UOC

UNIVERSITAT OBERTA DE CATALUNYA

Ingeniería Técnica de Telecomunicación

(Especialidad Telematica)

Diseño de un panel de LEDs

Alumno/a: Ignacio Francés Caballero

Dirigido por: Carlos Alberto Pacheco Navas

CURSO 2014-15 (Septiembre)

Agradecimientos

En primer lugar a toda la comunidad de la UOC que he ido encontrando en este largo camino.

Como no, un recuerdo para mis compañeros de trabajo Pablo y Jose Luis. Sin ellos las fuerzas me hubieran abandonado, pero sus ánimos siempre me motivaron para ir más allá.

Gracias a Carlos Alberto Pacheco por guiarme por el buen camino al inicio del TFC.

Y sobre todo, gracias a mis hijos Iñigo y Darío por respetar mis horas de estudio, a mi vida, María por su paciencia, comprensión y ánimos en los momentos duros que no han sido pocos en estos años.

Gracias a todos.

Presentación del autor del TFC

Este trabajo final de carrera ha sido realizado por **Ignacio Francés Caballero.**

Decidido a finalizar los estudios de ingeniería que comenzó a cursar en la Escuela Politécnica de La Almunia, emprendió hace seis años el largo trayecto de cursar Ingeniería Técnica de Telecomunicaciones por la Universidad Oberta Catalunya.

Actualmente, trabaja como técnico electrónico en el departamento de I+D de la empresa Azkoyen SA.

Su gran pasión, la tecnología, el deporte y un buen libro. Su debilidad, sus hijos y su esposa.

1 Índice

1.1 Índice de contenidos

1.2 Índice de tablas

1.3 Índice de ilustraciones

2 Introducción

2.1 Enunciado del TFC

El TFC se centra en diseñar un panel de *LEDs* capaz de mostrar cualquier información transmitida a través de una conexión RS232 y mediante el protocolo Modbus. Además la solución deberá ser eficiente energéticamente minimizando el consumo energético del producto.

El producto estará dividido en los siguientes bloques:

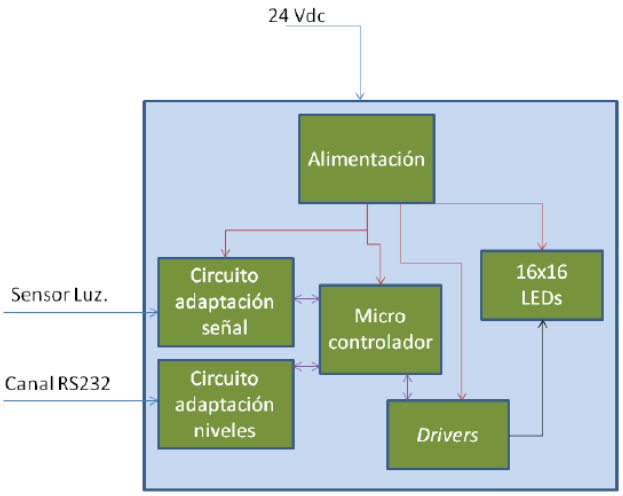


Ilustración 1 - Esquema de bloques

La función de cada bloque será la siguiente:

- **Alimentación:** se encargara de adaptar la tensión de entrada a los niveles de tensión necesarios de cada uno de los bloques.
- **Circuito de adaptación de señal:** mediante un sistema de amplificación adaptara la señal de un sensor de luz. La salida de este bloque ira conectada directamente a una entrada analógica del microprocesador. Dependiendo del nivel de luz los *LEDs* deberán lucir más o menos.
- **Circuito adaptación de niveles:** tiene que realizar la conversión de los niveles RS232 del PC a niveles TTL para que el microprocesador pueda procesarlos.
- **Microcontrolador:** mediante un algoritmo grabado en la Flash realizara el control de todas las funciones de control de *LEDs*, medición de sensores y comunicaciones.
- **Drivers:** es el conjunto de circuitos que controlara mediante un numero *n* de entradas y salidas el encendido y apagado de los *LEDs* y la luminosidad de estos.
- **16x16 *LEDs*:** matriz de *LEDs* en la que se deberá mostrar el mensaje transmitido a través de la línea serie.

2.2 Objetivos

Los objetivos se dividen en dos tipos:

- Los objetivos de la asignatura
- Los objetivos del TFC.

2.2.1 Objetivos de la asignatura

El objetivo de la asignatura es realizar un TFC en el que se deberán aplicar los conocimientos adquiridos en las asignaturas cursadas en la titulación **Ingeniería Técnica de Telecomunicaciones (Especialidad Telemática)**.
En concreto se deberán validarlas competencias adquiridas en las asignaturas relacionadas con las aplicaciones electromagnéticas y electrónicas.
Los objetivos identificados son:

- Identificar los objetivos específicos de un proyecto.
- Dar soluciones eficaces y adecuadas a los objetivos identificados.
- Realizar los ensayos y/o simulaciones de las soluciones aportadas y saber interpretar los resultados obtenidos.
- Emitir las conclusiones basándonos en los apartados anteriores y saber identificar las debilidades, fortalezas y oportunidades de la solución aportada. Todo esto debe ser el punto de partida para plantear líneas futuras de trabajo.

También será necesario poner en práctica los conocimientos adquiridos en las asignaturas dedicadas a la gestión y organización de proyectos. Así pues, se deberán adquirir las siguientes competencias:

- Saber identificar de manera clara que es lo que se ha de realizar en el proyecto. Detectar las necesidades del cliente e identificarlas.
- Tener capacidad para realizar un pliego de especificaciones real y poder dar una solución lo más ajustada posible tanto en tiempo como en precio.
- Demostrar que se conocen los métodos y procedimientos necesarios para realizar una planificación eficiente.
- Saber detectar los riesgos que están escondidos en la ejecución del proyecto y poder anticiparnos mediante planes de contingencia.
- Redactar una memoria final y un video de presentación en los cuales quede plasmada una síntesis del trabajo realizado. Mediante la lectura o visualización del video una persona ajena al TFC debe entender de manera clara el trabajo realizado y la solución aportada.

2.2.2 Objetivos del TFC

Los objetivos del TFC son medibles en función de la solución que se debe desarrollar. En el caso actual se pide desarrollar un **"Panel de *LEDs*".** Por lo tanto debemos de alcanzar las siguientes competencias:

- Buscar los componentes adecuados a las soluciones aportadas y familiarizarnos con las hojas técnicas de los mismos.
- Diseñar un circuito para el control de *LEDs* utilizando las menores señales posibles y la menor energía.
- Diseñar un circuito capaz de leer la luz ambiente.
- Conocer e implementar una solución software basada en microprocesador para el problema planteado.

- Calcular y diseñar una fuente de alimentación ajustada a las necesidades de la solución propuesta.
- Conocer e implementar el protocolo Modbus y adaptar las señales que provienen del RS232.
- Aprender a manejar software para el dibujo de esquemáticos y el ruteado de tarjetas electrónicas. La última fase del TFC será la de presentar un producto terminado listo para fabricar.

2.3 Calendario de trabajo

El calendario de trabajo que se plantea es en función de las horas libres que se dispone a lo largo de la semana debido a ocupaciones como la familia y el trabajo.

Tabla 1 - Disponibilidad de horas semanales

Día de la semana	Hora de inicio	Hora de fin	Número de horas
Lunes a Jueves	17:30h	20:30h	12
Viernes	16:00h	20:00h	4
Sábado	17:00h	20:00h	3
Domingo	17:00h	20:00h	3
Total horas semanales			22

A continuación se detallan los días en los que no se trabajara en el TFC.

Tabla 2 - Calendario

julio

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			

agosto

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

septiembre

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

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		

noviembre

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

diciembre

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**	**1**	2	3	4
5	**6**	**7**				

Leyenda:	Inicio del TFC	Días hábiles	Días no hábiles	Fin del TFC

Por lo tanto, la disponibilidad de horas totales para llevar a cabo la ejecución del proyecto es de **333 horas.**

2.4 Hitos

Los hitos del TFC quedaran directamente relacionados con las entregas planificadas.

Tabla 3 - Hitos

Hito	Fecha limite
Borrador plan de trabajo	27/09/2014
Corrección plan de trabajo	29/09/2014
Entrega plan de trabajo	30/09/2014
Borrador PEC2	20/10/2014
Corrección PEC2	21/10/2014
Entrega PEC2	22/10/2014
Borrador PEC3	02/12/2014
Corrección PEC3	03/12/2014
Entrega PEC3	04/12/2014
Borrador memoria TFC	14/12/2014
Corrección memoria TFC	15/12/2014
Entrega memoria TFC	07/01/2015

2.5 Planificación

A continuación se muestra la planificación del TFC desglosando la duración estimada de cada una de ellas.

Tabla 4 - Planificación de las tareas

Id	Tarea	Predecesoras	Duración	Comienzo	Fin
1	**Definición del TFC**				
1.1	Inicio de semestre		3 horas	17/09/14	17/09/14
1.2	Enunciado del TFC	1.1	2 horas	18/09/14	18/09/14
1.3	Videoconferencia	1.2	2 horas	20/09/14	20/09/14
2	**Plan de trabajo**				
2.1	Obtener documentación	1.3	1 hora	20/09/14	21/09/14
2.2	Leer documentación	2.1	3 horas	21/09/14	22/09/14
2.3	Buscar bibliografía	2.2	4 horas	22/09/14	23/09/14
2.4	Redactar índice del borrador	2.3	1 hora	23/09/14	23/09/14
2.5	Redactar borrador	2.4	6 horas	23/09/14	25/09/14
2.6	Planificar tareas	2.5	6 horas	25/09/14	27/09/14
2.7	Finalización del borrador	2.6	2 horas	27/09/14	28/09/14
2.8	Corrección del borrador	2.7	4 horas	28/09/14	29/09/14
2.9	Entregar el Plan de Trabajo	2.8	0,5 horas	30/09/14	30/09/14
3	**PEC2 (Diseño de hardware)**				
3.1	**Diseño de los drives de la matriz de *LEDs***				
3.1.1	Elección de los *LEDs*	2.9	3 horas	30/09/14	01/10/14
3.1.2	Estudio de la solución	3.1.1	3 horas	01/10/14	02/10/14
3.1.3	Implementación de la solución	3.1.2	4 horas	02/10/14	03/10/14
3.1.4	Memoria	3.1.3	2 horas	03/10/14	03/10/14
3.2	**Diseño de la matriz de *LEDs***				
3.2.1	Análisis de sistemas reales	3.1.4	2 horas	03/10/14	04/10/14
3.2.2	Estudio de la solución	3.2.1	2 horas	04/10/14	05/10/14
3.2.3	Implementación de la solución	3.2.2	2 horas	05/10/14	05/10/14
3.2.4	Memoria	3.2.3	2 horas	05/10/14	06/10/14

3.3	**Diseño del controlador de luz ambiente**				
3.3.1	Análisis de soluciones comerciales	3.2.4	1 hora	06/10/14	06/10/14
3.3.2	Análisis de amplificadores	3.3.1	5 horas	06/10/14	08/10/14
3.3.3	Implementación de la solución	3.3.2	6 horas	08/10/14	10/10/14
3.3.4	Memoria	3.3.3	2 horas	10/10/14	10/10/14
3.4	**Diseño del adaptador RS232**				
3.4.1	Estudio de mercado	3.3.4	2 horas	10/10/14	12/10/14
3.4.2	Implementación de la solución	3.4.1	1 hora	12/10/14	12/10/14
3.4.3	Memoria	3.4.2	1 hora	12/10/14	13/10/14
3.5	**Selección e implementación del microcontrolador**				
3.5.1	Estudio de requisitos	3.4.3	1 hora	13/10/14	13/10/14
3.5.2	Estudio de mercado	3.5.1	1 hora	13/10/14	13/10/14
3.5.3	Implementación de la solución	3.5.2	2 horas	13/10/14	14/10/14
3.5.4	Memoria	3.5.3	4 horas	14/10/14	15/10/14
3.6	**Diseño de la fuente de alimentación**				
3.6.1	Estudio de requisitos	3.5.4	2 horas	15/10/14	16/10/14
3.6.2	Estudio de mercado	3.6.1	3 horas	16/10/14	17/10/14
3.6.3	Implementación de la solución	3.6.2	6 horas	17/10/14	19/10/14
3.6.4	Memoria	3.6.3	3 horas	19/10/14	20/10/14
3.7	**Redacción y entrega PEC2**				
3.7.1	Completar la memoria de la PEC2.	3.6.4	4 horas	20/10/14	21/10/14
3.7.2	Enviar PEC2 al consultor para revisión.	3.7.1	0 horas	21/10/14	21/10/14
3.7.3	Revisión	3.7.2	2 horas	21/10/14	22/10/14
3.7.4	Correcciones propuestas	3.7.3	2 horas	22/10/14	22/10/14
3.7.5	Entregar la PEC2.	3.7.4	0,5 horas	22/10/14	22/10/14
4	**PEC3 (Diseño de firmware)**				
4.1	**Selección y aprendizaje de compilador**				
4.1.1	Selección de herramientas	3.7.5	2 horas	23/10/14	23/10/14
4.1.2	Aprendizaje del compilador	4.1.1	20 horas	23/10/14	29/10/14
4.2	**Desarrollo de las inicializaciones del microcontrolador**				
4.2.1	Estudio registros de control	4.1.1	5 horas	23/10/14	24/10/14
4.2.2	Detección de registros necesarios	4.2.1	1 hora	25/10/14	25/10/14
4.2.3	Implementación de la solución.	4.2.2	2 horas	25/10/14	25/10/14
4.2.4	Simulación	4.2.3	2 horas	26/10/14	26/10/14
4.2.5	Memoria	4.2.4	2 horas	26/10/14	27/10/14
4.3	**Desarrollo de las interrupciones del microcontrolador**				
4.3.1	Estudio registros de interrupción	4.2.5	2 horas	27/10/14	27/10/14
4.3.2	Detección de interrupciones necesarias	4.3.1	1 hora	28/10/14	28/10/14
4.3.3	Implementación de la solución.	4.3.2	2 horas	28/10/14	28/10/14
4.3.4	Simulación	4.3.3	2 horas	29/10/14	29/10/14
4.3.5	Memoria	4.3.4	1 hora	29/10/14	29/10/14
4.4	**Diseño del control de la matriz de *LEDs***				
4.4.1	Estudio de periféricos	4.3.5	2 horas	30/10/14	30/10/14
4.4.2	Flujo de programa.	4.4.1	2 horas	30/10/14	31/10/14
4.4.3	Implementación de la solución.	4.4.2	6 horas	31/10/14	02/11/14
4.4.4	Simulación.	4.4.3	2 horas	03/11/14	03/11/14
4.4.5	Memoria	4.4.4	3 horas	03/11/14	04/11/14
4.5	**Diseño del control de la luz ambiente**				

4.5.1	Estudio del conversor A/D del microcontrolador.	4.4.5	2 horas	04/11/14	05/11/14
4.5.2	Flujo de programa.	4.5.1	2 horas	05/11/14	05/11/14
4.5.3	Implementación de la solución.	4.5.2	4 horas	06/11/14	07/11/14
4.5.4	Simulación	4.5.3	6 horas	07/11/14	08/11/14
4.5.5	Análisis de resultados	4.5.4	6 horas	09/11/14	10/11/14
4.5.6	Implementación de la tabla o función matemática.	4.5.5	3 horas	11/11/14	11/11/14
4.5.7	Memoria	4.5.6	3 horas	12/11/14	12/11/14
4.6	**Creación del mapa de caracteres**				
4.6.1	Búsqueda	4.5.7	4 horas	13/11/14	14/11/14
4.6.2	Desarrollo o adaptación	4.6.1	2 horas	14/11/14	14/11/14
4.6.3	Memoria	4.6.2	2 horas	14/11/14	15/11/14
4.7	**Diseño del protocolo Modbus**				
4.7.1	Búsqueda	4.6.3	2 horas	15/11/14	15/11/14
4.7.2	Estudio del protocolo	4.7.1	2 horas	16/11/14	16/11/14
4.7.3	Estudio de necesidades	4.7.2	6 horas	16/11/14	18/11/14
4.7.4	Diseño del mapa de memoria	4.7.3	2 horas	18/11/14	19/11/14
4.7.5	Implementación del mapa de memoria.	4.7.4	2 horas	19/11/14	19/11/14
4.7.6	Estudio de la UART del microcontrolador	4.7.5	2 horas	20/11/14	20/11/14
4.7.7	Implementación del algoritmo de comunicaciones.	4.7.6	8 horas	20/11/14	22/11/14
4.7.8	Simulación	4.7.7	4 horas	23/11/14	24/11/14
4.7.9	Memoria	4.7.8	4 horas	24/11/14	25/11/14
4.8	**Diseño del programa principal**				
4.8.1	Flujo del programa principal.	4.7.9	3 horas	25/11/14	26/11/14
4.8.2	Implementación de la solución final.	4.8.1	6 horas	26/11/14	28/11/14
4.8.3	Simulación	4.8.2	6 horas	28/11/14	30/11/14
4.8.4	Memoria	4.8.3	4 horas	30/11/14	01/12/14
4.9	**Redacción y entrega PEC3**				
4.9.1	Completar la memoria de la PEC3.	4.8.4	4 horas	01/12/14	02/12/14
4.9.2	Enviar PEC3 al consultor para revisión.	4.9.1	0 horas	02/12/14	02/12/14
4.9.3	Revisión	4.9.2	2 horas	03/12/14	03/12/14
4.9.4	Correcciones propuestas	4.9.3	2 horas	03/12/14	04/12/14
4.9.5	Entregar la PEC3.	4.9.4	0,5 horas	04/12/14	04/12/14
5	**Memoria**				
5.1	**Industrialización del producto**				
5.1.1	Dibujar el esquema final	4.9.5	3 horas	04/12/14	05/12/14
5.1.2	Realizar el ruteado de la o las PCB	5.1.1	9 horas	05/12/14	08/12/14
5.1.3	Generar el fichero *Gerber*	5.1.2	2 horas	08/12/14	08/12/14
5.1.4	Listado de componentes	5.1.3	1 hora	08/12/14	09/12/14
5.1.5	Calculo del coste final de la solución	5.1.4	1 hora	09/12/14	09/12/14
5.2	**Memoria técnica del TFC**				
5.2.1	Recopilación de la información generada	5.1.5	2 horas	09/12/14	10/12/14
5.2.2	Integración de contenidos	5.2.1	2 horas	10/12/14	10/12/14
5.2.3	Redacción de conclusiones	5.2.2	3 horas	10/12/14	12/12/14
5.2.4	Redacción del borrador	5.2.3	6 horas	12/12/14	14/12/14
5.2.5	Envío al consultor para revisión	5.2.4	4 horas	14/12/14	15/12/14
5.2.6	Correcciones de los errores detectados por el consultor	5.2.5	4 horas	15/12/14	17/12/14

5.2.7	Revisión ortográfica, varias lecturas, etc…	5.2.6	4 horas	17/12/14	18/12/14
5.2.8	Entrega de la memoria	5.2.7	0,5 horas	07/01/15	07/01/15
5.3	**Video presentación del TFC**				
5.3.1	Confección de la presentación en PowerPoint	5.2.7	5 horas	18/12/14	19/12/14
5.3.2	Grabación de la locución de la exposición	5.3.1	5 horas	19/12/14	21/12/14
5.3.3	Maquetado, edición y montaje del video	5.3.2	3 horas	21/12/14	22/12/14
5.3.4	Entrega del video presentación	5.2.8	0,5 horas	07/01/15	07/01/15
5.4	**Ficheros, fuentes y anexos**				
5.4.1	Recopilación de ficheros generados (src, pcb, simulación, etc…)	5.3.3	2 horas	22/12/14	23/12/14
5.4.2	Compresión de los ficheros	5.4.1	0,5 horas	23/12/14	23/12/14
5.4.3	Entrega del fichero comprimido	5.3.4	0,5 horas	07/01/15	07/01/15
6	**Debate virtual**				
6.1	Recopilación de las preguntas solicitadas desde el tribunal		2 horas	28/01/15	28/01/15
6.2	Redacción y contestación a las cuestiones solicitadas	6.1	3 horas	28/01/15	29/01/15
6.3	Enviar documento con respuestas	6.2	0 horas	29/01/15	29/01/15
7	**Fin de TFC**			**29/01/15**	
Total horas planificadas			**316,5 horas**		

