per a les sortides s'utilitzarà dues senyals de polsos modulables, o *PWM*, i el que farem serà canviar el percentatge del cicle de treball, on la seva resolució és també de 10 bits.

Per a la comunicació *RS-485* utilitzarem un adaptador de *RS-232* a *RS-485* tal i com hem vist en el punt 2.2. Per a transmetre és necessari activar els pins DE i $\overline{RE}$, i per a rebre desactivar-los. Utilitzarem la mateixa sortida digital del *PIC* pels dos pins. Com que la freqüència de l'oscil·lador és de $8\ MHz$, màxim podem tenir una velocitat de $33600\ bauds$.

Per a la simulació i depuració del codi s'utilitzarà el programa *Real PIC Simulator*. Al punt 6.1 explicarem els motius d'elecció del microcontrolador ***PIC16F876A***[19], i en l'apartat 6.2 explicarem el codi del programa a implementar per part del microcontrolador *PIC*.

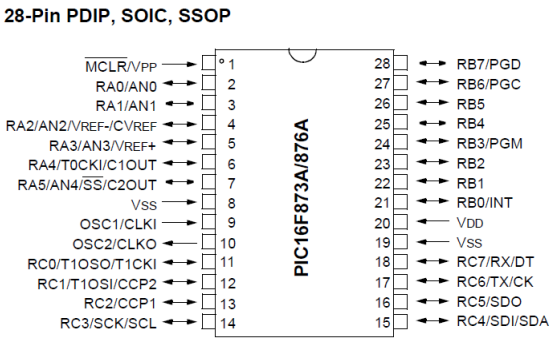


Figura 40 Diagrama de pins del PIC16F876A

6.1. Elecció microcontrolador

Al tenir quatre entrades i dues sortides analògiques necessitem un microcontrolador amb al menys quatre canals d'entrades analògiques i dues sortides de *PWM*. El *PIC* ha d'acceptar tensions en els seus pins de 0 a $5V$. Per a la conversió d'analògic a digital necessitem un convertidor de 10 bits. Per a definir la freqüència i el cicle de treball del *PWM* necessitem un temporitzador (*TIMER2*).

Al tenir que activar i desactivar el dispositiu *RS-485* necessitem una sortida digital que proporcioni $5V$. A més per a comunicar-se entre ells, el *PIC* ha de tenir dos pins per a la transmissió - recepció del bus *RS-232*. Per a saber quan hem rebut un datagrama ens farà falta un altre temporitzador, diferent del utilitzat pel *PWM* (*TIMER0*).

Amb tots aquests requisits anem a la pàgina de *Microchip*:

http://www.microchip.com/maps/microcontroller.aspx

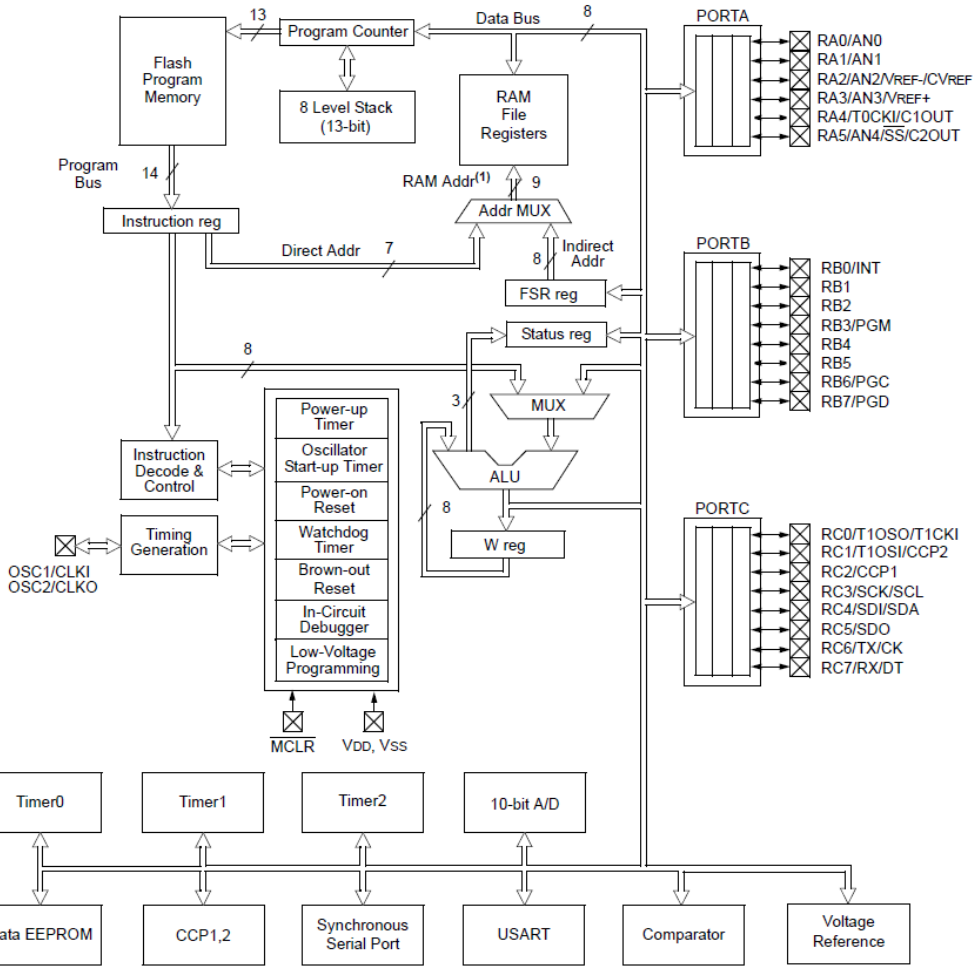
I de entre totes les opcions que ens faciliten escollim la família *PIC16F87XA*[19], què encara que sigui un *PIC* dels més senzills, compleix en tots els nostres requisits. A més de gaudir de gran popularitat per la gran quantitat d'aplicacions que s'han realitzat amb ell.

De entre els quatre models de la família descartem els pics PIC16F873A i PIC16F874A per tenir menys capacitat de memòria i de programa. La diferència entre el PIC16F876A I PIC16F877A és

que aquest últim té més entrades analògiques i digitals, el què implica que l'encapsulat tindrà més pins i per tant serà molt més gran i ocuparà més espai sobre la nostra *PCB*, veure apartat 7. El PIC16F877A disposa de 40 pins i el PIC16F876A de 28.

El microcontrolador escollit és el ***PIC16F876A***, el qual disposa de 5 entrades analògiques situades al port A, i després disposa de dos ports de 8 bits B i C per a sortides/entrades digitals. En el port C es troben les dues sortides *PWM* i els dos pins de la *USART*.

Per a funcionar necessiten un oscil·lador extern connectat als pins 9 i 10, l'oscil·lador serà de $8\ MHz$, tot per a poder donar una resolució de 10 bits a les sortides *PWM*, tal i com hem vist al punt 3.3. El diagrama de blocs del *PIC* és:



Device	Program Flash	Data Memory	Data EEPROM
PIC16F873A	4K words	192 Bytes	128 Bytes
PIC16F876A	8K words	368 Bytes	256 Bytes

Figura 41 Diagrama de blocs del PIC16F876A

Com podem veure disposa de 3 ports d'entrades i sortides. Té una memòria per a emmagatzemar el programa i una altra memòria per a emmagatzemar els registres del microcontrolador. Disposa de 3 temporitzadors (2 de 8 bits i 1 de 16 bits), un convertidor d'analògic a digital de 10 bits i una *USART* (entre altres). Tots comunicats per a un bus de 8 bits. A cada cicle de rellotge el microcontrolador dur a terme l'execució d'una funció, ja sigui llegir o escriure des d'un port, modificar algun registre, etc. Totes les funcions utilitzen algun registre per a establir els seus valors. Per exemple, el port B utilitza el registre *PORTB* per a

emmagatzemar l'estat del port. Però també utilitza el registre *TRISB* per a indicar quin pin és d'entrada i quin de sortida. Tots aquests registres es distribueixen en quatre bancs de registres (de 128 bytes cadascun), tal i com podem veure a la següent figura:

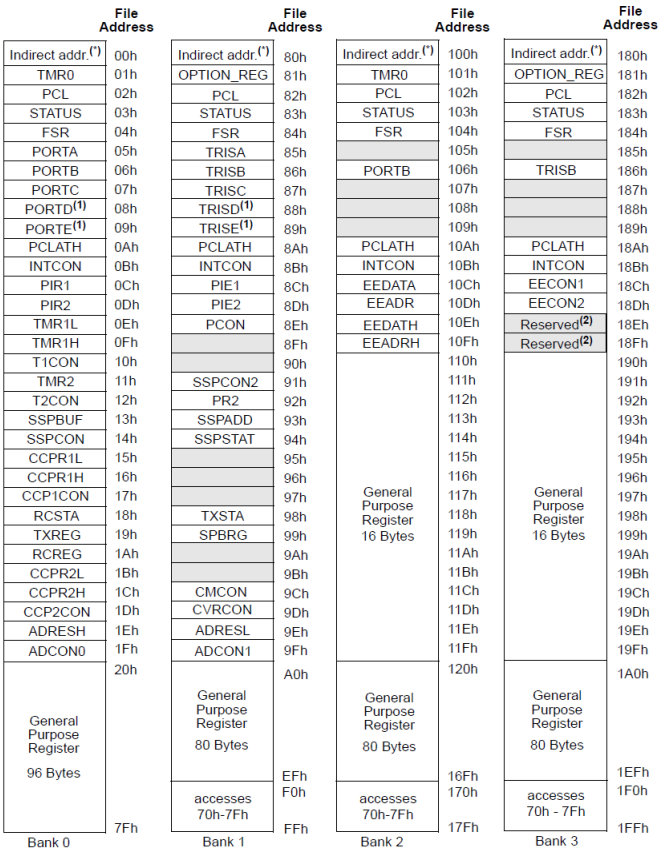


Figura 42 Adreces registres del PIC16F876A

Els microcontroladors *PIC* es poden programar amb *Assembler* o amb llenguatge d'alt nivell. En assemblador es programa a baix nivell fent referència a tots aquests registres. Però com que per a realitzar una tasca es pot fer feixuga la seva programació, pel tractament de tants de registres, nosaltres utilitzarem el llenguatge *C* per a programar el *PIC*, el qual facilita les coses.

