UNIVERSITAT OBERTA DE CATALUNYA

Ingeniería Técnica de Telecomunicaciones

(Especialidad en Telemática)

DISEÑO DE UN PANEL DE LEDES

Alumno/a: Marta Pérez Romero

Dirigido por: Carlos Alberto Pacheco Navas

CURSO 2014-15 (Septiembre/Enero)

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE ECUACIONES

1. PRESENTACIÓN DEL PROYECTO

El Trabajo Final de Carrera (TFC) constituye un repaso completo a tres cursos académicos dentro de la Ingeniería Técnica de Telecomunicaciones. En el área que nos ocupa, "Aplicaciones electromagnéticas y electrónicas", nuestros esfuerzos se centrarán en la rama de Electrónica.

A través de este TFC pondremos en práctica conocimientos que hemos ido adquiriendo a lo largo de la carrera. Especial relevancia adquieren, en este sentido, asignaturas como "Física", "Sistemas electrónicos digitales" o "Teoría de circuitos". El proyecto no se limita a revisar contenidos teóricos, sino que nos obligará a hacer uso de ellos a partir del diseño de un prototipo concreto, un panel informativo compuesto por ledes[1]. Igualmente, deberemos contextualizarlo dentro de una memoria global que incluirá un plan de proyecto, su desarrollo y sus conclusiones finales.

Si bien este aspecto está fuera del alcance del proyecto, también hemos construido físicamente el dispositivo. El paso del prototipo al montaje final hace que el trabajo adquiera una nueva dimensión, puesto que convierte en tangibles las distintas fases del diseño y afianza los conocimientos. A lo largo del documento se irán explicando en paralelo ambos modelos, incidiendo en cualquier diferencia de concepto o implementación que pueda haber entre el prototipo y el equipo físico.

Por otro lado, hemos estructurado el TFC en distintos apartados: después de una breve introducción y un repaso por la planificación, entraremos en explicaciones teóricas sobre la tecnología led y seleccionaremos el tipo de led que más nos interesa para el panel. Acto seguido, diseñaremos los circuitos de adaptación de las señales, que incluyen el sensor de luz, el canal serie RS232 y los *drivers* para controlar el encendido y apagado del sistema. Tras la explicación de los aspectos mecánicos, elegiremos un microprocesador y lo programaremos para que desarrolle las tareas oportunas. El circuito se completa con el diseño de la fuente de alimentación. Por último, crearemos la placa de circuito impreso (PCB), que materializará el panel de ledes.

1.1. INTRODUCCIÓN

En el mundo actual, queremos estar permanentemente informados de todo aquello que nos afecta a la vida cotidiana. Cada vez más valoramos el tiempo y es por ello que, especialmente, nos interesa saber los tiempos de espera previstos de las infraestructuras que utilizamos. Por ejemplo, nos interesa conocer los tiempos de espera en las paradas de autobús, o de metro, o de cola para hacer un trámite, etc.

Existen incluso aplicaciones móviles para poder tener acceso a esa información en remoto, pero dado que no todo el mundo posee aún acceso a las mismas, sigue siendo muy habitual tener paneles informativos que nos dan esa información. Los hay de muchos formatos y tecnologías, pero para este proyecto nos vamos a centrar en los paneles de ledes, como los que nos encontramos en marquesinas de autobús, metro, etc.

Los ledes son hoy en día un recurso muy utilizado para infinidad de aplicaciones. Precisamente el día 7 de octubre de 2014 se otorgaba a tres físicos japoneses el Premio Nobel de Física por su trabajo: "*La*

[1] Utilizamos "led" y "ledes" ya que son las formas normativas aceptadas por la Real Academia Española de la Lengua.

invención de los diodos emisores de luz azul que han permitido las fuentes de luz brillantes y de ahorro energético"[2]. Esto nos da una idea de la relevancia de esta tecnología.

Nuestro proyecto se centra en diseñar un panel de ledes que sea capaz de mostrar cualquier información y que, al mismo tiempo, lo haga de una manera eficiente, minimizando el consumo eléctrico del equipo.

1.1.1. Diseño conceptual de la aplicación

Para el diseño del panel de ledes combinaremos una serie de elementos, que sin entrar en detalles técnicos, serán principalmente los siguientes:

- Controlador: es el dispositivo que lleva todo el control del encendido/apago de los ledes y de la interfaz con el equipo maestro que proporciona la información que se ha de mostrar por pantalla.
- Ledes y drivers asociados: los ledes son el elemento que mediante su encendido y apagado mostrará en el panel la información deseada. Será necesario, además, escoger los drivers más adecuados para minimizar el consumo eléctrico.
- Sensor de luz: servirá para regular la luminosidad de los ledes y adaptarla al nivel de luz ambiente.
- Alimentación: elementos que permiten trabajar con alimentaciones habituales y adaptarlas a las necesidades concretas del equipo.

1.1.2. Diagrama de bloques

El sistema se diseñará a nivel de bloques tal y como se muestra en la siguiente figura:

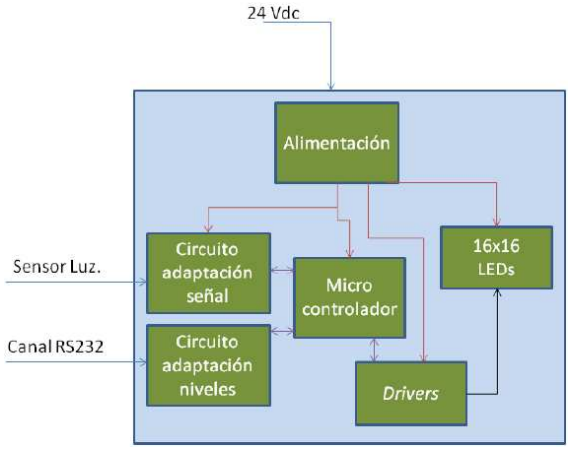


Figura 1. Diagrama de bloques del equipo

La tensión de alimentación será de $24\ Voltios$ de continua. Por su parte, el sensor de luz medirá la luz ambiental para ajustar la luminosidad de los ledes. El sistema recibirá la información que se quiere mostrar a través de una entrada de canal serie RS232.

El comportamiento del sistema será, en resumen, el siguiente: el equipo estará continuamente monitorizando la señal del sensor de luz, controlando el encendido y apagado de los ledes a nivel individual en función de la información que se quiere mostrar y supervisando la entrada del canal RS232 para aceptar un nuevo mensaje cuando llegue.

[2] Fuente: http://www.nobelprize.org/nobel_prizes/physics/laureates/2014/

1.2. OBJETIVOS Y COMPETENCIAS

Tal y como hemos indicado en la Introducción, la realización de este TFC nos llevará a la consecución de diversos objetivos y competencias. Todos ellos son evaluables y medibles, por lo que a la finalización del proyecto podrá comprobarse su grado de alcance:

Competencias generales

- Comunicación escrita: comunicarse con facilidad por escrito, estructurando el contenido del texto y los apoyos gráficos para facilitar la comprensión del lector en escritos de extensión mediana.

Competencias específicas

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

Objetivos generales

- Aplicar e integrar los conocimientos adquiridos en las diferentes materias de la titulación.
- Adquirir una dimensión práctica y tangible sobre el trabajo con prototipos electrónicos.

Objetivos específicos

- Diseñar un panel de ledes.
- Diseñar la fuente de alimentación y los circuitos de adaptación de las entradas y salidas a niveles del microcontrolador y simular/hacer mediciones de diferentes circuitos con parámetros diversos.
- Programar un microcontrolador.
- Implementar el protocolo MODBUS para la comunicación serie sobre canal RS232.
- Diseñar la placa de circuito impreso (PCB).
- Adquirir conocimientos prácticos de los componentes electrónicos implicados en el diseño del panel.

1.3. PLANIFICACIÓN

1.3.1. Calendario de trabajo

El TFC se desarrollará entre el 17/09/2014 (fecha de inicio del semestre) y el 30/01/2015 (fecha de fin del debate virtual). Disponemos pues, de unos cuatro meses, durante los cuales habrá que tener en cuenta las fechas de entrega del proyecto y de otra asignatura con evaluación continua.

Visualmente el calendario se estructurará de la siguiente manera:

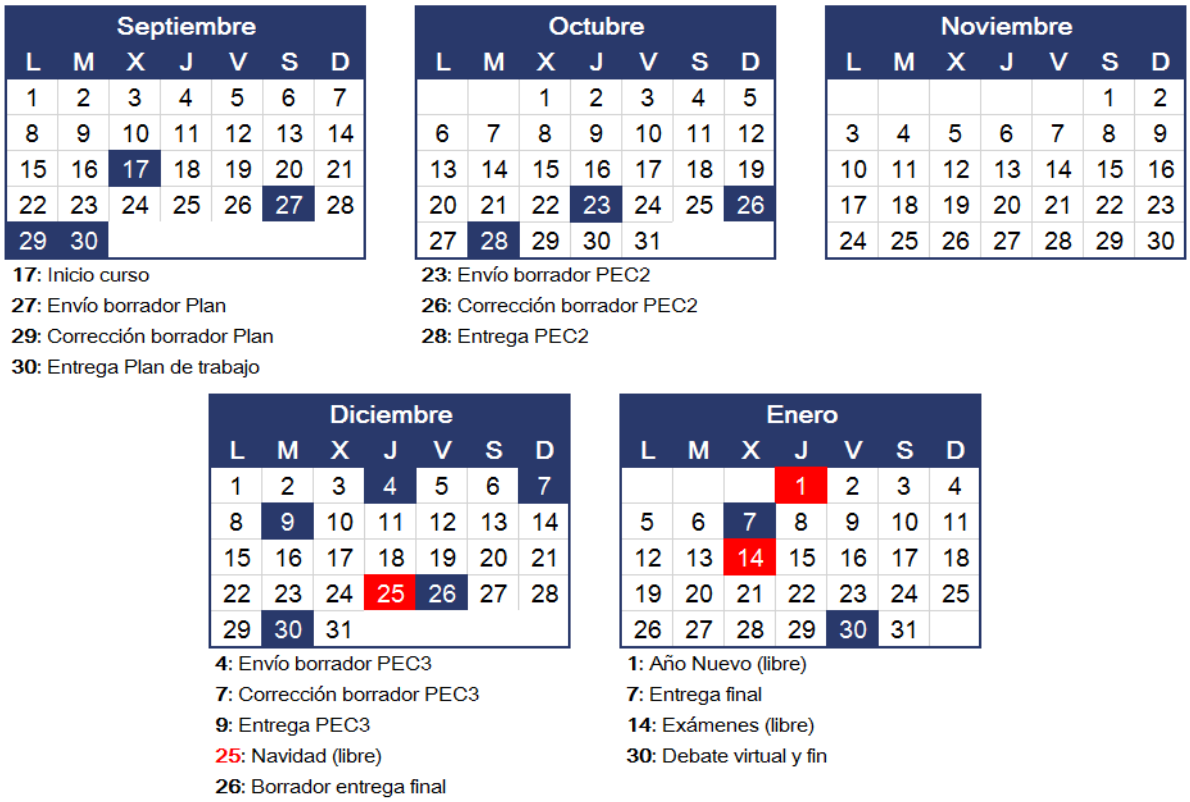


Figura 2. Calendario del proyecto

Se ha previsto una dedicación completa al proyecto a excepción de los días de Navidad y Año Nuevo.

Teniendo en cuenta lo anterior, la disponibilidad diaria entre semana será de 4 horas y de 3 en fin de semana, tal y como queda reflejado en la tabla inferior. El número de días se ha calculado desde el 17/09/14 hasta el 7/01/15. Del 7/01/15 al 28/01/15 no serán horas computables, al estar el Tribunal evaluando. Igualmente, la suma total corresponde a la suma de los días por el número de horas más 6,5 horas que se dedicarán entre el 28/01/15 y el 30/01/15 a la lectura, elaboración y envío de las preguntas del Debate Virtual.

Día Semana	Nº Días	Nº Horas	Horas totales
Lunes	16	4	64
Martes	16	4	64
Miércoles	15	4	60
Jueves	14	4	56
Viernes	16	4	64
Sábado	16	3	48
Domingo	16	3	48
		Total	410,5

Tabla 1. Horas de trabajo disponibles (estimado)

1.3.2. Hitos

Los hitos corresponden a las entregas parciales, tanto borrador como versión final, y a la entrega de la memoria. Añadimos también como hitos la fecha de inicio del proyecto y del debate virtual:

Grupo	Hito	Fecha
	Inicio del curso	17/09/2014
PEC1	Envío del borrador al consultor	27/09/2014
	Elaboración del documento definitivo (correcciones)	29/09/2014
	Entrega de la PEC1: Plan de trabajo	30/09/2014
PEC2	Envío borrador PEC2 al consultor	23/10/2014
	Revisión de comentarios del consultor y sus correcciones	26/10/2014
	Entrega de la PEC2: Primera parte del TFC	28/10/2014
PEC3	Envío borrador PEC3 al consultor	04/12/2014
	Revisión de comentarios del consultor y sus correcciones	07/12/2014
	Entrega de la PEC3: Segunda parte del TFC	09/12/2014
Entrega final	Envío del borrador memoria al consultor	27/12/2014
	Revisión de comentarios del consultor y sus correcciones	30/12/2014
	Entrega final: memoria, archivos y vídeo-presentación	07/01/2015
Debate virtual	Envío de las respuestas al Tribunal de evaluación	30/01/2015

Tabla 2. Hitos

1.3.3. Planificación de tareas

Considerando el enunciado del proyecto, el calendario, las horas de trabajo disponibles y las fechas de entrega, se ha desglosado el desarrollo del trabajo en las siguientes tareas y actividades. Mostramos las fechas de realización, su duración estimada en horas y sus predecesoras.