2.6 Diagramas de Gantt

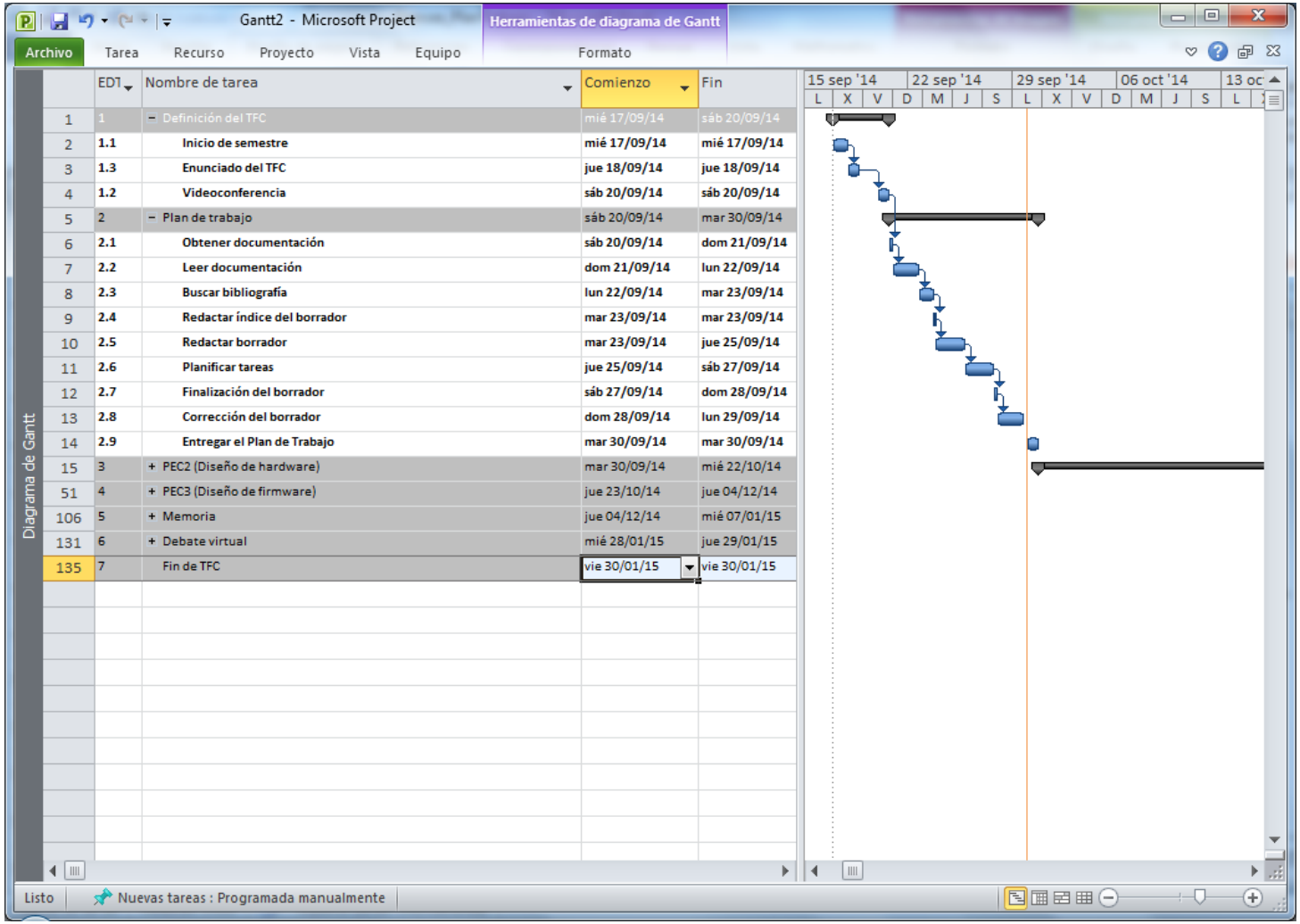


Ilustración 2 - Gantt plan de trabajo

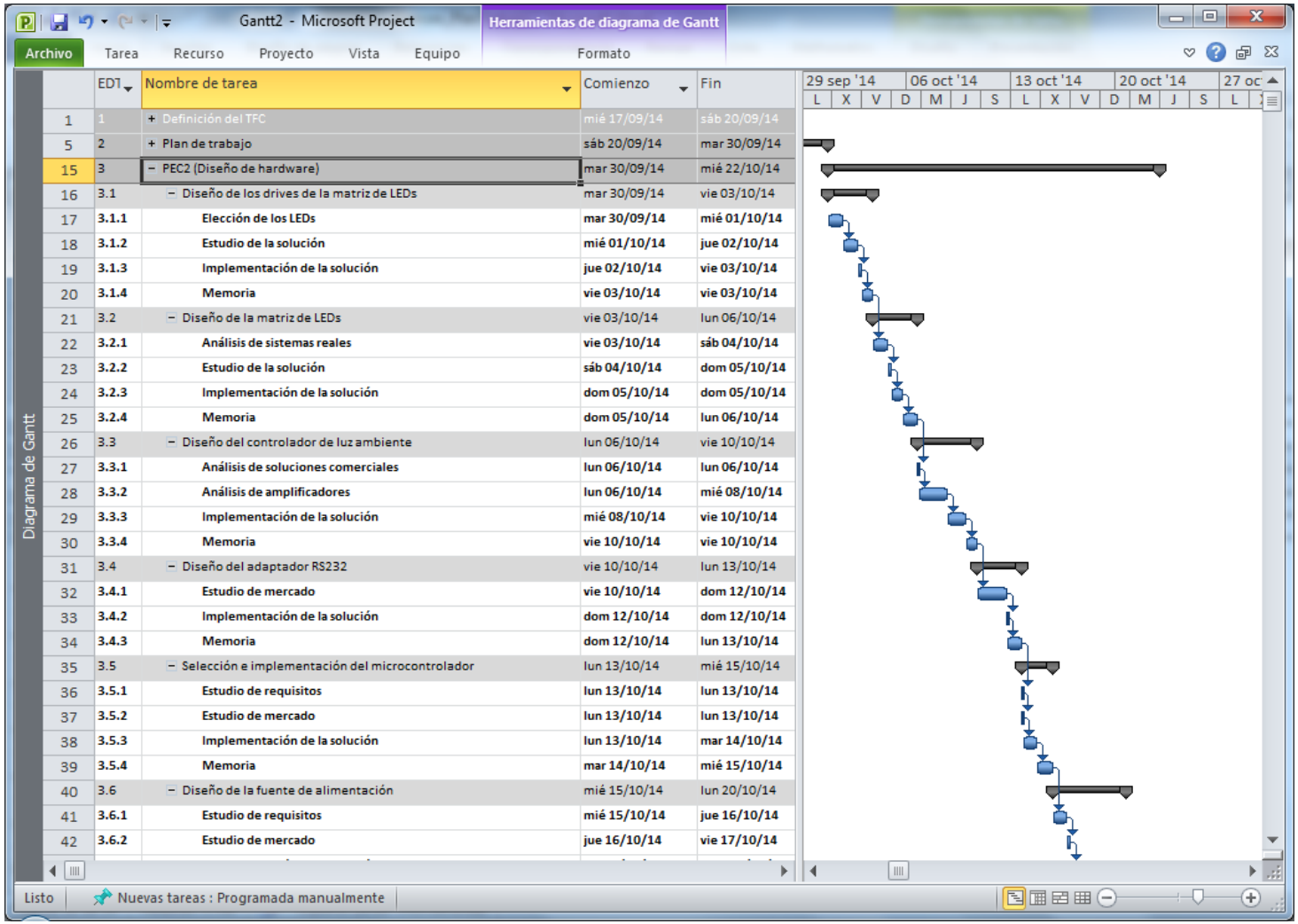
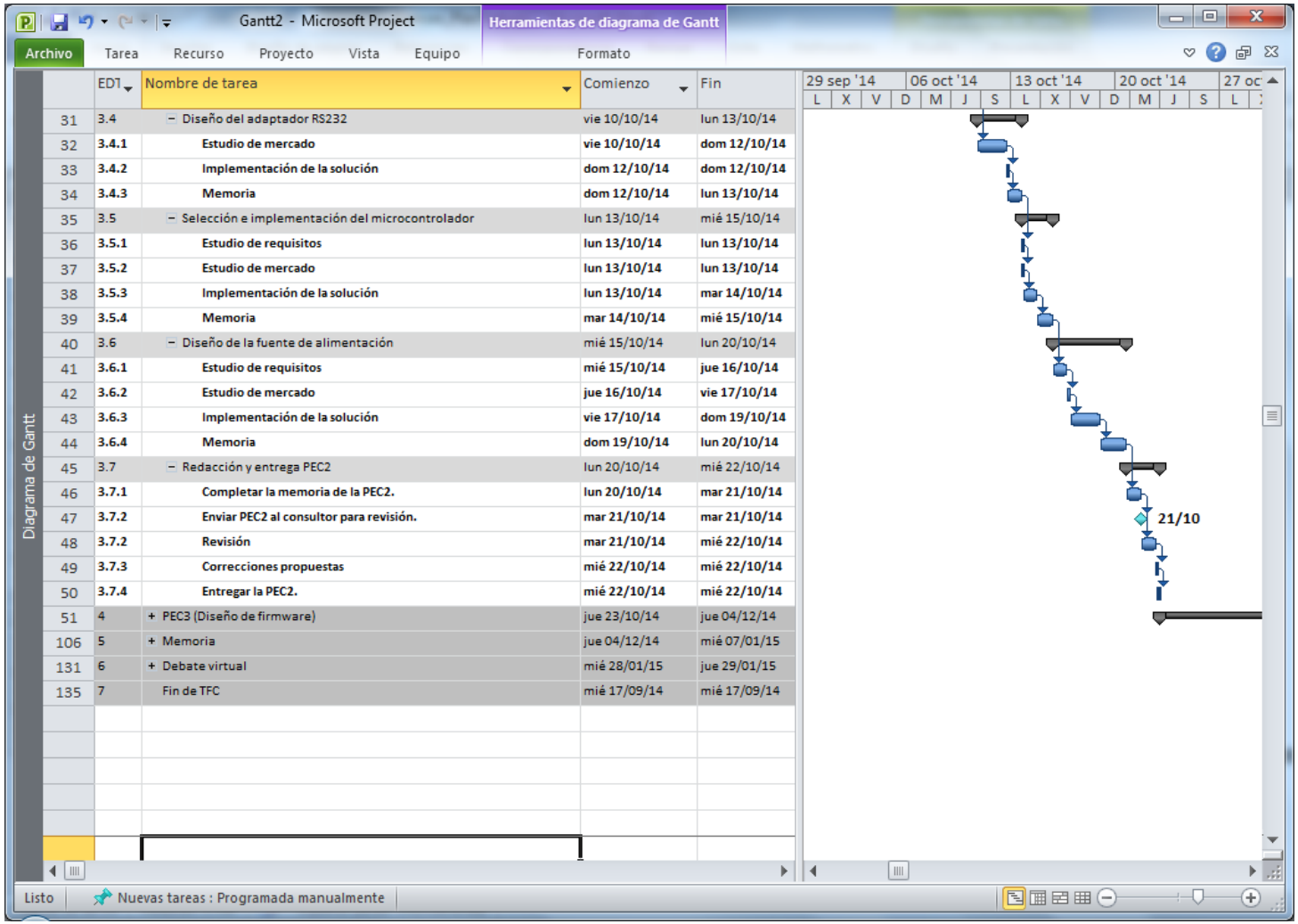


Ilustración 3 - Gantt PEC2 parte 1



	EDT	Nombre de tarea	Comienzo	Fin
31	3.4	− Diseño del adaptador RS232	vie 10/10/14	lun 13/10/14
32	3.4.1	Estudio de mercado	vie 10/10/14	dom 12/10/14
33	3.4.2	Implementación de la solución	dom 12/10/14	dom 12/10/14
34	3.4.3	Memoria	dom 12/10/14	lun 13/10/14
35	3.5	− Selección e implementación del microcontrolador	lun 13/10/14	mié 15/10/14
36	3.5.1	Estudio de requisitos	lun 13/10/14	lun 13/10/14
37	3.5.2	Estudio de mercado	lun 13/10/14	lun 13/10/14
38	3.5.3	Implementación de la solución	lun 13/10/14	mar 14/10/14
39	3.5.4	Memoria	mar 14/10/14	mié 15/10/14
40	3.6	− Diseño de la fuente de alimentación	mié 15/10/14	lun 20/10/14
41	3.6.1	Estudio de requisitos	mié 15/10/14	jue 16/10/14
42	3.6.2	Estudio de mercado	jue 16/10/14	vie 17/10/14
43	3.6.3	Implementación de la solución	vie 17/10/14	dom 19/10/14
44	3.6.4	Memoria	dom 19/10/14	lun 20/10/14
45	3.7	− Redacción y entrega PEC2	lun 20/10/14	mié 22/10/14
46	3.7.1	Completar la memoria de la PEC2.	lun 20/10/14	mar 21/10/14
47	3.7.2	Enviar PEC2 al consultor para revisión.	mar 21/10/14	mar 21/10/14
48	3.7.2	Revisión	mar 21/10/14	mié 22/10/14
49	3.7.3	Correcciones propuestas	mié 22/10/14	mié 22/10/14
50	3.7.4	Entregar la PEC2.	mié 22/10/14	mié 22/10/14
51	4	+ PEC3 (Diseño de firmware)	jue 23/10/14	jue 04/12/14
106	5	+ Memoria	jue 04/12/14	mié 07/01/15
131	6	+ Debate virtual	mié 28/01/15	jue 29/01/15
135	7	Fin de TFC	mié 17/09/14	mié 17/09/14

Ilustración 4 - Gantt PEC2 parte 2

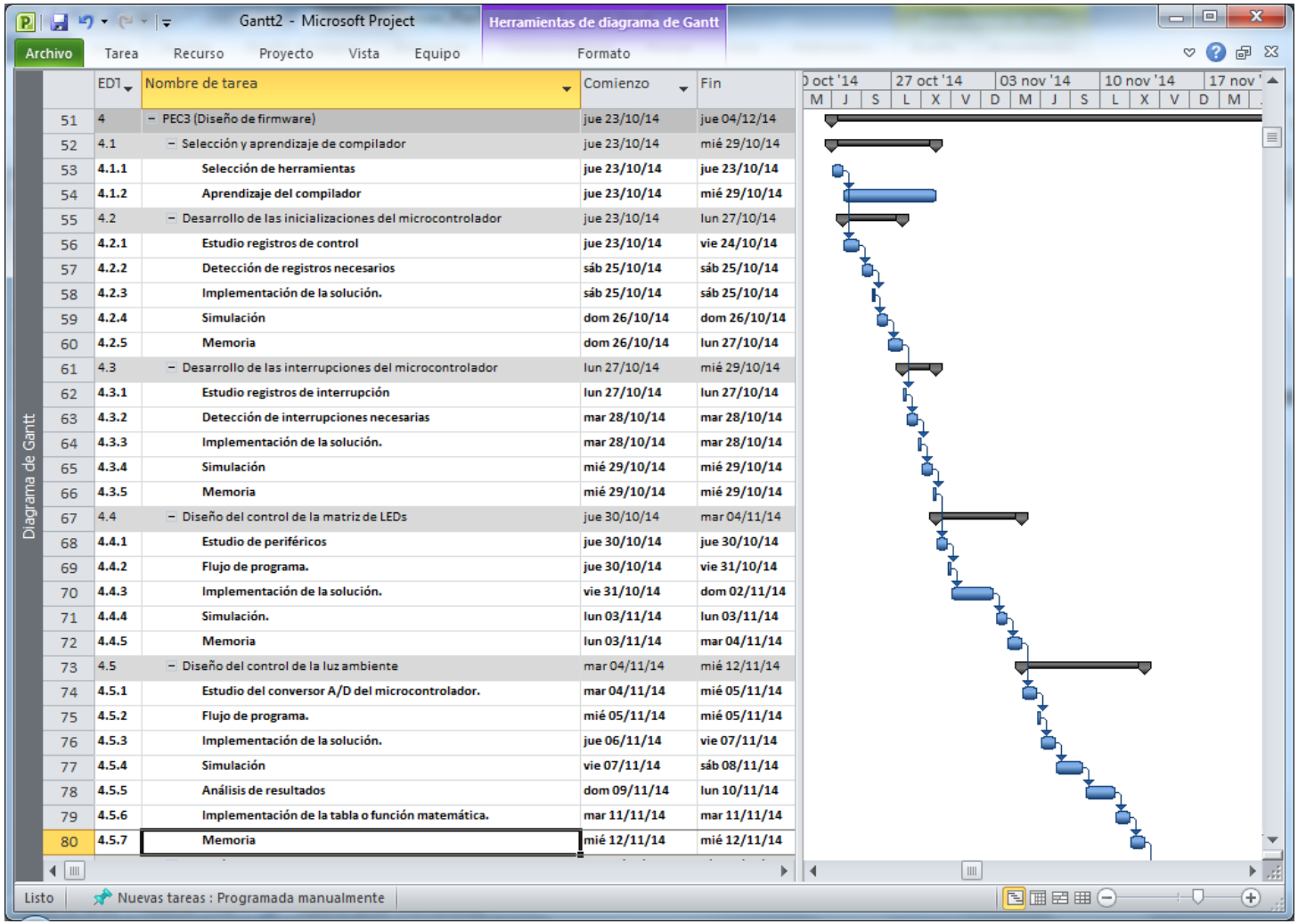


Ilustración 5 - Gantt PEC3 parte 1

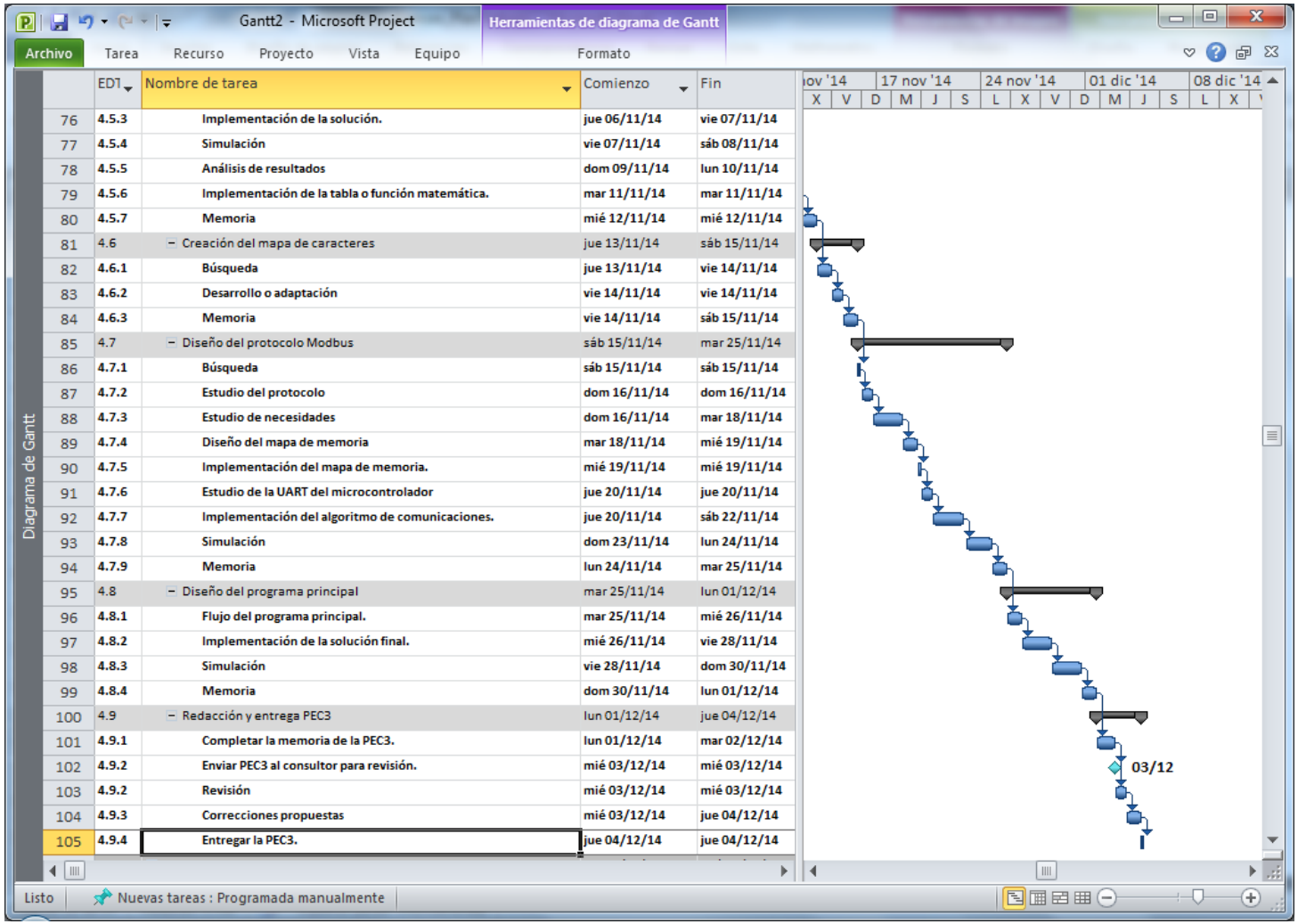
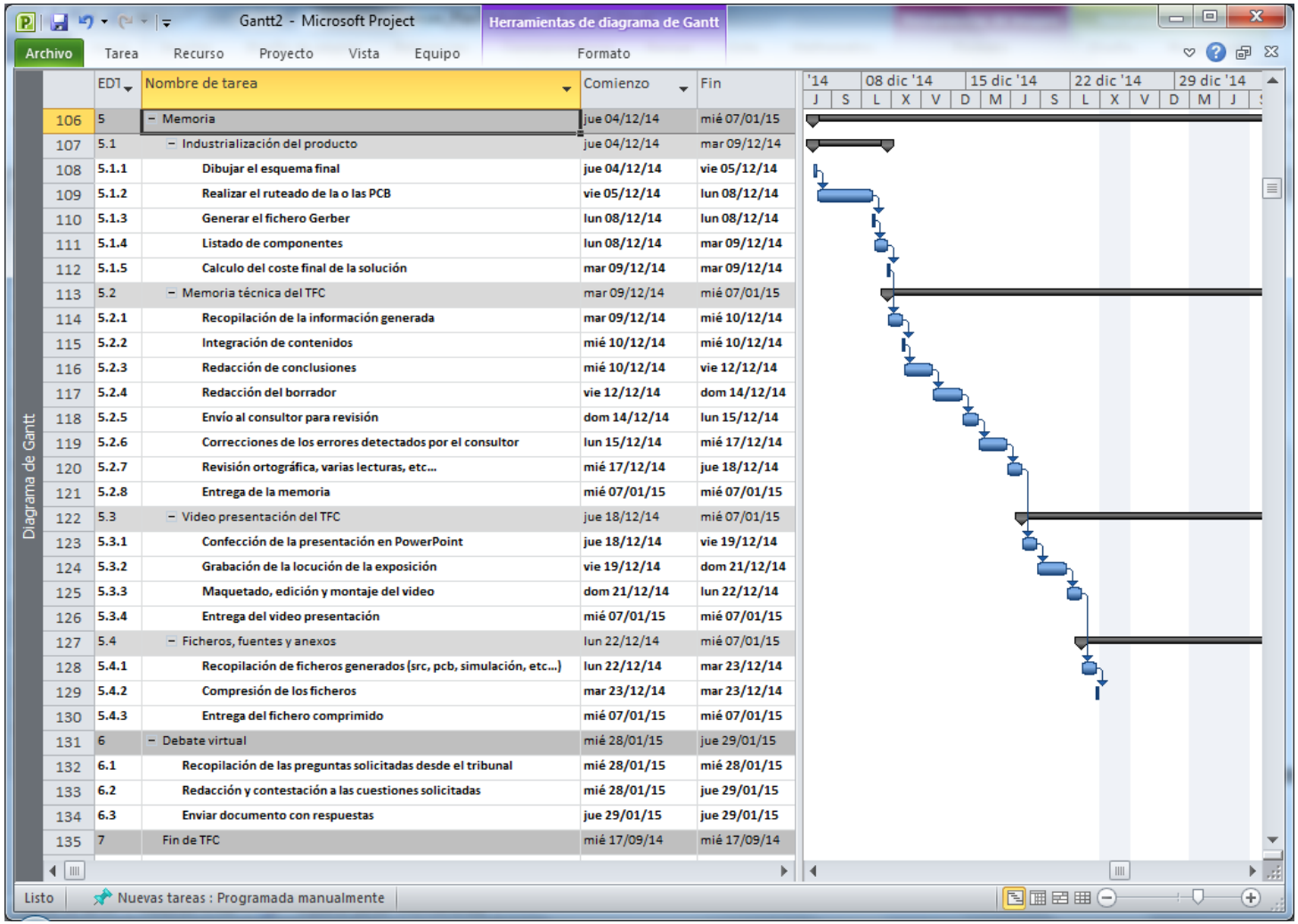


Ilustración 6 - Gantt PEC3 parte 2



	EDT	Nombre de tarea	Comienzo	Fin
106	5	Memoria	jue 04/12/14	mié 07/01/15
107	5.1	Industrialización del producto	jue 04/12/14	mar 09/12/14
108	5.1.1	Dibujar el esquema final	jue 04/12/14	vie 05/12/14
109	5.1.2	Realizar el ruteado de la o las PCB	vie 05/12/14	lun 08/12/14
110	5.1.3	Generar el fichero Gerber	lun 08/12/14	lun 08/12/14
111	5.1.4	Listado de componentes	lun 08/12/14	mar 09/12/14
112	5.1.5	Calculo del coste final de la solución	mar 09/12/14	mar 09/12/14
113	5.2	Memoria técnica del TFC	mar 09/12/14	mié 07/01/15
114	5.2.1	Recopilación de la información generada	mar 09/12/14	mié 10/12/14
115	5.2.2	Integración de contenidos	mié 10/12/14	mié 10/12/14
116	5.2.3	Redacción de conclusiones	mié 10/12/14	vie 12/12/14
117	5.2.4	Redacción del borrador	vie 12/12/14	dom 14/12/14
118	5.2.5	Envío al consultor para revisión	dom 14/12/14	lun 15/12/14
119	5.2.6	Correcciones de los errores detectados por el consultor	lun 15/12/14	mié 17/12/14
120	5.2.7	Revisión ortográfica, varias lecturas, etc...	mié 17/12/14	jue 18/12/14
121	5.2.8	Entrega de la memoria	mié 07/01/15	mié 07/01/15
122	5.3	Video presentación del TFC	jue 18/12/14	mié 07/01/15
123	5.3.1	Confección de la presentación en PowerPoint	jue 18/12/14	vie 19/12/14
124	5.3.2	Grabación de la locución de la exposición	vie 19/12/14	dom 21/12/14
125	5.3.3	Maquetado, edición y montaje del video	dom 21/12/14	lun 22/12/14
126	5.3.4	Entrega del video presentación	mié 07/01/15	mié 07/01/15
127	5.4	Ficheros, fuentes y anexos	lun 22/12/14	mié 07/01/15
128	5.4.1	Recopilación de ficheros generados (src, pcb, simulación, etc...)	lun 22/12/14	mar 23/12/14
129	5.4.2	Compresión de los ficheros	mar 23/12/14	mar 23/12/14
130	5.4.3	Entrega del fichero comprimido	mié 07/01/15	mié 07/01/15
131	6	Debate virtual	mié 28/01/15	jue 29/01/15
132	6.1	Recopilación de las preguntas solicitadas desde el tribunal	mié 28/01/15	mié 28/01/15
133	6.2	Redacción y contestación a las cuestiones solicitadas	mié 28/01/15	jue 29/01/15
134	6.3	Enviar documento con respuestas	jue 29/01/15	jue 29/01/15
135	7	Fin de TFC	mié 17/09/14	mié 17/09/14

Ilustración 7 - Gantt memoria

3 Diseño de hardware

Como punto de partida del diseño del hardware, se quiere aclarar que el propósito es cumplir las especificaciones marcadas en el enunciado del TFC y además realizar una solución que tenga el menor coste posible.