6.2. Programa de control

El programa de control té varies parts diferenciades. Per una banda tenim la recepció de datagrames *ModBus*. Després tenim el tractament de la informació rebuda, segons el codi de funció que ens demanen haurem de convertir a digital el valor d'una entrada analògica, o modificar el valor d'una sortida analògica, o bé haurem de canviar l'adreça del dispositiu esclau, o finalment canviar la velocitat de transmissió del bus *RS-485*. I per altra banda haurem d'enviar un datagrama *ModBus* amb la resposta a la funció executada, o amb l'error produït durant la mateixa.

En la conversió[3] analògica a digital el que es fa és quantificar un senyal elèctric en una sèrie de N nivells. Després aquests nivells es codifiquen en binari assignant a cada nivell un codi digital únic. El número de nivells, o de codis, depèn del número de bits que utilitzem $2^n \geq N$. En el nostre cas utilitzem $n = 10\ bits$, tant per a les entrades com per a les sortides analògiques.

Aquesta quantificació provoca que una tensió de 0 a $5V$ es codifiqui de la mateixa manera que un senyal de 0 a $10V$. Per tant, els valors a transmetre de les entrades al estar codificades a 10 bits no les hem de tractar, ja que nosaltres codificarem la tensió de 0 a $5V$ i el receptor ho interpretarà com una tensió de 0 a $10V$. Al igual amb les sortides, el mestre enviarà una codificació per a una tensió de 0 a $10V$ i nosaltres activarem la sortida entre 0 a $5V$, l'adaptació de la sortida serà la què proporcionarà una tensió de 0 a $10V$.

Tal i com hem vist al punt 3.3 per a modificar una sortida analògica el que hem de fer és modificar el cicle de treball del senyal *PWM*. La freqüència necessària per a obtenir un bon resultat (10 bits) vam veure que era de $f_{PWM} = 7{,}81\ kHz$.

El codi del programa està estructurat en dos arxius, un arxiu de capçalera ***16F876A.h*** que conté les declaracions i configuracions del nostre microcontrolador i un arxiu ***piris_modbus.c*** que conté tot el codi principal. En aquest arxiu hi ha totes les declaracions de variables, la configuració inicial del microcontrolador, la funció i taules per a calcular el *CRC*, una funció per a llegir les entrades analògiques, una funció per a modificar el cicle de treball de les dues sortides analògiques, les funcions de les respostes a enviar segons la funció rebuda o error produït. També hi ha definida la funció *main()* i les dues funcions d'interrupcions, una per a la interrupció de byte rebut per la *USART* i una altra per a la interrupció del *TIMER0*, utilitzat per a saber si l'últim byte rebut és el final del datagrama *ModBus*.

Per a configurar el *PWM* hem de modificar el registre *PR2* del temporitzador *TIMER2* per tal d'establir aquesta f_{PWM}. Per a establir el percentatge del cicle de treball hem d'escriure en el registre *CCPR1L* i en els bits 5 i 4 del registre *CCP1CON*, sumats donen els 10 bits de resolució de la sortida analògica. Per a indicar que els pins 12 i 13 del port C són de sortida hem de posar a 0 els bits corresponents del registre *TRISC*. Per a configurar el *TIMER2* ho fem en el registre *T2CON*. Per fer tot això amb les dues sortides analògiques, utilitzant el compilador *CCS C* emprarem les següents funcions:

```
/* Configurem les sortides analògiques */
setup_ccp1(CCP_PWM);                  //Mode PWM.
setup_ccp2(CCP_PWM);
setup_comparator(NC_NC_NC_NC);        //No mode comparador.
setup_timer_2(T2_DIV_BY_1,255,1);     //Fpwm ~ 7.812,5 Hz.
duty[0] = 0;                          //Sortida 1 a 0V al inici del sistema.
set_pwm1_duty(duty[0]);
duty[1] = 0;                          //Sortida 2 a 0V al inici del sistema.
set_pwm2_duty(duty[1]);
```

Per a les entrades analògiques hem de posar a 1 els bits corresponents del registre *TRISA* per a configurar-los com a entrades. Hem de configurar els canals que utilitzarem en el registre *ADCON0*. En el registre *ADCON1* hem d'especificar el rellotge de conversió així com la configuració dels ports d'entrada analògics. El nostre dispositiu només accepta 9 configuracions, i com que només volem 4 entrades analògiques haurem d'escollir la configuració 0011, amb la qual els canals d'entrada són el AN0, AN1, AN2 i AN4, el canal AN3 s'utilitzarà com a referència i anirà connectat a $5V$ positius. Recordar que només es pot convertir d'analògic a digital un canal a la vegada, ja que només es disposa d'un *ADC*. Finalment en els registres *ADRESL* i *ADRESH* s'emmagatzema el valor de la conversió, en el registre *ADRESH* només es guarden els 2 bits més significatius, d'aquesta manera s'aconsegueix una resolució de 10 bits. Utilitzant el compilador *CCS C* emprarem les següents funcions:

```
/* Definim el convertidor analògic-digital a 10 bits */
#device adc=10

/* Configurem les entrades analògiques */
setup_adc_ports(AN0_AN1_AN2_AN4_VSS_VREF);   //L'entrada AN3 anirà connectada a +5V.
setup_adc(ADC_CLOCK_INTERNAL);
```

Per a utilitzar la *USART* hem de configurar els registres *TXSTA* que condiciona la transmissió, *RCSTA* que defineix la recepció i *SPBRG*, aquest últim és el que defineix la velocitat del bus. El registre *TXREG* és on posem el byte a enviar. En el registre *RCREG* és on s'emmagatzemen els bytes rebuts, aquest registre disposa d'una memòria *FIFO* què pot emmagatzemar fins a dos bytes. Per a enviar o rebre des de *RS-485* hem d'activar o desactivar una sortida digital del *PIC*. Quan tenim un byte disponible per a ser llegit es dispara una interrupció. Utilitzant el compilador *CCS C* emprarem les següents funcions:

```
/* Activem la UART amb la configuració següent */
#use rs232(UART1, baud=9600, parity=N, bits=8, stream=PORT1)

/* Configurem el port B com a sortides digitals */
set_tris_b(0x00);
output_low(PIN_B1);    //Aquest pin si està activat és per a transmetre

/* Activem les interrupcions del sistema */
enable_interrupts(GLOBAL);
enable_interrupts(INT_RDA);
```

La funció d'interrupció per byte rebut per la *USART* consta d'una variable booleana i una variable d'un byte per emmagatzemar el byte rebut. El que fem és agafar aquest byte rebut per buidar el registre *RCREG* i poder rebre un altre byte.

Després tenim una condició que si no hem rebut cap datagrama anirem llegint els bytes entrants, i si ja hem rebut un datagrama només anem buidant el registre *RCREG* sense cap més acció. Dintre del *IF* anem esbrinant on hem de guardar el byte rebut, primer guardem l'adreça de l'esclau, després el codi de la funció, i finalment anem guardant tots els bytes de dades, més els dos bytes de *CRC* dintre de *l'array datagrama.dades[]*.

Tot seguit tenim un *SWITCH* que segons el codi de funció rebut anirem calculant el *CRC* amb els bytes rebuts, però descartant sempre els dos últims bytes que són els bytes *CRC* rebuts. Per això hi ha tot un seguit de condicions, que tenen en compte la longitud del datagrama segons la funció rebuda i el número d'entrades o sortides a tractar.

Un cop calculat el *CRC* del byte rebut activem el *TIMER0* per a començar a comptar, més o menys, un temps de 3,5 caràcters per tal de saber si hem rebut tot un datagrama o encara no. El temporitzador *TIMER2* té més capacitat de configuració, però al ser utilitzat pel *PWM* ens veiem obligats a utilitzar el temporitzador *TIMER0*. Aquesta interrupció queda així:

```
/* Interrupció de byte rebut per la USART, emmagatzemem bytes i calculem CRC */
#int_RDA
void  RDA_isr(void){
   int8 c;
   int1 h = FALSE;
   c = getc();    //Traiem el byte del registre d'entrada de la USART.

   /* Si no hem rebut un datagrama, el guardem dintre de DATAGRAMA */
   if(!datagrama_In){
      if(event==0){
         mesDades = 0;           //Inicialitzem valors per un nou datagrama.
         modbus_crc.d = 0xFFFF;
         datagrama.address = c;  //El primer byte és l'adreça.
         calcul_crc(c);          //Calculem CRC.
         event++;
      }else if(event == 1){
         datagrama.funcio = c;   //El segon byte és la funció.
         calcul_crc(c);          //Actualitzem CRC.
         event++;
      /* El tercer i altres bytes són les dades de les funcions */
      }else if(event == 2){
         h = TRUE;               //Per actualitzar CRC.
         datagrama.dades[mesDades] = c;
         mesDades++;
      }
```

```
      /* Segons la funció rebuda anem calculant el CRC sense calcular
      els 2 bytes finals del datagrama */
      switch(datagrama.funcio){
         case 0x03:
            /* Aquesta petició només pot tenir fins a 4 bytes de dades */
            if(h && (mesDades<5)) calcul_crc(c);
            break;
         case 0x06:
            /* Aquesta petició només pot tenir fins a 4 bytes de dades */
            if(h && (mesDades<5)) calcul_crc(c);
            break;
         case 0x10:
            /* Aquesta petició com a mínim ha de tenir 7 bytes de dades */
            if(h && mesDades<8){
               calcul_crc(c);
            /* I com a màxim 9 bytes de dades, ja que només tenim 2 sortides */
            }else if(h && (mesDades<10) && (datagrama.dades[4]==0x04)){
               calcul_crc(c);
            }
            break;
         case 0x17:
            /* Aquesta petició com a mínim ha de tenir 11 bytes de dades */
            if(h && mesDades<12){
               calcul_crc(c);
            /* I com a màxim 13 bytes de dades, ja que només tenim 2 sortides */
            }else if(h && (mesDades<14) && (datagrama.dades[8]==0x04)){
               calcul_crc(c);
            }
            break;
         case 0x65:
            /* Aquesta petició només pot tenir 2 bytes de dades */
            if(h && (mesDades<3)) calcul_crc(c);
            break;
         default:
            break;
      }
      /* Per saber si és l'últim byte del datagrama activem el temporitzador TIMER0 i si
       passa el temps de més de 3,5 caràcters sense rebre cap byte és que hem rebut un
       datagrama MODBUS */
      set_timer0(0);
      count = 0;
      enable_interrupts(INT_TIMER0);   //Habilitar interrupció.
   }
}
```