Igualmente, incluimos la carga de trabajo por cada una de las tareas (a nivel global), en función del tiempo requerido. De esta manera, podemos calcular que la PEC2 supondrá un 24% del total, mientras que para la PEC3 dedicaremos un 44% del tiempo (tendrá más peso por incluir todo el tiempo dedicado a la programación). Debemos indicar que se han distribuido las tareas de manera coherente, y que los porcentajes cuadran a nivel global con la temporización general y con las fechas de entrega.

Por último, hemos de señalar que este plan estima unas 315 horas de trabajo, algo que es posible teniendo en cuenta que en el calendario de disponibilidad tenemos un total de 410,5. En la sección de “Incidencias personales” veremos que se han reservado deliberadamente para disponer de al menos un 20% de tiempo de margen en caso de ser necesario.

En las próximas tres páginas podemos encontrar la lista completa de tareas, donde además de lo previamente citado, hemos añadido una breve descripción de cada grupo de tareas.

Nota: cada tarea viene identificada por un "Id. de tarea”, además de mantener una nomenclatura propia en función del grupo y subgrupo de tareas al que pertenece. El fin de este sistema es garantizar la consistencia respecto al modo en que *Microsoft Project* numera las tareas.

Id. Tarea		APARTADO / TAREA	HORAS DEDICADAS	INICIO	FIN	PREDECESORAS	DESCRIPCIÓN / CARGA HORAS
1	1	TRABAJO DE FIN DE CARRERA	315,5	17/09/14	30/01/15		100,00%
2	2	DEFINICIÓN Y CONTEXTUALIZACIÓN DEL TFC		17/09/14	22/09/14		3,02%
3	2.1	**Lectura del enunciado del proyecto**	2	17/09/14	17/09/14		Primer contacto con el enunciado del proyecto, resolución de dudas y definición global de la metodología a partir de los ejemplos proporcionados.
4	2.2	**Lectura pormenorizada del enunciado y extracción de dudas**	3	18/09/14	18/09/14	3	
5	2.3	**Video conferencia de inicio con el consultor**	2	20/09/14	20/09/14		
6	2.4	**Lectura del plan de trabajo de ejemplo**	2	21/09/14	21/09/14	5	
7	2.5	**Revisión del documento resumen de la reunión**	2	22/09/14	22/09/14	5	
8	3	PEC1: PLAN DE TRABAJO		23/09/14	30/09/14		7,41%
9	3.1	**Definición del guión del plan de trabajo**	**4**	**23/09/14**	**23/09/14**	**6,7**	Creación del plan del proyecto, definición de objetivos, planificación completa, recursos necesarios para su desarrollo y establecimiento de líneas de trabajo y planes de contingencia. Al finalizar este grupo de tareas estaremos listos para el inicio del desarrollo del proyecto.
10	3.2	**Borrador secciones: primera aproximación**		**24/09/14**	**26/09/14**		
11	3.2.1	Presentación e introducción	2	24/09/14	24/09/14	9	
12	3.2.2	Objetivos y competencias	2	24/09/14	24/09/14	9	
13	3.2.3	Estructura TFC	1	25/09/14	25/09/14	9	
14	3.2.4	Planificación: calendario, hitos y tareas	3	25/09/14	25/09/14	9	
15	3.2.5	Diagrama de Gannt y análisis DAFO	1	25/09/14	25/09/14	9	
16	3.2.6	Materiales y bibliografía	2	26/09/14	26/09/14	9	
17	3.2.7	Riesgos y plan de contingencia	2	26/09/14	26/09/14	9	
18	3.3	**Redacción final del borrador**		**27/09/14**	**27/09/14**		
19	3.3.1	Revisión por sección y mejoras	2	27/09/14	27/09/14	10	
20	3.3.2	Envío del borrador al consultor	0,5	27/09/14	27/09/14	10	
21	3.4	**Creación del documento final**		**28/09/14**	**29/09/14**		
22	3.4.1	Repaso de formato y mejoras	3	28/09/14	28/09/14	19	
23	3.4.2	Revisión de comentarios del consultor y sus correcciones	2	29/09/14	29/09/14	20	
24	3.4.3	Elaboración del documento definitivo	2	29/09/14	29/09/14	23	
25	3.5	**Entrega de la PEC1: Plan de trabajo**	**0,5**	**30/09/14**	**30/09/14**	**9,20,24**	
26	4	PUNTO DE PARTIDA E INSTALACIÓN DE SOFTWARE		30/09/14	30/09/14		1,10%
27	4.1	**Revisión de los equipos de trabajo y puesta a punto de los mismos**	1,5	30/09/14	30/09/14		Provisión de software y hardware. Revisión y prueba de las instalaciones.
28	4.2	**Selección del software que se va a utilizar**	1	30/09/14	30/09/14		
29	4.3	**Instalación del software y pruebas de uso pertinentes**	1,5	30/09/14	30/09/14	27,28	
30	5	PEC2: PRIMERA PARTE DEL TFC		01/10/14	28/10/14		24,14%
31	5.1	**Revisión y repaso de conocimientos necesarios para abordar la PEC2**		**01/10/14**	**02/10/14**		Desarrollo de la PEC2, que supondrá un 24% de la carga total. Breve repaso de conceptos necesarios. Estudio, diseño y
32	5.1.1	Lectura de materiales de asignaturas relevantes	4	01/10/14	01/10/14		
33	5.1.2	Investigación de recursos por internet	2	02/10/14	02/10/14	32	

34	5.1.3	Primer contacto y provisión de recursos para el software pertinente	2	02/10/14	02/10/14	33	simulación de los siguientes componentes: LEDs, sensor de luz, circuitos de adaptación, circuito de excitación de los LEDs y drivers.
35	**5.2**	**Estudio de mercado de los LEDs**		**03/10/14**	**07/10/14**		
36	5.2.1	Introducción teórica a la tecnología LED	4	03/10/14	03/10/14	31	
37	5.2.2	Estudio de mercado de las diferentes opciones de LED	6	04/10/14	05/10/14	36	
38	5.2.3	Selección de la mejor opción de LED	4	06/10/14	06/10/14	37	
39	5.2.4	Documentación: redacción introducción y análisis de opciones	2	07/10/14	07/10/14	36,37,38	
40	**5.3**	**Diseño de los circuitos de adaptación**		**08/10/14**	**15/10/14**		
41	5.3.1	Estudio del sensor de luz	8	08/10/14	09/10/14	31	
42	5.3.2	Diseño del circuito eléctrico	7	10/10/14	11/10/14	41	
43	5.3.3	Búsqueda del circuito integrado	7	12/10/14	13/10/14	41,42	
44	5.3.4	Documentación: sensor, circuito y componentes seleccionados	8	14/10/14	15/10/14	41,42,43	
45	**5.4**	**Diseño del circuito de excitación de los LEDs y drivers asociados**		**16/10/14**	**22/10/14**		
46	5.4.1	Diseño del circuito analógico y/o digital	8	16/10/14	17/10/14	31	
47	5.4.2	Simulación del circuito	10	18/10/14	20/10/14	46	
48	5.4.3	Documentación	8	21/10/14	22/10/14	46,47	
49	**5.5**	**PEC2: Borrador**		**23/10/14**	**26/10/14**		
50	5.5.1	Redacción del borrador de la PEC2	4	23/10/14	23/10/14	39,44,48	
51	5.5.2	Envío borrador PEC2 al consultor	0,5	23/10/14	23/10/14	50	
52	5.5.3	Revisión de comentarios del consultor y sus correcciones	3	26/10/14	26/10/14	51	
53	**5.6**	**Entrega de la PEC2: Primera parte del TFC**	**0,5**	**28/10/14**	**28/10/14**	**52**	
54	**6**	**PEC3: SEGUNDA PARTE DEL TFC**		**24/10/14**	**09/12/14**		**44,44%**
55	**6.1**	**Revisión y repaso de conocimientos necesarios para abordar la PEC3**		**24/10/14**	**25/10/14**		
56	6.1.1	Lectura de materiales de asignaturas relevantes	4	24/10/14	24/10/14		Desarrollo de la PEC3, que implicará un 44% de la carga total. Breve repaso de conceptos necesarios. Estudio, selección, diseño y simulación de los siguientes componentes: microprocesador, fuente de alimentación y placa de circuito impreso (PCB). Establecimiento de conclusiones y análisis de posibles ampliaciones. Al finalizar este grupo de tareas el grueso del proyecto estará prácticamente terminado.
57	6.1.2	Investigación de recursos por internet	2	25/10/14	25/10/14	56	
58	6.1.3	Primer contacto y provisión de recursos para el software pertinente	1	25/10/14	25/10/14	57	
59	**6.2**	**Elección del procesador e implementación del programa de control**		**27/10/14**	**18/11/14**		
60	6.2.1	Estudio de microcontroladores	8	27/10/14	28/10/14	55	
61	6.2.2	Análisis de los requisitos de nuestro sistema	4	29/10/14	29/10/14		
62	6.2.3	Programación del micro controlador	60	30/10/14	14/11/14	60,61	
63	6.2.4	Simulación PIC	7	15/11/14	16/11/14	62	
64	6.2.5	Documentación	8	17/11/14	18/11/14	60,61,62,63	
65	**6.3**	**Diseño de la fuente de Alimentación**		**19/11/14**	**24/11/14**		
66	6.3.1	Estudio de la fuente de alimentación	4	19/11/14	19/11/14	55	
67	6.3.2	Diseño del circuito de la fuente de alimentación	7	20/11/14	21/11/14	66	
68	6.3.3	Simulación del circuito diseñado	7	22/11/14	23/11/14	67	

69	6.3.4	Documentación	4	24/11/14	24/11/14	66,67,68	
70	**6.4**	**Diseño del layout de la placa PCB**		**25/11/14**	**01/12/14**		
71	6.4.1	Estudio de los componentes de la placa	4	25/11/14	25/11/14	55	
72	6.4.2	Diseño y simulación de la placa PCB: archivo con pistas	18	26/11/14	30/11/14	71	
73	6.4.3	Documentación	4	01/12/14	01/12/14	71,72	
74	**6.5**	**Conclusiones y ampliaciones**		**02/12/14**	**03/12/14**	8,30,59,65,70	
75	6.5.1	Reflexión sobre objetivos, problemas y soluciones	2	02/12/14	02/12/14		
76	6.5.2	Análisis de ampliaciones futuras	2	02/12/14	02/12/14		
77	6.5.3	Documentación	4	03/12/14	03/12/14	75,76	
78	**6.6**	**PEC3: Borrador**		**04/12/14**	**08/12/14**		
79	6.6.1	Redacción del borrador de la PEC3	4	04/12/14	04/12/14	59,65,70,74	
80	6.6.2	Envío borrador PEC3 al consultor	0,5	04/12/14	04/12/14	79	
81	6.6.3	Revisión de comentarios del consultor y sus correcciones	7	07/12/14	08/12/14	80	
82	**6.7**	**Entrega de la PEC3: Segunda parte del TFC**	**0,5**	**09/12/14**	**09/12/14**	**81**	
83	**7**	**ENTREGA FINAL**		**12/12/14**	**30/01/15**		**19,89%**
84	**7.1**	**Recopilación de todo el trabajo realizado hasta la fecha**	4	12/12/14	12/12/14	8,30,54	
85	**7.2**	**Memoria final**		**13/12/14**	**07/01/15**		
86	7.2.1	Elaboración del borrador de la memoria	46	13/12/14	26/12/14	8,30,54,84	
87	7.2.2	Envío del borrador al consultor	0,5	26/12/14	26/12/14	86	
88	7.2.3	Revisión de comentarios del consultor y sus correcciones	4	30/12/14	30/12/14	87	
89	7.2.4	Entrega de la memoria final	0,5	07/01/15	07/01/15	88,94,97	Últimos retoques al proyecto. Elaboración formal del documento de la memoria. Realización del vídeo presentación. Revisión de todos los archivos generados durante las simulaciones y del código de programación. Entrega final incluyendo debate virtual. Al finalizar este grupo de tareas el proyecto habrá llegado a su fin.
90	**7.3**	**Presentación audiovisual**		**27/12/14**	**07/01/15**		
91	7.3.1	Revisión y comprobación de uso del software Camtasia	1	27/12/14	27/12/14		
92	7.3.2	Preparación guión del vídeo	2	27/12/14	27/12/14		
93	7.3.3	Grabación del vídeo	3	04/01/15	04/01/15	91,92	
94	7.3.4	Entrega de la presentación audiovisual	0,5	07/01/15	07/01/15	93	
95	**7.4**	**Archivos fuente**		**05/01/15**	**07/01/15**		
96	7.4.1	Recopilación de los archivos fuente y de las simulaciones realizadas	4	05/01/15	05/01/15	35,40,45,59,65,70	
97	7.4.2	Envío de los archivos fuente	0,5	07/01/15	07/01/15	96	
98	**7.5**	**Debate virtual**		**28/01/15**	**30/01/15**		
99	7.5.1	Recepción y lectura de las preguntas del Tribunal	2	28/01/15	28/01/15		
100	7.5.2	Elaboración de las respuestas	4	29/01/15	29/01/15	99	
101	7.5.3	Envío de las respuestas al Tribunal de evaluación	0,5	30/01/15	30/01/15	100	
		TOTAL HORAS	**315,5**				