3.1 Driver de la matriz de *LEDs*

Se considera driver de la matriz de *LEDs* a la parte hardware que proporcionara la potencia necesaria a los diodos *LEDs*.

3.1.1 Estudio de la etapa de potencia del *LED*

Tal y como se especifica en el enunciado se debe poder variar la potencia del *LED* en función de la intensidad lumínica que exista en el entorno del panel. Para conseguirlo es necesario modificar la corriente que pasa a través del *LED*. Existen varias técnicas para conseguirlo pero todas se basan en dos métodos: regular la tensión de alimentación o regular la intensidad.

El primer caso queda desestimado por la variación de la tensión directa del *LED*[1]. Dentro de un mismo fabricante se indica que este parámetro es muy variable en la fabricación, por lo que si se utiliza el método de variación de tensión no se conseguirá que todos los *LEDs* tengan la misma corriente y por consiguiente luzcan con la misma intensidad.

La segunda técnica nombrada se basa en regular la intensidad del *LED*. Se consigue mediante una fuente de corriente y la variación de la tensión de la base del transistor. Se ha realizado una simulación con *PsPice* para mostrar el resultado. La simulación es un *DC-Sweep*[2] con una variación de V2 desde 0V hasta 5V.

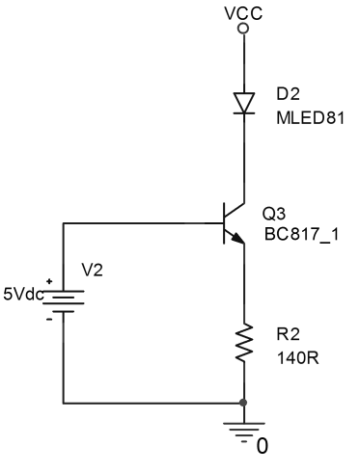


Ilustración 8 - Fuente de corriente

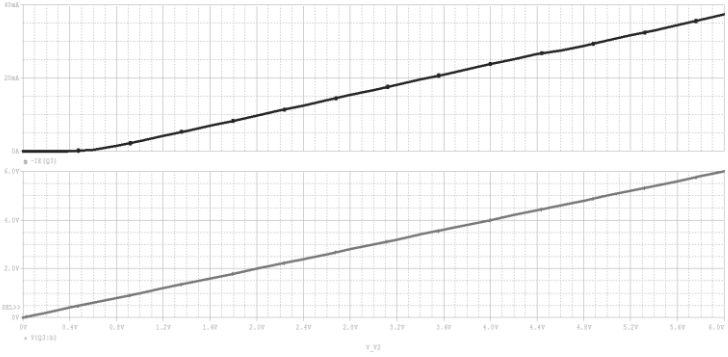


Ilustración 9 - Tensión de la base vs Intensidad *LED*

Para conseguir la variación de la tensión en la base del transistor se podrá utilizar una señal PWM (*Pulse Width Modulation)* con un filtro paso bajo basado en RC. Ahora se utiliza una fuente de tensión continua.

[1] Ver nota de aplicación de Osram, pagina 4: "*Behaviour of InGaN LEDs in Parallel Circuits*"
[2] *DC-Sweep* es un tipo de análisis en *PsPice.* Se trata de un barrido de tensión o corriente.

3.1.2 Selección del *LED*

La selección del *LED* depende en gran parte de la aplicación final.

Se ha especificado un panel de *LEDs* de 16x16 unidades y el tamaño será de aproximadamente 8x10 cm. Con estas medidas se da por sentado que el panel podrá ser observado desde una distancia corta. Por lo tanto, la intensidad lumínica que emitan los *LEDs* no deberá ser muy elevada.

Otro parámetro importante es que tiene que ser una solución energéticamente eficiente. Para lograrlo hay que seleccionar un *LED* de alto rendimiento y que tenga un consumo moderado o bajo.

Por último, hay que tener en cuenta que se dispone de un panel con 256 unidades y que para ser energéticamente eficientes no se pueden conectar todos los *LEDs* al mismo tiempo. De hecho, la solución óptima pasa por encender los *LEDs* de uno en uno. Teniendo en cuenta que la frecuencia mínima a la que el ojo humano es capaz de detectar variaciones son 50Hz o 20ms, y que durante ese tiempo se han de encender todos los *LEDs* en el peor de los casos, hay que buscar un *LED* que pueda aplicar el máximo pulso de corriente durante una pequeña fracción de tiempo para conseguir la máxima luminosidad del panel. Esa fracción de tiempo será el *PWM* permitido por el fabricante del *LED* a la máxima potencia.

El *LED* seleccionado es el modelo **LS T67K de Osram** en formato *PLCC2*. El fabricante lo recomienda en su *Datasheet*[3] para aplicaciones de señalizado y paneles informativos. La corriente directa nominal es de 20mA y la máxima de 100mA. La tensión directa nominal es de 1,8V y la máxima de 2,2V. Las deviaciones de producción no afectaran en nuestro diseño.

En las siguientes ecuaciones se observa que cumple con nuestros requisitos de máxima intensidad y tiempos de refresco.

$$D_{esp} = \frac{1}{256} \cong 0.004$$

$$t_{p\,esp} = \frac{20ms}{256} = \cong 70\mu s$$

$$D_{esp} < D$$

$$t_{p\,esp} < t_p$$

$$Max\ I_F Permitida = \mathbf{0.10A}$$

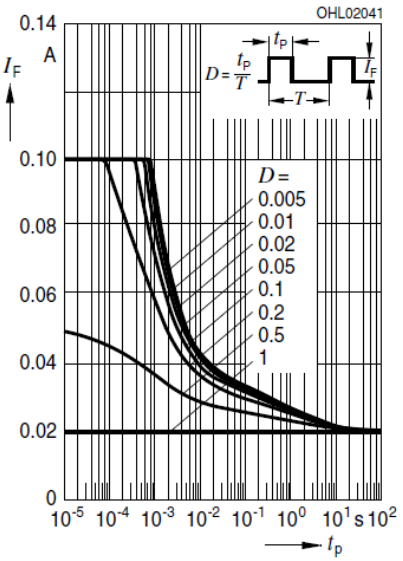


Ilustración 10 - Corriente directa vs T máx. ON

Queda por reseñar que es un *LED* estándar en el mercado y que varios fabricantes disponen de dispositivos con las mismas características como *Everlight* o *Wurth*.

[3] Ver hoja de especificaciones del fabricante "*LS T67K – TOPLED*"

3.1.3 Solución del driver *LED*

Con la solución propuesta en el apartado 3.1.1 se encuentra el primer escoyo. Hay que inyectar una corriente por la base del transistor de 100mA para conseguir el efecto deseado. El problema no es aprovisionar de esta corriente, sino que desde el pin del microprocesador, que es por donde saldrá el *PWM,* no podemos proveer esa corriente. Para conseguirlo se montara el transistor en la configuración emisor común. Mediante esta configuración se conseguirá generar una alta corriente de colector inyectando una muy baja en la base del transistor.

El transistor que se montara en modo emisor común será el **BC817**[4] del fabricante *NXP.* Su formato de montaje es *SOT23* en *SMD*. Este transistor tiene un corriente de colector de 500mA y una corriente de pico máxima en la base de 200mA. Con estos parámetros cumple los requisitos necesarios sin estar sobredimensionado.

Para que la configuración en emisor común sea eficiente y la corriente de base sea pequeña, debemos alimentar el ramal del *LED* con una tensión constante y que tenga un suministro de corriente fiable. Alimentamos con 5V. Después se verá la parte de la fuente de alimentación.

Los cálculos del circuito son los siguientes:

$$V_{R1} = V_{in} - Vf_{D1} - V_{ce\,Q1} = 5V - 2{,}2V - 0.2V = \mathbf{2,6V}$$

$$R1 = \frac{V_{R1}}{I_{Cmax}Q1} = \frac{2{,}6V}{0{,}1A} = 26\Omega \approx \mathbf{25\Omega}^{5}$$

Se ha aplicado una corriente máxima de colector Q1 de 100mA para conseguir máxima corriente necesaria en el *LED*.

Para el cálculo de la resistencia de base se ha obtenido la β[6] del transistor en las condiciones deseadas para aplicar al *LED* la máxima potencia. Tiene un valor de 100 con I_C=100mA V_{BE}=1V. Estos valores se han obtenido de la hoja de especificaciones del transistor.

$$I_{bQ1} = \frac{I_{cQ1}}{\beta} = \frac{0.1A}{100} = 0.001A$$

$$R2 = \frac{V_{maxPWM} - V_{BE}}{I_{bQ1}} = \frac{5V - 1V}{0{,}001A} = 4000\Omega \approx \mathbf{3K9\Omega}^{7}$$

Como se puede apreciar en los cálculos, la corriente de base del transistor necesaria para generar los 100mA necesarios será muy baja y permitirá montar cualquier circuito de control para realizar activaciones del *LED*.

Con todos los valores obtenidos se ha dibujado la solución final del driver *LED* y se ha realizado una simulación mediante un barrido de tensión en *PsPice aplicando* de 0V a 5V en la base del transistor.

[4] Ver hoja de especificaciones del fabricante "*BC817 Datasheet*"
[5] R1 será de 25 ohmios para normalizar al estándar del mercado.
[6] Es la *beta* del transistor. Se trata de la ganancia que existe entre corriente de base y corriente de colector.
[7] R1 será de 3K9 ohmios para normalizar al estándar del mercado.

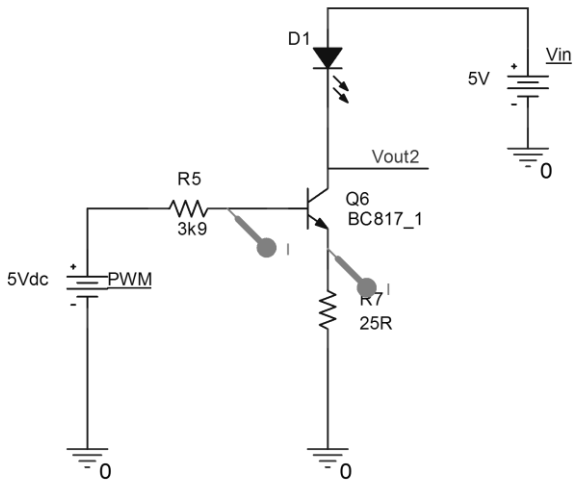


Ilustración 11 - Driver del *LED*

A continuación la simulación realizada en *PsPice:*

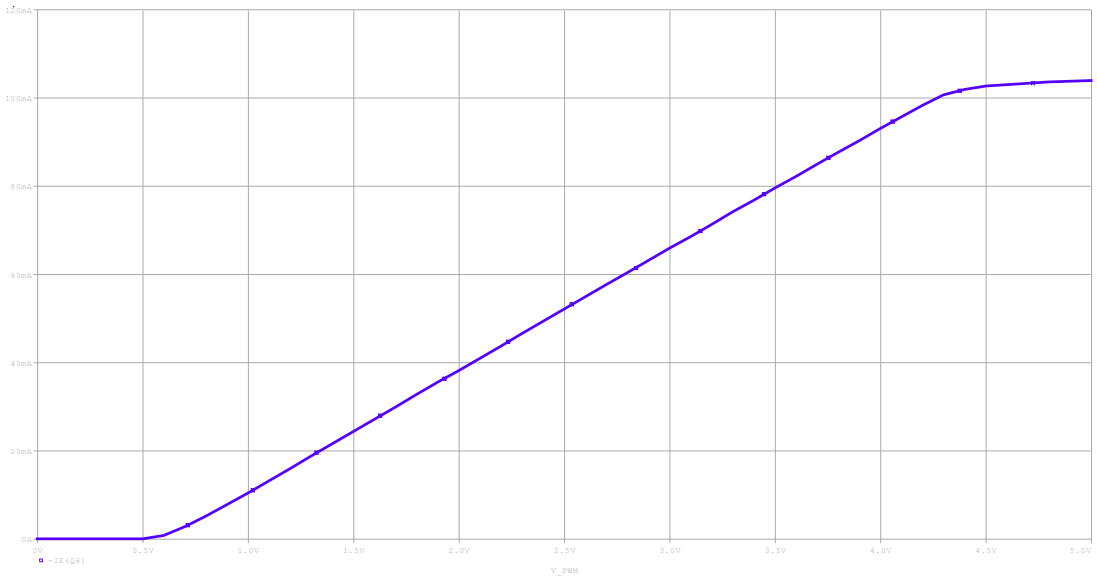


Ilustración 12 – Simulación *PsPice*. Tensión base Q1 vs Ic Q1

Como conclusión de la simulación y diseño del driver tenemos los siguientes valores para el circuito:

Máxima I_c Q1	R1	Q1	R2
100mA	25Ω 1206 1/4W	BC817 SOT23	3k9Ω 1206 1/4W

3.2 Control de la matriz de *LEDs*

Se considera control de la matriz de *LEDs* a la parte hardware que activara o desactivara los *LEDs*.

3.2.1 Estudio de soluciones

En el mercado existen distintas soluciones para manejar una cantidad determinada de *LEDs*. La mayoría de ellas están basadas en *chips*[8] que controlan tanto la activación y desactivación de los *LEDs*, como la potencia que se les debe aplicar. El manejo de estos *chips* se realiza mediante un bus de comunicaciones *I2C* o *SPI.*

Para el TFC, en vez de utilizar estos *chips,* se ha tomado la determinación de diseñar un circuito de características similares a los que existen y que cubra todos los requisitos especificados. El reto que se ha planteado es que para realizar el control del panel solamente se utilicen cuatro señales: *Clock_Counter, Reset_Counter, Data_Counter* y *PWM_Led.*

3.2.2 Desarrollo de la solución

Para lograr el objetivo marcado, el panel de 16x16 *LEDs* quedará dividido en 16 señales de activación de filas y 16 señales de activación de columnas. Cada fila se activara de manera única y cada columna también. De este modo siempre existirá una única fila activada y una única columna activada. Es decir, solamente se podrá activar un *LED* de manera simultánea. Este circuito se desarrollara mediante contadores decimales.

El ciclo de paso por cada fila se realizara mediante una señal de *clock* y cada ciclo será de 70µs. Se procesaran 16 ciclos de *clock* y se habrán recorrido las 16 filas. El ciclo de paso por cada columna será mediante el procesado de las 16 filas. Es decir, cada 16 filas avanzaremos una columna que será el equivalente a 1,120ms. Para controlar las columnas se utilizara un transistor *PNP* y hará las funciones de un interruptor conectando a 5V. Para controlar las filas, como se ha mencionado en el capítulo anterior utilizaremos un transistor *NPN* en configuración emisor común, y además colocaremos un *switch* analógico (**CD4066**[9]) para dejar pasar la señal de *PWM* o no.

Se ha dibujado la solución para un único *LED* y se ha realizado una simulación en *PsPice.* En la simulación se ha introducido un *clock* en el pin *dato* para que cada 20µs se realice una activación y desactivación del valor de *PWM* que se desea escribir. Además se ha introducido otro *clock* por las conexiones de fila y columna simulando una activación del *LED.*

En la simulación se ha observado que el circuito funciona correctamente y que consigue entregar la corriente necesaria para que el *LED* trabaje a su máxima potencia. Además se muestra que los circuitos de activación y desactivación de las filas, columnas y el dato funcionan correctamente y que los tiempos de respuesta de los transistores son adecuados.

A continuación las imágenes del circuito eléctrico con la solución final y la simulación en *PsPice* con los resultados obtenidos.

[8] Ver ejemplo de *chip*: "*Driver Leds SP-DM13H*"
[9] Ver hoja de especificaciones del fabricante "*CD4066*"

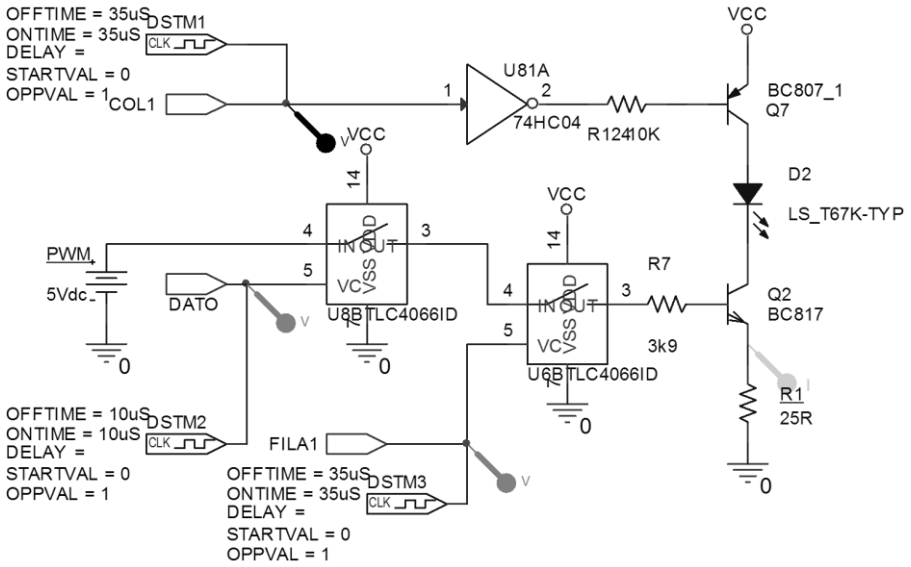


Ilustración 13 - Ejemplo de manejo del driver de un *LED*

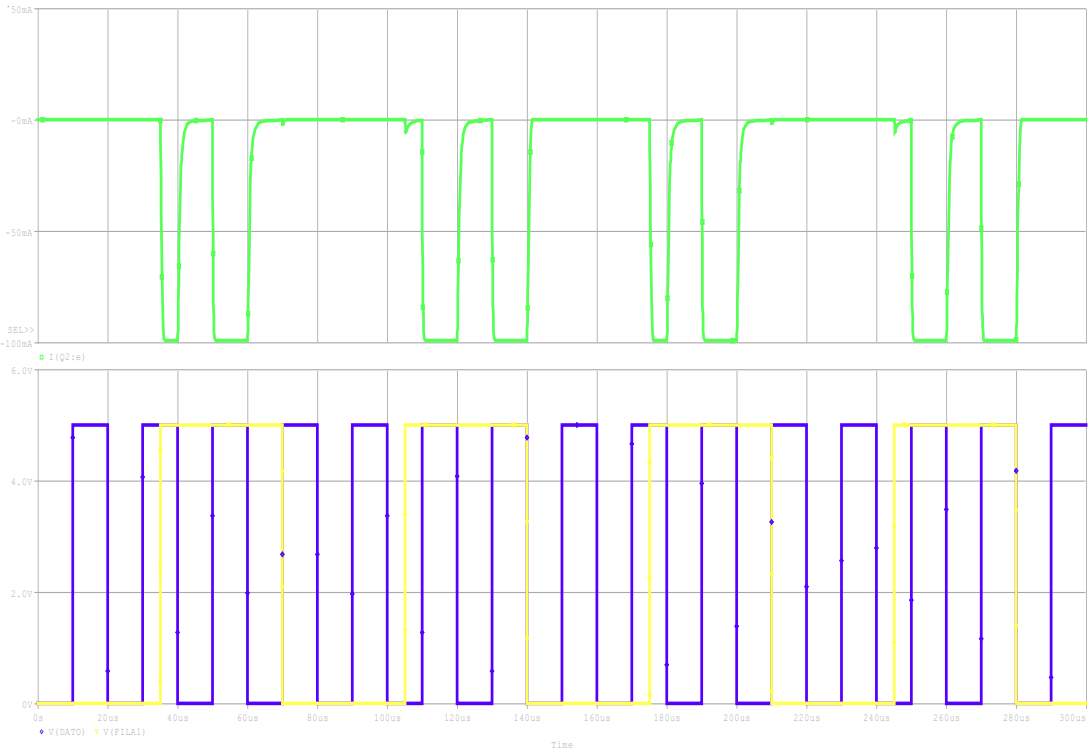


Ilustración 14 - Simulación *PsPice* del driver. Activaciones vs Corriente del *LED*

En la anterior simulación se demuestra que como el sistema es capaz de realizar la regulación de la corriente a la vez que se va activando y desactivando el led. Se han introducido varias señales de *clock* que son asíncronas para que se vaya activando y desactivando en distintos periodos de tiempo el led. En la señal superior se muestra la corriente del led y en las inferiores las señales de activación.

El diseño del circuito de activación de las filas y las columnas se realizara mediante una señal de *Clock_Counter, Reset_Counter* y cuatro circuitos contadores decimales (**CD4017B**[10]). En la figura 15 se puede observar como quedara el diseño:

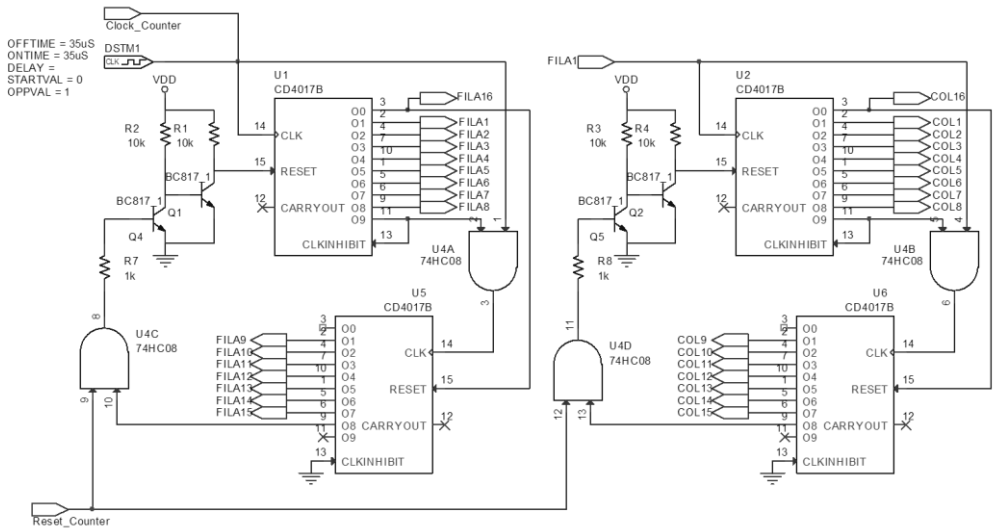


Ilustración 15 – Circuito de contadores para activación filas y columnas

Para la siguiente simulación realizada en *PsPice* se ha colocado un *clock* de aproximadamente 14KHz.