La funció d'interrupció per temps del *TIMER0* és aproximadament cada $128\mu s$, i com que nosaltres volem esperar un temps mínim de 3,5 caràcters, utilitzem les variables *STOP_COUNT* i *COUNT* per comptar el número de cops que hem entrat en aquesta interrupció. Si ha passat el temps activem una variable booleana per indicar que hem rebut un datagrama, també desactivem el *TIMER0*. Aquesta interrupció queda així:

```
/* Interrupció del TIMER0 per desbordament */
#int_TIMER0
void  TIMER0_isr(void){
   /* Contem els cops que s'ha disparat aquesta interrupció per saber si ha passat el
   temps de més de 3,5 caràcters sense rebre cap byte, sí ha passat aquest temps és que
   hem rebut un datagrama ModBus.
   STOP_COUNT depèn de la velocitat (bauds) de la USART, veure MAIN */
   if(count>stop_count){
      datagrama_In = TRUE;             //Activem datagrama rebut.
      count = 0;
      disable_interrupts(INT_TIMER0);  //Desactivem la interrupció del TIMER0.
   }
   /*Comptem els cops que hem entrat a la interrupció des de la última inicialització */
   count++;
}
```

Dintre de la funció *main()* configurem les entrades analògiques, digitals, les sortides *PWM* i habilitem les interrupcions, tal i com hem explicat anteriorment. Dintre d'aquesta funció hi ha un bucle infinit mitjançant un *WHILE(TRUE)* el qual anirà executant el programa

indefinidament. Només sortirem del bucle quan hi hagi una interrupció per byte rebut o per la interrupció del *TIMER0*.

Dintre del bucle tenim una condició per saber si el datagrama va dirigit a l'adreça del nostre dispositiu o l'adreça de *broadcast* (0x00), si és així seguim amb l'execució del codi de funció. Però si no va dirigit cap al nostre dispositiu, inicialitzem les variables per a poder rebre un nou datagrama i continuar amb l'execució del programa.

Si el datagrama és pel nostre dispositiu comprovarem quin codi de funció ens han enviat, mitjançant un *SWITCH*. Si el codi no és suportat pel dispositiu enviarem una resposta d'error amb l'excepció de funció no reconeguda *ILLEGAL_FUNCTION*.

Tal i com hem explicat a l'apartat 5, el nostre dispositiu només acceptarà els codis de funció 0x03, 0x06, 0x10, 0x17 i 0x65. L'estructura de les diferents funcions és la mateixa, primer comprovem si els bytes de *CRC* rebuts són iguals als que hem anat calculant en la seva recepció. Si són diferents enviarem una resposta d'error amb l'excepció de dades no vàlides *ILLEGAL_DATA_VALUE*.

Si el datagrama rebut és correcte continuem, i comprovem si l'adreça del registre que ens demanen és correcte, és a dir, correspon amb les opcions de les diferents funcions. Si no és correcte enviarem una resposta d'error amb l'excepció d'adreça no vàlida *ILLEGAL_DATA_ADDRESS*.

Si l'adreça del registre és correcte hem de comprovar si les dades enviades estan dintre del marge acceptat, si són correctes o no. Si no són correctes enviarem una resposta d'error amb l'excepció de dades no vàlides *ILLEGAL_DATA_VALUE*.

Segons la funció cridarem a una funció per a llegir les entrades i/o una funció per a escriure a les sortides. A part tenim la funció per a canviar els paràmetres, la qual modificarà l'adreça del dispositiu de la variable global *SLAVE_ADDRESS* o les variables *BAUDS* i *STOP_COUNT* per a modificar la velocitat del bus.

Dintre de la funció per a llegir les entrades analògiques podem destacar que només podem llegir els canals 0, 1, 2 i 4, ja que el canal 3 s'utilitza com a referència a $+5V$. Per a llegir una entrada primer hem d'activar el canal que volem llegir, per després agafar la conversió de 10 bits. Aquesta conversió l'emmagatzemem dintre d'un *array*, primer guardem el byte de major ordre i després el byte de menor ordre:

```
/* Funció per llegir N entrades */
void llegir_entrades(int8 canal, int8 quantitat){
   int16 A = 0;            //Per memoritzar conversió ADC.
   int index_entrades = 0; //Índex del vector de bytes d'entrades.
   /* Llegim les N entrades analògiques, si quantitat = 0 no llegim cap entrada */
   for(i=0; i<quantitat; i++){
      /* Com que l'entrada D està connectada al canal 4: */
      if(canal==3){
         /* Activem canal 4 per llegir quarta entrada */
         set_adc_channel(4);
      }else{
         /* Activem els canals 0, 1 o 2 de les entrades 1 a 3 */
         set_adc_channel(canal);
      }
      /* Convertim d'analògic a digital el canal seleccionat */
      A = read_adc();
      /* Guardem el valor convertit, primer el byte MSB i després el byte LSB */
      vector_bytes_entrades[index_entrades] = make8(A,1);
      vector_bytes_entrades[index_entrades+1] = make8(A,0);
      index_entrades = index_entrades + 2;   //Incrementem en 2 unitats l'índex.
      canal++;    //Incrementem canal a llegir.
   }
}
```

Dintre de la funció d'escriure a les sortides analògiques actualitzarem la variable del cicle de treball de les sortides segons les dades rebudes. Pot ser en un canal dels dos o tots dos alhora:

```
/* Funció per escriure a una sortida o a les dues sortides */
void escriure_sortides(int8 canal, int8 quantitat, int16 valor_Out1, int16 valor_Out2){
   /* Segons el número de sortides a escriure escollim entre 1 o 2 sortides */
   switch(quantitat){
      case 0x01:
         /* Si volem escriure a la sortida 1: */
         if(canal==0x10){
            duty[0] = valor_Out1;    //Actualitzem el cicle de treball de la sortida 1.
            set_pwm1_duty(duty[0]);
         /* Sinó és que volem escriure a la sortida 2 */
         }else{
            duty[1] = valor_Out1;    //Actualitzem el cicle de treball de la sortida 2.
            set_pwm2_duty(duty[1]);
         }
         break;
      case 0x02:
         /* Com que volem escriure a les 2 sortides executem: */
         duty[0] = valor_Out1;   //Actualitzem el cicle de treball de la sortida 1.
         set_pwm1_duty(duty[0]);
         duty[1] = valor_Out2;   //Actualitzem el cicle de treball de la sortida 2.
         set_pwm2_duty(duty[1]);
         break;
      default:
         break;
   }
}
```

Finalment existeixen tres funcions per a enviar les respostes a les funcions de llegir, escriure o canviar paràmetres, més una funció per a enviar els errors. Totes aquestes funcions presenten la mateixa estructura general. A mode d'exemple es mostra la resposta a una lectura d'entrades analògiques.

Primer inicialitzem la variable del *CRC* a 0xFFFF, i després anem calculant els dos bytes de *CRC* segons la quantitat de bytes a enviar. Començant primer per l'adreça de l'esclau, el codi de la funció executada més tots els bytes de les dades a enviar. Un cop calculat el *CRC* activem el bit *PIN_B1* per a transmetre pel bus el datagrama, i anem posant a la *USART* els bytes en l'ordre adequat. Les funcions de resposta per lectura, escriptura o canvi de paràmetres tenen una condició inicial que si hem rebut un *broadcast* no enviarem cap resposta. En cas d'error sí.

```
/* Enviar resposta a les funcions FUNC_LLEGIR_ENTRADES i FUNC_LLEGIR_ESCRIURE_N. */
void enviar_resposta_llegir_entrades(int8 address, int8 funcio,
                                     int8 nBytes, int8 dades[30]){
   int index=0;
   /* Si és un broadcast no contestem */
   if(datagrama.address!=0x00){
      /* Inicialitzem a 1 la variable global CRC */
      modbus_crc.d = 0xFFFF;
      /*Anem calculant el CRC amb les dades a enviar, i en l'ordre que seran enviades */
      calcul_crc(address);
      calcul_crc(funcio);
      calcul_crc(nBytes);
      /* Calculem el CRC amb totes les lectures a enviar */
      for(index=0; index<nBytes; index++){
         calcul_crc(dades[index]);
      }
      /* Esperem un temps de retard que equival al interval de silenci de l'inici
      de la transmissió del datagrama */
      delay_us(retard);
      /* Enviem les dades que formen la resposta, activem PIN_B1 per habilitar la
       transmissió RS-485, esperem un petit retard i seguim */
      output_high(PIN_B1);
      delay_us(10);
      putc(address);
      putc(funcio);
      putc(nBytes);
      /* Enviem totes les lectures realitzades */
```

```
      for(index=0; index<nBytes; index++){
         putc(dades[index]);
      }
      putc(modbus_crc.b[0]);  //Primer el byte LSB.
      putc(modbus_crc.b[1]);  //Segon el byte MSB.
      /* Esperem un temps de retard que equival al interval de silenci del final
      de la transmissió del datagrama */
      delay_us(retard);
   }
}
```

Un cop enviat el datagrama inicialitzem variables per a rebre i tractar un nou datagrama.