Tabla 3. Planificación por tarea

1.3.4. Diagramas de Gannt

Toda la planificación explicada en los apartados anteriores se puede resumir gráficamente a través de una serie de diagramas de Gannt. Mostramos la vista general y a continuación, el detalle por grupo de tareas, correspondientes a las tres entregas de evaluación continua y a la entrega final:

	Nombre de tarea	Comienzo	Fin
1	1 TRABAJO DE FIN DE CARRERA	mié 17/09/14	vie 30/01/15
2	2 DEFINICIÓN Y CONTEXTUALIZACIÓN DEL TFC	mié 17/09/14	lun 22/09/14
8	3 PEC1: PLAN DE TRABAJO	mar 23/09/14	mar 30/09/14
26	4 PUNTO DE PARTIDA E INSTALACIÓN DE SOFTWARE	mar 30/09/14	mar 30/09/14
30	5 PEC2: PRIMERA PARTE DEL TFC	mié 01/10/14	mar 28/10/14
54	6 PEC3: SEGUNDA PARTE DEL TFC	vie 24/10/14	mar 09/12/14
83	7 ENTREGA FINAL	vie 12/12/14	vie 30/01/15

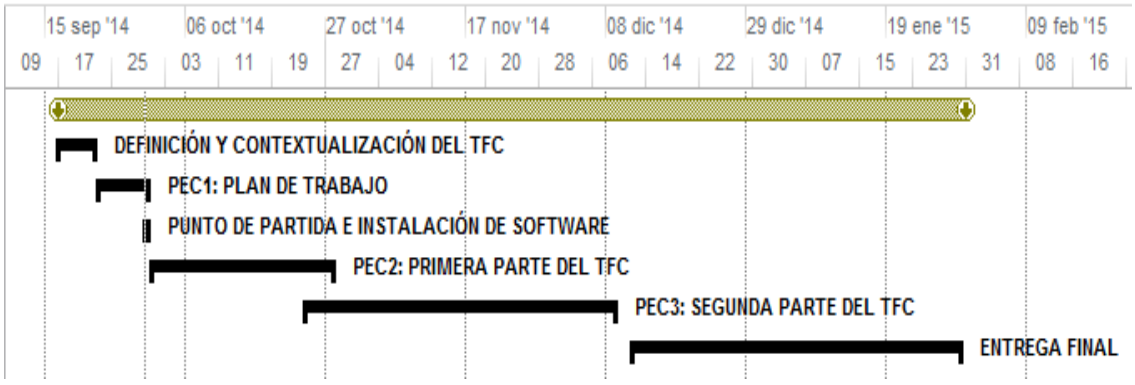


Figura 3. Diagrama de Gannt - Vista general del proyecto

Id	Nombre de tarea
2	**2 DEFINICIÓN Y CONTEXTUALIZACIÓN DEL TFC**
3	2.1 Lectura del enunciado del proyecto
4	2.2 Lectura pormenorizada del enunciado y extracción de dudas
5	2.3 Video conferencia de inicio con el consultor
6	2.4 Lectura del plan de trabajo de ejemplo
7	2.5 Revisión del documento resumen de la reunión
8	**3 PEC1: PLAN DE TRABAJO**
9	**3.1 Definición del guión del plan de trabajo**
10	**3.2 Borrador secciones: primera aproximación**
11	3.2.1 Presentación e introducción
12	3.2.2 Objetivos y competencias
13	3.2.3 Estructura TFC
14	3.2.4 Planificación: calendario, hitos y tareas
15	3.2.5 Diagrama de Gannt y análisis DAFO
16	3.2.6 Materiales y bibliografía
17	3.2.7 Riesgos y plan de contingencia
18	**3.3 Redacción final del borrador**
19	3.3.1 Revisión por sección y mejoras
20	3.3.2 Envío del borrador al consultor
21	**3.4 Creación del documento final**
22	3.4.1 Repaso de formato y mejoras
23	3.4.2 Revisión de comentarios del consultor y sus correcciones
24	3.4.3 Elaboración del documento definitivo
25	**3.5 Entrega de la PEC1: Plan de trabajo**
26	**4 PUNTO DE PARTIDA E INSTALACIÓN DE SOFTWARE**
27	4.1 Revisión de los equipos de trabajo y puesta a punto de los mismos
28	4.2 Selección de todo el software que se va a utilizar
29	4.3 Instalación del software y pruebas de uso pertinentes

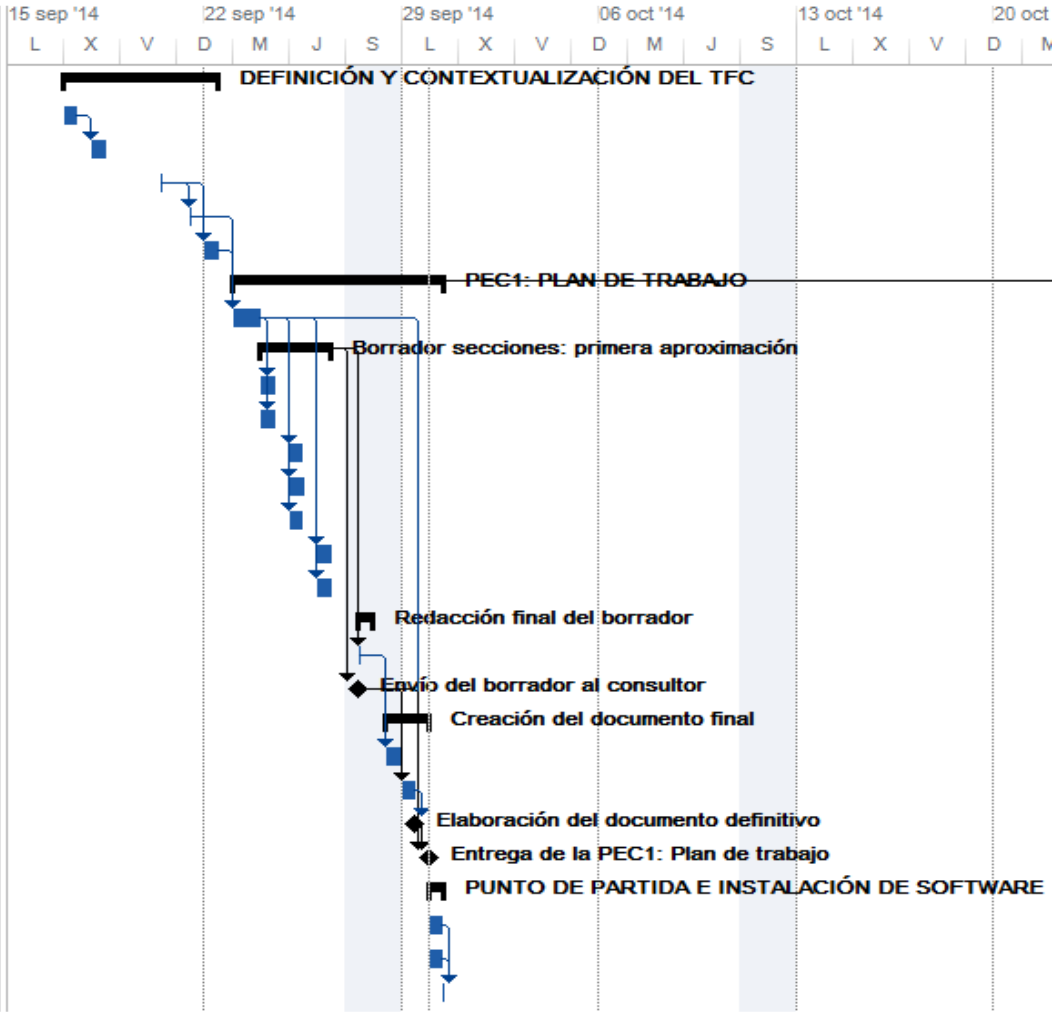


Figura 4. Diagrama de Gannt - Definición y contextualización, PEC1 y Punto de Partida.

Id	Nombre de tarea
30	**5 PEC2: PRIMERA PARTE DEL TFC**
31	**5.1 Revisión y repaso de conocimientos necesarios para abordar la PEC2**
32	5.1.1 Lectura de materiales de asignaturas relevantes
33	5.1.2 Investigación de recursos por internet
34	5.1.3 Primer contacto y provisión de recursos para el software pertinente
35	**5.2 Estudio de mercado de los ledes**
36	5.2.1 Introducción teórica a la tecnología led
37	5.2.2 Estudio de mercado de las diferentes opciones de led
38	5.2.3 Selección de la mejor opción de led
39	5.2.4 Documentación: redacción introducción y análisis de opciones
40	**5.3 Diseño de los circuitos de adaptación**
41	5.3.1 Estudio del sensor de luz
42	5.3.2 Diseño del circuito eléctrico
43	5.3.3 Diseño del circuito integrado para la señal RS232
44	5.3.4 Documentación: sensor, circuito y componentes seleccionados
45	**5.4 Diseño del circuito de excitación de los ledes y drivers asociados**
46	5.4.1 Diseño del circuito analógico y/o digital
47	5.4.2 Simulación del circuito
48	5.4.3 Documentación
49	**5.5 PEC2: Borrador**
50	5.5.1 Redacción del borrador de la PEC2
51	5.5.2 Envío borrador PEC2 al consultor
52	5.5.3 Revisión de comentarios del consultor y sus correcciones
53	**5.6 Entrega de la PEC2: Primera parte del TFC**

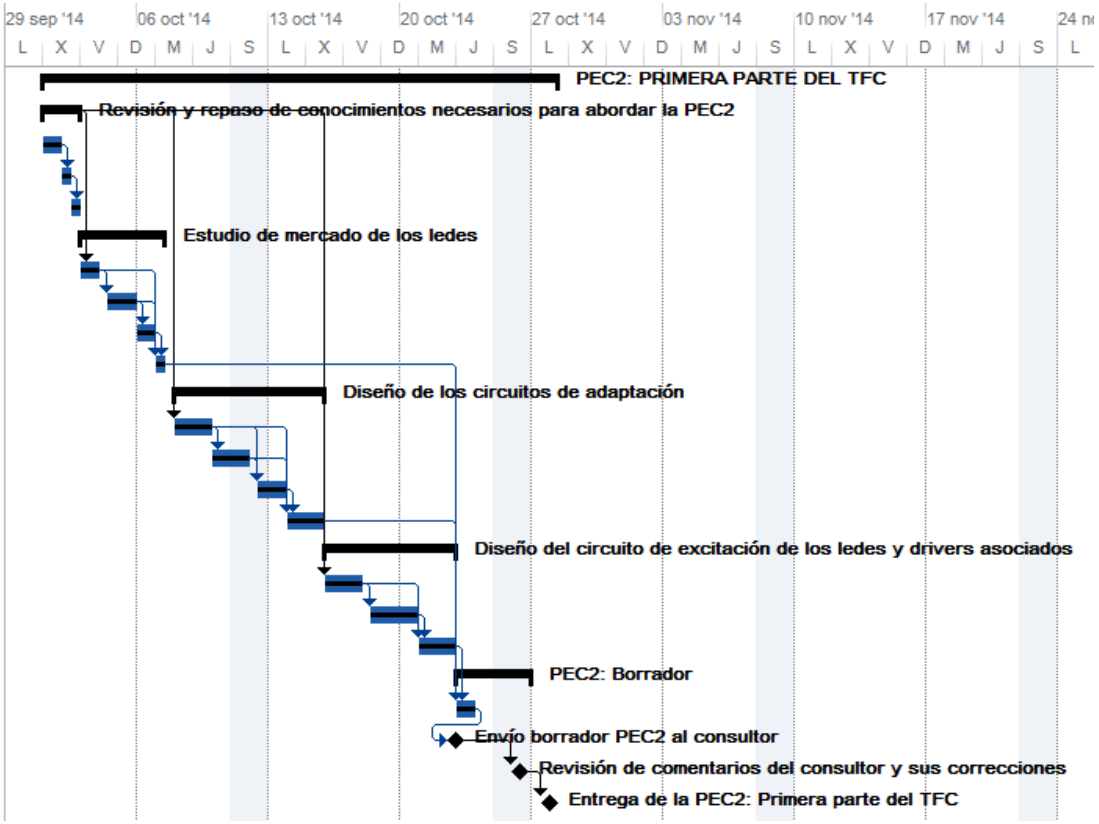


Figura 5. Diagrama de Gannt - PEC2: Primera parte del TFC

Id	Nombre de tarea
54	**6 PEC3: SEGUNDA PARTE DEL TFC**
55	**6.1 Revisión y repaso de conocimientos necesarios para abordar la PEC3**
56	6.1.1 Lectura de materiales de asignaturas relevantes
57	6.1.2 Investigación de recursos por internet
58	6.1.3 Primer contacto y provisión de recursos para el software pertinente
59	**6.2 Elección del procesador e implementación del programa de control**
60	6.2.1 Estudio de microcontroladores
61	6.2.2 Análisis de los requisitos de nuestro sistema
62	6.2.3 Programación del microcontrolador
63	6.2.4 Simulación PIC
64	6.2.5 Documentación
65	**6.3 Diseño de la fuente de Alimentación**
66	6.3.1 Estudio de la fuente de alimentación
67	6.3.2 Diseño del circuito de la fuente de alimentación
68	6.3.3 Simulación del circuito diseñado
69	6.3.4 Documentación
70	**6.4 Diseño del layout de la placa PCB**
71	6.4.1 Estudio de los componentes de la placa
72	6.4.2 Diseño y simulación de la placa PCB: archivo con pistas
73	6.4.3 Documentación
74	**6.5 Conclusiones y ampliaciones**
75	6.5.1 Reflexión sobre objetivos, problemas y soluciones
76	6.5.2 Análisis de ampliaciones futuras
77	6.5.3 Documentación
78	**6.6 PEC3: Borrador**
79	6.6.1 Redacción del borrador de la PEC3
80	6.6.2 Envío borrador PEC3 al consultor
81	6.6.3 Revisión de comentarios del consultor y sus correcciones
82	**6.7 Entrega de la PEC3: Segunda parte del TFC**