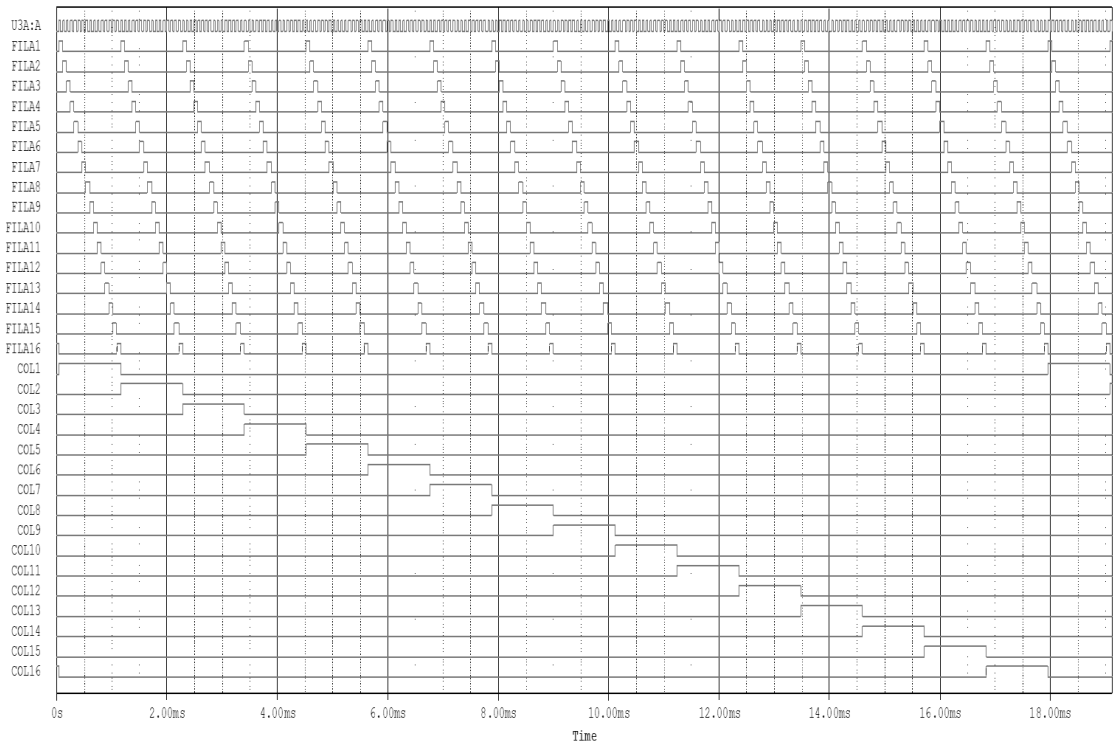


Ilustración 16 – Simulación *PsPice*. Activación de filas y columnas

El circuito de control de la matriz de *LEDs* quedaría como en la siguiente imagen:

[10] Ver hoja de especificaciones del fabricante "*CD4017B*"

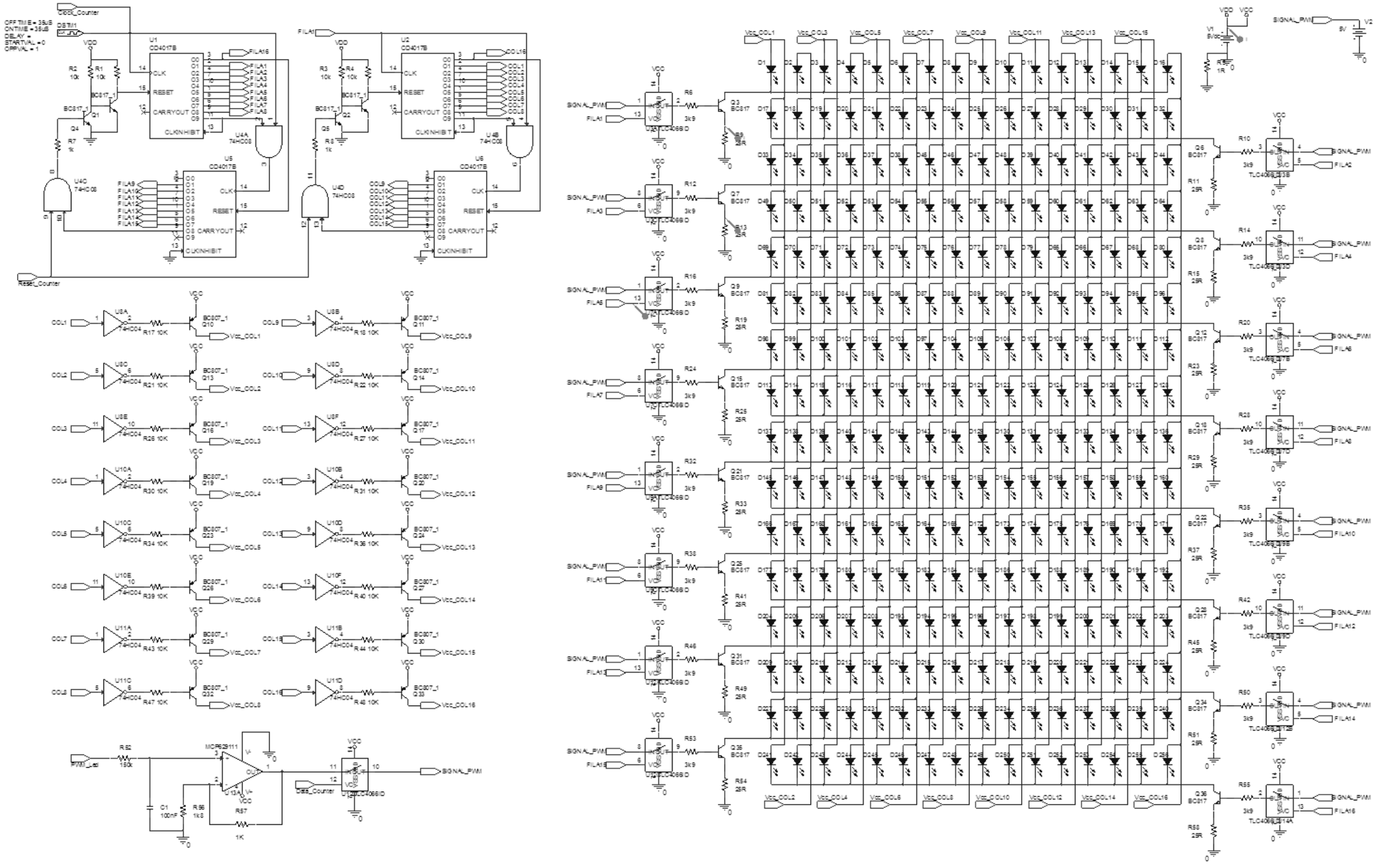


Ilustración 17 - Esquema eléctrico de la matriz y control de 16x16 *LEDs*

A continuación se muestran dos simulaciones en *PsPice* del circuito anterior. Se muestra la corriente en dos *LEDs* y la suministrada por la fuente de alimentación.

La primera aplicando un *PWM* máximo (5V):

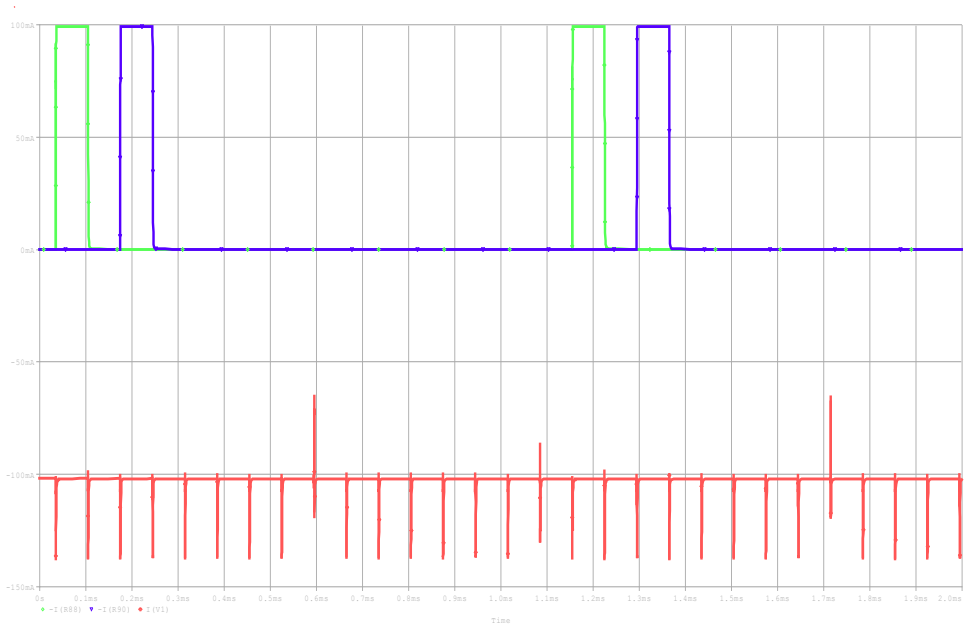


Ilustración 18 - Simulación *PsPice.* PWM aplicado a la matriz de 5V.

La segunda aplicando un *PWM* medio (2.5V):

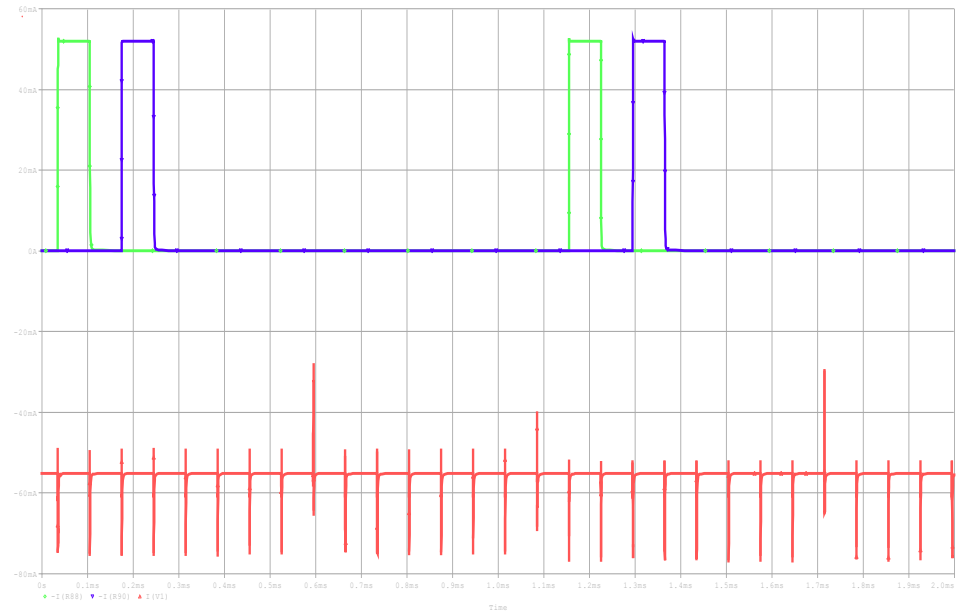


Ilustración 19 - Simulación *PsPice.* PWM aplicado a la matriz de 2,5V.

Se puede observar que el circuito regula la corriente perfectamente y realiza las activaciones. Además se puede observar el consumo del circuito cuando todos los *LEDs* están activados. Los pulsos de corriente que aparecen en la simulación se deben a los tiempos de conmutación de las puertas y los transistores de activación.

3.3 Control de corriente del led

Para realizar el control de corriente del led, y tal y como se ha explicado anteriormente, se aplicara una señal de *PWM* mediante la cual regularemos la corriente.

El microcontrolador tendrá un nivel de salida de 3,3V por lo que será necesario amplificar la tensión de salida hasta los 5V. Para conseguirlo se montara en primer lugar un filtro RC que convertirá la señal *PWM* en CC y después se colocara un amplificador no inversor con la ganancia necesaria.

Para los circuitos amplificadores no inversores la ganancia viene determinada por la siguiente ecuación:

$$V_o = V_i(1 + \frac{R3}{R2})$$

Como en el sistema necesitamos pasa de 3,3V a 5V necesitamos una ganancia de aproximadamente 1,5. Por lo tanto podemos montar R3 =1k y R2=1k8. De este modo nos quedara una ganancia de 1,55.

El circuito quedara de la siguiente manera:

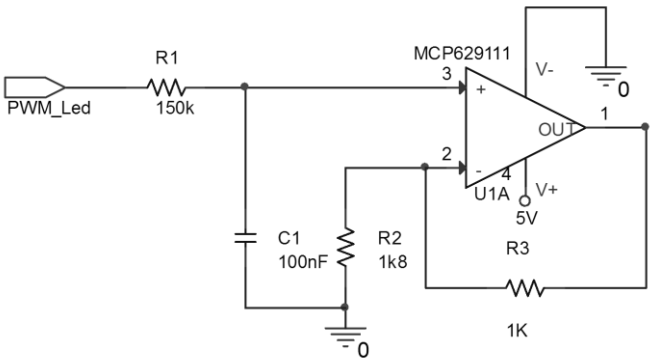


Ilustración 20 - Filtro *PWM* y adaptador de 3,3V a 5V

3.4 Sensor de luz

Existen diversos fabricantes que desarrollan soluciones finales para medir tanto propiedades como valores absolutos relacionados con la luz. En concreto, se puede medir el color de la luz, la cantidad de luz, la calidad, etc… Estas soluciones están formadas por varios bloques:

- Fotodiodo o diodo *pin:* es el sensor en cuestión que transforma la intensidad de la luz en corriente.
- Etapa de amplificación: se trata de un amplificador de transconductancia que transforma la corriente en tensión.
- Conversor A/D: transforma la tensión en una palabra binaria para poder ser tratada.
- Comunicaciones: mediante un bus *SPI* o *I2C* y unos comandos definidos por el fabricante se pueden extraer las medidas realizadas por los bloques anteriores.

En el desarrollo se ha creído interesante realizar un desarrollo completo de la solución. Como es lógico, no se abordaran todos los bloques mencionados ya que al disponer de un microcontrolador se dispone del conversor y el módulo de comunicaciones.

3.4.1 Selección del sensor

El sensor para realizar la medición de la luz ambiente tiene que ser sensible tanto a niveles de luz muy pequeños, como a la iluminación de un sistema artificial como un fluorescente o la luz solar. El problema es que la caracterización de este sensor pasaría por una función logarítmica, ya que la luz del sol es mucho más superior que la de cualquier sistema artificial y los sensores son todos lineales.

Otro aspecto importante es la longitud de onda a la que reacciona el sensor. No resulta interesante que reaccione fuera del espectro visible, ya que de ser así no estaríamos midiendo condiciones de luz reales. Por estos motivos se ha elegido el diodo pin **ALS-PD70-01C/TR7**[11] del fabricante *Everlight.* Las características principales del sensor son:

- Corriente en oscuridad (0 Lux): 2nA.
- Corriente en iluminación (1000 Lux): 12µA.
- Rango de longitud de onda (ʎ): 390nm a 700nm.

En la siguiente grafica adjunta se puede observar la respuesta en corriente a la luz ambiente:

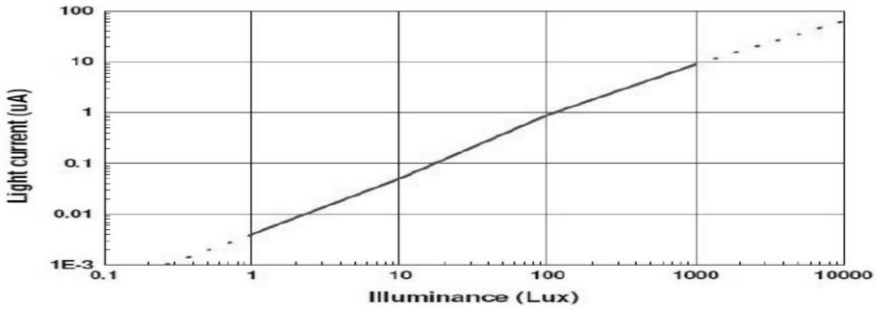


Ilustración 21 - Corriente vs Iluminación

Cumple perfectamente con los requisitos marcados anteriormente.

3.4.2 Diseño del amplificador

La función del amplificador será transformar la corriente generada por el sensor en tensión, para que el conversor A/D del microcontrolador lo pueda transformar en datos. Para esta labor existe el modelo del amplificador de transconductancia. Como ya se ha mencionado, transforma la corriente en sus entradas en tensión.

Antes de realizar el diseño, hay que seleccionar el amplificador operacional. Para este propósito hay que tener en cuenta distintos parámetros del fabricante:

- Se van a medir pequeñas variaciones de corriente a la entrada del amplificador, por lo cual deberá tener *low bias input current*. Es un parámetro especificado por los fabricantes que determina los valores mínimos de variación en la entrada.

[11] Ver hoja de especificaciones del "*ALS-PD70-01C/TR7*"

- Para aprovechar al máximo la capacidad de amplificación del sistema deberá será *rail-to-rail.* Significa que es capaz de sacar hasta el nivel de alimentación. Normalmente los amplificadores operacionales tienen perdidas (*Drop-down)*.

Todos estos parámetros los cumple el amplificador operacional **MCP6291**[12] de *Microchip.* Esta especialmente diseñado para aplicaciones con fotodiodos.

La tensión de alimentación del amplificador será de 3,3V, la misma que la del conversor ADC del microcontrolador, por lo que será deseable que el nivel de saturación sea de 3,3V. En el apartado de selección del sensor se ha especificado que la corriente generada por un fluorescente era de 12µA, pero en el diseño se realizara un cálculo con hasta 20µA para no fijar la mínima iluminación de los *LEDs* en el nivel de un fluorescente. Esta corriente es equivalente a aproximadamente 2000 Lux. Para realizar el cálculo de los componentes del amplificador nos basamos en:

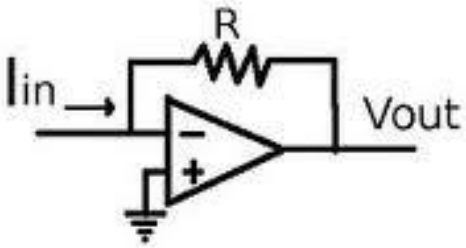


Ilustración 22 - Modelo amplificador transconductancia

$$V_{out} = R \cdot I_{in}$$

$$R = \frac{V_{out}}{I_{in}} = \frac{3{,}3V}{20\mu A} = 165K$$

A continuación se muestra una simulación en *PsPice.* Se trata de un barrido en corriente, desde el valor mínimo fijado hasta el máximo (20µA).

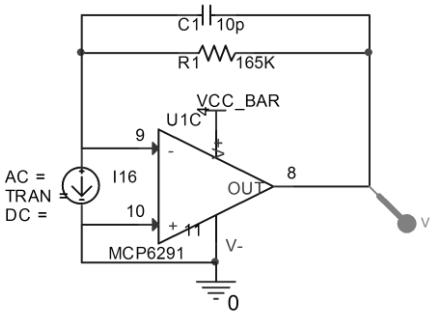


Ilustración 23 – Esquema eléctrico amplificador transconductancia

[12] Ver hoja de especificaciones de "*MCP6291*"

En la imagen se muestra como se ha sustituido el diodo pin o sensor por una fuente de corriente de *PsPice*. También se ha incluido un condensador para eliminar ruidos de la señal de salida.

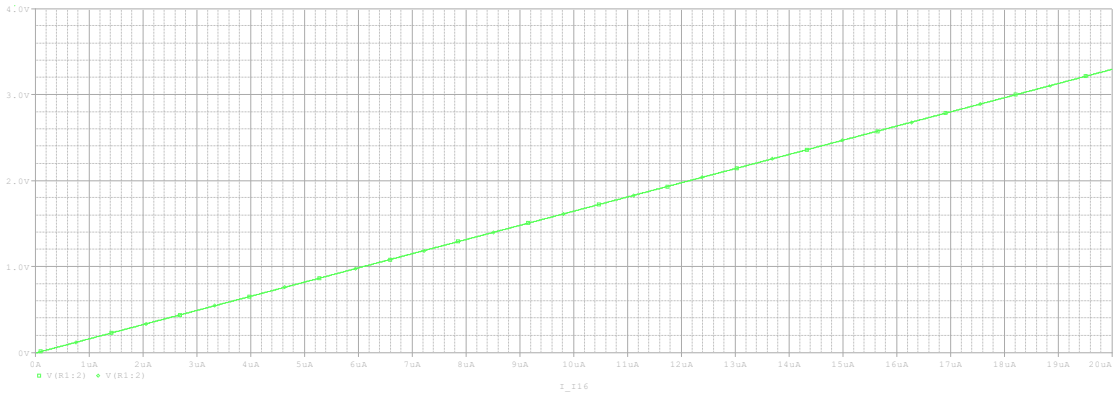


Ilustración 24 – Simulación *PsPice*. Corriente vs Tensión de salida

Tras realizar la simulación y observar los resultados se puede afirmar que el sistema está trabajando correctamente. Se observa como con la corriente máxima se obtienen a la salida del amplificador los 3,3V deseados.

3.5 Adaptación de señales RS232

En este apartado hay que buscar una solución a la incompatibilidad que existe entre los niveles de tensión proporcionados por el *PC* para el *RS232* y los niveles necesarios para el buen funcionamiento de la *UART* del microcontrolador. También hay que analizar el tipo de conexión física que existe en los aparatos estándar y adaptarlo al diseño mediante un cable o interface.

3.5.1 Análisis de la capa física RS232

RS232 es un estándar en comunicaciones. Existen distintas tipologías de conexión respecto al conector y el cable que une los dos elementos que se desean comunicar.

El más utilizado en la actualidad es el DB-9. Lo utilizan tanto los equipos como los adaptadores de USB-RS232. Por este motivo, seleccionaremos el DB-9 como la conexión del diseño e incluiremos una referencia del cable de unión entre master y esclavo en el listado final de componentes y en el precio de la solución definitiva.

Ilustración 25 - Cable conexión RS232 DB9-DB9

Ilustración 26 - Cable conexión RS232 USB-DB9

3.5.2 Desarrollo del adaptador RS232

RS232 proporciona señales referenciadas a masa. Estas señales están comprendidas entre +3V/+15V y -3V/-15V. Los microprocesadores no llevan incorporada ninguna tecnología que

sea capaz de trabajar con estos niveles de tensión por lo que hay que realizar una adaptación mediante un *chip*. Se ha elegido el *chip* **MAX3232**[13] de *Texas Instruments*, que es capaz de trabajar con un diferencial de 30V y salidas de 3,3V.

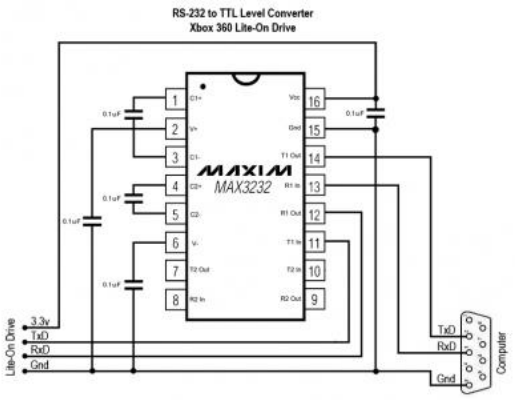


Ilustración 27 - Esquema eléctrico adaptación RS232

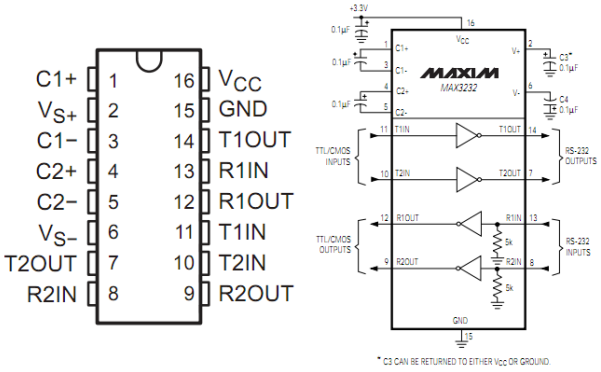


Ilustración 28 - *Pin out* MAX3232

3.6 Microprocesador

En primer lugar, y para poder seleccionar el microprocesador, hay que especificar las necesidades y requisitos que debe cumplir. Gran parte de ellos viene determinados por las condiciones del proyecto, y otros en previsión de futuras ampliaciones y un correcto funcionamiento del sistema.