Finalment, el bucle infinit dintre de la funció *main()* queda així:

```
/* No sortim del bucle mentre no hi hagi una interrupció */
while(TRUE){
   /* Si hem rebut un datagrama i és dirigeix al nostre dispositiu o adreça 0x00: */
   if(datagrama_In && (slave_Address==datagrama.address || datagrama.address==0x00)){
      switch(datagrama.funcio){
         case FUNC_LLEGIR_ENTRADES:
            /* Comprovem el CRC calculat amb el rebut */
            if((modbus_crc.b[0]!=datagrama.dades[4])||
               (modbus_crc.b[1]!=datagrama.dades[5])){
               enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
            }else{
               /* Comprovem que l'adreça i la quantitat de registres són correctes */
               if((datagrama.dades[0] || datagrama.dades[2] !=0x00) ||
                       (datagrama.dades[1]>=0x04)||
                       ((datagrama.dades[1]+datagrama.dades[3])>=0x05)){
                  enviar_excepcio(slave_Address,datagrama.funcio, ILLEGAL_DATA_ADDRESS);
               }else{
                  /* Tenim 4 entrades analògiques enumerades 0, 1, 2 i 4 */
                  BC = 2*datagrama.dades[3];  //Memoritzar nº bytes a llegir.
                  /* Cridem funció per llegir les N entrades */
                  llegir_entrades(datagrama.dades[1], datagrama.dades[3]);
                  /* Enviem la resposta a la petició. */
                  enviar_resposta_llegir_entrades(slave_Address, datagrama.funcio, BC,
                                                  vector_bytes_entrades);
               }
            }
            break;
         case FUNC_ESCRIURE_SORTIDA:
            /* Comprovem el CRC calculat amb el rebut */
            if((modbus_crc.b[0]!=datagrama.dades[4]) ||
               (modbus_crc.b[1]!=datagrama.dades[5])){
               enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
            }else{
               /* Comprovem que l'adreça de la sortida a escriure comença per 0x00 */
               if((datagrama.dades[0] !=0x00) || ((datagrama.dades[1]!=0x10) &&
                  (datagrama.dades[1]!=0x11))){
                  enviar_excepcio(slave_Address,datagrama.funcio, ILLEGAL_DATA_ADDRESS);
               }else{
                  /* Memoritzem el valor de la sortida a escriure */
                  valor_sortida1 = make16(datagrama.dades[2], datagrama.dades[3]);
                  valor_sortida2 = 0;  //No l'utilitzem en la crida següent.
                  /* Comprovem que el valor està entre 0 i 1023 (0% i 100%).
                  Sinó enviem excepció Valor Incorrecte */
                  if((valor_sortida1!=0 && valor_sortida1>=1024)){
                     enviar_excepcio(slave_Address,datagrama.funcio,ILLEGAL_DATA_VALUE);
                  }else{
                     /* Tenim 2 sortides analògiques enumerades 0x10 i 0x11.*/
                     /* Cridem funció per escriure 1 sortida */
                     escriure_sortides(datagrama.dades[1],0x01,
                                       valor_sortida1,valor_sortida2);
                     /* Enviem la resposta a la petició. */
                     enviar_resposta_escriure_sortides(slave_Address, datagrama.funcio,
                             datagrama.dades[0],datagrama.dades[1],
                             datagrama.dades[2],datagrama.dades[3]);
                  }
               }
            }
            break;
```

```
case FUNC_ESCRIURE_SORTIDES:
   /* Comprovem el CRC calculat amb el rebut */
   /* Si només volem escriure una sortida: */
   if((datagrama.dades[4]==2) && ((modbus_crc.b[0]!= datagrama.dades[7]) ||
         (modbus_crc.b[1]!=datagrama.dades[8]))){
      enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
   /* Si volem escriure més d'una sortida */
   }else if((datagrama.dades[4]==4) && ((modbus_crc.b[0]!= datagrama.dades[9])
            || (modbus_crc.b[1]!=datagrama.dades[10]))){
      enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
   }else{
      /* Comprovem que l'adreça i la quantitat de registres comencen per 0 */
      if((datagrama.dades[0] || datagrama.dades[2] != 0x00) ||
           ((datagrama.dades[1]!=0x10) && (datagrama.dades[1]!=0x11)) ||
           ((datagrama.dades[1]+datagrama.dades[3]) > 0x12)){
         enviar_excepcio(slave_Address,datagrama.funcio, ILLEGAL_DATA_ADDRESS);
      /* El número de sortides a escriure ha de ser correcte */
      }else if(datagrama.dades[3]*2 != datagrama.dades[4]){
         enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
      }else{
         /* Memoritzem els valors de les sortides a escriure */
         valor_sortida1 = make16(datagrama.dades[5], datagrama.dades[6]);
         valor_sortida2 = make16(datagrama.dades[7], datagrama.dades[8]);
         /* Comprovem que els valors estan entre 0 i 1023 (0% i 100%).
         Sinó enviem excepció Valor Incorrecte */
         if((valor_sortida1!=0 && valor_sortida1>=1024) || (valor_sortida2!=0
                  && valor_sortida2>=1024 && datagrama.dades[3]==4)){
            enviar_excepcio(slave_Address,datagrama.funcio,ILLEGAL_DATA_VALUE);
         }else{
            /* Tenim 2 sortides analògiques enumerades 0x10 i 0x11.*/
            /* Cridem funció per escriure les N sortides */
            escriure_sortides(datagrama.dades[1], datagrama.dades[3],
                              valor_sortida1, valor_sortida2);
            /* Enviem la resposta a la petició. */
            enviar_resposta_escriure_sortides(slave_Address, datagrama.funcio,
                  datagrama.dades[0], datagrama.dades[1], datagrama.dades[2],
                     datagrama.dades[3]);
         }
      }
   }
   break;
case FUNC_LLEGIR_ESCRIURE_N:
   /* Comprovem el CRC calculat amb el rebut */
   /* Si només volem escriure una sortida: */
   if((datagrama.dades[8]==2) && ((modbus_crc.b[0]!= datagrama.dades[11]) ||
            (modbus_crc.b[1]!=datagrama.dades[12]))){
      enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
   /* Si volem escriure més d'una sortida */
   }else if((datagrama.dades[8]==4) && ((modbus_crc.b[0]!= datagrama.dades[13])
            || (modbus_crc.b[1]!=datagrama.dades[14]))){
      enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
   }else{
      /* Aquesta funció segueix igual que la FUNC_LLEGIR_ENTRADES */
      if((datagrama.dades[0] || datagrama.dades[2] !=0x00) ||
            (datagrama.dades[1]>=0x04)||
            ((datagrama.dades[1]+datagrama.dades[3])>=0x05)){
         enviar_excepcio(slave_Address,datagrama.funcio, ILLEGAL_DATA_ADDRESS);
      }else{
         /* Continuem llegint les entrades */
         BC = 2*datagrama.dades[3];  //Memoritzar N° de bytes a llegir.
         /* Cridem funció per llegir les N entrades */
         llegir_entrades(datagrama.dades[1], datagrama.dades[3]);
      }
      /* Seguim el programa igual que FUNC_ESCRIURE_SORTIDES */
      if((datagrama.dades[4] || datagrama.dades[6] != 0x00) ||
                ((datagrama.dades[5]!=0x10) && (datagrama.dades[5]!=0x11)) ||
                ((datagrama.dades[5]+datagrama.dades[7]) > 0x12)){
         enviar_excepcio(slave_Address,datagrama.funcio, ILLEGAL_DATA_ADDRESS);
      /* El número de sortides a escriure ha de ser correcte */
      }else if(datagrama.dades[7]*2 != datagrama.dades[8]){
         enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
      }else{
         /* Memoritzem els valors de les sortides a escriure */
         valor_sortida1 = make16(datagrama.dades[9], datagrama.dades[10]);
         valor_sortida2 = make16(datagrama.dades[11], datagrama.dades[12]);
         /* Comprovem que els valors estan entre 0 i 1023 (0% i 100%).
         Sinó enviem excepció Valor Incorrecte */
```

```
            if((valor_sortida1!=0 && valor_sortida1>=1024) || (valor_sortida2!=0
                && valor_sortida2>=1024 && datagrama.dades[8]==4)){
               enviar_excepcio(slave_Address,datagrama.funcio,ILLEGAL_DATA_VALUE);
            }else{
               /* Tenim 2 sortides analògiques enumerades 0x10 i 0x11.*/
               /* Cridem funció per escriure les N sortides */
               escriure_sortides(datagrama.dades[5], datagrama.dades[7],
                                 valor_sortida1, valor_sortida2);
               /* Enviem la resposta a la petició */
               enviar_resposta_llegir_entrades(slave_Address,datagrama.funcio, BC,
                                               vector_bytes_entrades);
            }
         }
      }
      break;
   case FUNC_CANVI_PARAMETRES:
      /* Comprovem el CRC calculat amb el rebut */
      if((modbus_crc.b[0]!=datagrama.dades[2])||
         (modbus_crc.b[1]!=datagrama.dades[3])){
         enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_DATA_VALUE);
      }else{
         /* Comprovem si volem canviar l'adreça o modificar la velocitat */
         if(datagrama.dades[0]==0x00){
            /* Comprovem que la nova adreça és vàlida */
            if((datagrama.dades[1]>=1) && (datagrama.dades[1]<=247)){
               int8 old_Address = slave_Address;  //Memoritzem antiga adreça.
               slave_Address = datagrama.dades[1];//Actualitzem la variable global
               /* Enviem la resposta a la petició. */
               enviar_resposta_parametres(slave_Address,datagrama.funcio,
                                          datagrama.dades[0], old_Address);
            }else{
               /* Si la nova adreça és 0 o major a 247 enviem */
               enviar_excepcio(slave_Address,datagrama.funcio,ILLEGAL_DATA_VALUE);
            }
         }else if(datagrama.dades[0]==0x01){
            /* Segons el codi rebut, actualitzem la UART */
            switch(datagrama.dades[1]){
               /* 0 -> 1200bauds; 1 -> 2400bauds; 2 -> 9600bauds; 3 -> 19200bauds;
                  4 -> 28800bauds; 5 -> 33600bauds; altres valors no són vàlids
               Segons la velocitat de la UART haurem de contar X interrupcions.
               Més o menys és el temps de 3,5 caràcters entre datagrames.
               STOP_COUNT = 3,5*8bits/bauds/128,0us  -> 128,0 és el TIMER0 */
               case 0:
                  bauds = 1200;           //Actualitzem variable global.
                  stop_count = 183;       //Actualitzem nº de INT_TIMER0.
                  set_uart_speed(1200);   //Actualitzem velocitat USART.
                  break;
               case 1:
                  bauds = 2400;
                  stop_count = 91;
                  set_uart_speed(2400);
                  break;
               case 2:
                  bauds = 9600;
                  stop_count = 23;
                  set_uart_speed(9600);
                  break;
               case 3:
                  bauds = 19200;
                  stop_count = 12;
                  set_uart_speed(19200);
                  break;
               case 4:
                  bauds = 28800;
                  stop_count = 8;
                  set_uart_speed(28800);
                  break;
               case 5:
                  bauds = 33600;
                  stop_count = 7;
                  set_uart_speed(33600);
                  break;
               default:
                  /* Si la nova velocitat no està definida */
                  enviar_excepcio(slave_Address,datagrama.funcio,
                                  ILLEGAL_DATA_VALUE);
                  break;
```

```
                  }
                  /* Enviem la resposta a la petició i actualitzem variables. */
                  retard = (35000000/bauds);
                  enviar_resposta_parametres(slave_Address,datagrama.funcio,
                                             datagrama.dades[0],datagrama.dades[1]);
               }else{
                  /* Si volem canviar un altre paràmetre no */
                  enviar_excepcio(slave_Address,datagrama.funcio, ILLEGAL_DATA_ADDRESS);
               }
            }
            break;
         default:
            /* Si la funció no està implementada en el dispositiu,
            enviem excepció de Funció Incorrecte */
            enviar_excepcio(slave_Address, datagrama.funcio, ILLEGAL_FUNCTION);
            break;
      }
      /* Inicialitzem variables per rebre una nova petició o datagrama */
      event = 0;
      mesDades = 0;
      datagrama_In = FALSE;
      /* Ja hem acabat de transmetre la trama i per tant desactivem
      PIN_B1 per a poder rebre noves trames */
      output_low(PIN_B1);
   /* Si el datagrama no va dirigit al nostre dispositiu no actuem i inicialitzem */
   }else if(datagrama_In && (slave_Address!=datagrama.address && datagrama.address!=0)){
      event = 0;
      mesDades = 0;
      datagrama_In = FALSE;
   }
}
```