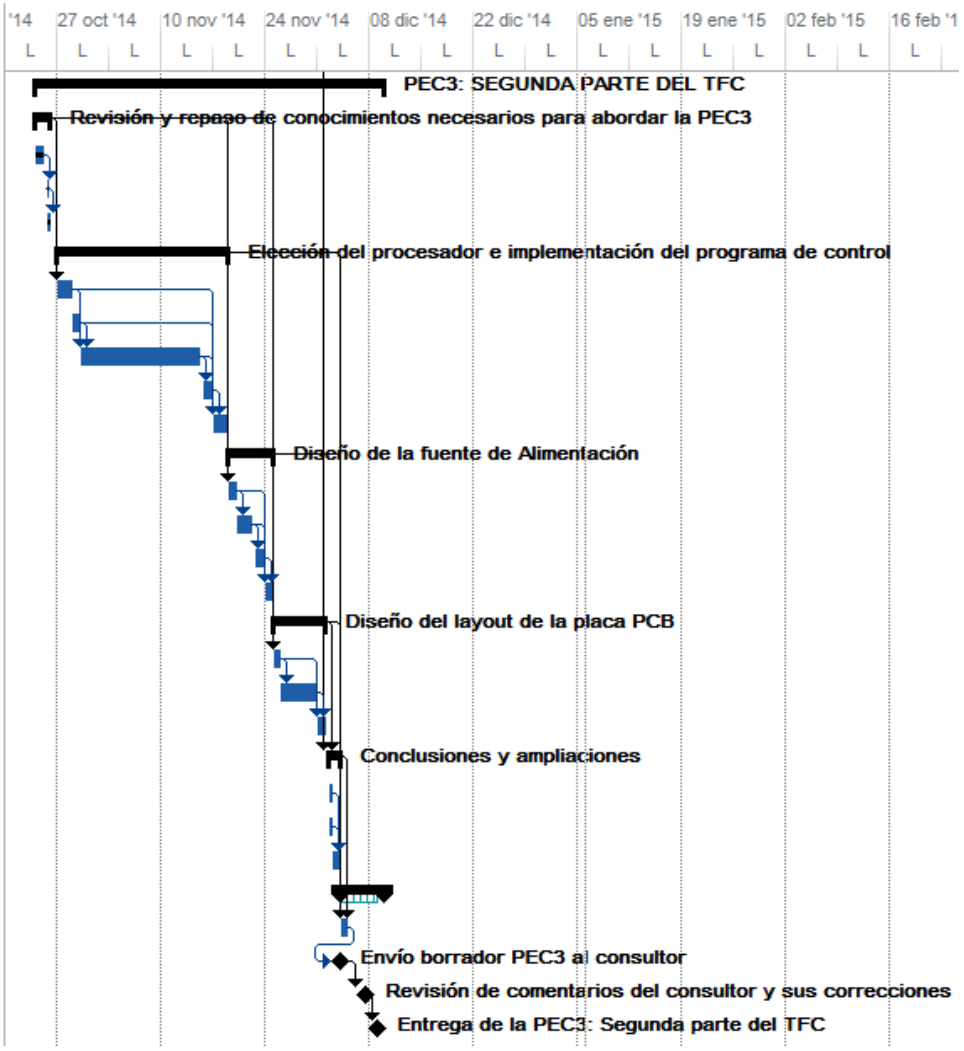


Figura 6. Diagrama de Gannt – PEC3: Segunda parte del TFC

	Nombre de tarea
83	**7 ENTREGA FINAL**
84	**7.1 Recopilación de todo el trabajo realizado hasta la fecha**
85	**7.2 Memoria final**
86	7.2.1 Elaboración del borrador de la memoria
87	7.2.2 Envío del borrador al consultor
88	7.2.3 Revisión de comentarios del consultor y sus correcciones
89	7.2.4 Entrega de la memoria final
90	**7.3 Presentación audiovisual**
91	7.3.1 Revisión y comprobación de uso del software Camtasia
92	7.3.2 Preparación guión del vídeo
93	7.3.3 Grabación del vídeo
94	7.3.4 Entrega de la presentación audiovisual
95	**7.4 Archivos fuente**
96	7.4.1 Recopilación de todos los programas
97	7.4.2 Envío de los archivos fuente
98	**7.5 Debate virtual**
99	7.5.1 Recepción y lectura de las preguntas del Tribunal
100	7.5.2 Elaboración de las respuestas
101	7.5.3 Envío de las respuestas al Tribunal de evaluación

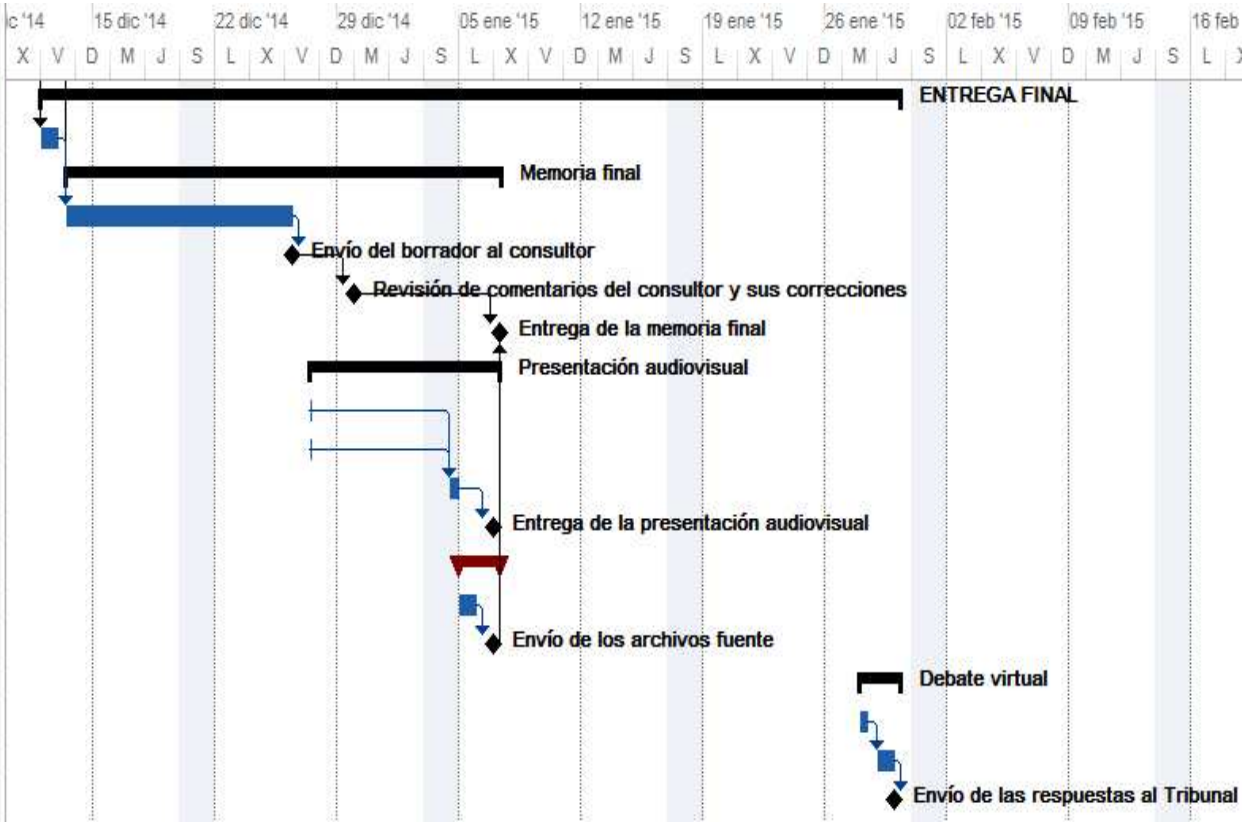


Figura 7. Diagrama de Gannt – Entrega final

1.4. MATERIALES

Para la elaboración del proyecto emplearemos una serie de recursos de *hardware* y *software*, que en el momento de la redacción de este plan de trabajo se encuentran ya disponibles. La lista de aplicaciones está sujeta a revisión en caso de ser necesario.

Hardware

El recurso principal es un portátil donde tenemos instalados todos los programas. También se ha habilitado un equipo de sobremesa, que nos servirá de alternativa en caso de que el equipo principal sufra cualquier eventualidad:

- Equipo principal: portátil *MacBook Pro* con procesador Intel Core i5 a 2,5GHz, 8Gb de memoria RAM y HD de 500MB. El sistema operativo principal es MacOS X 10.8.5 y el secundario es Windows 8.1.
- Equipo de *backup*: sobremesa con procesador AMD a 1,8GHz, 6Gb de RAM, HD de 1TB y Windows 7.

Software

Los equipos disponen de los siguientes programas:

- Microsoft Project 2013: utilizado para la planificación del proyecto y su seguimiento.
- Suite de Office 2013: para la redacción de documentos y elaboración de presentaciones.
- Proteus 8:
  - Isis: para el diseño y simulación de circuitos analógicos/digitales.
  - Ares: para el diseño de la placa PCB.
- CCS 5.0.15: compilador para el código del microprocesador.
- Multiway V10: para la simulación de envío de tramas MODBUS.
- Virtual Serial Port Driver 7.2: para la creación de puertos virtuales.
- LochMaster 4.0: para el diseño PCB sobre placas perforadas para el dispositivo físico.
- Camtasia: con el que grabaremos el vídeo con la presentación final al Tribunal.

1.5. RIESGOS Y PLAN DE CONTINGENCIA

Es de esperar que a lo largo del proyecto puedan surgir eventualidades no previstas. Estas incidencias se pueden clasificar en dos grupos, y para cada una de ellas añadimos su plan de contingencia:

Incidencias técnicas

- Problemas con los recursos de *hardware*: es posible que alguno de los equipos no responda como se espera. Por ello contamos con dos ordenadores independientes. En caso de que ambos se estropeen, disponemos de soluciones alternativas que incluyen el uso de terminales de familiares.
- Incompatibilidad de los recursos de *software*: se puede dar el caso de que no podamos desarrollar el proyecto completo sobre una única plataforma, por cuestiones de incompatibilidad. Es por ello que disponemos de los tres sistemas operativos principales, Windows, MacOS y Linux, de modo que nos aseguraremos de que todos los programas podrán funcionar en uno u otro.
- Pérdida de datos: ante la posibilidad de que se pierdan datos, algo no deseable, se hará una copia de seguridad diaria de la carpeta de trabajo, que se guardará tanto en una localización en la nube como en un disco duro portátil y en un dispositivo USB. Se garantiza la sincronización de los datos.

- Dificultades en la utilización del *software* especializado: al ser programas nuevos será necesario superar una cierta curva de aprendizaje. Para ello hemos previsto unas horas de lectura de documentación que nos ayudarán a avanzar más rápido cuando trabajemos con dichos programas.

Incidencias personales

- Simultaneidad con otra asignatura: para evitar que el estudio de otra asignatura pueda interferir en el desarrollo del TFC se ha aprovechado el verano para prepararla y se han tenido en cuenta las fechas de entrega de la evaluación continua y del examen a la hora de preparar el plan del proyecto.
- Dificultades en la realización: este proyecto supone un reto, puesto que gran cantidad de los contenidos son relativamente nuevos. Es posible que haya puntos a los que deba dedicarse más tiempo del previsto. Para que ello no suponga un problema tenemos un margen de un 20% de horas que se podrán utilizar para recuperar tiempo en caso de que sea necesario.
- Enfermedad o indisposición: este 20% extra de horas contemplado en el plan nos será también de utilidad en caso de que alguna enfermedad nos obligue a replantear los horarios.
- El plan de contingencia principal para evitar cualquier riesgo se basa en hacer un seguimiento continuado del plan de proyecto. A través de ese ejercicio se puede comprobar día a día el grado de cumplimiento y si es necesario hacer algún tipo de restructuración o aplicar alguna medida correctora.

1.6. ANÁLISIS DAFO

A modo de recapitulación vamos a utilizar un análisis DAFO. Esta herramienta nos ayudará a situar de forma gráfica los puntos fuertes y débiles del proyecto:

ANALISIS INTERNO	
Fortalezas	**Debilidades**
Dedicación plena al proyecto durante el semestre	Escasa experiencia práctica en diseño de circuitos y prototipos
Varios años de experiencia profesional en la dirección de proyectos técnicos	Desconocimiento de los programas de diseño que se utilizarán
Perseverancia sin límite	
Buen margen de maniobra para gestionar imprevistos	
ANALISIS EXTERNO	
Oportunidades	**Amenazas**
Riqueza de recursos en Internet, tanto teóricos como prácticos	Riesgo de necesitar más tiempo del estimado
Disposición del consultor para hacer seguimiento y servir de guía	Riesgo de enfermedad durante el proyecto
	Imposibilidad de prever fallos técnicos de los equipos utilizados

Tabla 4. Análisis DAFO

2. ESTUDIO DE MERCADO DE LOS LEDES

2.1. INTRODUCCIÓN TEÓRICA A LA TECNOLOGÍA LED

Nos adentramos en el mundo de la tecnología led con el fin de diseñar un panel informativo. Como punto de partida describiremos qué es un led, unidad mínima que compone nuestro panel, y posteriormente haremos un repaso de los distintos tipos y usos hasta llegar a la solución seleccionada para el proyecto.