3.6.1 Requisitos que debe cumplir

El microprocesador deberá implementar los siguientes periféricos:

- *PWM*: deberá tener al menos un pin de salida que implemente el *PWM* para poder controlar la potencia que se le asigna al diodo *LED*.
- Conversor A/D: por lo menos un canal con una resolución de 10 *bits* reales para poder evaluar la señal obtenida por el sensor de luz.
- *UART*: un puerto serie para comunicarse con el master y que pueda ir a una velocidad de hasta 100kbps.
- Entradas/Salidas: por lo menos deberá contar con 3 pines dedicados a funciones de I/O y que serán utilizados como *Clock_Counter, Reset_Counter* y *Data_Counter.*
- 1024 bytes de memoria *RAM* y al menos 16kB de memoria *FLASH* para el programa.

Las funciones descritas en el apartado anterior son las necesarias para cumplir con las especificaciones del TFC. Solamente queda definir la tensión de trabajo del microcontrolador.

Lo lógico serie dotar a todo el sistema con una única fuente de alimentación y que todos los dispositivos estuvieran alimentados a la misma tensión. La alimentación de los *LEDs* debe ser 5V para sacar el máximo rendimiento al sistema, por lo que el microcontrolador también estará alimentado a esa tensión.

[13] Ver hoja de especificaciones del fabricante "*MAX3232*"

3.6.2 Selección del microcontrolador

Existen diversas familias de microcontroladores que cumplen los requisitos mencionados en el apartado anterior. En nuestro caso, estamos familiarizados con los *ARM Kinetis* de *Freescale.* En concreto con la familia *Cortex M0+*. Las principales características de estos microcontroladores son bajo precio, alta velocidad y variedad de periféricos.

Para el TFC se ha pensado en el microcontrolador **MKL25Z128VLK4**[14]. Tiene un precio de 0.90€ que es muy apropiado para la solución. El encapsulado es LQFP 60. Uno de los motivos principales por los que se ha seleccionado este microcontrolador es que tiene una plataforma hardware de desarrollo y *debug* que es asequible a cualquiera.

Además tiene las siguientes características:

- 128KB de memoria Flash para las rutinas del programa.
- 32KB de memoria RAM para las variables del programa.
- 3 *UART* de hasta 5Mbps.
- 1 canal *ADC SAR* (aproximaciones sucesivas) de 12 bits con hasta dieciséis entradas.
- Hasta ocho salidas *PWM*.
- Hasta 60 pines configurables como entrada y salida.
- Hasta 8 pines de entrada capaces de generar interrupción ante cambios y transiciones.
- Velocidad de 48MHz.
- Tensión de trabajo desde 2,7V hasta 3,3V.
- Modos de trabajo de bajo consumo.
- *Watchdog* o controlador de pérdidas en la ejecución del código.

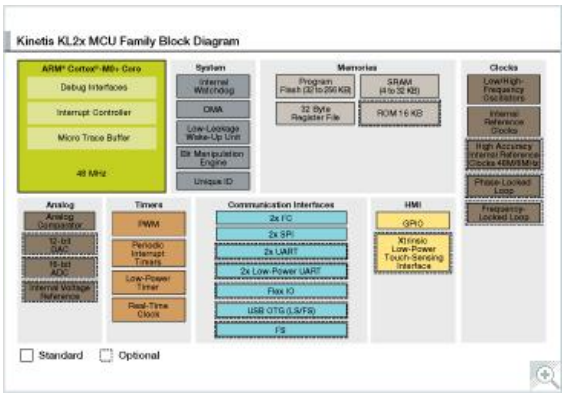


Ilustración 29 - Esquema de bloques del microcontrolador MKL25Z128

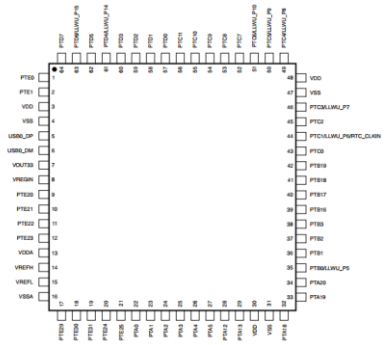


Ilustración 30 - Encapsulado y pines de MKL25Z128

3.6.3 Desarrollo del microcontrolador

Para realizar el desarrollo del apartado dedicado al microprocesador se dispone de todas las variables necesarias. El bloque del microcontrolador quedaría de la siguiente manera:

[14] Ver hoja de especificaciones de "**MKL25Z128VLK4**"

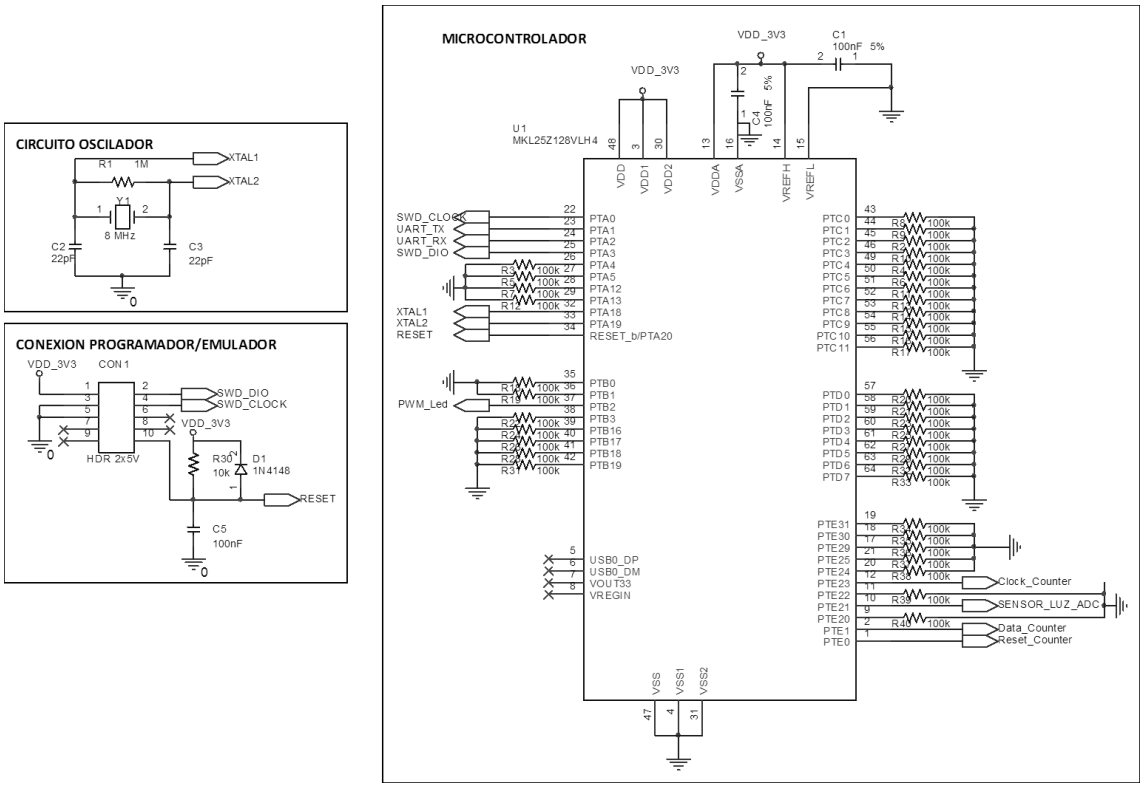


Ilustración 31 - Esquema eléctrico microcontrolador

En la anterior figura se puede observar que se ha colocado un cristal de 8MHz para proporcionar *clock* al microcontrolador. También se han adaptado las señales no utilizadas del microcontrolador mediante *pull-down*. Por último se ha añadido el conector y los componentes necesarios para grabar la *Flash* y poder emular el código fuente.

3.7 Fuente de alimentación

En este apartado se debe construir la fuente de alimentación que convierta la tensión de entrada (24V) a la tensión necesaria en nuestros circuitos. Como ya se ha comentado anteriormente esta tensión será de 5V para todos los circuitos del TFC y de 3,3V para el microcontrolador.

3.7.1 Análisis de los tipos de fuentes de alimentación

Existen distintos tipos de fuentes de alimentación.

- Cuando el consumo del diseño es muy bajo se suelen montar **fuentes de alimentación lineales**. Estas fuentes disipan los diferenciales de tensión entre la entrada y la salida en forma de calor. Como en este diseño el consumo es medio y la diferencia de potencial entre la entrada y la salida es alta, no es aconsejable montar una fuente de este tipo ya que será necesario montar un radiador en el conversor para disipar el calor.
- Otro tipo son las **fuentes conmutadas**. Estas cuentan con un interruptor de potencia, un inductor y un diodo para transferir la energía de la entrada a la salida. El único problema de estas fuentes es que pueden inducir ruido eléctrico al sistema debido a la velocidad de conmutación. Existen diferentes topologías de fuentes conmutadas. Esto es debido a la diferencia de potencial entre la entrada y la salida. Si esta diferencia es

positiva se denominan *reductores* y si es negativa *elevadores*. En las siguientes figuras se muestran:

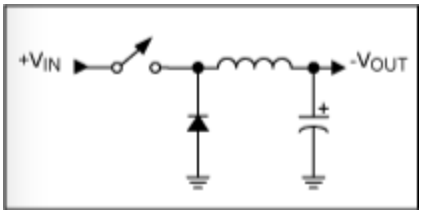


Ilustración 32 - Reductor *buck*

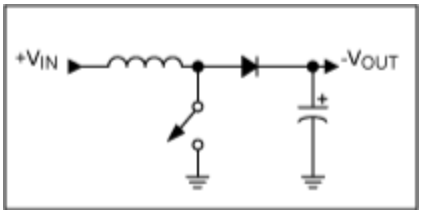


Ilustración 33 - Elevador *boost*

En el diseño de la fuente de 5V se utilizara el tipo reductor *buck* debido a que la diferencia de potencial entre la entrada y para la de 3,3V se utilizara una lineal.

3.7.2 Diseño de la fuente de alimentación 5V

El primer paso para diseñar la fuente de alimentación es conocer la corriente que debe suministrar al circuito para que no tenga problemas de rendimiento. Como se ha comentado en apartados anteriores, la matriz de *LEDs* tendrá un consumo máximo de 100mA, el microcontrolador aproximadamente 10mA y la circuitería de control unos 20mA. Por lo tanto, será necesaria una fuente de alimentación que suministre 5V y unos 150mA aproximadamente. Con estos parámetros se puede proceder al diseño.

Para realizar el diseño se ha utilizado la herramienta de *Texas Instruments Webench Design Center*[15]. En esta página, una vez seleccionado el tipo de asistente *Power* (existen para varios tipos de circuitos), solamente queda introducir los parámetros necesarios. En nuestro caso seleccionamos:

- Tensión de entrada: mínima 12V y máxima 24V.
- Corriente de salida: 1A. Se ha sobredimensionado pensando en futuras ampliaciones del sistema.
- Tensión de salida: 5V.

Una vez se han proporcionado todos los parámetros, la aplicación nos muestra una serie de opciones finales para nuestro diseño. En cada una de ellas se especifica el coste, el tamaño y el rendimiento. En este caso se ha seleccionado el sistema con menor coste y más eficiencia.

A continuación se muestra el diseño generado mediante la aplicación *Webench*:

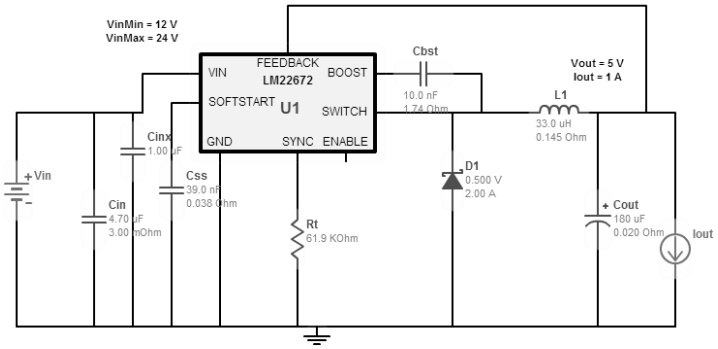


Ilustración 34 - Esquema eléctrico de la fuente de alimentación

[15] Ver herramienta para el cálculo de fuentes de alimentación "*Webench*"

3.7.3 Diseño de la fuente de alimentación 3,3V

Para el diseño de la fuente lineal de 3,3V se utilizara un regulador lineal llamado ***FAN2504***[16]**.** Este regulador puede suministrar hasta 200mA y tiene un encapsulado muy pequeño (*SOT23)*. Además como se va a alimentar a 5V no tiene problemas de disipación. Es un regulador ideal para el microcontrolador.

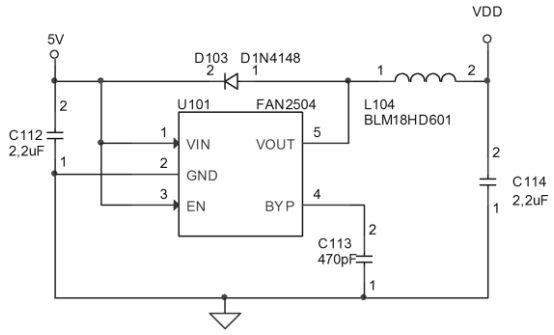


Ilustración 35 - Regulador lineal con salida 3,3V

4 Diseño del firmware

4.1 Selección del compilador

Para realizar el diseño y desarrollo del firmware se utilizara el software gratuito *Kinetis Design Studio* de *Freescale.* Es un compilador *GNU ARM* integrado en el entorno *Eclipse.* La aplicación cuenta con un asistente denominado *Processor Expert* que está pensado para facilitar la configuración de los periféricos del microcontrolador y el propio núcleo. En el desarrollo se utilizara esta herramienta para configurar todos los periféricos.

Esta herramienta permite la configuración de todos los aspectos necesarios para realizar tanto la inicialización de la CPU seleccionada como de sus periféricos. Para todos ellos, permite cambiar de modo sencillo las propiedades del dispositivo seleccionado. Mediante estas propiedades se configura el dispositivo en cuestión y se adapta a las necesidades de cada desarrollo. Además se puede configurar si será necesaria una señal de interrupción o no, si es necesario que el periférico o dispositivo sea inicializado después de un *reset*, o los métodos que es necesario que el asistente añada. Para cada dispositivo configurado se dispone de tres pestañas de configuración:

- Propiedades: muestra las propiedades que es posible modificar para el dispositivo, así como seleccionar el pin o pines del microcontrolador que estarán ligados al dispositivo configurado.
- Métodos: propone una serie de funciones para generar de manera automática. Se pueden añadir o quitar cuantas se quieran.
- Eventos o interrupciones: se puede configurar la interrupción que generara el dispositivo y su nivel de prioridad.

[16] Ver nota de aplicación FAN2504

A continuación se muestra una imagen del aspecto de *Processor Expert.*

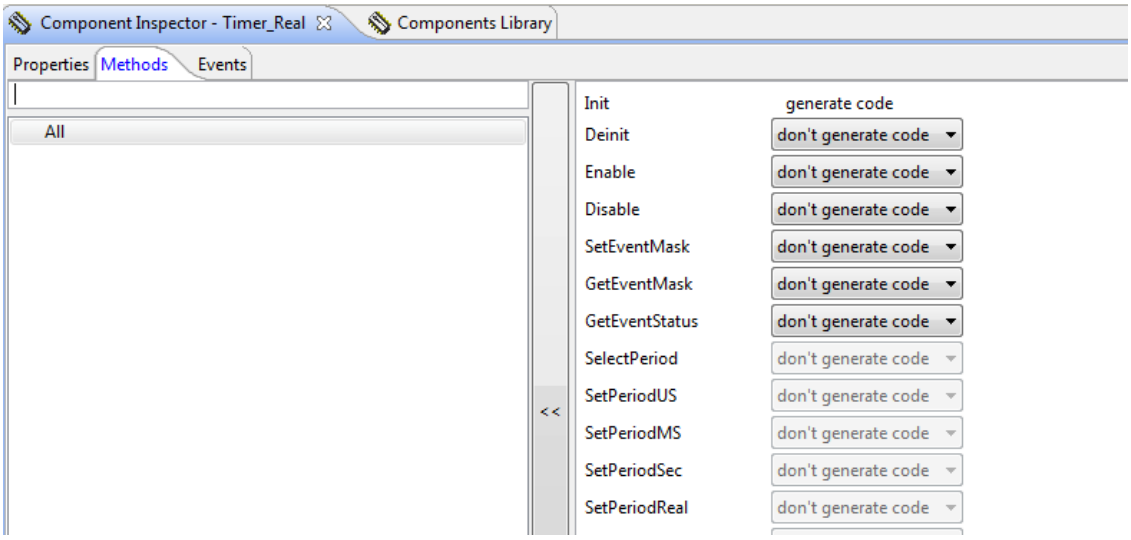


Ilustración 36 - Muestra de *Processor Expert*

Para realizar el *debug o simulación* del software se utilizara una tarjeta de demostración denominada *Freedom KL25.* En dicha tarjeta están disponibles todos los pines del microcontrolador por los que se simularan muchos de los comportamientos del software.

4.2 Inicializaciones del microcontrolador

Como ya se comentó en el apartado dedicado a la selección del microcontrolador, la velocidad máxima que soporta es de 48MHz. Para conseguir esta velocidad el dispositivo debe contar con un cristal externo que suministre un *clock* al núcleo. En este apartado se procederá a configurar mediante *Processor Expert* los parámetros dedicados al *clock* interno del microcontrolador. En la siguiente imagen se aprecian los parámetros seleccionados:

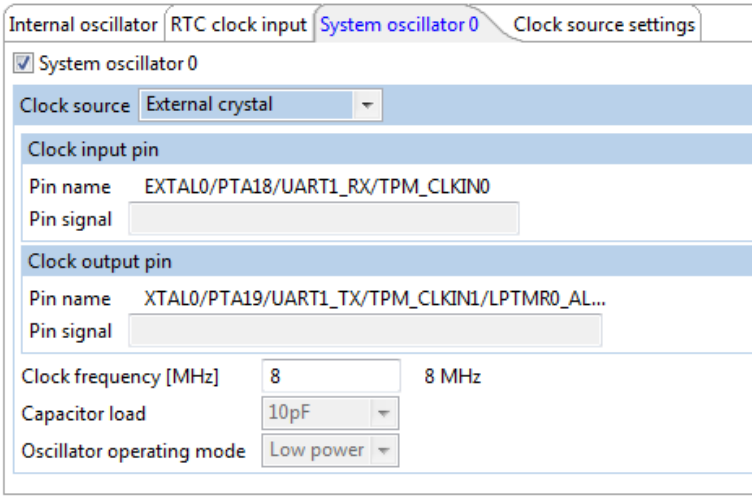


Ilustración 37 - *Processor Expert.* Configuración del *clock* interno

Una vez parametrizado el *clock* del microcontrolador el asistente se encarga de inicializar los registros adecuadamente.

Los demás parámetros de inicialización de la CPU son los que vienen por defecto en el asistente. Esto quiere decir que no trabajaremos con modos especiales de bajo consumo, no

realizamos mapeos de la *Flash* distintos de los propuestos por el compilador, no se protege la *Flash* contra lectura externa, etc…

El compilador generara el código fuente de manera automática. No es necesario escribir ningún fichero para gestionar la *Flash*, las interrupciones o las inicializaciones básicas del microcontrolador. Todos los ficheros necesarios son generados por *Processor Expert*. A continuación se muestra el fragmento de código en el que se realiza la inicialización de la CPU:

```
/*** ### MKL25Z128VLK4 "Cpu" init code ... ***/
/*** PE initialization code after reset ***/
SCB_VTOR = (uint32_t)(&__vect_table); /* Set the interrupt vector table position */
/* Disable the WDOG module */
/* SIM_COPC: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=
SIM_COPC = SIM_COPC_COPT(0x00);

/* System clock initialization */
/* SIM_CLKDIV1: OUTDIV1=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,OUTDIV4=3,??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
SIM_CLKDIV1 = (SIM_CLKDIV1_OUTDIV1(0x00) | SIM_CLKDIV1_OUTDIV4(0x03)); /* Set the system prescalers to safe value */
/* SIM_SCGC5: PORTE=1,PORTB=1,PORTA=1 */
SIM_SCGC5 |= SIM_SCGC5_PORTE_MASK |
```

Ilustración 38 - Fragmento de código fuente para la inicialización de la CPU

4.3 Gestión de interrupciones

Para el proyecto es necesaria la ejecución de interrupciones en el microcontrolador. Las interrupciones permiten atender peticiones de sistemas externos o simplemente abstraer la programación de consultas continuas a periféricos o registros internos del microcontrolador. En este proyecto será necesario gestionar cuatro tipos de interrupciones:

- Modbus o *UART:* existirá una interrupción dedicada a la escucha del puerto serie del microcontrolador. Cuando se reciba un *byte* por la línea serie se ejecutara una interrupción de manera automática y el código comprobara si la información que está siendo recibida es correcta o no.
- *Clock* de la matriz de LEDs: cada 35µs se ejecutara una interrupción de *timer*. En su ejecución será necesario generar los flancos de subida y de bajada de la señal ***Clock_Counter***. Además, y dependiendo del flanco que se disponga a aplicar será necesario cargar en el pin ***Data_Counter*** el valor que activara el led o no.
- Base de tiempos: el sistema debe contar con una base de tiempos que proporcione un valor medible en las escalas correspondientes. En este caso, se contara con un *timer* que cada 1ms ejecutara una interrupción. En dicha interrupción incrementaremos una variable que servirá como referencia temporal para las distintas temporizaciones que deban ser ejecutadas.
- Conversor *ADC:* cada vez que sea necesario ejecutar una conversión analógico-digital la CPU forzara el inicio de la misma. Para no tener que estar esperando la finalización de la conversión se configurara de tal manera que cuando haya terminado genere un evento o interrupción.

Con estas cuatro interrupciones se pretende mejorar la ejecución del código del panel de LEDs en cuanto a esperas y rendimiento del software.

Cada una de estas interrupciones ira ligada a una función específica que será configurada en *Processor Expert.*

4.4 Programa principal

El programa principal o *main()*, es la línea de ejecución de esta aplicación. Se divide en dos apartados:

- Inicialización de CPU, periféricos y módulos: se realizaran las llamadas necesarias para inicializar las principales funciones del microcontrolador y los módulos o máquinas de estado de cada apartado (control matriz, control luz ambiente, comunicaciones Modbus).
- Ejecución de módulos: se realiza una llamada a cada uno de los módulos. Estas llamadas son ejecuciones de la máquina de estados de cada uno de ellos. Cada máquina de estados ejecutara una acción y devolverá el control al programa principal.

4.4.1 Módulos del programa principal

Los módulos que se ejecutaran en el programa principal serán los siguientes:

- **Comunicaciones Modbus** (*Modbus.c*): en este módulo se procesan los comandos recibidos a través de la línea serie. Estos comandos deberán cumplir la especificación del protocolo Modbus. Si la recepción es correcta se ejecutara la acción requerida y este módulo será el encargado de emitir la respuesta al dispositivo maestro.
- **Control de la matriz** (*ControlMatriz.c*): se trata del módulo que controla las señales de activación y desactivación de la matriz de LEDs. En concreto, es el modulo encargado de generar el *Clock* y de activar o desactivar el led que este seleccionado en la matriz. Además controlara los distintos modos de visualización de la matriz y será el encargado de procesar los mensajes recibidos a través de Modbus.
- **Control de luz ambiente** (*LuzAmbiente.c*): su tarea es realizar medidas de la luz ambiente y en función del resultado calcular y aplicar una señal de *PWM* correcta para controlar la iluminación del panel de LEDs.
- **Temporizador** (*Temporizador.c*): este módulo se ejecuta en segundo plano mediante una interrupción. Cada milisegundo se ejecutara e incrementara una variable que es utilizada como un contador. Este módulo se utilizara de apoyo para realizar temporizaciones y mediciones de tiempo en el programa.