Per a calcular el CRC s'utilitza la següent funció, la seva explicació es troba al punt 5.1:

```
/* Variable global per emmagatzemar el valor del CRC rebut o enviat */
union{
   int8 b[2];  //2 Bytes.
   int16 d;    //Actualitza els dos bytes a l'hora.
} modbus_crc;

/* Taula dels valors CRC per al byte de major ordre */
const unsigned char modbus_auchCRCHi[] = {
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,
0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40
};

/* Taula dels valors CRC per al byte de menor ordre */
const char modbus_auchCRCLo[] = {
0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06,0x07,0xC7,0x05,0xC5,0xC4,
0x04,0xCC,0x0C,0x0D,0xCD,0x0F,0xCF,0xCE,0x0E,0x0A,0xCA,0xCB,0x0B,0xC9,0x09,
0x08,0xC8,0xD8,0x18,0x19,0xD9,0x1B,0xDB,0xDA,0x1A,0x1E,0xDE,0xDF,0x1F,0xDD,
0x1D,0x1C,0xDC,0x14,0xD4,0xD5,0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3,
0x11,0xD1,0xD0,0x10,0xF0,0x30,0x31,0xF1,0x33,0xF3,0xF2,0x32,0x36,0xF6,0xF7,
0x37,0xF5,0x35,0x34,0xF4,0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A,
0x3B,0xFB,0x39,0xF9,0xF8,0x38,0x28,0xE8,0xE9,0x29,0xEB,0x2B,0x2A,0xEA,0xEE,
0x2E,0x2F,0xEF,0x2D,0xED,0xEC,0x2C,0xE4,0x24,0x25,0xE5,0x27,0xE7,0xE6,0x26,
0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,0xA0,0x60,0x61,0xA1,0x63,0xA3,0xA2,
0x62,0x66,0xA6,0xA7,0x67,0xA5,0x65,0x64,0xA4,0x6C,0xAC,0xAD,0x6D,0xAF,0x6F,
0x6E,0xAE,0xAA,0x6A,0x6B,0xAB,0x69,0xA9,0xA8,0x68,0x78,0xB8,0xB9,0x79,0xBB,
```

```
0x7B,0x7A,0xBA,0xBE,0x7E,0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,0xB4,0x74,0x75,0xB5,
0x77,0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71,0x70,0xB0,0x50,0x90,0x91,
0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,0x9C,0x5C,
0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B,0x99,0x59,0x58,0x98,0x88,
0x48,0x49,0x89,0x4B,0x8B,0x8A,0x4A,0x4E,0x8E,0x8F,0x4F,0x8D,0x4D,0x4C,0x8C,
0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,
0x40
};

/* Calculem el CRC de la dada rebuda o a enviar i actualitzem el CRC global del missatge
rebut/enviat */
void calcul_crc(int8 data){
  int8 uIndex ; //Índex per referenciar-nos a les taules anteriors.

  uIndex = (modbus_crc.b[1]) ^ data; //Calculem el CRC.
  modbus_crc.b[1] = (modbus_crc.b[0]) ^ modbus_auchCRCHi[uIndex];
  modbus_crc.b[0] = modbus_auchCRCLo[uIndex];
}
```

Per a més detalls consultar l'arxiu ***piris_modbus.c*** entregat amb aquesta memòria. Realitzant les oportunes simulacions amb el programa *Real Pic Simulator* s'ha anat depurant el codi. Una captura de pantalla d'una simulació utilitzant la funció 0x17, per a llegir les quatre entrades i activar la sortida 1 a un 50% i la sortida 2 a un 25%, és:

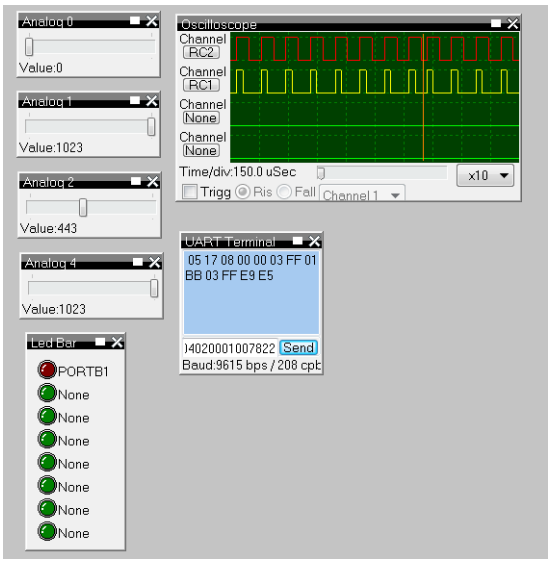


Figura 43 Simulació PIC

La petició que hem enviat és:

05 17 00 00 00 04 00 10 00 02 04 02 00 01 00 78 22

I la seva resposta és:

05 17 08 00 00 03 FF 01 BB 03 FF E9 E5

Com podem veure ens ha contestat el dispositiu 0x05, a la funció 0x17, on la primera entrada analògica està a zero, la segona entrada a $10V$, la tercera entrada aproximadament a $2{,}16V$, finalment la quarta entrada també està a $10V$.

Els últims 2 bytes corresponen al CRC. Per comprovar el seu valor utilitzarem la següent web:

http://www.lammertbies.nl/comm/info/crc-calculation.html

Com podem veure els bytes 05 17 08 00 00 03 FF 01 BB 03 FF tenen com a *CRC* E9 E5.

I els polsos de les dues sortides podem veure com es corresponen amb un 50% i un 25% del cicle de treball, tal i com havíem demanat a la petició.

7. Disseny del layout de la placa PCB

En aquest apartat exposarem la realització dels *layout* de les dues PCB a realitzar. La versió *freeware* del programa *Eagle PCB v6.3.0*, de *Cadsoftusa*, només ens permet fer dissenys de plaques d'una superfície de $100\ mm$ d'ample per $80\ mm$ d'alçada. A més, només ens permet fer el disseny utilitzant les capes *top* i *bottom*. Per aquests motius s'ha decidit fer el disseny en dues parts.

Una primera part conté totes les etapes d'adaptació de les quatre entrades i les dues sortides analògiques, l'etapa de comunicació entre la *USART* del *PIC* i el bus *RS-485*, i finalment l'etapa de control que està formada pel *PIC16F876A*.

I una segona part conté la font d'alimentació que admet una entrada de 12 a $48V$ i té unes sortides de $+15V$, $-15V$, $+5V$, $-1V$ i massa, tal i com ja hem vist a l'apartat 4.

En primer lloc farem dues taules amb el llistat de tots els elements necessaris per a dissenyar les *PCB*, per després fer l'esquemàtic de les dues plaques (què es poden veure a l'annex 11 d'aquesta memòria).

Finalment, seguint les indicacions del mòdul 1 de la documentació *Aplicacions electromagnètiques i electròniques i introducció al càlcul d'errors*[3] i el manual del programa, hem procedit a fer el disseny de les dues plaques. S'han tingut en compte temes d'espai entre components (per temes de dissipació de calor), i pistes, així com l'amplada segons quin tipus de càrrega havia de suportar cada pista. S'ha utilitzat tant la cara superior com la inferior degut a la gran quantitat d'elements a utilitzar.

7.1. Llistat diferents elements de la placa PCB

Primer exposem la llista d'elements de la primera *PCB* que conté la comunicació, les entrades i sortides, i l'etapa de control de l'expansor:

Codi	Valor Referència	Descripció	Encapsulat
C1,C2,C3,C4	10 nF	Filtratge PWM	C025
IN_1, IN_2, IN_3, IN_4	OPA4277	Amplificadors Entrades Analògiques	DIP14
JP1,JP2,JP3,JP4, JP5,JP6,JP8	1x2	Connexions per a 2 cables	1x02
JP7	1x5	Connexions per a 5 cables	1x05
MICRO	PIC16F876A	Microchip: Controlador expansor	DIP28
MODBUS	ISO3088	Texas: Adaptador UART-RS485	SOIC16
OUT_1,OUT_2	OPA551	Amplificadors Sortides Analògiques	DIP8
QF1	8MHz	Oscil·lador extern per a PIC	UM1
R1,R5,R14A,R14B,R15A, R21A,R21B,R22A	2K	Resistències per a realimentacions amplificadors operacionals	0204V
R2,R3,R4,R6,R7,R8,R10, R11,R12,R13,R17,R18,R19, R20,R25,R26,R29,R30	1K	Resistències per a realimentacions amplificadors operacionals	0204V
R9,R16	250 Ohms	Convertidor intensitat-tensió	0204V
R15B,R22B	3K	Resistències per a realimentacions amplificadors operacionals	0204V
R23,R24,R27,R28	3K9	Filtratge PWM	0204V

Taula 14 Llistat elements PCB1.