Un led es, en resumen, un diodo que tiene la capacidad de emitir luz. Un diodo, como sabemos, es un componente semiconductor que permite el paso de la corriente en un único sentido. No es nuestro cometido estudiarlo en detalle pero sí es importante contextualizar el componente que nos interesa: el led.

El led (del inglés "*LED*" - "*Light Emitting Diode*") puede emitir todos los colores del espectro visible desde el azul hasta el rojo. Tiene muchas aplicaciones prácticas. Entre ellas, se utilizan como indicadores, como emisores de infrarrojos (por ejemplo, en los mandos a distancia) o como elementos de iluminación.

Frente a otras fuentes de luz proporcionan una iluminación muy direccional, y son más rápidos, ya que pueden encenderse y apagarse instantáneamente sin necesidad de calentamiento previo.

Desde un punto de vista técnico, un led emite luz cuando se encuentra en polarización directa y se comporta como un interruptor abierto cuando está en polarización inversa. Por lo tanto, si aplicamos tensión en un sentido se establece una circulación de corriente y si la tensión se invierte la corriente se interrumpe.

Para entender el funcionamiento de la tecnología es importante entender cómo produce luz un led. En la conducción electrónica de cualquier diodo intervienen dos portadores de carga: los electrones (con carga negativa) y los huecos (espacios vacíos que quedan en los átomos de la estructura que han perdido electrones; su carga se considera positiva). Por tanto, la corriente en el interior de un semiconductor está formada por un desplazamiento de electrones en un sentido y de huecos en sentido contrario.

Cuando aplicamos un campo eléctrico a un semiconductor el electrón se libera y se mueve buscando el hueco libre más próximo. Con ello se transmite parte de la energía del campo eléctrico al electrón y este se mueve. Cuando el electrón encuentra un hueco y lo ocupa (fenómeno que conocemos como "recombinación"), se interrumpe su movimiento y pierde la energía transmitida. Esta energía no vuelve al campo eléctrico sino que se transforma en calor y luz.

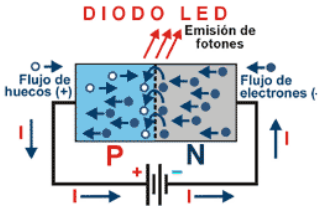


Figura 8. Movimiento de electrones – Diodo led

En el caso de los ledes, sus materiales de fabricación (galio, arsénico o fósforo, por ejemplo) permiten que la luz generada se irradie al exterior, ya que son transparentes, y además, la unión de sus contactos externos con el material semiconductor no se interpone en su camino.

El color de la luz que emite un led depende del material de fabricación y se corresponde con una frecuencia del espectro electromagnético, con una longitud de onda que para el rojo es de unos $630nm$.

Un led no puede conectarse directamente a la fuente de alimentación, sino que necesita de una resistencia para limitar la corriente. En caso contrario, el led se quemará.

El circuito más básico estaría formado por una fuente de alimentación, una resistencia limitadora y un led, y el valor de la resistencia se puede obtener aplicando la Ecuación 1.

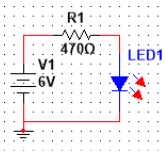


Figura 9. Circuito básico led

$$R = \frac{[V(IN) - V(LED)]}{I(LED)}$$

Ecuación 1. Cálculo resistencia led

Siempre que queramos usar un led tendremos que calcular la resistencia necesaria y remitirnos a la hoja de especificaciones del fabricante, que nos dará sus valores mínimos y máximos.

Para obtener una buena intensidad luminosa debe escogerse bien la corriente que atraviesa el led. El voltaje de operación para el led rojo es de unos $2V$ (posteriormente veremos los valores concretos para el modelo elegido). La intensidad que debe circular por él varía según su aplicación. Para un indicador el valor típico es de $10mA$ y para una lámpara de alta luminosidad, de $40mA$. Nuestro panel debe ser eficiente, por lo que tendremos que optimizar su funcionamiento. En general, se entiende que la eficiencia es mayor cuanto menor es la intensidad que circula por el led. Con este fin, haremos que la luminosidad dependa del nivel lumínico ambiente: cuanta menos luz externa haya, menos tendrá que brillar nuestro led, para lo cual, lo tendremos encendido más o menos tiempo. Esta duración la controlaremos modificando el ciclo de trabajo con modulación PWM. Así, si la luz exterior es escasa y el led ha de estar encendido el 25% del tiempo, el ciclo de trabajo será del 25%. En el apartado "Estudio del sensor de luz" explicaremos esta idea con algo más de detalle, y en la sección "Programación del microcontrolador" veremos cómo hemos implementado la lógica de funcionamiento y cómo gestionaremos la señal PWM y los ciclos de trabajo.

2.2. ESTUDIO DE MERCADO DE LAS DIFERENTES OPCIONES DE LED

Al iniciar esta introducción hemos mencionado que los ledes tienen diversas aplicaciones. Para poder seleccionar el más adecuado debemos tener en cuenta el uso que se le va a dar. Podemos relacionar directamente los tipos de ledes con su aplicación más frecuente, y esto nos llevará en el siguiente apartado a la selección del más apropiado para el proyecto.

Los principales tipos de diodos led que podemos encontrar en el mercado son:

Figura 10. Diodo led común

- **Estándar o común.** Se utiliza principalmente como indicador o señalizador. Su potencia lumínica es escasa, por lo que no es adecuado para aplicaciones como la iluminación doméstica.

Figura 11. Led infrarrojo

- **Diodo infrarrojo**. Se usa en aplicaciones de control remoto para la transmisión de datos entre dispositivos. Encontramos este tipo en mandos a distancia, electrodomésticos o equipos de música. No es un led apto para nuestro proyecto, ya que emite infrarrojos.

Figura 12. Led alta luminosidad

- **De alta luminosidad**. Similar al estándar pero con mayor intensidad lumínica. Su encapsulado suele ser transparente y puede emitir luz roja, anaranjada, amarilla, verde, azul y blanca. Se utiliza como fuente de iluminación o para paneles informativos. Este será el tipo de led que utilizaremos en el proyecto.

Figura 13. Led de potencia

- **Led de potencia**. Ofrece mayor potencia lumínica que los ledes de alta luminosidad pero con mayor consumo, además de requerir de una disipación térmica muy buena. Se utiliza en linternas potentes o como fuentes de luz. Debido a su mayor consumo no seleccionaremos este tipo para nuestro panel.

2.3. SELECCIÓN DE LA MEJOR OPCIÓN DE LED

Los ledes se pueden adquirir individualmente o agrupados. Para el prototipo serán individuales, ya que a nivel de simulación los programas nos permiten definir mejor sus características. En cambio, para el equipo físico usaremos 4 matrices de 8x8, ya que no se dispone del tiempo suficiente para soldar todos los ledes.

Puesto que vamos a diseñar un panel informativo, tenemos unos requisitos concretos que nos llevarán a buscar un tipo de led de alta luminosidad, que es el más utilizado en carteles.

Si analizamos las opciones que nos ofrece el mercado veremos que hay una larga lista de posibilidades, así que nos centraremos en los siguientes criterios básicos para elegir uno:

- El color será rojo (longitud de onda de alrededor de $630nm$).
- Intensidad luminosa adecuada. Para un panel informativo se considera que el mínimo ha de ser de $500mcd$. Para no excedernos en el consumo debemos encontrar un equilibrio entre ambos parámetros.
- Rango de intensidad luminosa amplio, para que el ajuste en función de la luz ambiente sea perceptible.
- Corriente directa de alrededor de $20mA$. para asegurar que el consumo del equipo sea eficiente.

Atendiendo a estos criterios se ha seleccionado el *Kingbright KPTL-3216SURCK-01*, que cumple nuestros requisitos. Sus principales características, extraídas de la hoja de del fabricante, son:

Parámetro	Mín.	Típ.	Máx.
Intensidad luminosa (a 20mA)	400mcd	550mcd	
Angulo de visión		70º	
Corriente directa	10mA	20mA	30mA
Tensión directa		1,95V	2,5V
Longitud de onda		630nm	

Tabla 5. Características led Kingbright KPTL-3216SURCK-01

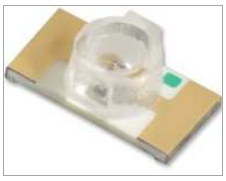
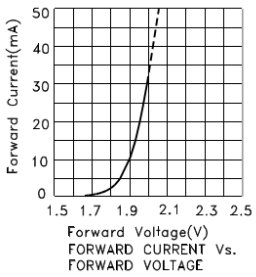


Figura 14. Kingbright KPTL-3216SURCK-01

Durante el proyecto tendremos en cuenta sus curvas características, con las que podremos definir la corriente en función del voltaje, y comprobar cuál es la intensidad luminosa relativa respecto a la corriente:



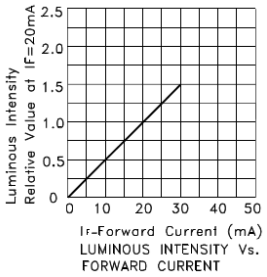
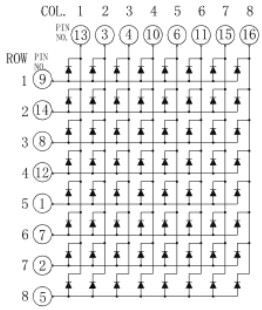
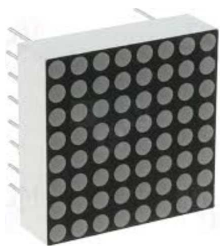


Figura 15. Curvas características KPTL-3216SURCK-01

Para el equipo físico se utilizan 4 matrices de alto brillo de la casa *Lukylight*, soldando sus pines según los esquemas de conexión. A continuación podemos verlos, junto a la matriz individual y al panel final:





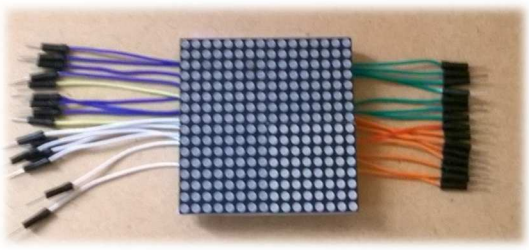


Figura 16. Detalle de matrices 8x8 Luckylight

3. DISEÑO DE LOS CIRCUITOS DE ADAPTACIÓN

En el Diagrama de bloques expuesto en la "Introducción" hemos visto cuáles son los módulos que conforman el sistema. Cada uno de ellos tiene unas señales de entrada y de salida que tendremos que tratar y adaptar convenientemente para que puedan comunicarse con el resto de los bloques.

En esta sección analizaremos en concreto los circuitos del sensor de luz y de la señal RS232.

3.1. ESTUDIO DEL SENSOR DE LUZ

La luminosidad de los ledes dependerá del nivel de luz exterior, por lo que nuestro equipo cuenta con un sensor de luz que envía la señal al microprocesador. Este adquiere los valores y determina cuánto tiempo han de estar encendidos los ledes. Así es como controlaremos que brillen más o menos.

Necesitamos, por lo tanto, un dispositivo sensor y un circuito de adaptación de señal que haga las conversiones necesarias entre la señal analógica que se recibe y el microprocesador. Posteriormente, programaremos el microprocesador para que envíe las señales oportunas al bloque de *drivers* de los ledes.

Nos corresponde, en primer lugar, entender qué opciones tenemos en cuanto a sensores. Dentro de las posibles, nos hemos centrado en dos:

- Uso de un circuito integrado que capte la luz y la convierta a una magnitud adecuada. Aunque a priori podría parecer una solución directa, implica necesariamente el desarrollo de un *firmware* para la comunicación con el microprocesador. Al valorar esta opción se ha analizado el convertidor *TSL235R* de TAOS. Su modo de funcionamiento consiste en la conversión de luz a frecuencia. Responde al rango de $320nm$ a $1.050nm$, se alimenta con entre $2{,}7$ y $5{,}5V$ y consume $2mA$. No requiere componentes externos. Sin embargo, ha sido descartado por necesitar de un *firmware* para convertir la frecuencia.

Figura 17. TSL235R

- Uso de un sensor LDR. El LDR (*light dependent resistor*) es una resistencia que varía su valor en función de la luz; cuanta más luz recibe, menor es su resistencia. Está fabricado con un semiconductor de alta resistencia (por ejemplo, sulfuro de cadmio). Si la luz que incide en el dispositivo es de alta frecuencia, los fotones son absorbidos por la elasticidad del semiconductor dando a los electrones la suficiente energía para saltar la banda de conducción. El electrón libre resultante conduce electricidad, por lo que disminuye la resistencia.

Figura 18. LDR

Se ha decidido optar por su uso porque se trata de un componente muy versátil, sencillo de implementar y con un coste muy reducido. Aunque requiere el desarrollo de un circuito analógico para su adaptación al microprocesador, en el próximo apartado veremos que se trata de un circuito relativamente simple.