4.4.2 Flujo del programa principal

El flujo del programa es muy básico. Como ya se ha comentado está dividido en dos partes. La primera se ejecuta después de un reset y la segunda es un bucle de ejecución continuo.

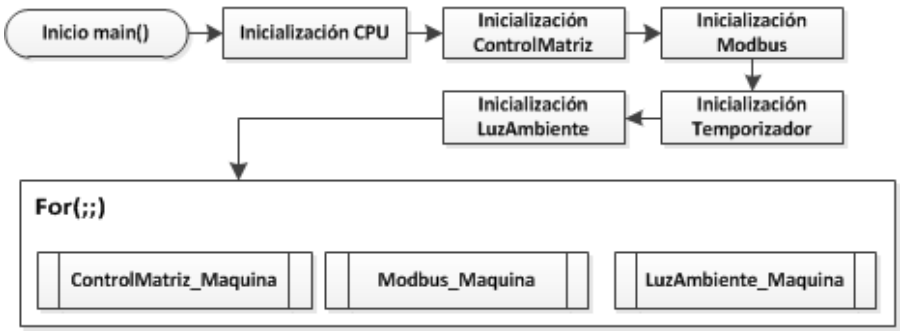


Ilustración 39 - Flujo del programa principal *main()*

4.5 Comunicaciones mediante protocolo Modbus

En este apartado se realizara una introducción al protocolo Modbus y a sus aspectos más generales. Además se explicara cómo se ha adaptado para la funcionalidad necesaria en el TFC.

4.5.1 Introducción al protocolo

Modbus es un protocolo de comunicaciones situado en el nivel 7 del *Modelo OSI*[17]. Mediante este protocolo es posible realizar comunicaciones tipo cliente-servidor o maestro-esclavo. En este proyecto realizaremos comunicaciones maestro-esclavo. Esto significa que el maestro siempre tomara la iniciativa en las comunicaciones, por lo que el esclavo solamente se limitara a responder o ejecutar los comandos solicitados por el maestro.

Modbus puede ir implementado en distintos medios físicos. Puede funcionar sobre *TCP-IP, RS485 o RS232.* En este último medio será en el que se utilizara en este proyecto.

4.5.2 Funcionamiento Maestro/Esclavo

En este tipo de funcionamiento existen dos tipos de dispositivos:

- Maestro: es quien toma la iniciativa en las comunicaciones y solamente existe uno. Puede enviar una petición a un dispositivo en concreto (*unicast)* o dirigirse a todos los dispositivos (*broadcast*).
- Esclavo: pueden existir hasta 247 dispositivos conectados en el bus de comunicaciones.

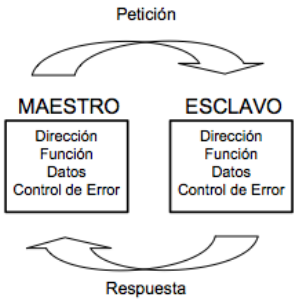


Ilustración 40 - Comunicaciones Maestro/Esclavo en Modbus

4.5.3 Direccionamiento y tramas

El direccionamiento en Modbus se realiza mediante un *byte* en el cual se pueden dar hasta 256 direcciones distintas. La dirección 0 está reservada para las peticiones *broadcast* y el maestro y de la 248 a la 255 están reservadas por el protocolo. Cada una de estas direcciones debe identificar de manera única a un dispositivo.

Tabla 5 - Direccionamiento en Modbus

0	1 a 247	248 a 255
Dirección de maestro	Dirección de cada esclavo	Reservadas por Modbus

El formato de las tramas en Modbus viene determinado por una unidad de datos única denominada *PDU (Protocol Data Unit).* Esta unidad consta únicamente del comando o código de función a ejecutar y de los datos.

[17] Ver modelo OSI

Tabla 6 - Trama Modbus. *Protocol Data Unit*

PDU (Protocolo Data Unit)	
Código de función o comando	Datos

Dependiendo del medio físico, redes o buses, se complementa el *PDU* con más información. Esta información es necesaria para identificar al emisor o al receptor o para darle seguridad a los datos enviados. Así pues, en el caso del RS232 se complementa añadiendo un campo de dirección y otro de *CRC.*

Tabla 7 - Formato de trama en Modbus

	PDU (Protocolo Data Unit)		
Dirección	Código de función o comando	Datos	CRC

4.5.4 Modos de transmisión serie

En Modbus existen dos tipos de transmisión serie: modo RTU (*Remote Terminal Unit)* y modo ASCII. En este proyecto se utilizara el modo RTU. No se va a explicar el funcionamiento en modo ASCII ya que no es de interés en este momento pero si hay que matizar que todos los dispositivos conectados a un bus deben tener el mismo modo de transmisión serie.

MODO RTU

En el modo RTU se envían *arrays* de *bytes*. Cada *byte* puede ser enviado con un formato de comunicación serie estándar. En este proyecto se enviara por cada *byte* 1 *bit* de *start,* 8 *bit* de datos y 2 *bits* de *stop* o uno dependiendo si se trabaja con paridad o no. La velocidad de comunicación recomendada es de 9600bps.

En este modo de funcionamiento el máximo tamaño de trama será de 256 *bytes* por lo que el formato de la trama quedara de la siguiente manera:

Tabla 8 - Formato de trama en modo RTU

	PDU (Protocolo Data Unit)		
Dirección	Código de función o comando	Datos	CRC
1 *byte*	1 *byte*	0 a 252 *bytes*	2 *bytes*

El cálculo del CRC se realiza mediante un algoritmo proporcionado en la especificación Modbus[18].

Los errores de comunicación vienen controlados en primer lugar por los dispositivos hardware y en segundo lugar por la verificación del CRC y temporizaciones entre *bytes* de la trama y tiempo entre la petición del maestro y la respuesta del esclavo.

En el siguiente flujo se puede observar el envío, respuesta y control de errores en modo RTU:

[18] Link a especificación Modbus.

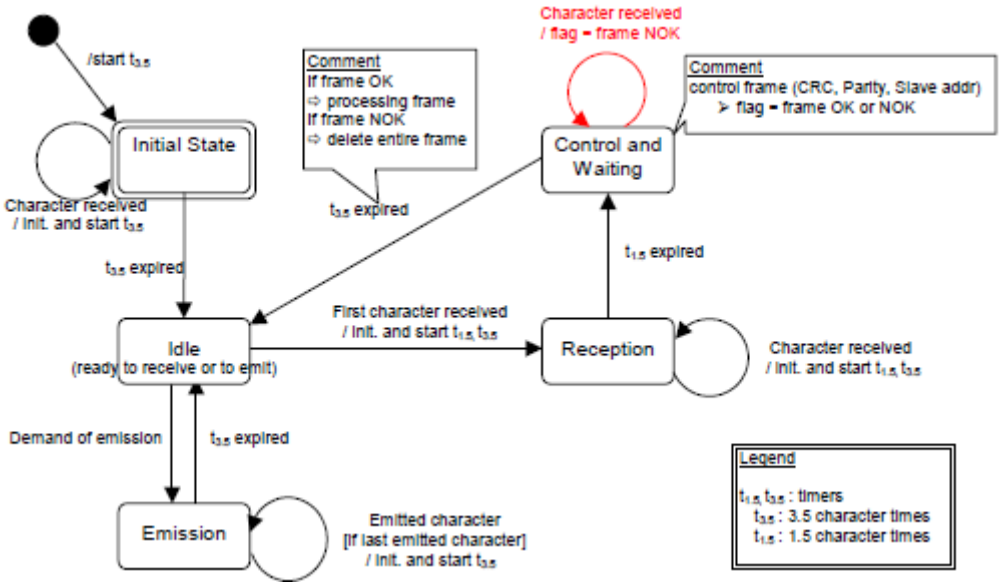


Ilustración 41- Flujo de comunicaciones en modo RTU

4.5.5 El mapa de memoria

Para poder elaborar el mapa de memoria del proyecto en primer lugar se deberá aclarar el tipo de datos que utilizaremos en nuestra aplicación y los rangos de memoria que tenemos disponles.

Los tipos de datos que implementa Modbus son los siguientes:

- Entrada o salida digital: variable de tamaño 1 *bit* que es utilizada para la activación de dispositivos o la lectura de sus entradas.
- Registros de entrada o salida: variable de tamaño 16 *bits* utilizada para la escritura o lectura de señales tipo analógico o registros de la aplicación.

Para nuestra aplicación, y dado que no es necesario que trabajemos directamente con señales de entrada o salida tipo *bit*, trabajaremos directamente con los registros de entrada o salida. En concreto trabajaremos con los denominados *analog inputs* para la lectura del nivel de luz ambiente y los *holding registers* para la escritura de los parámetros de control y mensajes del panel de LEDs.

Para poder terminar con el mapa de memoria es necesario definir exactamente los datos que se desearan leer o escribir en la aplicación.

Como entradas analógicas será necesario conocer el valor del nivel de luz ambiente.

Como registros de propósito general manejaremos los siguientes datos:

- **Registro estado comunicaciones**: mediante este registro se puede modificar la velocidad de comunicación del dispositivo. Las opciones son 9600 *baudios* o 19200 *baudios.*
- **Registro control**: mediante este registro se podrá enviar la orden de inicio, parada o reposo. Con valor 0 estará en reposo o después de un *reset*, con valor 1 se iniciara el

proceso de mostrar el mensaje en el panel y con valor 2 se detendrá el proceso de mostrar el mensaje.

- **Registro de modo**: mediante este registro se configuran los modos de visualización de mensaje en el panel. Con valor 0 se muestra el mensaje en modo estático, con valor 1 en modo estático temporizado y con valor 3 en modo desplazamiento. Estos modos se explicaran más adelante.
- **Registro temporización**: mediante este registro se configura la temporización si la existe de los modos de trabajo del panel.
- **Registro longitud mensaje**: indicamos la longitud del mensaje que vamos amostrar en el panel.
- **Registros mensaje**: almacena el mensaje a mostrar. Puede tener un tamaño máximo de 250 caracteres.

El mapa de memoria quedaría de la siguiente manera:

Tabla 9 - Mapa de memoria de la aplicación

Dirección	MSB	LSB	Tipo
@30001	Nivel de luz ambiente		Entrada analógica
@40001	Registro de estado comunicaciones		Propósito general
@40002	Registro de control		Propósito general
@40003	Registro de modo		Propósito general
@40004	Registro temporización		Propósito general
@40005	Registro longitud mensaje		Propósito general
@40006	Carácter 1	Carácter 2	Propósito general
hasta			
@40131	Carácter 251	Carácter 252	Propósito general

4.5.6 Comandos o funciones

En Modbus existen tres tipos de comandos o funciones definidos por la especificación. Están las funciones públicas, las definidas por el usuario y las funciones reservadas. En concreto y para el TFC solamente será necesario utilizar alguna de las funciones públicas:

Tabla 10 - Comandos utilizados en la aplicación

Comando	Código	Descripción
Read input registers	04	Lectura de la entrada analógica
Read single register	03	Lectura de registro
Write multiple registers	16	Escritura de registros

El formato de petición y respuesta para cada uno de los comandos será:

Comando 16 (escritura de registros):

Tabla 11 - Solicitud maestro. Comando 16.

Solicitud maestro												
Dirección esclavo	Comando	Dirección registro 1		Nº registros		Nº bytes	Valor registro 1		Valor registro N		CRC	
1 byte	1 byte	MSB	LSB	MSB	LSB	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Tabla 12 - Respuesta esclavo. Comando 16.

Respuesta esclavo							
Dirección esclavo	Comando	Dirección registro 1		Nº bytes escritos		CRC	
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Comando 04 - 03 (lectura de registros analógicos y de propósito general):

Tabla 13 - Solicitud maestro. Comando 04 -03.

Solicitud maestro							
Dirección esclavo	Comando	Dirección registro 1		Nº registros		CRC	
1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

Tabla 14 - Respuesta esclavo. Comando 04 - 03.

Respuesta esclavo								
Dirección esclavo	Comando	Nº bytes leídos	Valor registro 1		Valor registro N		CRC	
1 byte	1 byte	1 byte	MSB	LSB	MSB	LSB	MSB	LSB

4.5.7 Códigos de error en la respuesta

En el apartado anterior se trataban los comandos y su respuesta para cuando no existe ningún problema en la transmisión del mensaje, los datos solicitados o la ejecución de la petición. Si cualquiera de estos errores ocurriese, el protocolo ha definido una serie de respuestas para cada situación:

Tabla 15 - Códigos de error en Modbus

Código de error	Tipo de error	Descripción
01	Función no valida	El comando solicitado no existe.
02	Dirección no valida	Dirección fuera del rango.
03	Dato o parámetro no valido	Dato o parámetro no permitido para ese registro.
04	Error de ejecución	No se ha podido ejecutar el comando o hay un error en el dispositivo esclavo.
05	*ACK* o reconocimiento	Se está procesando la petición.
06	*Busy* u ocupado	El dispositivo está ocupado con otro comando o función.
16	*NACK* o error de transmisión	Error en la transmisión.

4.5.8 Inicialización de periféricos UART

La inicialización de la UART que será utilizada en las comunicaciones se ha realizado mediante la herramienta *Processor Expert*. Solamente ha sido necesario indicar que periférico se desea utilizar y los parámetros necesarios para la comunicación (*baudios, bits de start-stop,*etc…)

A continuación los parámetros seleccionados:

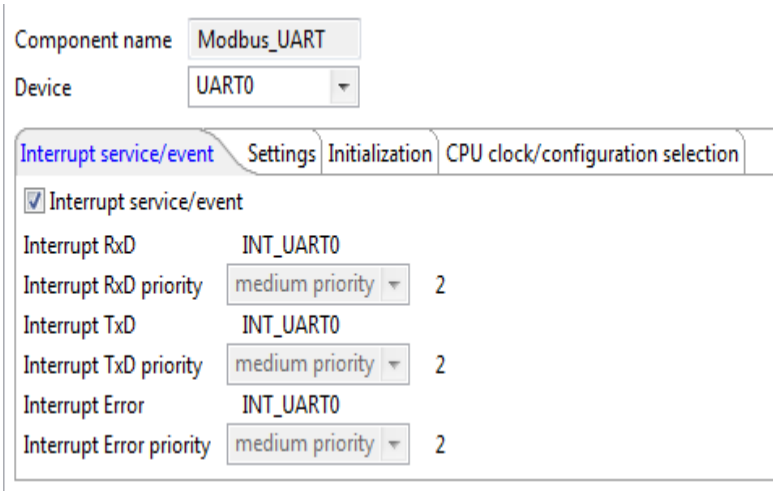


Ilustración 42 - *Processor Expert*. Interrupciones de la UART

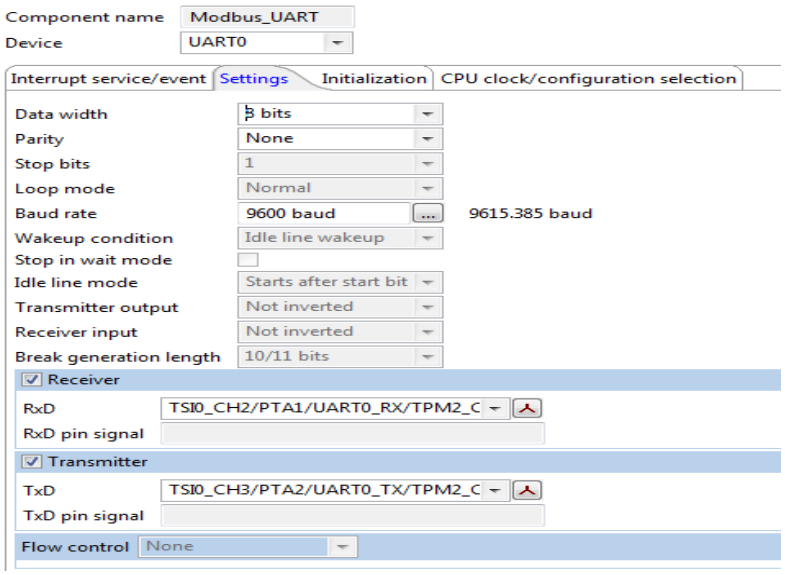


Ilustración 43 - *Processor Expert*. Inicialización de la UART

4.5.9 Desarrollo del software

A parte del código generado por *Processor Expert*, se he desarrollado el módulo *Modbus.c* donde se realiza el control de las comunicaciones y procesado de los comandos enviados a través de la *UART.* En concreto, su función es la de procesar los caracteres recibidos y comprobar que se tratan de comandos correctos, dirigidos a la dirección correcta y con los parámetros adecuados. Si no es así, generara el error correspondiente y si es necesario lo transmitirá al maestro. Si el comando es correcto será el encargado de almacenar o leer de los registros la información solicitada.

Además de controlar los datos que llegan por el puerto serie, también es el modulo encargado de controlar las temporizaciones entre *bytes* recibidos. Si se cumple un tiempo determinado entre bytes se reinician los *bufferes* de comunicaciones para comenzar a almacenar otra trama. El flujo de las comunicaciones es muy sencillo. Existen solamente cuatro estados: inicio, esperando, procesando comando y enviando respuesta.

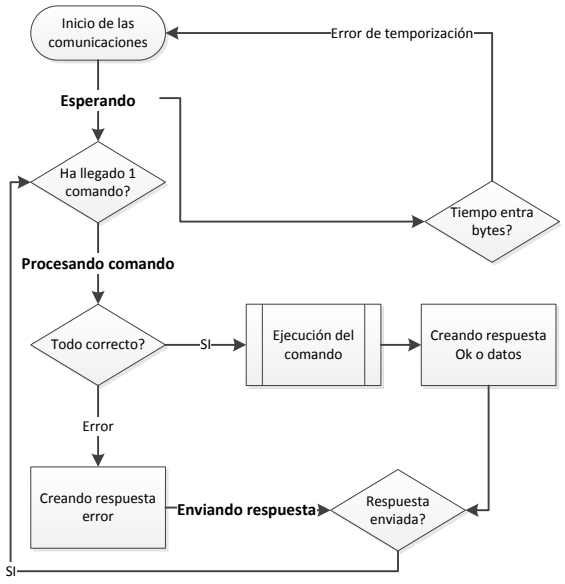


Ilustración 44 - Flujo de comunicaciones Modbus

4.5.10 Simulación del protocolo Modbus

Para realiza la simulación del funcionamiento del protocolo Modbus en el proyecto, se ha utilizado como dispositivo maestro una aplicación (*Modbus poll)* suministrada desde la página Web de Modbus[19]. Esta aplicación permite enviar comandos de lectura y escritura de registros de propósito general. Además de la aplicación se ha cargado el software desarrollado en la tarjeta utilizada para realizar *debug*. Como ya se comentó, esta tarjeta es la *Freedom KL25* de Freescale. Para la captura de las señales se ha utilizado un Osciloscopio de PC (Picoscope) vía USB.

En primer lugar se ha capturado el comando de lectura (0x03) de dos registros a partir de la dirección @40001. El sistema devuelve valor 0 y valor 1 para cada uno respectivamente.

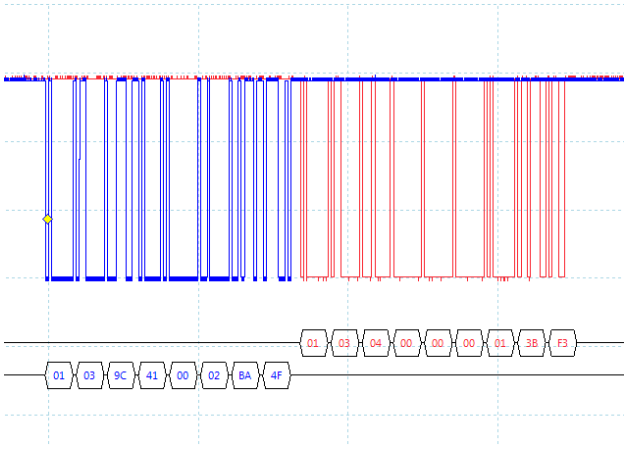


Ilustración 45 - Simulación del comando 0x03 de Modbus

[19] Se puede descargar en el siguiente link.

La siguiente captura es la escritura (0x10) de un registro en la dirección @40004. En concreto estamos escribiendo en el registro de temporizaciones un tiempo de 1000. El sistema responde que ha escrito dos *bytes.*

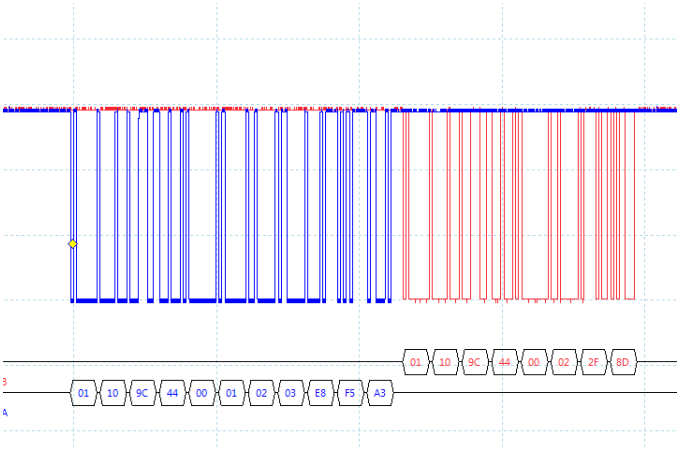


Ilustración 46 - Simulación del comando 0x10 de Modbus

4.6 Control de la matriz de LEDs

Se denomina control de la matriz de LEDs a la parte del software que está dedicada a transformar los mensajes que llegan mediante Modbus en señales eléctricas. Estas señales serán las que mostraran el mensaje en los LEDs. En esta transformación se tendrán en cuenta los distintos parámetros seleccionados por el usuario.

Se denominan parámetros seleccionados por el usuario a los registros que leeremos o escribiremos mediante las comunicaciones vía línea seria y protocolo Modbus. Para un correcto funcionamiento del panel o matriz de LEDs es necesario seguir una secuencia de inicialización correcta de los registros:

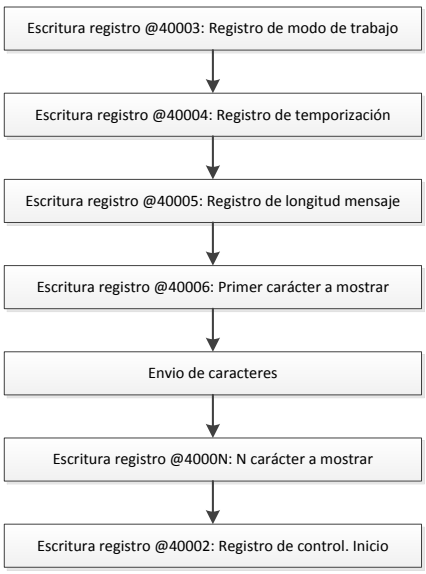


Ilustración 47 - Secuencia de inicio de un nuevo mensaje

Existirán distintos modos de trabajo del panel (registro @40003):

- **Estático (valor 0):** se muestra un único carácter en la matriz de LEDs y para cambiarlo es necesario enviar otra secuencia de mensajes a través de Modbus.
- **Estático temporizado (valor 1):** se puede mostrar un mensaje o array de caracteres. Los caracteres van cambiando cuando se cumple la temporización (registro @40004) enviada a través de Modbus.
- **Desplazamiento (valor 2):** se puede mostrar un mensaje o array de caracteres. El mensaje va deslizándose por la matriz y la velocidad de deslizamiento (registro @40004) es enviada por Modbus.

El código fuente de este módulo está contenido en el fichero *ControlMatriz.c* y la definición de variables y funciones públicas en el fichero *ControlMatriz.h*. El control se realiza mediante un máquina de estados denominada *ControlMatriz_Maquina()* que es llamada desde el programa principal y la función de interrupción *ControlMatriz_Interrupcion35us()* que es ejecutada cada 35µ por la interrupción del *Timer_Clock*.