I aquesta és la llista de la *PCB* de la font d'alimentació:

Codi	Valor Referència	Descripció	Encapsulat
C1,C2,CD,CIN	2,2 uF	Filtratge i compensació senyals	C025
CC1	10 pF	Filtratge i compensació senyals	C025
CC2	8,2 nF	Filtratge i compensació senyals	C025
CCIN, CPIN	1 uF	Filtratge i compensació senyals	C025
CI	0,32 uF	Filtre entrada LM7805	C025
CO	0,1 uf	Filtre sortida LM7805	C025
COUT1	470 uF	Condensador sortida +15V	C025
COUT2	220 uF	Condensador sortida -15V	C025
D1,D2	1A, 90V	Regular sortides de +-15V	SMB
IC1	LM7805	Regulador tensió de 5V	TO220V
IC2	OPA277	A.O. per a tensió de -1V	SO-8
JP1	1x2	Connexions per a 2 cables	1x02
JP2	1x5	Connexions per a 5 cables	1x05
L1,L2	18 uH	Inductor per a sortida de +-15V	CMS1-1-R
Q1	FDMC86106LZ	Transistor FET	SO-8
Q2	80V, 300mW	Transistor NPN regular tensió	SOT-23
R1	15K	Resistència per a OPA277	0204V
R2	1K	Resistència per a OPA277	0204V
RB1	64,9K	Regular tensió entrada	0204V
RB2	866 Ohms	Regular tensió entrada	0204V
RC1	649K	Compensació senyals	0204V
RD	0,36 Ohms	Compensació senyals	0204V
RF1	49K9	Controlar sortida font	0204V
RF2	4K42	Controlar sortida font	0204V
RFREQ	60K4	Freqüència de l'oscil·lador	0204V
RIN	1 Ohm	Compensació senyals	0204V
RS	634 Ohms	Compensació senyals	0204V
RSEN	0,022 Ohms	Compensació senyals	0204V
U1	ADP1621	DC-DC Analog Devices	MSOP-10
Z1	4,7V 500mW	Diode Zener de 4,7V	SOD80C

Taula 15 Llistat elements PCB2.

Com podem veure a les dues taules, hi ha diversos tipus d'encapsulats, alguns són de muntatge superficial (es col·loquen i solden a la cara superior de la *PCB*) i alguns altres són de muntatge superficial però es solden per la cara de baix, el què implica que s'han de fer perforacions a les *PCB* per a poder col·locar aquests elements. Les resistències perquè ocupin menys espai s'instal·laran en vertical, en lloc d'horitzontal, encara que sigui més difícil el seu muntatge ens permetrà més espai a la placa.

Al tenir el disseny en dues *PCB* hem hagut de dotar a aquestes de connectors per tal de poder connectar les entrades i sortides analògiques, l'alimentació interna i externa del dispositiu i la connexió del bus *RS-485*.

7.2. Disseny de la placa PCB

Un cop llistats tots els elements necessaris, tant la quantitat com el seu valor, també és important saber quin tipus de encapsulat presenten, ja què d'ell depèn el seu muntatge sobre la *PCB*.

Pel disseny de la primera *PCB* hem dotat a la mateixa de 6 connectors dobles anomenats *JPx* per a poder connectar tant les entrades com les sortides analògiques. El primer pin del connector és l'entrada o la sortida dels senyals a tractar, i el segon pin és la connexió a massa.

Un altre connector doble és per a la connexió del bus *RS-485*. El primer pin del connector és pel cable A del bus (o $D0$), i el segon pin pel cable B (o $D1$).

Finalment tenim un connector de 5 pins per a l'alimentació dels circuits. El primer pin és per a la massa del circuit, el segon i tercer pin aporten els $+15V$ i $-15V$ pels amplificadors operacionals i les sortides analògiques. El quart pin és per alimentar amb $+5V$ el microcontrolador i l'adaptador de *USART* a *RS-485*. Finalment el cinquè pin és per a la petita alimentació dels $-1V$ per a poder adaptar les entrades analògiques d'intensitat.

L'esquema del circuit final el podem veure a la figura 46 de l'apartat 11 d'annex. Tenint en compte les consideracions de disseny exposades a la introducció d'aquest apartat, i que el color blau indica la capa *bottom*, el color vermell la capa *top* i que el color verd són perforacions a la *PCB*, el *layout* del disseny final d'aquesta placa *PCB* és:

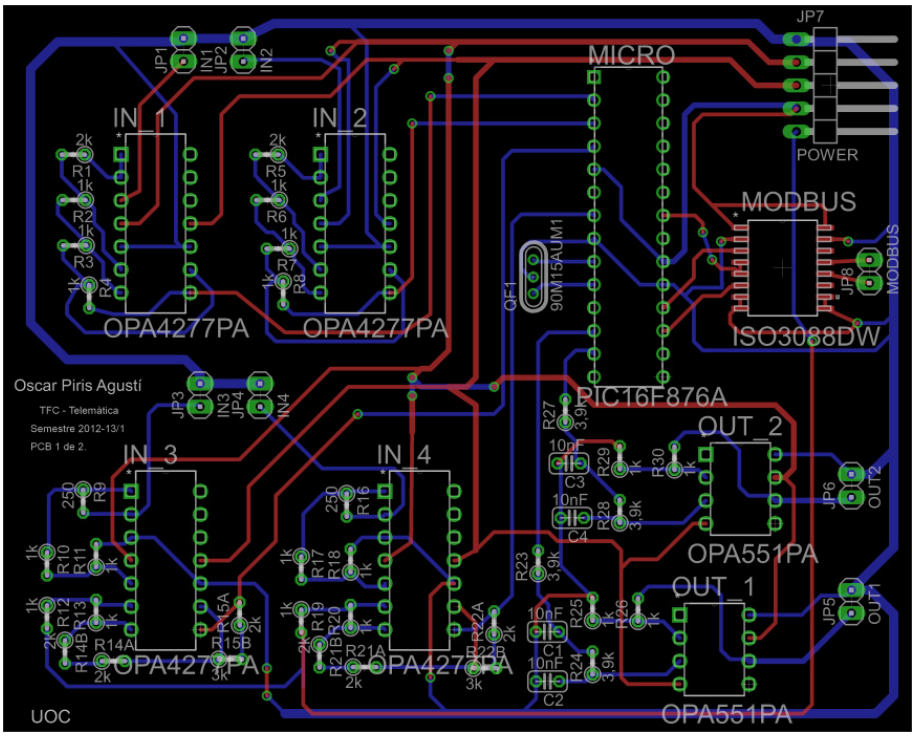


Figura 44 Layout PCB1.

Com podem veure, la quantitat d'elements és importat, el què ens obliga a utilitzar les dues capes i fer vies per anar d'un punt a l'altre. L'oscil·lador al funcionar a $8MHz$ s'ha col·locat el més pròxim al *PIC* possible i separat de les entrades per tal d'evitar interferències en els senyals d'entrada. L'adaptador de *USART* a *RS-485* al utilitzar velocitat elevades, també podria afectar als senyals d'entrada, per aquest motiu s'ha deixat a un costat de la placa. Les entrades es troben a l'esquerra i finalment les sortides davall del *PIC*.

Per tal de donar bona cobertura s'ha dissenyat dues pistes principals de massa, les quals són més amples que la resta de pistes. Les dimensions d'aquesta *PCB* és de $100\ mm$ d'ample i de $80\ mm$ d'alçada.

Pel disseny de la segona *PCB* hem dotat a la mateixa d'un connector doble per a la connexió de l'alimentació exterior de l'expansor. El primer pin del connector és per al voltatge positiu, i el segon pin per a la massa. Al igual que l'altra *PCB*, també s'ha dotat d'un connector per a

l'alimentació d'aquella placa, on en aquest cas al estar instal·lat a 180º respecte de l'altre, els valors dels seus pins canvien. El primer pin és ara pels $-1V$, el segon pin és pels $+5V$, els pins tres i quatre són pels voltatges de $-15V$ i $+15V$, i finalment el cinquè pin és massa.

L'esquema del circuit final també el podem veure a la figura 47 de l'apartat 11 d'annex. La dificultat que he tingut per aquest disseny és que no he trobat els models de *Eagle* de tots els dispositius que necessitava, així que he utilitzat altres elements amb el mateix encapsulat. D'aquesta manera al situar l'element real sobre l'empremta encaixarà perfectament. Concretament el *DC-DC ADP1621* de *Analog Devices* té un encapsulat *MSOP-10*, i l'element utilitzat per a realitzar l'esquemàtic no correspon amb el nom dels seus pins, sí però amb les connexions físiques. El *layout* del disseny final d'aquesta placa *PCB* és:

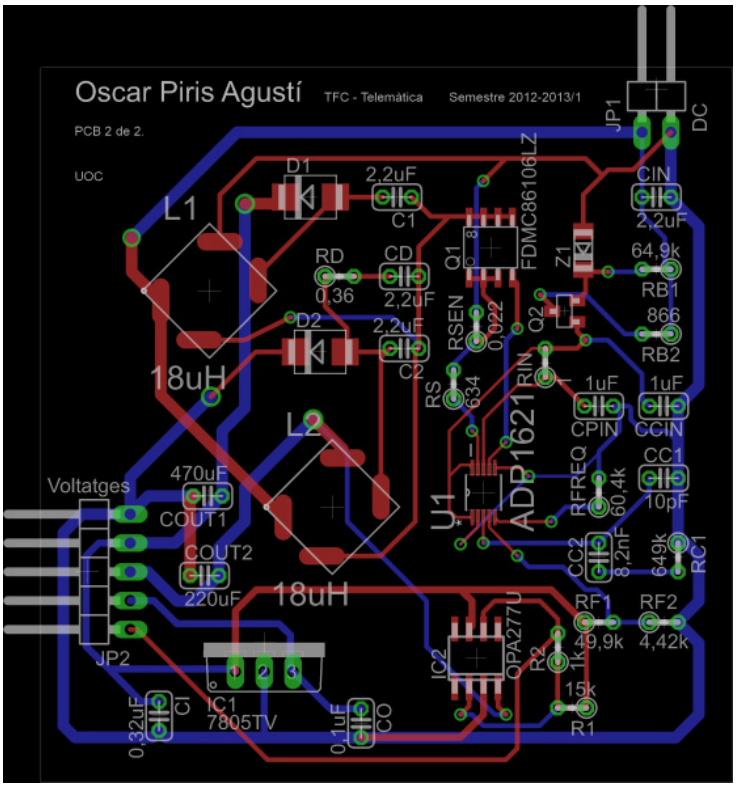


Figura 45 Layout PCB2.