3.2. DISEÑO DEL CIRCUITO ELÉCTRICO DEL SENSOR DE LUZ

El LDR que vamos a utilizar varía su resistencia en función de la cantidad de luz que recibe.

El circuito de adaptación consistirá en un divisor de tensión entre el LDR y otra resistencia, de donde saldrá la señal que finalmente conectaremos a una entrada analógica del microprocesador.

Colocaremos el LDR como resistencia superior, por lo que cuanta más luz haya, menos resistencia ejercerá el LDR y mayor tensión tendremos a la entrada del PIC. Por lo tanto, en una situación de oscuridad completa el divisor nos dará un valor cercano a los $0V$ y en otra de luminosidad absoluta, cercano a los $5V$.

Podemos adquirir un LDR genérico para este fin, y de hecho, consultando las especificaciones de varios de ellos, vemos que sus características no difieren significativamente a no ser que optemos por un LDR de alta precisión (cuyo precio es ostensiblemente mayor y cuyas ventajas no notaríamos demasiado en nuestro proyecto). Es por ello que nos decantamos por el LDR *GL5516*.

Al consultar la hoja de datos podemos ver cuál es su nivel de sensibilidad, cuánto tiempo tarda en reaccionar a un cambio de luz, cuál es el voltaje máximo que admite, cuáles los valores mínimos y máximos de luz que puede captar y lo que más nos interesa, cuál es la relación entre la resistencia y la iluminación. Para este modelo estos datos son:

Type	Max. Voltage	Max. power	Environmental temp.
GL5516	150	90	-30~+70

Light resistance (10Lux) (KΩ)	Dark resistance (MΩ)	γ_{10}^{100}	Response time (ms)	
			Increase	Decrease
5-10	0.5	0.5	30	30
10-20	1	0.6	20	30
20-30	2	0.6	20	30
30-50	3	0.7	20	30
50-100	5	0.8	20	30
100-200	10	0.9	20	30

Tabla 6. Características LDR GL5516

La fórmula matemática que expresa la relación resistencia-iluminación es la siguiente:

$$R = A \cdot E^{a}$$

Ecuación 2. Relación resistencia-luminosidad

Donde R es la resistencia del LDR, E es la luminosidad en lux., y A y a son constantes que dependen del material de construcción del dispositivo (entre $0{,}7$ y $0{,}9$).

En la hoja de especificaciones también tenemos disponible la gráfica que relaciona ambos parámetros, y con ayuda de esta y de la ecuación anterior podemos obtener la siguiente tabla, que a partir de unos valores típicos de iluminación nos indica cuál sería la salida del divisor de tensión. Así comprobamos que el sensor cubre todo el rango del ADC del PIC y que posteriormente el microprocesador podrá obtener la luz a partir del voltaje que recibe por la entrada:

Situación	Lux	Salida V
Oscuridad	0.1	$\sim 0V$
Penumbra	10	$\sim 2V$
Luz intensa	100	$\sim 4V$
Plena luz	>200	$\sim 5V$

Tabla 7. Valores voltaje salida en función de luz

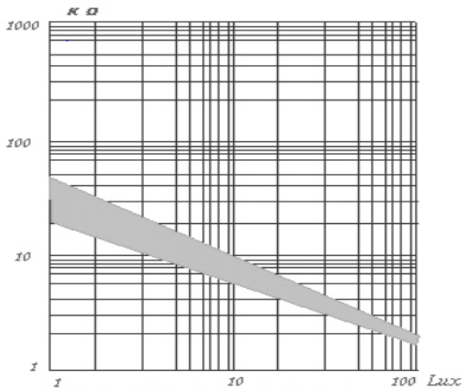


Figura 19. Gráfica R-Lux LDR

Una vez hemos analizado el funcionamiento del sensor y el significado de la señal que recibirá el PIC, procedemos a completar el circuito eléctrico. Además del sensor propiamente dicho, necesitamos también incluir un seguidor de tensión (el *LM324*). El motivo es que debemos tener en cuenta la impedancia de entrada del ADC del PIC. Esta puede desvirtuar las medidas en caso de ser muy baja, por lo que para garantizar que el valor que obtenemos es el correcto añadiremos un seguidor que utilizaremos para adaptar impedancias diferentes (conectar un dispositivo con gran impedancia a otro con una impedancia pequeña o viceversa).

El seguidor de tensión es una de las aplicaciones del amplificador operacional (AO), circuito integrado cuya salida es la diferencia de las dos entradas multiplicada por un factor G o ganancia. En el caso del seguidor de tensión la ganancia es 1, por lo que la tensión de salida será igual a la tensión de entrada.

Con el circuito ya configurado, la lógica de funcionamiento es simple. La señal de salida del seguidor de tensión va conectada a un pin de entrada analógica del PIC. Este lee el valor de tensión y hace las conversiones oportunas para interpretar la señal en términos de luminosidad (a partir de las ecuaciones facilitadas por el fabricante podemos obtener el valor de la luz), y con ello determina qué porcentaje del tiempo total del ciclo de trabajo debe estar encendido el led que corresponda. Este valor lo tendremos en la salida PWM del PIC, que estará conectada al pin "*enable*" del circuito de drivers de los ledes, y hará que el sistema se comporte con normalidad (en caso de estar dentro del tiempo de encendido) o que se limpien las salidas de los integrados que gestionan los ledes y por lo tanto, estos se apaguen (cuando se haya superado el tiempo de encendido). En el apartado "Programación del microcontrolador" explicaremos con detalle este aspecto y todos los cálculos seguidos.

Inicialmente se había contemplado la opción de manejar la señal PWM por *software*, pero la dificultad de obtener la precisión necesaria nos obliga a variar el planteamiento y hacerlo por *hardware*.

Con todos los elementos explicados ya podemos diseñar el circuito de adaptación del sensor de luz. A continuación podemos ver el circuito en *Proteus* y el montaje, tanto en *protoboard*[3] como en la placa final:

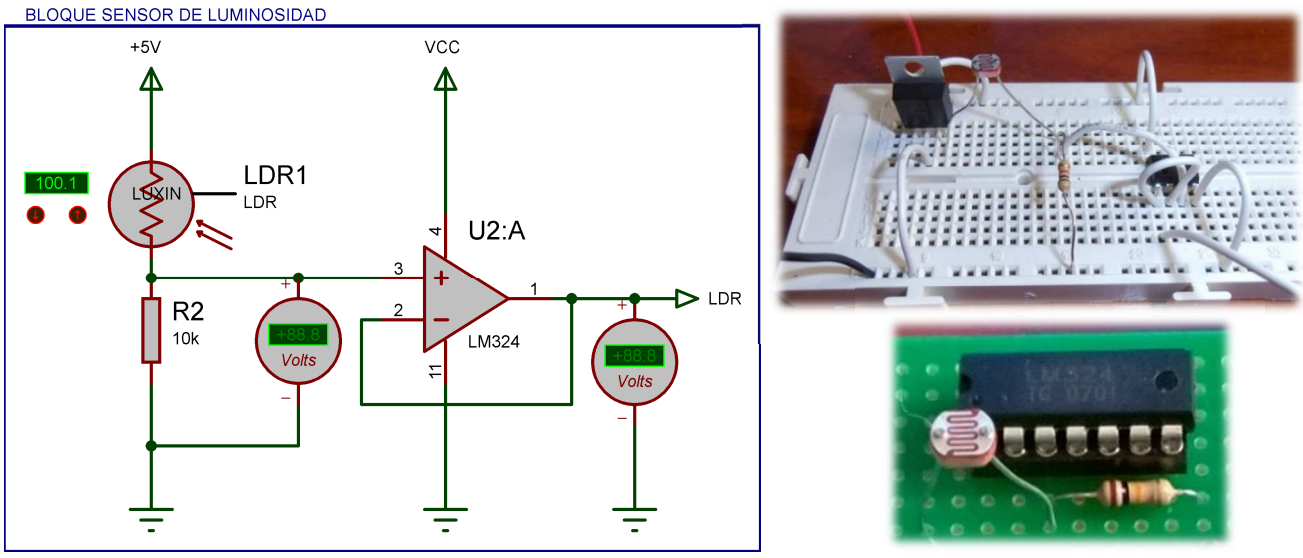


Figura 20. Circuito sensor de luz en Proteus, protoboard y placa

[3] En la *protoboard* se ha añadido un regulador de voltaje 7805 para regular la entrada al divisor de tensión.

La resistencia que completa el divisor de tensión es de $10k$, que es un valor que nos permite adaptar la salida del AO a todo el rango de ADC del PIC. La alimentación del circuito se especificará en el apartado "Diseño de la fuente de alimentación". En el circuito en *Proteus* hemos añadido dos voltímetros solo para simulación.

Con estos esquemas vamos a simular dos escenarios, oscuridad y luz intensa, y comparar los resultados teóricos según los datos del fabricante con los medidos en *Proteus* y con el multímetro en la *protoboard*:

Escenario	Lux	Salida V		
		Valor teórico	Proteus	*Protoboard*
Oscuridad	0,1	$\sim 0V$	$0{,}05V$	$0{,}14V$
Luz intensa	100	$\sim 4V$	$4{,}16V$	$4{,}13V$

Tabla 8. Simulación sensor de luz

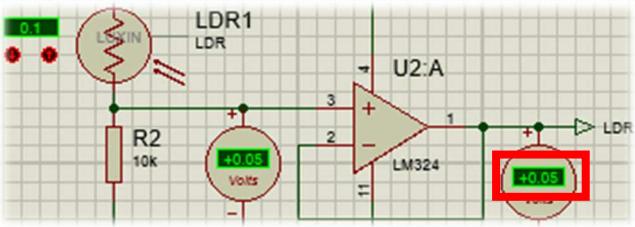
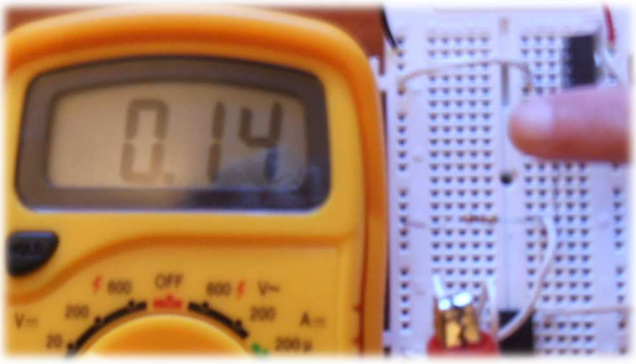
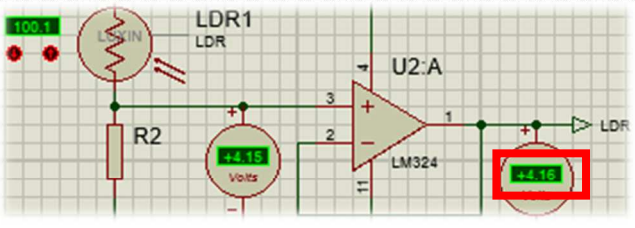
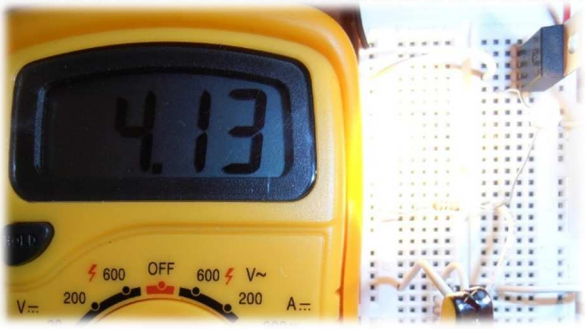
	Simulación en *Proteus* para escenario de oscuridad $(0{,}1\ lux)$ = $\mathbf{0,05V}$
	Medida con multímetro para escenario de oscuridad (tapando el LDR con un dedo) = $\mathbf{0,14V}$
	Simulación en *Proteus* para escenario de luz intensa $(100\ lux)$ = $\mathbf{4,16V}$
	Medida con multímetro para escenario de luz intensa (aplicando luz de linterna) = $\mathbf{4,13V}$

Figura 21. Simulaciones sensor luz en Proteus y protoboard

3.3. ADAPTACIÓN DE LA SEÑAL RS232

Nuestro panel de ledes mostrará la información que llegue a través de otro equipo conectado por medio de una entrada de canal serie RS232. Por lo tanto, el sistema estará continuamente pendiente de este canal para aceptar el nuevo mensaje que se ha de mostrar.

La señal RS232 deberá, en consecuencia, ser adaptada a los niveles del microprocesador. Con este fin, debemos buscar un circuito integrado que realice dicha adaptación.

Para esta tarea hemos seleccionado el *MAX232* de *Maxim*, que es un circuito integrado que se utiliza precisamente para adaptar los niveles requeridos en una conexión donde interviene un dispositivo que maneja niveles de tensión TTL (como el microcontrolador que utilizaremos) y otro que trabaja bajo la norma $EIA/TIA-232E$ y las $V.28/V.24$ (el RS232). Los niveles de tensión de nuestro PIC operan entre los $0V$ y $5V$ y la norma RS232 entre los $-12V$ y los $+12V$. La ventaja del *MAX232* es que está especialmente diseñado para trabajar en equipos que utilicen bajos niveles de tensión ($5V$), y además necesita pocos componentes externos para lograr un funcionamiento óptimo. Por otro lado, la hoja de producto incluye todos los esquemas y valores de los componentes externos necesarios, lo cual sin duda facilita nuestra tarea.