El flujo de ejecución de la máquina de estados *ControlMatriz_Maquina()* será el siguiente:

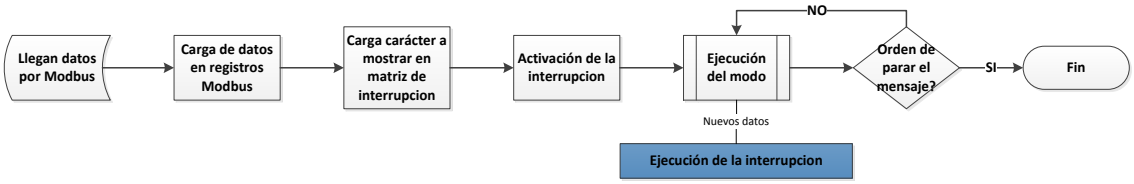


Ilustración 48 - Flujo de *ControlMatriz_Maquina()*

En el anterior flujo de programa se observa el traspaso de datos entre los registros de Modbus y la interrupción. Este traspaso de datos es necesario para controlar los distintos modos de trabajo configurados. En el traspaso de datos se realizara la traslación del carácter ASCII (registros @40006-@40131) a los datos formateados para mostrar en la matriz. En este módulo se ha incluido la tabla de la fuente *swiss721_outline*[20] en la que cada carácter tiene un formato de 16x16 bits que se ajusta perfectamente al panel de LEDs.

Para el modo estático y estático temporizado se cargan los datos directamente en una matriz de 16x16 bits.

$$ASCII \rightarrow \begin{bmatrix} b_{16:1} & \cdots & b_{1:1} \\ \vdots & \ddots & \vdots \\ b_{16:16} & \cdots & b_{1:16} \end{bmatrix}$$

Para el modo de desplazamiento se cargara el primer carácter a mostrar en la parte derecha de una matriz de 32x16 bits. Cada vez que se cumpla la temporización (registro @40004) se desplazaran las columnas una posición hacia la izquierda. Se seguirá este proceso hasta que se haya desplazado 16 veces que será el momento en el que se realice la carga del siguiente carácter a mostrar. En ese momento estará siendo visualizado el carácter completamente en la matriz de LEDs y mediante los siguientes desplazamientos desaparecerá por la parte izquierda.

[20] La fuente *swiss 721 outline* se ha obtenido del siguiente link.

$$ASCII \rightarrow \begin{bmatrix} b_{32:1} & \cdots & b_{1:1} \\ \vdots & \ddots & \vdots \\ b_{32:16} & \cdots & b_{1:16} \end{bmatrix}$$

4.6.1 Inicialización de los periféricos

En este apartado se manejaran en concreto tres pines o señales del microcontrolador y una interrupción de *timer*. Se han utilizado pines I/O de propósito general para estas señales. Las señales son:

- *Clock_Counter:* proporciona el *clock* al circuito que genera el barrido de las filas y columnas.
- *Reset_Counter:* poniendo esta señal a nivel alto se genera un *reset* o reinicio en los contadores de filas y columnas.
- *Data_Counter:* poniendo esta señal a nivel se aplica al led que esta seleccionado una corriente establecida por la señal *PWM_Led.*

Para la inicialización de los periféricos se ha utilizado *Processor Expert.* A continuación se muestran las capturas realizadas con las configuraciones de cada uno de ellos.

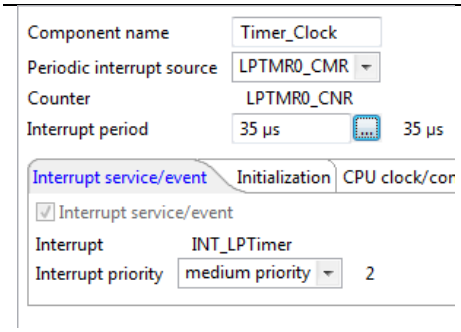


Ilustración 49 - *Processor Expert.* Inicialización del *timer* de 35us

Timer Clock
Configurado para ejecutar una interrupción cada 35µs.
Prioridad asignada a la interrupción de tipo medio.

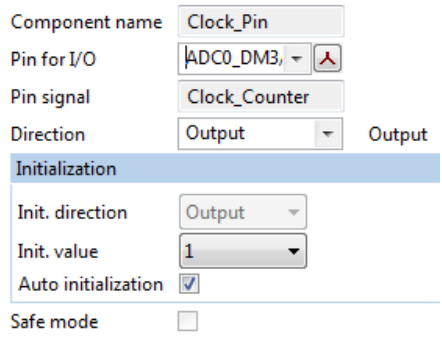


Ilustración 50 - *Processor Expert.* Inicialización *Clock_Counter*

Clock_Counter
Se ha utilizado el pin PTE23 del microcontrolador. Se configura como salida y se le asigna un valor de inicio de nivel alto.

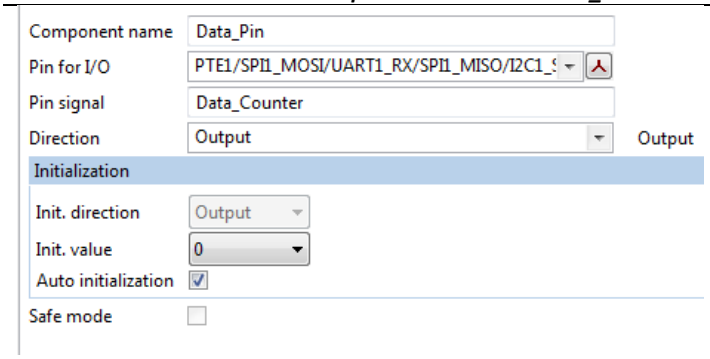
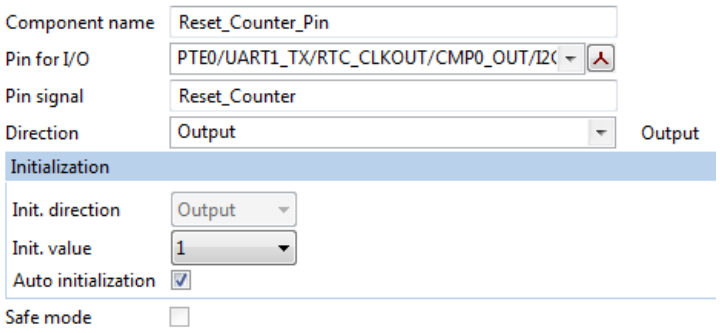


Ilustración 51 - *Processor Expert*. Inicialización *Data_Counter*

Data_Counter
Se ha utilizado el pin PTE1 del microcontrolador. Se configura como salida y se le asigna un valor de inicio de nivel bajo.



Reset_Counter
Se ha utilizado el pin PTE0 del microcontrolador. Se configura como salida y se le asigna un valor de inicio de nivel alto.

Ilustración 52 - *Processor Expert.* Inicialización *Reset_Counter*

Definiendo estas simples propiedades en los periféricos, quedan completamente configurados y listos para su uso.

4.6.2 Interrupción del *Clock*

La interrupción que se ejecutara cada 35µs estará dividida en dos partes: la aplicación del flanco de subida y la aplicación del flanco de bajada. Cada una de las partes estará separada por 35 µs y se ejecutaran distintas tareas necesarias para el control de la matriz. En el flanco de bajada prepararemos el dato para encender o apagar el led siguiente. En el flanco de subida cargaremos los valores anteriores en la matriz. Mediante esta separación también se consigue que no se colapse la ejecución en un ciclo por exceso de trabajo. En el siguiente flujo de programa podemos observar el funcionamiento.

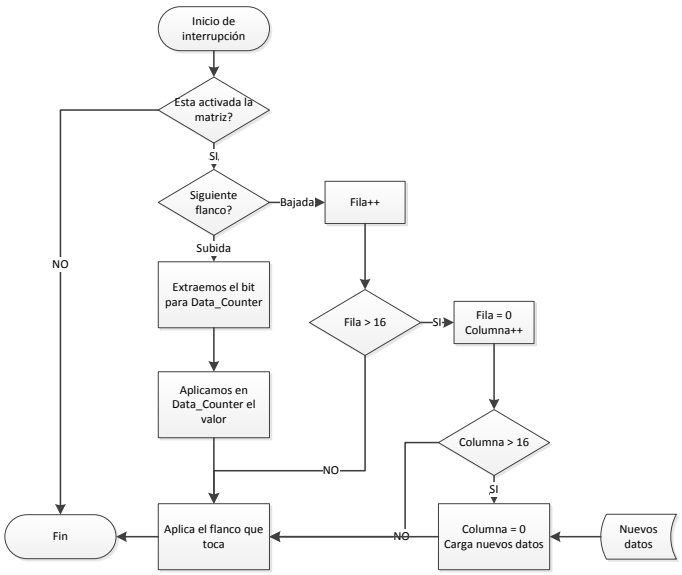


Ilustración 53 - Flujo de la interrupción de 35us

En el flujo de la interrupción los datos no se refrescan hasta que no se han procesado los 256 LEDs de la matriz. El proceso consiste en realizar un barrido mediante los contadores explicados en el hardware. Se sincronizan los pulsos de la señal *Clock_Counter* con las variables de control *Fila* y *Columna*. Cuando se desea activar el control de la matriz se reinician los contadores mediante la señal *Reset_Counter* y a su vez se inicializan las variables de control. De este modo se consigue una sincronización exacta entre software y hardware.

4.6.3 Simulación

Para la simulación se ha realizado un ejercicio sencillo. Hemos forzado en el software que se muestren en la matriz de LEDs distintos datos y se ha conectado el osciloscopio para comprobar que el resultado es el esperado. Las señales que se visualizan son: *D0* es *Data_Counter, D1* es *Clock_Counter* y *D2* es *Reset_Counter.*

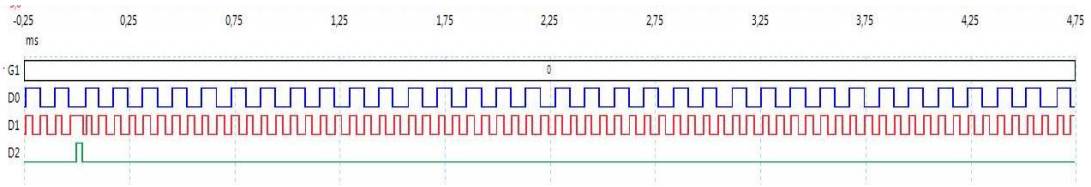


Ilustración 54 - Simulación datos tipo 1 en matriz, primer tramo

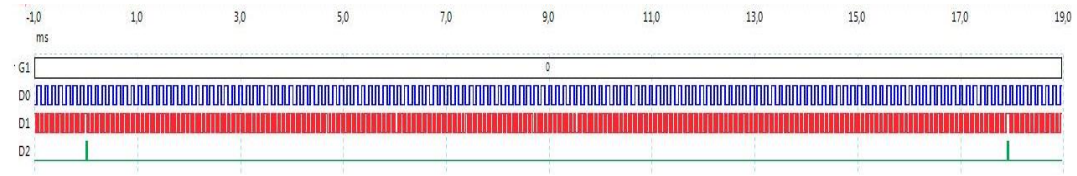


Ilustración 55 - Simulación datos tipo 1 en matriz, barrido completo

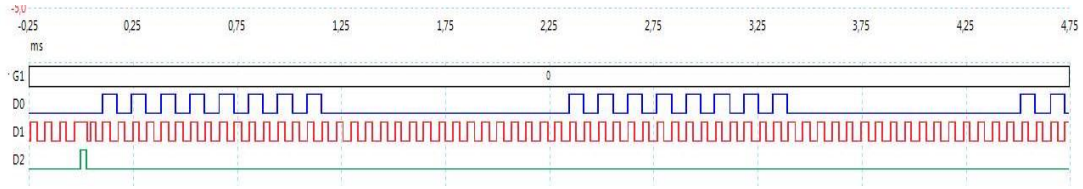


Ilustración 56 - Simulación datos tipo 2 en matriz, primer tramo

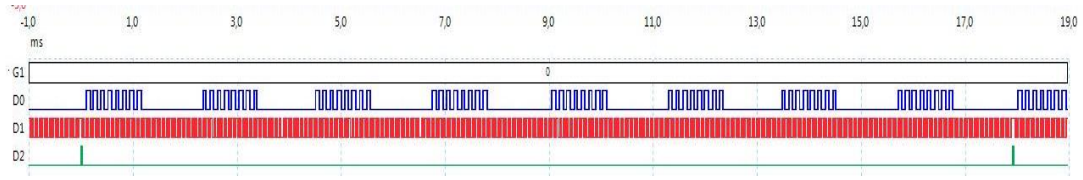


Ilustración 57 - Simulación datos tipo 2 en matriz, barrido completo

Simulación datos tipo 1: En la matriz de datos cargamos un valor 0xFFFFFFFF para las filas impares y de 0x00000000 para las pares. Como se puede observar, el resultado es que durante el barrido vamos alternando el valor 0 y 1 en el pin *Data_Counter* ya que como se ha comentado se selecciona una columna y luego se barren todas las filas.

Simulación datos tipo 2: En la matriz de datos cargamos un valor de 0x50505050 para las filas impares y de 0x05050505 para las filas pares. Como se puede observar, el resultado es que durante el barrido vamos mostrando el valor de *Data_Counter* de manera simétrica. Se han elegido precisamente estos datos para mostrar el correcto funcionamiento por su simetría.

Con estas dos simulaciones se deja claro el correcto funcionamiento del apartado de software dedicado al barrido de la matriz de LEDs. Además si se observan los tiempos queda claro que se cumplen los objetivos de refresco de la matriz de LEDs en los tiempos en que el ojo humano no es capaz de captar cambios. Recordemos que eran 20ms.

4.7 Control de la luz ambiente

La principal función de este módulo es la de calcular el valor optimo de *PWM* para aplicar al panel de LEDs. En resumen, cuando existe mucha luz ambiente el panel deberá funcionar con una alta luminosidad y por el contrario, cuando haya poca luz ambiente o sea nula, el panel deberá funcionar con una luminosidad baja.

Para conseguir este efecto el hardware consta con un medidor de luz ambiente. La luz se medirá mediante un amplificador de transconductancia conectado a una entrada del conversor analógico-digital del microcontrolador. También será necesario contar con una salida de *PWM* del microcontrolador. Este pin será el encargado de aplicar el nivel correcto de luminosidad al panel.

4.7.1 Periféricos para controlar la luz ambiente

Para el control de la luz ambiente se utilizaran dos periféricos del microcontrolador. Un canal del conversor analógico-digital y una salida de *PWM*.

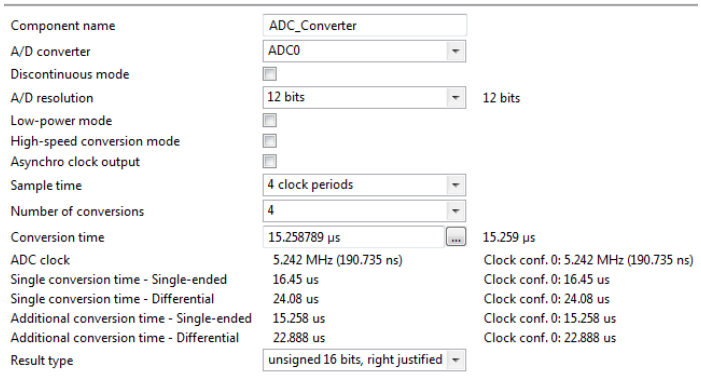


Ilustración 58 - *Processor Expert*. Inicialización *ADC_Converter*

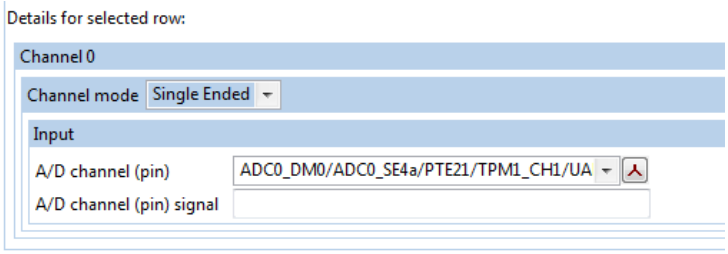


Ilustración 59 - *Processor Expert.* Selección del pin para *ADC_Converter*

ADC_Converter

Se ha utilizado el pin PTE21 del microcontrolador. Se ha seleccionado una resolución de 12 *bits* y un tiempo de conversión de aproximadamente 15µs.

También se ha configurado una interrupción al final de la medida del conversor. De este modo no es necesario estar a la espera de la finalización de la conversión A/D.

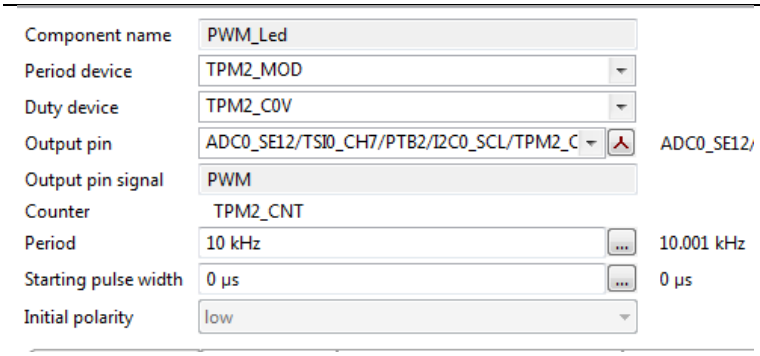


Ilustración 60 - *Processor Expert.* Inicialización de *PWM_Led*

PWM_Led

Se ha utilizado el pin PTB2 del microcontrolador. Se ha configurado con una frecuencia de 10KHz.

4.7.2 Desarrollo de la solución

Se ha desarrollado una máquina de estados en la cual se están realizando medidas de la luz ambiente cada 100 ms. Cada una de estas medidas se almacena en un *buffer* de ocho posiciones. Por último, se medían las ocho posiciones para tener una respuesta más lenta a los cambios de luz. El propósito de este algoritmo es que las variaciones de luz ambiente bruscas no hagan fluctuar la intensidad del display.

En cuanto al cálculo del valor de *PWM* que es necesario aplicar a los LEDs, se realiza una traslación mediante la siguiente función:

$$f(Valor_{Luz}) = \begin{cases} PWM_{Led} = 60735, & Valor_{Luz} < 300 \\ PWM_{Led} = 56000 - (Valor_{Luz} \ll 4), & 300 < Valor_{Luz} < 3500 \\ PWM_{Led} = 0, & 3500 < Valor_{Luz} \end{cases}$$

En primer lugar y para poder explicar esta función, hay que aclarar que el 100% del *PWM* o lo que es lo mismo, la máxima corriente de excitación de los LEDs se consigue con la carga del valor 0 en *PWM_Led*. Por consiguiente, con 65535 se tendrá la mínima corriente de excitación de los LEDs. También se debe tener en cuenta, que el valor medido en el conversor es un valor de 12 *bits.* Por este motivo se realiza la rotación de 4 posiciones a la izquierda en los cálculos para eliminar los valores mínimos e igualarlo a 16 *bits* como el *PWM_Led.*

La función propuesta tiene tres rangos de valores:

- Cuando la luz ambiente es menor de 300 puntos del conversor significa que no existe apenas luz por lo que no será necesario aplicar una corriente elevada al led para que sea visible. En este punto aplicaremos el valor mínimo para que los LEDs luzcan con su mínima intensidad.

$$60735 = 300 \ll 4 \ \rightarrow 8mA$$

- Cuando la luz ambiente es superior a 300 e inferior a 3500 puntos (equivalente a máxima luz ambiente en interior) aplicaremos la siguiente ecuación:

$$PWM_{Led} = 56000 - (Valor_{Luz} \ll 4)$$

$$56000 = \ 3500 \ll 4 \ \rightarrow Desde\ 8mA\ hasta\ 100mA$$

- Por último, cuando la luz ambiente supera el valor 3500 aplicaremos la máxima intensidad a los LEDs

4.7.3 Simulación de la solución

Como en apartados anteriores, para realizar la simulación se ha utilizado la tarjeta *Freedom KL25* de *Freescale.* Ha esta tarjeta se le ha conectado un potenciómetro en la entrada A/D correspondiente. Con este sistema se simula la lectura de la luz ambiente. Se ha utilizado el Osciloscopio de PC para capturar las dos señales correspondientes: la entrada analógica y la salida de *PWM_Led.*

En todas las mediciones se han añadido medidas de referencia como por ejemplo la tensión media proporcionada por la señal de *PWM,* el porcentaje de *PWM* aplicado y los valores de medida mínimos y máximos de cada uno de los canales capturados.

Por último se han realizado tres simulaciones:

- Sin nivel de luz (0V).
- Nivel de luz bajo (975mV).
- Nivel de luz alto pero sin saturar (2,5V).

A continuación los resultados obtenidos:

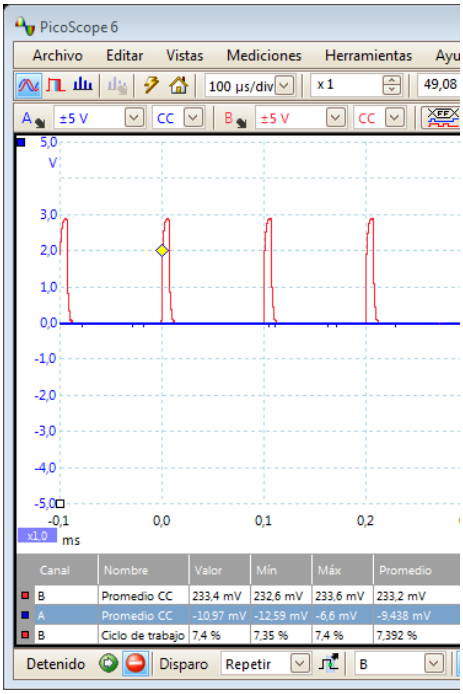


Ilustración 61 - Simulación luz ambiente. Luz nula.

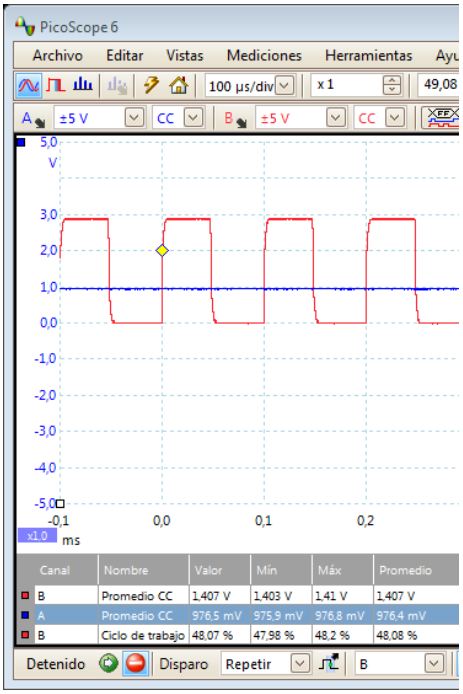


Ilustración 62 - Simulación luz ambiente. Luz muy débil.