Com que la corrent que ha de suportar aquest circuit és elevada, s'han dissenyat les pistes d'alimentació i de sortida més amples que la resta de pistes. Tant els inductors com el regulador de cinc volts, són elements que s'escalfen molt durant la conversió de tensions. Una altre element que també s'escalfa bastant és el convertidor de corrent continua en corrent contínua. Tenint en compte aquests escalfaments s'han repartit per a tota la superfície de la placa per tal de donar-los espai suficient per a dissipar la calor i/o per a poder instal·lar uns dissipadors a sobre. Com que la majoria de resistències i condensadors són per a configurar el *ADP1621* estan distribuïts de forma conjunta en un costat de la placa. Tots els elements de control de *L1* i *L2* estan distribuïts per la part alta de la placa. Finalment la part central i la part esquerra és per a on passen les pistes d'alimentació de sortida i per a instal·lar el connector.

Encara que no s'aprecia molt be a la figura, les dimensions d'aquesta *PCB* és de $60\ mm$ d'ample i de $75\ mm$ d'alçada.

8. Conclusions

Tal i com hem pogut anar comprovant al llarg de tots els apartats, aquest TFC és bastant complert ja que abasta varies temàtiques. Hem tractat en un primer apartat tota la planificació de les tasques i fites per a realitzar aquest TFC.

A l'apartat 3 hem adaptat tant les sortides com les entrades analògiques, per tant, hem fet ús de tots els coneixements d'electrònica analògica, tant pel que fa a amplificadors operacionals com pel que fa a *PWM* i filtres de primer ordre.

A l'apartat 4 hem creat una font d'alimentació que admet a la seva entrada tensions de 12 a 48 volts, i té unes sortides de +15V, -15V, +5V, -1V i massa. Hem pogut observar com es poden utilitzar diversos dispositius per a fer-ho realitat. En aquest TFC hem utilitzat un convertidor *DC-DC* per a transformar la tensió d'entrada de 12-48V a ±15V. A partir de la tensió de +15V obtenim la tensió de +5V mitjançant un regulador de tensió, i també mitjançant un amplificador operacional obtenim els -1V.

Un cop hem fet els dissenys electrònics tant de les adaptacions com de la font d'alimentació, a l'apartat 7 hem fet el disseny dels *layouts* de les *PCB* del nostre expansor. Per a fer aquests dissenys, hem fet un llistat de tots els components necessaris tenint en compte els seus encapsulats.

Als apartats 2 i 5 hem explicat el funcionament del protocol *ModBus* i el bus *RS-485*. Com hem pogut veure, *ModBus* defineix els datagrames a enviar/rebre, així com les funcions que s'envien, i els codis d'error o excepcions. En canvi el bus *RS-485* defineix les característiques elèctriques de funcionament del bus.

Finalment, per a controlar les entrades i sortides analògiques i la comunicació, el nostre expansor utilitza un microcontrolador PIC16F876A. A l'apartat 6 s'han explicat les seves principals característiques i funcionament. També hem explicat el programa de control de l'expansor d'entrades i sortides analògiques per *ModBus RTU* sobre *RS-485*.

És a dir, hem tractat els temes de gestió de projectes, d'electrònica analògica, d'electrònica digital, de microcontroladors, les capes física, d'enllaç i d'aplicació del model *OSI*, i de disseny de *PCB*.

A l'apartat 5 hem vist tot un llistat de diferents funcions que aporta el protocol *ModBus*, i que nosaltres només hem aplicat les principals. Una futura línea de continuació seria la implementació de la resta de funcions, ja que només seria anar afegint diferents opcions dintre del bucle principal del programa, concretament dintre del *SWITCH*.

Aquestes funcions podrien ser les funcions 0x08 i 0x07, les quals s'utilitzen per a recollir informació sobre l'estat de la xarxa i/o dels dispositius. La funció 0x08 defineix diferents subcodis de les diferents funcions que és capaç de realitzar.

Encara que la principal línea de continuació d'aquest TFC seria la del muntatge real tant de la font d'alimentació com dels circuits d'adaptació d'entrades i sortides analògiques, i de comunicació i control.

Nosaltres en aquest TFC hem fet el disseny i la simulació a nivell de software, i per tant ens queda pendent la simulació dels circuits reals. Primerament realitzaríem els muntatges en una *protoboard* per anar simulant els circuits electrònics i anar fent mesures de les diferents tensions, intensitats, i temperatures de tots els components electrònics.

Un cop estem segurs que els circuits electrònics no presenten errades greus de disseny, ja podem procedir a la creació i muntatge dels dissenys *PCB* realitzats en aquest TFC.

Un cop muntats els circuits reals del nostre expansor, hem de simular el programa de control, tot comprovant les entrades com les sortides. Aquesta part segurament seria la més laboriosa, ja que abans de posar al mercat el nostre producte ens hem d'assegurar que realitza correctament totes les funcions dissenyades, i també ha de ser capaç de donar solució a totes les possibles fallades del sistema, etc.

Per tant, una futura línia de continuació del TFC és la realització del muntatge real.

Finalment, farem una petita valoració econòmica del nostre expansor d'entrades i sortides analògiques per *ModBus RTU* sobre *RS-485*. Totes les resistències les agruparem al mateix preu, al igual que els condensadors, ja que les diferencies de preu no són significatives. Els preus s'han consultat el dia 16 de desembre de 2012 de la pàgina http://www.micropik.com/.

Component	***Quantitat***	***Preu Unitari***	***Total***	***Total amb IVA***
Resistències	42	0,042€	1,76€	2,13€
Condensadors	16	0,120€	1,92€	2,32€
Connector 2 Pin	8	0,070€	0,56€	0,68€
Connector 5 Pin	2	0,160€	0,32€	0,39€
Díodes	3	0,330€	0,99€	1,20€
Transistor NPN	1	0,450€	0,45€	0,54€
Transistor MOSFET	1	2,270€	2,27€	2,75€
Inductors	2	4,150€	8,30€	10,04€
Amplificadors OPA4277	4	0,950€	3,80€	4,60€
Amplificadors OPA551	2	1,590€	3,18€	3,85€
Amplificador OPA277	1	0,750€	0,75€	0,91€
Regulador LM7805	1	0,890€	0,89€	1,08€
Cristall 8 MHz	1	0,840€	0,84€	1,02€
PIC16F876A	1	8,110€	8,11€	9,81€
ISO3088 RS-485	1	5,010€	5,01€	6,06€
Convertidor DC-DC ADP1621	1	1,860€	1,86€	2,25€
Hores Enginyeria	329,8	30,000€	9894,00€	11.971,74€
			Total	**12.021,37€**

Taula 16 Valoració econòmica

Com podem veure el disseny d'aquest prototip està valorat amb *12.021,37€* aplicant un IVA del 21%. Si no tenim en compte aquestes hores de disseny, el prototip sense muntatge puja a un total de *49,63€*. Per tant, el seu valor en el mercat serà aquest preu més les hores del seu muntatge i prorratejant les hores de disseny entre la quantitat de dispositius a fabricar.

Per a una producció d'uns 1000 dispositius i unes 5 hores de muntatge per dispositiu, el preu final de l'expansor podria ser d'uns *211,60€*, encara que aquí faltaria afegir les despeses generals (normalment un 13%) més els beneficis industrials (normalment un 6%). Per tant, estaríem parlant d'un preu final, de venta al públic, aproximat de:

$$\boxed{\mathbf{251,80}€}$$

9. Glossari

ACII: *American Standard Code for Information Interchange*, és un codi de caràcters llatí utilitzat en informàtica. Cada caràcter ocupa 1 byte.

AO: Amplificador operacional.

Bauds: És una unitat de mesura utilitzada en telecomunicacions que indica quantes vegades canvia d'estat un senyal per segon.

Bottom: Cara inferior d'una PCB.

CAD: Convertidor de analògic a digital.

CRC: Comprovació de redundància cíclica, és un codi de detecció d'errors.

CDA: Convertidor de digital a analògic.

CÙK: Convertidor de corrent continu a corrent continu.

Daisy-Chain: És una instal·lació on els dispositius estan connectats entre ells un a un.

Duty Cicle: Cicle de treball d'un senyal *PWM*.

ESR: Resistència sèrie equivalent d'un condensador.

FIFO: *First In First Out*, memòria on la primera dada rebuda és la primera que surt.

ModBus RTU: Protocol de comunicacions situat al nivell 7 del model *OSI*.

MOSFET: Transistor d'efecte de camp metall-òxid-semiconductor.

NPN: És un dels dos tipus de transistors bipolars, on té dues capes dopades N i una capa dopada P.

OSI: Model d'interconnexió de sistemes oberts creat per la *ISO*.

PAC: Prova d'avaluació continuada de les assignatures de la UOC.

PCB: Placa de circuit imprès. Placa on s'instal·laran tots els components electrònics.

PDU: Unitat de dades de protocol.

Pull-up: Resistència de polarització que eleva la tensió de sortida d'un dispositiu digital.

Pull-down: Resistència de polarització que disminueix la tensió de sortida d'un dispositiu digital.

PWM: Senyal d'ampla de banda modulat.

SEPIC: *Single-ended primary-inductor converter*. Convertidor de tensió a base de inductors.

TFC: Treball final de carrera.

Top: Cara superior d'una PCB.

RS-485: Especificacions elèctriques d'un bus sèrie que només afecta a la capa física del model *OSI*.

UOC: Universitat Oberta de Catalunya.

USART: Control mitjançant bus sèrie d'un *PIC*.