Lo que hace internamente el *MAX232* es obtener las tensiones que se requieren a ambos lados de los circuitos que enlaza y cuenta con cuatro adaptadores-inversores de tensión: dos son utilizados para la conversión TTL – RS232 y los otros dos para la operación inversa.

En el siguiente apartado mostraremos un esquema del circuito donde vemos la conexión entre los puertos del PIC y el RS232 a través del *MAX232*.

3.4. DISEÑO DEL CIRCUITO ELÉCTRICO PARA LA ADAPTACIÓN DE LA SEÑAL RS232

Como se ha comentado en el apartado anterior, vamos a utilizar un circuito integrado *MAX232* para realizar la adaptación de los niveles del PIC a los niveles de tensión del dispositivo RS232.

El *datasheet* del fabricante nos explica con detalle las características eléctricas del dispositivo, e incluye un esquema con los pines y valores apropiados de los capacitadores, como vemos en la imagen:

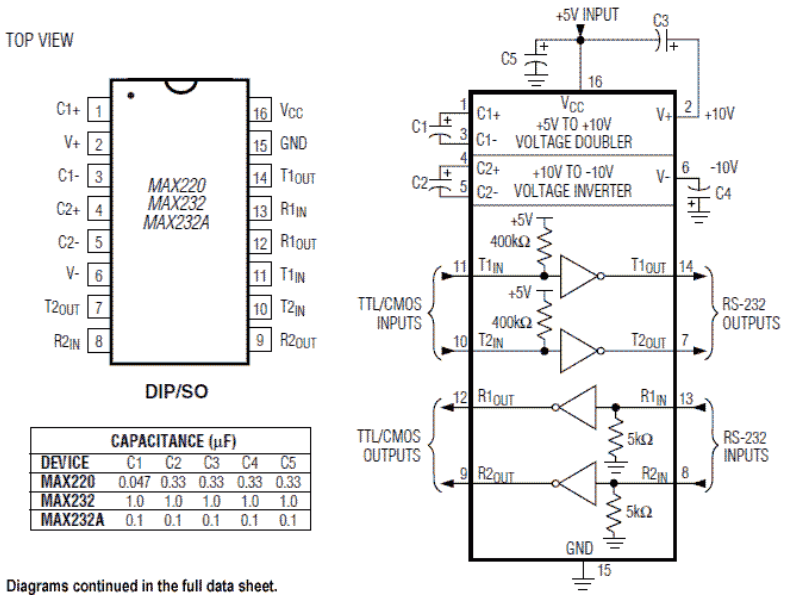


Figura 22. Diagrama Maxim MAX232.

De modo que el circuito en el que conectaríamos el *MAX232* al canal R232 y al PIC, sería:

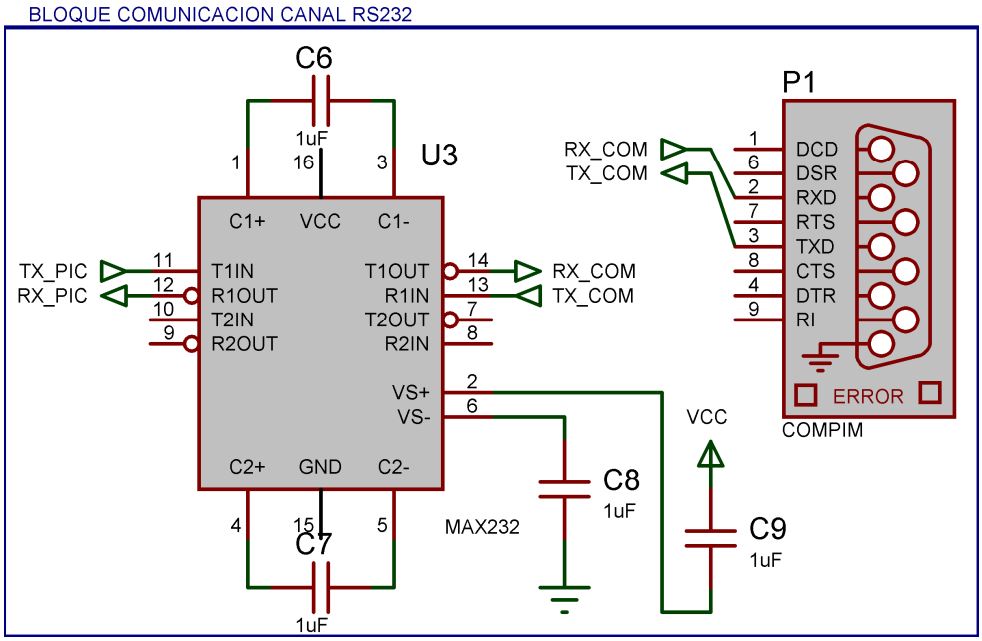


Figura 23. Circuito adaptación RS232.

Los circuitos equivalentes en *protoboard* y en placa definitiva los podemos ver a continuación:

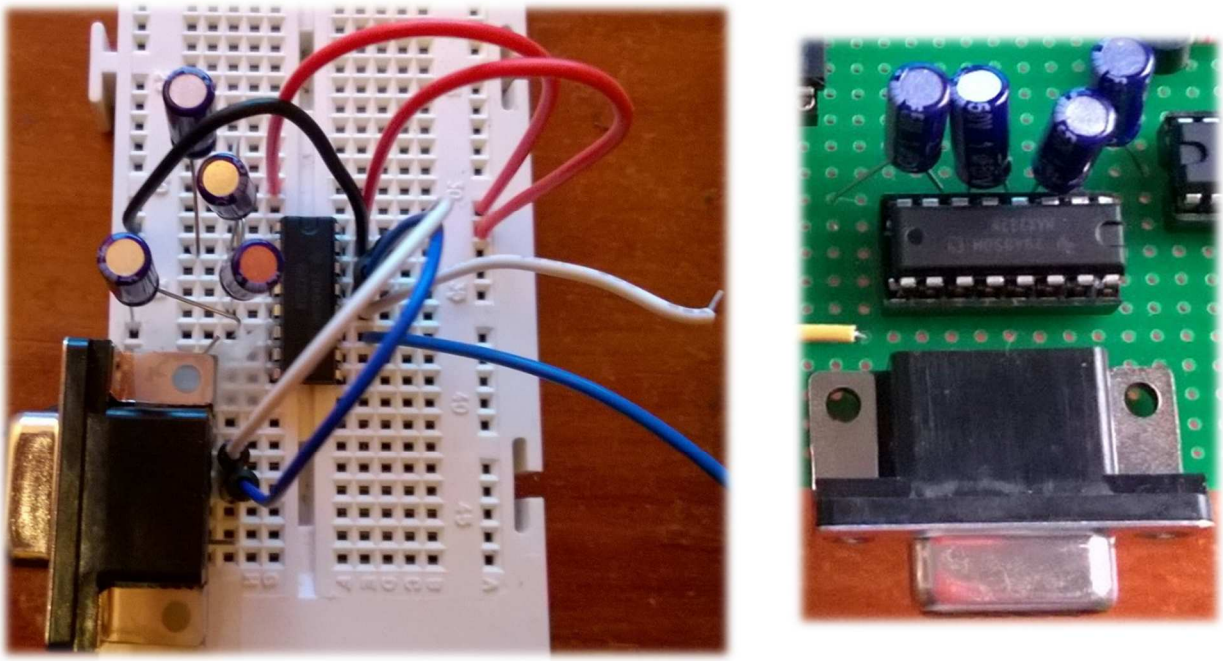


Figura 24. Circuito adaptación RS232 en protoboard y placa

Las simulaciones oportunas se realizarán cuando programemos el PIC en la sección “Programación del microcontrolador”.

4. CIRCUITO DE EXCITACIÓN DE LOS LEDES Y DRIVERS

El panel va a ser una matriz de 16 filas por 16 columnas de ledes de color rojo. Por lo tanto, tendremos un total de 256. Debemos diseñar el circuito de excitación y los *drivers* apropiados para poder controlarlos.

Si siguiéramos el esquema básico de funcionamiento del led con un microcontrolador, tendríamos que conectar el ánodo al PIC y el cátodo a una resistencia. El led encendería cuando el estado del pin del PIC fuera 1 y se apagaría cuando fuera 0. En nuestro caso esto no es una posibilidad, ya que necesitaríamos un pin para cada led, es decir, un PIC con 256 pines de salida.

Para evitarlo vamos a recurrir al multiplexado, técnica con la que utilizando unos pocos pines del microcontrolador podemos controlar una serie de circuitos integrados que se encarguen de excitar los ledes.

Estos integrados gestionarán la matriz a dos niveles: filas y columnas. En nuestro panel los ánodos están conectados en serie entre sí en grupos de 16 filas y los cátodos en grupos de 16 columnas. De esta forma, para que un led encienda, tiene que recibir un 1 por su fila y un 0 por su columna.

En estas imágenes tenemos un ejemplo de cómo están conectadas las filas y las columnas:

En cada una de las tres filas ($F1, F2, F3$) tenemos tres ledes con sus ánodos conectados en serie a su fila.

A su vez, tenemos tres columnas ($C1, C2, C3$), y en cada una de ellas hay tres ledes con sus cátodos conectados en serie a su columna.

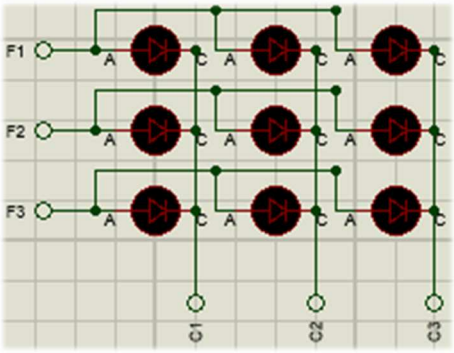


Figura 25. Detalle conexión filas y columnas

Para encender el led de la fila 2 - columna 2, por ejemplo, tendríamos que enviar un 1 a la fila 2 (o $5V$) y un 0 a la columna 2 (o conectarlo a masa).

Si en el mismo escenario enviáramos un 0 a las tres columnas (o las conectáramos a masa), los tres ledes de la fila 2 se encenderían.

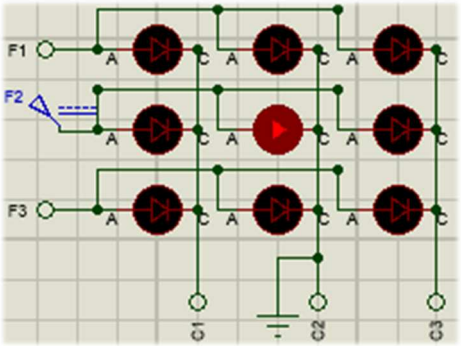


Figura 26. Detalle de led encendido

Para poder representar una imagen, forma o carácter lo que haremos será ir encendiendo las filas, una a una (enviando un 1), colocando un 0 en las columnas cuyos ledes corresponda encender (el modo de hacerlo lo explicamos más adelante en esta sección).

Haciendo esto a la velocidad apropiada no se notarán parpadeos. Si nos acogemos al principio por el cual el ojo humano mantiene la última imagen vista durante unos $25ms$, podemos ir mostrando fila a fila pero con la suficiente rapidez para que parezca que todas las filas del panel están encendidas al mismo tiempo.

Respecto al brillo de los ledes, hemos visto que el modelo seleccionado en el apartado "Selección de la mejor opción de led" se alimenta con unos $2V$ y requiere unos $20mA$. Como en la matriz tenemos 16 filas, cada led solo estará encendido una dieciseisava parte del tiempo, por lo que su brillo será dieciséis veces inferior al normal. Por ello, podríamos hacer circular una corriente mayor a la nominal sin que se dañen, ya que el tiempo que estará encendido cada uno será muy escaso. Además, el nivel de luminosidad captado por nuestro sensor LDR va a determinar también el tiempo de encendido, que solo en situación de plena luz se acercará al 100% del ciclo. En el apartado "Programación del microcontrolador" veremos cómo hemos implementado el control de tiempos en función del sensor de luz.

Pasamos a explicar en detalle cada uno de los niveles para entender el funcionamiento del panel:

Filas: tenemos 16 filas de 16 ledes cada una, con los ánodos conectados en serie a su fila correspondiente. En cada momento solo puede haber encendida (o activada) una única fila.

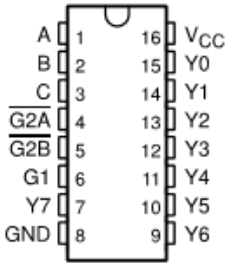
Para controlarlas utilizaremos un decodificador/demultiplexor, en concreto el *74HC138* de *Texas Instruments*. Inicialmente se planteó conectar cada fila a un pin de salida del PIC. Sin embargo, y aunque se pueden elegir microprocesadores con pines suficientes, la solución no sería escalable. Si posteriormente quisiéramos ampliar el tamaño de la matriz y colocar más filas, esto podría ser un problema. Con el demultiplexor solo necesitamos un total de cinco patillas.

Tiene 8 bits de salida para 3 entradas, por lo que colocaremos dos de ellos para manejar las 16 filas. Sus características principales son:

- Opera a temperaturas de entre $0℃$ y $+70℃$.
- El rango del voltaje es de $4{,}5V$ a $5{,}5V$.
- Tiene 8 pines de salida.
- Se pueden conectar varias unidades al PIC.
- Tiempo de respuesta muy rápido.