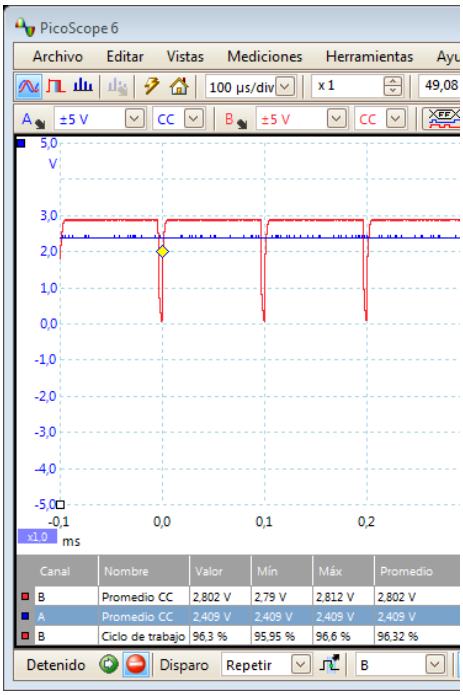


Ilustración 63 - Simulación de luz ambiente. Luz de interior máxima

5 Industrialización de la solución

5.1 Esquema eléctrico del TFC

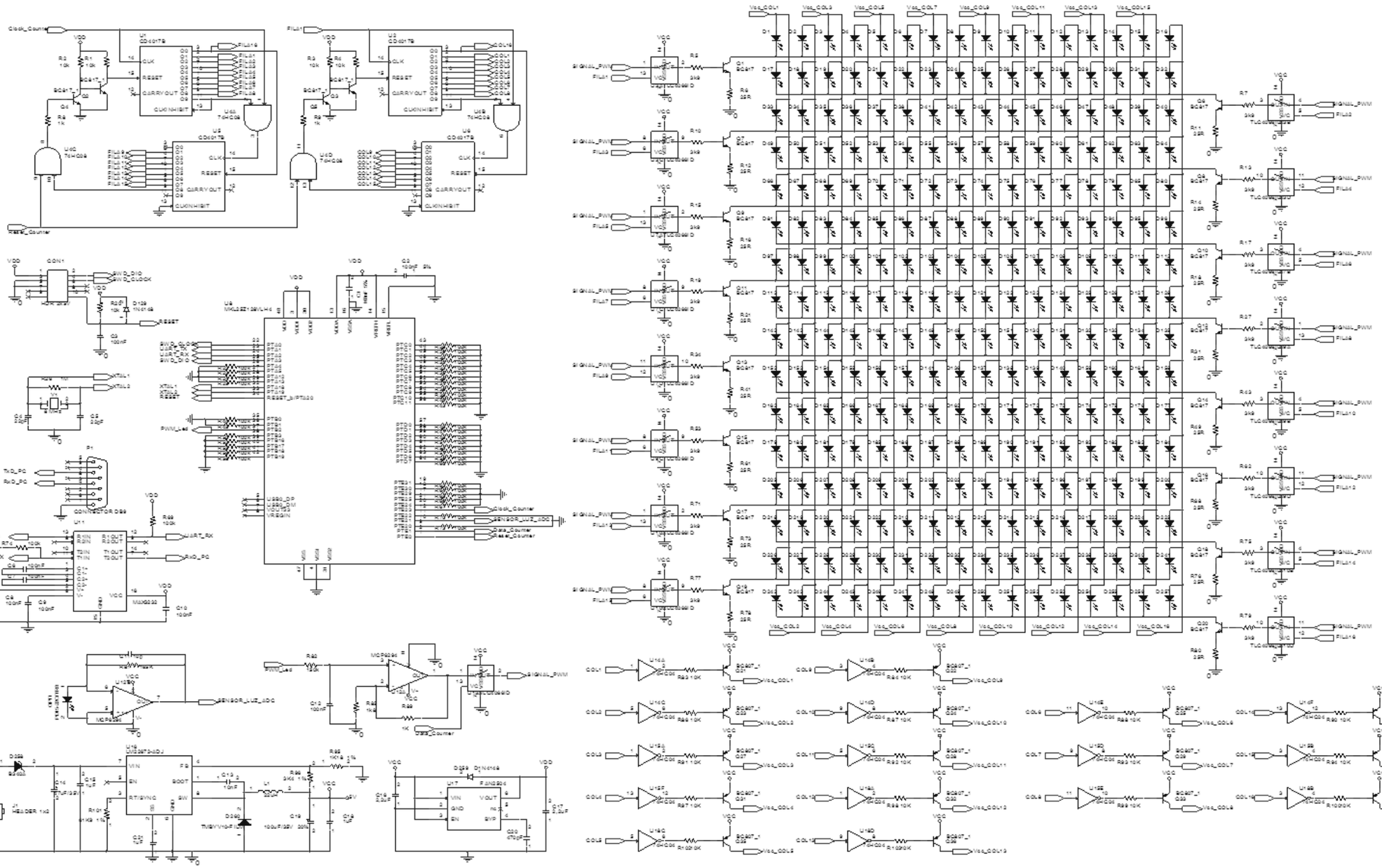


Ilustración 64 - Esquema eléctrico final del TFC

5.2 Ruteado de la PCB

Para realizar el ruteado de la PCB o *layout* final se ha utilizado la herramienta *Cadstat 13*. Se ha partido del esuqema electrico dibujado en *Orcad* y exportado en formato *Rinf*. El *layout* se ha realizado en dos capas. En la capa superior o *top* se han situado los componentes. En ambas caras, superior e inferior, se han añadido planos de masa. Alrededor de las fuentes de alimentación se han añadido planos de las señales de potencia o alimentacion. En ambos casos, el proposito de estos planos de cobre es el de prevenir emisiones electromagneticas de las pistas. Con estos planos cerramos las lineas de flujo magnetico de las pistas. El resultado es el que se muestra a continuación:

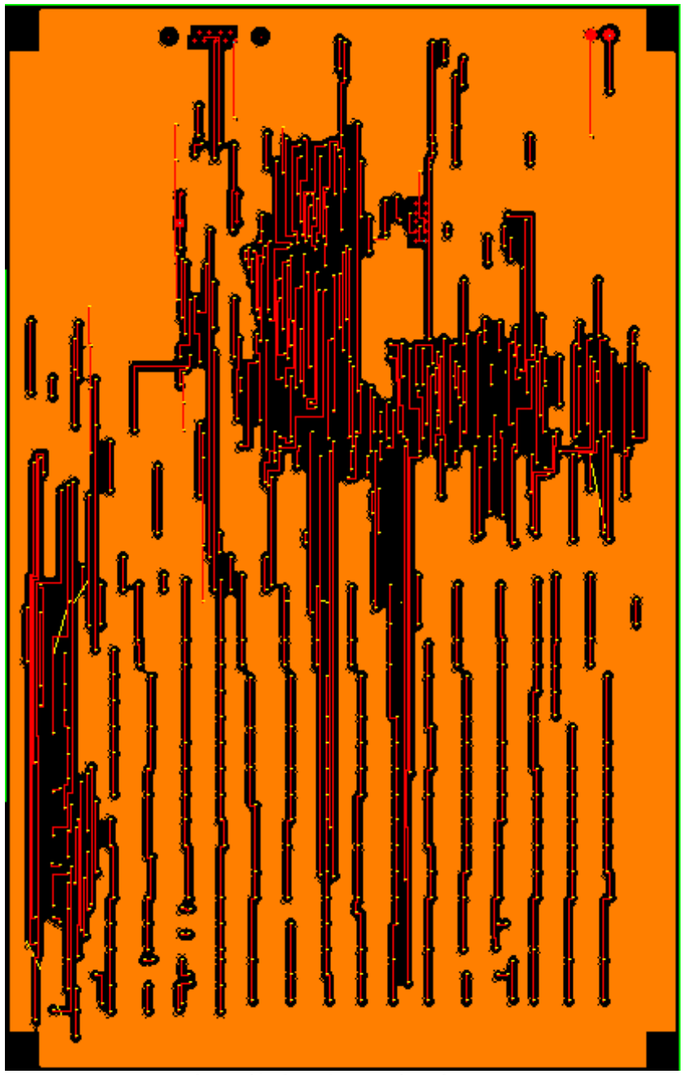


Ilustración 65 - PCB cara pistas *botton*

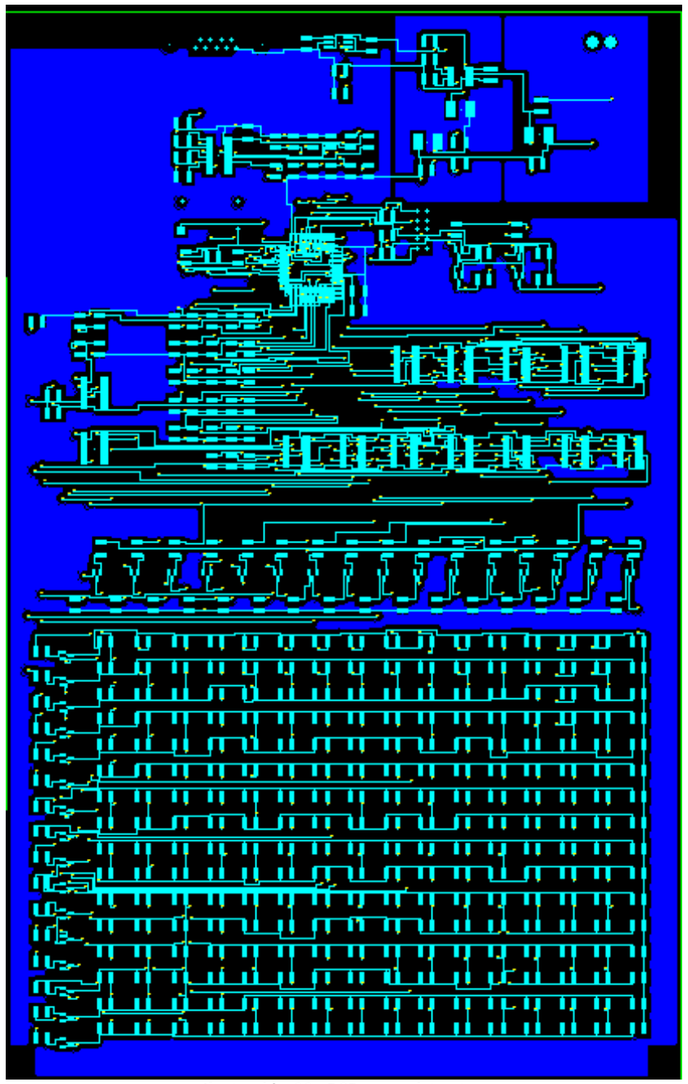


Ilustración 66 - PCB cara pistas *top*

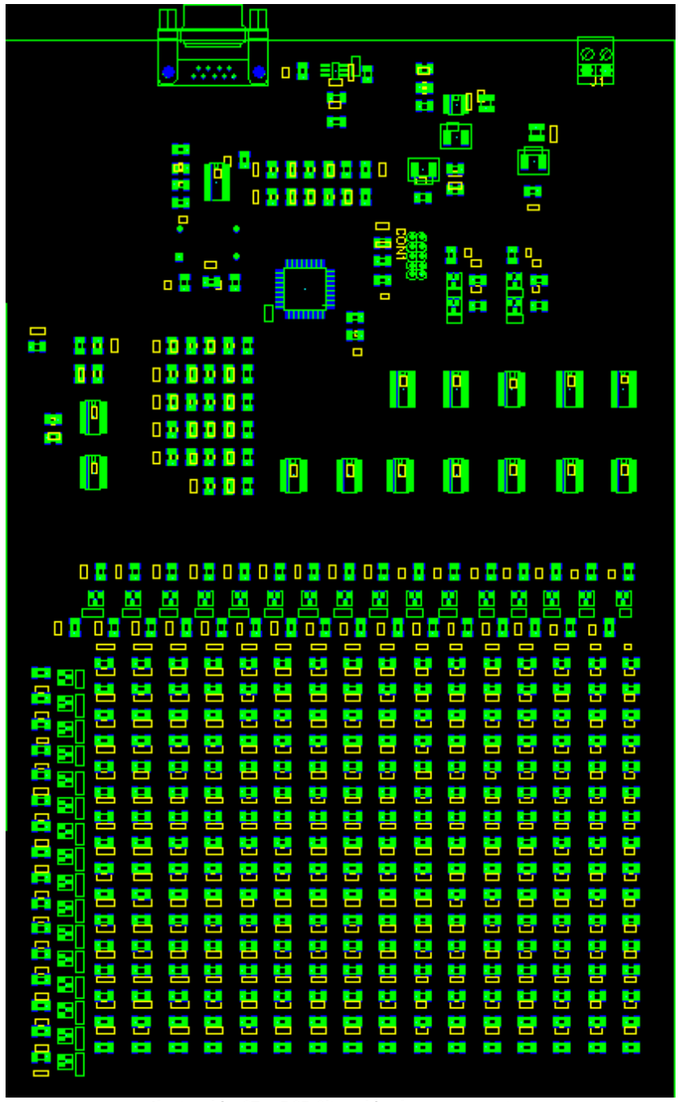


Ilustración 67 - PCB Situación de componentes

5.3 Listado de componentes

Ítem	Cantidad	Referencia	Denominación	Precio	Total
1	1	CON1	HDR 2x5V	0,882€	0,882 €
3	9	C1,C2, C3,C6,C7,C8,C9,C10,C12	100nF 1206	0,067€	0,603 €
4	2	C4,C5	22pF 1206	0,039€	0,078 €
5	1	C11	10pF 1206	0,226€	0,226 €
6	1	C13	10nF 1206	0,061€	0,061 €
7	1	C14	47uF/35V Radial	0,70€	0,700 €
8	3	C15,C18,C21	1uF 1206	0,052€	0,156 €
9	2	C16,C17	2,2uF 1206	0,15€	0,300 €
10	1	C19	100uF/35V Radial	0,70€	0,700 €
11	1	C20	470pF 1206	0,115€	0,115 €
12	256	D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15,D16,D17,D18,D19,D20,D21,D22,D23,D24,D25,D26,D27,D28,D29,D30,D31,D32,D33,D34,D35,D36,D37,D38,D39,D40,D41,D42,D43,D44,D45,D46,D47,D48,D49,D50,D51,D52,D53,D54,D55,D56,D57,D58,D59,D60,D61,D62,D63,D64,D65,D66,D67,D68,D69,D70,D71,D72,D73,D74,D75,D76,D77,D78,D79,D80,D81,D82,D83,D84,D85,D86,D87,D88,D89,D90,D91,D92,D93,D94,D95,D96,D97,D98,D99,D100,D101,D102,D103,D104,D105,D106,D107,D108,D109,D110,D111,D112,D113,D114,D115,D116,D117,D118,D119,D120,D121,D122,D123,D124,D125,D126,D127,D128,D130,D131,D132,D133,D134,D135,D136,D137,D138,D139,D140,D141,D142,D143,D144,D145,D146,D147,D148,D149,D150,D151,D152,D153,D154,D155,D156,D157,D158,D159,D160,D161,D162,D163,D164,D165,D166,D167,D168,D169,D170,D171,D172,D173,D174,D175,D176,D177,D178,D179,D180,D181,D182,D183,D184,D185,D186,D187,D188,D189,D190,D191,D192,D193,D194,D195,D196,D197,D198,D199,D200,D201,D202,D203,D204,D205,D206,D207,D208,D209,D210,D211,D212,D213,D214,D215,D216,D217,D218,D219,D220,D221,D222,D223,D224,D225,D226,D227,D228,D229,D230,D231,D232,D233,D234,D235,D236,D237,D238,D239,D240,D241,D242,D243,D244,D245,D246,D247,D248,D249,D250,D251,D252,D253,D254,D255,D256,D257	LS_T67K-TYP PLCC2	0,09€	23,040 €
13	1	D129, D259	1N4148 1206	0,033€	0,033 €
14	1	D258	B340A	0,128€	0,128 €
16	1	D260	TMBYV10-FILM	0,265€	0,265 €
17	1	J1	HEADER 1x2		0,000 €
18	1	L1	22uH SMD	2,06€	2,060 €
19	1	OPT1	PD15-22C/TR8	0,23€	0,230 €
20	1	P1	CONNECTOR DB9	1,3€	1,300 €
21	20	Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10,Q11,Q12,Q13,Q14,Q15,Q16,Q17,Q18,Q19,Q20	BC817 SOT23	0,025€	0,500 €
22	16	Q21,Q22,Q23,Q24,Q25,Q26,Q27,Q28,Q29,Q30,Q31,Q32,Q33,Q34,Q35,Q36	BC807 SOT23	0,039€	0,624 €
23	21	R1,R2,R3,R4,R20,R83,R84,R86,R87,R88,R90,R91,R92,R93,R94,R97,R98,R99,R100,R102,R103	10K 1206	0,003€	0,063 €
24	16	R5,R7,R10,R13,R15,R17,R19,R27,R34,R43,R53,R62,R71,R75,R77,R79	3k9 1206	0,003€	0,048 €
25	16	R6,R11,R12,R14,R16,R18,R21,R31,R41,R49,R61,R68,R73,R76,R78,R80	25R 1206	0,003€	0,048 €
26	3	R8,R9,R89	1K 1206	0,003€	0,009 €
27	40	R22,R23,R24,R25,R28,R29,R30,R32,R33,R35,R36,R37,R38,R39,R40,R42,R44,R45,R46,R47,R48,R50,R51,R52,R54,R55,R56,R57,R58,R59,R60,R63,R64,R65,R66,R67,R69,R70,R72,R74	100k 1206	0,003€	0,120 €
28	1	R26	1M 1206	0,003€	0,003 €
29	1	R81	165K 1206	0,03€	0,030 €
30	1	R82	150k 1206	0,03€	0,030 €
31	1	R85	1k8 1206	0,03€	0,030 €
32	1	R95	1K18 1% 1206	0,04€	0,040 €
33	1	R96	3K4 1% 1206	0,04€	0,040 €

34	1	R101	61K9 1% 1206	0,04€	0,040 €
35	4	U1,U2,U5,U6	CD4017B SOP16	0,159€	0,636 €
36	5	U3,U7,U9,U10,U13	TLC4066ID SOP14	0,198€	0,990 €
37	1	U4	74HC08 SOP14	0,271€	0,271 €
38	1	U8	MKL25Z128VLH4 LQFP64	0,90€	0,90€
39	1	U11	MAX3232 SOP16	1,236€	1,236 €
40	1	U12	MCP6294 SOP14	1,23€	1,230 €
41	3	U14,U15,U18	74HC04 SOP14	0,244€	0,732 €
42	1	U16	LM22672-ADJ	1,84€	1,840 €
43	1	U17	FAN2504 SOP8	0,403€	0,403 €
44	1	Y1	8 MHz	2,42€	2,420 €
***Todos los precios se han obtenido de RS *Amidata* y se pueden obtener mucho más baratos en otros distribuidores.**			Total		43,14€

6 Concluciones

Se puede asegurar que durante la planificación y ejecucion del TFC se han conseguido todos los objetivos marcados. Durante este tiempo tambien se han sufrido desviaciones sobre lo planificado por motivos tecnicos y por falta de tiempo. Los principales motivos de las desviaciones han sido:

- Mala selección del microprocesador en el apartado del hardarwe. En el seleccionado en primer lugar no cabia el firmware al añadir las tablas de traslación de caracteres.
- Problemas familiares de salud que han ido robando tiempo poco a poco.

Para mitigar estas desviaciones se tomaron medidas de contingencia como la mayor dedicacion durante los fines de semana y el uso de herramientas mas conocidas como Orcad y Codewarrior.

En cuanto al aspecto tecnico se tomo una solucion hardware ligeramente atrevida. Se decidio esta via de trabajo porque en cuanto a lo que al desarrollo de firmware se refiere, ya se tiene un alto nivel y se deseaba realizar un esfuerzo en la solucion del hardware. Durante el diseño del hardware se encontraron dificultades sobre todo en el control de las corrientes de los LEDs y en las simulaciones. Hay que tener muy claro cual es el objetivo final para poder ir definiendo poco a poco las diferentes etapas del hardware. Estoy encantado con la solucion hardware que se ha conseguido. Con solamente cuatro lineas se ha conseguido el manejo de los 256 LEDs.

Otro de los retos del desarrollo ha sido la industrializacion de la solucion final. Para el diseño de la *PCB* se pensaba realizar todo el proceso con *Proteus* pero al final se decidio realizarlo con Orcad y con Cadstart por problemas en la definicion de huellas y en la simulacion con PsPice. Este es el punto mas debil del desarrollo. En mi opinion se debe trabajar mas a fondo el ruteado de las pistas.

Me hubiera gustado montar un prototipo con la solucion final pero la falta de tiempo no me lo permitió. Como lineas de trabajo futuras se piensa en dividir el diseño en dos bloques: la fuente de alimentacion y el microprocesador, y por otra parte el control de la matriz de LEDs. De este modo se podrá ampliar la solucion a *n* matrices de LEDs que seran controladas por un solo microcontrolador.

Para finalizar, se ha disfrutado mucho realizando el TFC. Como ya se ha comentado, sobre todo en la parte del hardware que ha supuesto un reto. En mi opinion, ha sido bastante ingeniosa. Se podia haber optado por montar dispositivos *I2C* y cargar todo el peso del desarrollo en la parte del firmware, pero con una buena solucion hardware como esta se eliminan los problemas de comunicaciones y las latencias entre comandos, ademas de una uniformidad en la luminosidad de la matriz ya que todos los LEDs estan alimentados por la misma fuente de corriente.

Una gran experiencia que enriquece los estudios cursados en la UOC. Gracias.

7 Bibliografía

Photodiode/Phototransistor Application Circuit. (1999). Obtenido de Sharp: http://physlab.lums.edu.pk/images/1/10/Photodiode_circuit.pdf

ALS-PD70-01C-TR7.pdf. (2014). Obtenido de Everlight: http://www.everlight.com/file/ProductFile/201407061524148948.pdf

BC817.pdf. (2014). Obtenido de NXP: http://www.nxp.com/documents/data_sheet/BC817_BC817W_BC337.pdf

CD4017B. (2014). Obtenido de Texas Instruments: http://www.ti.com/lit/ds/symlink/cd4017b.pdf

CD4066.pdf. (2014). Obtenido de Texas Instruments: http://www.ti.com/lit/ds/symlink/cd4066b.pdf

Information family MKL25Z128.pdf. (2014). Obtenido de Freescale: http://cache.freescale.com/files/32bit/doc/ref_manual/KL25P80M48SF0RM.pdf?fasp=1&WT_TYPE=Reference%20Manuals&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation&fileExt=.pdf

LS T67K - TOPLED.pdf. (2014). Obtenido de Osram: http://www.osram-os.com/Graphics/XPic4/00099844_0.pdf/LS%20T67K%20-%20TOPLED.pdf

Max3232.pdf. (2014). Obtenido de Maxin: http://datasheets.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf

MCP6291.pdf. (2014). Obtenido de Microchip: http://ww1.microchip.com/downloads/en/DeviceDoc/21812e.pdf

Reference manual MKE04Z8VTG4.pdf. (2014). Obtenido de Freescale: http://cache.freescale.com/files/microcontrollers/doc/ref_manual/MKE04P24M48SF0RM.pdf

BC807.pdf. (s.f.). Obtenido de NXP: http://www.nxp.com/documents/data_sheet/BC807_BC807W_BC327.pdf

Coughlin, R. F. (1993). *Amplificadores operacionales y circuitos integrados lineales.*

Díaz, A. B. (s.f.). *Gestión y desarrollo de proyectos.* UOC.

FAN2504.ped. (s.f.). Obtenido de Fairchild: https://www.fairchildsemi.com/datasheets/FA/FAN2504.pdf

Guerrero, F. T. (s.f.). *Nociones básicas de transistores.*

LM22672.pdf. (s.f.). Obtenido de Texas instruments: http://www.ti.com/lit/ds/symlink/lm22672.pdf

Modbus.org. (s.f.). Obtenido de Modbus: http://www.modbus.org/tech.php

Montagut, R. B. (s.f.). *Presentación de documentos y elaboración de presentaciones.* UOC.

Pacheco Navas, C. A. (2005). PlanTrabajo_ejemplo.pdf. UOC.

Pacheco Navas, C. A. (2014). 89038_20141_AplicacionsElectro_cpacheco.pdf. UOC.

Sáenz Higueras, N., & Vidal Oltra, R. (s.f.). *Redacción de textos científico-técnicos.* UOC.