µC: Microcontrolador.

V_{RMS}: Valor mitjà d'un senyal.

10. Bibliografia

1. VV.AA., "Sistemes electrònics digitals" FUOC. Barcelona. 2006.
2. VV.AA., "Teoria de circuits" FUOC. Barcelona. 2010.
3. VV.AA., "Aplicacions electromagnètiques i electròniques i introducció al càlcul d'errors". FUOC. Barcelona. 2010.
4. VV.AA., "Gestió de projectes" FUOC. Barcelona. 2010.
5. VV.AA., "Treball final de carrera" FUOC. Barcelona. 2008.
6. VV.AA., "Competència comunicativa per a professionals de les TIC" FUOC. Barcelona. 2010.
7. VV.AA., "Modbus over Serial Line" Modbus.org [en línia] [data consulta: 02-10-2012] http://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf
8. VV.AA., "Isolated Half-duplex RS-485 Transceivers" Texas Instruments [en línia] [data consulta: 03-10-2012] http://www.ti.com/lit/ds/symlink/iso3080.pdf
9. ANGULO USATEGUI, J.M., "Enciclopedia de Electrónica Moderna" Paraninfo, SA. Madrid. 1995.
10. VV.AA., "High Precision Operation Amplifiers" Burr-Brown [en línia] [data consulta: 08-10-2012] http://www.ti.com/lit/ds/symlink/opa4277.pdf
11. VV.AA., " High Current Operation Amplifiers" Burr-Brown [en línia] [data consulta: 11-10-2012] http://www.ti.com/lit/ds/symlink/opa552.pdf
12. PALACHERLA, A., "AN538 - Using PWM to Generate Analog Output" Microchip Technology Inc. [en línia] http://ww1.microchip.com/downloads/en/AppNotes/00538c.pdf . 1997
13. ALTER, D.M., "SPRAA88A - Using PWM Output as a DAC" Texas Instruments. [en línia] [data consulta: 08-10-2012] http://www.ti.com/lit/an/spraa88a/spraa88a.pdf
14. BRUCE CARLSON, A., "Teoría de Circuitos" Paraninfo, SA. Madrid. 2002.
15. VV.AA., "Constant-Frequency, Current-Mode Step-Up DC/DC Controller ADP1621" Analog Devices [en línia] [data consulta 25-10-2012] http://www.analog.com/static/imported-files/data_sheets/ADP1621.pdf
16. VV.AA., "Dual-Output ADP1621 Sepic-Cùk" Analog Devices [en línia] [data consulta 25-10-2012] http://www.analog.com/static/imported-files/pwr_mgmt/RD/PRD1101_15V0_DualPolarityADP1621Sepic-Cuk.pdf
17. VV.AA., "LM78XX 3-Terminal 1A Positive Voltage Regulator" Fairchild Semiconductor [en línia] [26-10-2012] http://www.fairchildsemi.com/ds/LM/LM7805.pdf
18. VV.AA., "Modbus Application Protocol Specification" Modbus.org [en línia] [data consulta: 06-11-2012] http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf
19. VV.AA., "PIC16F87XA Data Sheet" Microchip [en línia] [data consulta: 09-11-2012] http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf
20. VV. AA., "PCW C Compiler Reference Manual" Custom Computer Services, Inc. [en línia] [data consulta: 18-11-2012] http://www.ccsinfo.com/downloads/ccs_c_manual.pdf

11. Annex

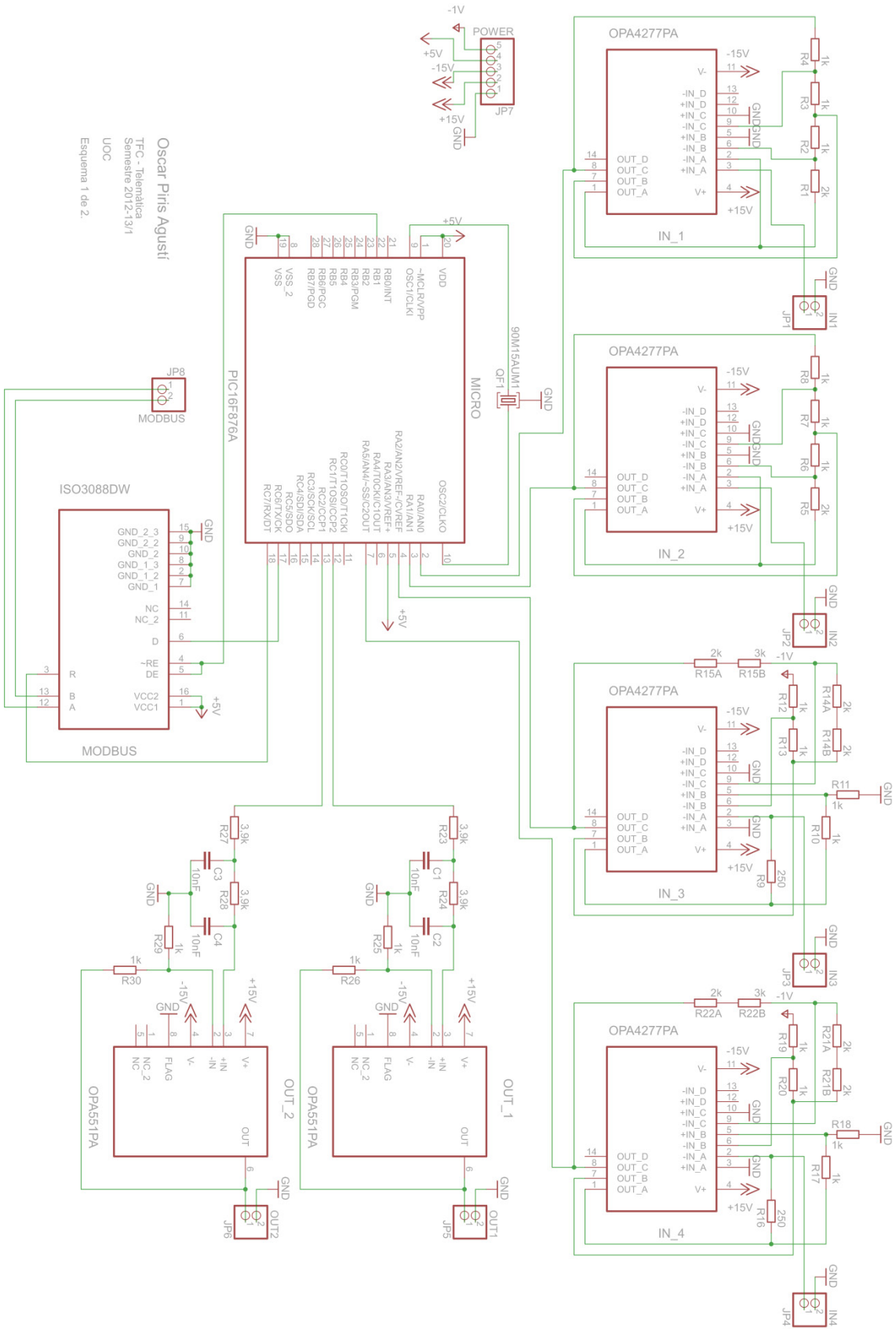


Figura 46 Esquema PCB1.

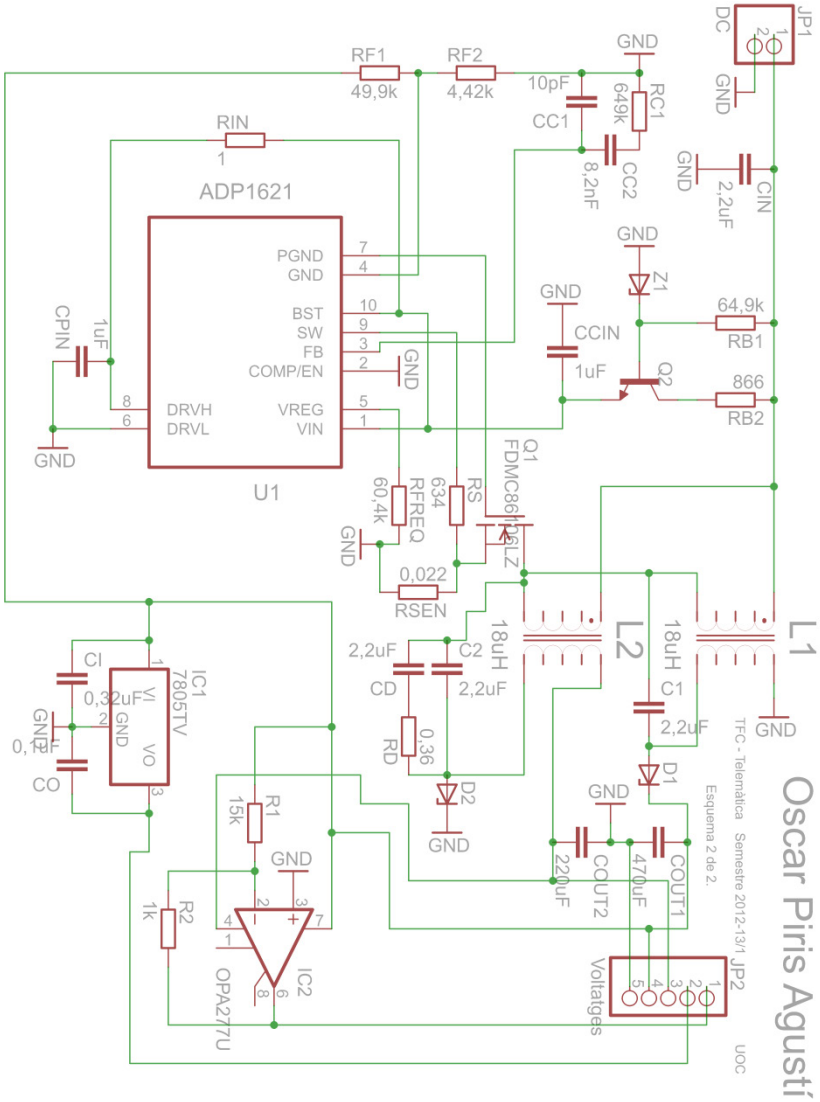


Figura 47 Esquema PCB2.