Figura 27. TI 74HC138.

La relación de pines y su correspondencia la podemos consultar en la hoja de producto:



Símbolo	Pin	Descripción
$G1 - \overline{G2A} - \overline{G2B}$	4-5-6	Enable
A-B-C	1-2-3	Selección
Y0 a Y7	7;9-15	Salidas
GND	8	Tierra (0V)
Vcc	16	Entrada de voltaje

Figura 28. 74HC138. Pines.

La utilidad de estos integrados en nuestro circuito es que nos van a servir para expandir puertos. Conectando las entradas A, B y C a tres pines del PIC podemos hacer la selección de fila, por lo que a partir de tres patillas controlamos ocho filas. En la sección "Programación del microcontrolador" veremos el algoritmo de control, y cómo utilizando un solo pin para cada entrada (uno para A, otro para B y un tercero para C) podemos controlar directamente los dos integrados y las 16 filas.

La tabla de verdad del decodificador nos ayudará a entender mejor su funcionamiento:

INPUTS						OUTPUTS							
ENABLE			SELECT										
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

Figura 29. 74HC138. Tabla de verdad.

Por lo tanto, tiene tres entradas de activación (*Enable*), dos de ellas activas a nivel bajo y una activa a nivel alto. Cuando está activo ($G1$ a nivel alto y tanto $\overline{G2A}$ como $\overline{G2B}$ a nivel bajo), a partir de tres bits de entrada (A, B, C) ofrece ocho salidas mutuamente excluyentes, activas a nivel bajo. La salida que esté a nivel bajo determinará la selección de fila. Para nuestro equipo no necesitamos utilizar las tres entradas de habilitación, sino solo una. Las otras dos las tendremos conectadas a masa.

El motivo por el que elegimos un componente con salida activa a nivel bajo es porque entre el integrado y el ánodo de la fila vamos a añadir una etapa de potencia a través de transistores PNP, y estos solo conducen cuando no hay corriente en su base. Por lo tanto, aunque lo que hay en la salida de la fila activa es un 0, lo que llega al ánodo es un 1. La etapa de potencia es necesaria para garantizar que el circuito podrá manejar la corriente requerida, teniendo en cuenta que existe la posibilidad de que en algún momento estén encendidos varios ledes al mismo tiempo. Para este fin utilizaremos un transistor en cada una de las filas.

El modelo elegido para el proyecto es el *BC327*, de tipo PNP, y entre sus características destacamos:

- Tensión emisor-colector hasta $45V$.
- Corriente de colector máxima $500mA$.
- Disipación máxima $625mW$.
- Trabaja en frecuencias de hasta $100MHz$.
- hFE de 100.

Figura 30. Transistor BC327.

Además, y para proteger los ledes, hemos de añadir resistencias entre el demultiplexor y el transistor. Su valor lo hemos calculado al final de esta sección (Ecuación 3).

Columnas: tenemos 16 columnas de 16 ledes cada una, con los cátodos conectados en serie a su columna correspondiente. En cada momento solo puede haber encendido un led de cada columna (puesto que el panel encenderá fila a fila). Para que al encender una fila determinada podamos tener distintos valores en las columnas (unos o ceros, según qué ledes deban encenderse), hemos de utilizar unos integrados en los que podamos introducir los datos que correspondan.

Este multiplexado se puede implementar de varias maneras y entre ellas hemos valorado el registro de desplazamiento y el *latch*. Aunque en un primer momento se implementó el circuito con registros (en concreto el *74HC164*), finalmente nos quedamos con el uso de un dispositivo con *latch*.

El inconveniente del registro de desplazamiento tradicional es que introduce y muestra los datos bit a bit, por lo que si el encendido y apagado de filas no se hace suficientemente rápido se pueden notar desfases

en la imagen o parpadeos. Utilizando un *latch* (como por ejemplo, el integrado *595*) se puede evitar este riesgo, ya que solo muestra la salida cuando están todos los datos introducidos. El integrado que nos interesa es el de tipo serial-paralelo, donde los bits entran uno a uno y salen de forma paralela.

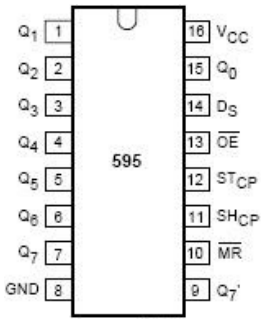
Esta lógica es muy adecuada para el proyecto, así que para nuestro panel utilizaremos el circuito *74HC595* de *NXP*, que es un registro de desplazamiento con *latch* de 8 bits. Con ellos podemos manejar una fila de 8 columnas. Como nuestro panel tiene 16 columnas, utilizaremos dos unidades en cascada.

Las características principales del *74HC595* y que nos interesan, son:

- Opera a temperaturas de entre $-40℃$ y $+125℃$.
- El rango del voltaje es de $2V$ a $6V$.
- Tiene 8 pines de salida.
- Se pueden conectar varias unidades en cascada.
- Tiene pin de "*enable*" que conectaremos a la señal PWM para el control de brillo de los ledes.

Figura 31. NXP 74HC595

De la hoja de producto podemos extraer el esquema de pines y ver su correspondencia:



Símbolo	Pin	Descripción
Q0 a Q7	1-7;15	Salidas paralelas
GND	8	Tierra (0V)
Q7´	9	Salida serie
$\overline{MR}$	10	*Master Reset*
SHCP	11	Señal reloj registro
STCP	12	Señal reloj latch
$\overline{OE}$	13	*Enable*
DS	14	Entrada de datos
Vcc	16	Entrada de voltaje

Figura 32. Pines 74HC595

Puesto que vamos a utilizar dos registros, la manera de conectarlos uno a continuación del otro será uniendo el pin D_S del primer *latch* con el pin $Q_{7'}$ del siguiente:

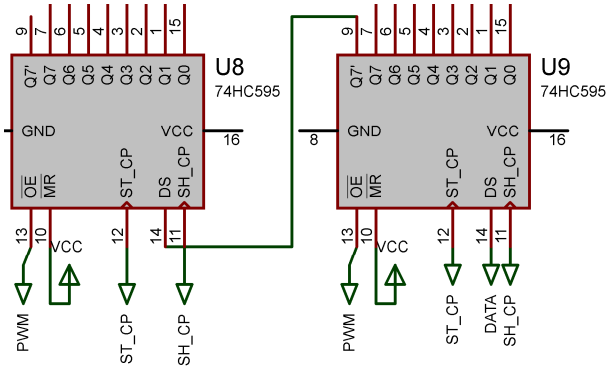


Figura 33. Conexión 74HC595 en cascada

Siguiendo este esquema, programaremos el PIC para encender los ledes correspondientes introduciendo ceros y unos en los integrados. De esta forma controlaremos las 16 columnas de una fila.

Su modo de funcionamiento es el siguiente: los datos ingresan de forma serial por el pin 14(D_S) cada vez que el pin SH_CP pasa de estado bajo a alto. Quedan guardados en la memoria del integrado hasta que la señal ST_CP pasa de bajo a alto, momento en que las salidas $Q_0 - Q_7$ muestran el estado de cada bit.

Por lo tanto, para que aparezca un byte en las salidas, el algoritmo, complementado con la imagen, es:

1. Se pone el pin D_S en el estado del bit que se quiera ingresar. Como el cátodo va conectado a la columna, un 0 hará que el led se encienda y un 1 que esté apagado.
2. Se coloca el pin SH_CP en bajo.
3. Se coloca el pin SH_CP en alto.
4. Se repite el proceso hasta enviar los 8 bits.
5. Se coloca el pin ST_CP en bajo.
6. Se coloca el pin ST_CP en alto.

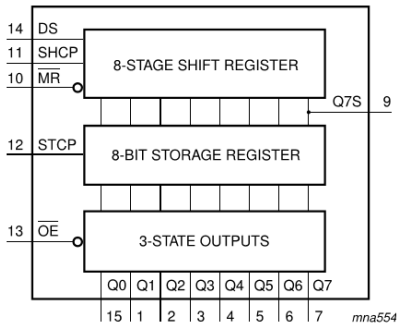


Figura 34. Diagrama funcional 74HC595

Al igual que hicimos con las filas, también debemos proteger los ledes con resistencias. Su valor lo hemos calculado más adelante en la Ecuación 4.

Una vez hemos definido el circuito a nivel físico, y aunque lo veremos con total detalle en el apartado "Programación del microcontrolador", resumimos la lógica de control del panel en estos pasos:

1. El PIC limpia todas las filas y todas las columnas.
2. Para cada carácter que queramos mostrar introducimos su representación en valores binarios en los registros. Cuando están todos los valores introducidos se activa la señal del *latch* y pasa a las salidas.
3. Se activa el decodificador del grupo de filas que corresponda y se envían a sus puertos $A - B - C$ los tres bits que determinan la fila que debe encenderse (a nivel bajo).
4. Los dos pasos anteriores se repiten para todas las filas mientras haya caracteres que representar.

Resumen del circuito:

- El panel está formado por 256 ledes *Kingbright KPTL-3216SURCK-01*, dispuestos en 16 filas y 16 columnas. Los ánodos están conectados a los decodificadores que controlan las filas a través de una resistencia y un transistor y los cátodos están conectados a los registros a través de una resistencia.
- Las filas de la matriz están conectadas al PIC a través de dos decodificadores (*74HC138*), con salida activa a nivel bajo. Para controlar el voltaje y proteger el circuito la conexión entre la fila y el *74HC138* se hará con un transistor PNP (*BC327*), que invierte la señal del *138* y que proporciona la corriente suficiente para encender los ledes de la fila que correspondan. Además, debemos colocar una resistencia entre ellos. En la Tabla 9 de este apartado hemos calculado que el consumo máximo posible, en caso de que todos los ledes de una fila estén encendidos al mismo tiempo con un consumo de $20mA$ cada uno, es de $320mA$. También sabemos por la hoja de producto que la caída de tensión sobre el transistor es de $0{,}7V$ y que su hFe (ganancia) es de 100, por lo que:

$$R = \frac{V - V_{BE}}{\frac{I}{hFe}} = \frac{5 - 0.7}{\frac{0{,}32}{100}} = 1.344\Omega$$

Ecuación 3. Cálculo valor resistencia transistor

De modo que la resistencia que colocaremos será de $1\text{K}5\Omega$.

- Las columnas de la matriz se gestionan mediante dos registros de desplazamiento con *latch* (*74HC595*) de 8 bits de longitud cada uno, conectados en cascada. El PIC genera los pulsos de reloj y datos y los envía al integrado. Este los guarda en su memoria intermedia hasta que se llena, momento en que con la

señal *latch* pasan a las salidas, que seleccionan las columnas que deben estar activas. Un 0 enciende el led y un 1 lo mantiene apagado. Cada salida de los *74HC595* se conecta a una columna de la matriz a través de una resistencia. Tenemos en cuenta que debido al multiplexado, a nivel de columna solo puede haber encendido al mismo tiempo un led, ya que las filas se encienden una a una, por lo que el valor calculado de la resistencia es:

$$R = \frac{V}{I} = \frac{5 - 1{,}95}{0.020} = 152{,}5\ \Omega$$

Ecuación 4. Cálculo valor resistencia columnas

Atendiendo al valor de las resistencias comerciales, incluiremos una de $180\ \Omega$.

- El pin D_S del registro/*latch* se controla desde un pin de salida del PIC que definiremos en su momento y los pulsos de reloj los proporcionarán otros dos pines del PIC.
- Los ledes brillarán con mayor o menor intensidad en función de la luz ambiente. Esto lo controlaremos encendiéndolos más o menos tiempo. El módulo PWM del PIC generará un pulso que marcará cuánto tiempo debe estar encendido el led y estará conectado al pin *enable* del *latch*. Cuando la señal de salida sea alta, el sistema funcionará con normalidad y el led estará encendido. Cuando la señal de salida sea 0 se limpiará el contenido del *latch* y el led se apagará. Como el *enable* del *latch* se activa a nivel bajo, incluimos una puerta inversora a la salida PWM del PIC (un integrado *74HC14*).
- El circuito de alimentación lo desarrollaremos en el apartado "Diseño de la fuente de alimentación", pero previamente nos hemos asegurado de que el sistema podrá aguantar. El multiplexado nos garantiza que en cada momento solo puede haber una fila encendida, y generalmente nunca estarán todos los ledes de esa fila encendidos simultáneamente. Pero aún en este caso hipotético el consumo calculado sería:

Mínimo	Medio	Máximo
0.010 x 16 = 160mA	0.020 x 16 = 320mA	0.030 x 20 = 480mA

Tabla 9. Consumo mínimo, medio y máximo por fila de ledes

Definimos el consumo mínimo, medio y máximo en función de las características de nuestro led, y teniendo en cuenta que dependiendo de la luminosidad captada por el sensor LDR, el led estará encendido más o menos tiempo.

- La distancia entre ledes se ha definido teniendo en cuenta que el tamaño y la distancia a la que se ha de ver un panel es proporcional a la intensidad luminosa del píxel y a la distancia entre puntos. Así por ejemplo, un panel cuyos ledes están a $15mm$ se pueden ver aproximadamente a $15\ metros$.
- Para la construcción del equipo físico, a excepción de las matrices de 8x8, el resto de los componentes utilizados son los mismos que los detallados en esta sección.
- A continuación mostramos el esquema completo del bloque de drivers en *Proteus* y el montaje en *protoboard* y placa:

BLOQUE DRIVERS DE LEDES

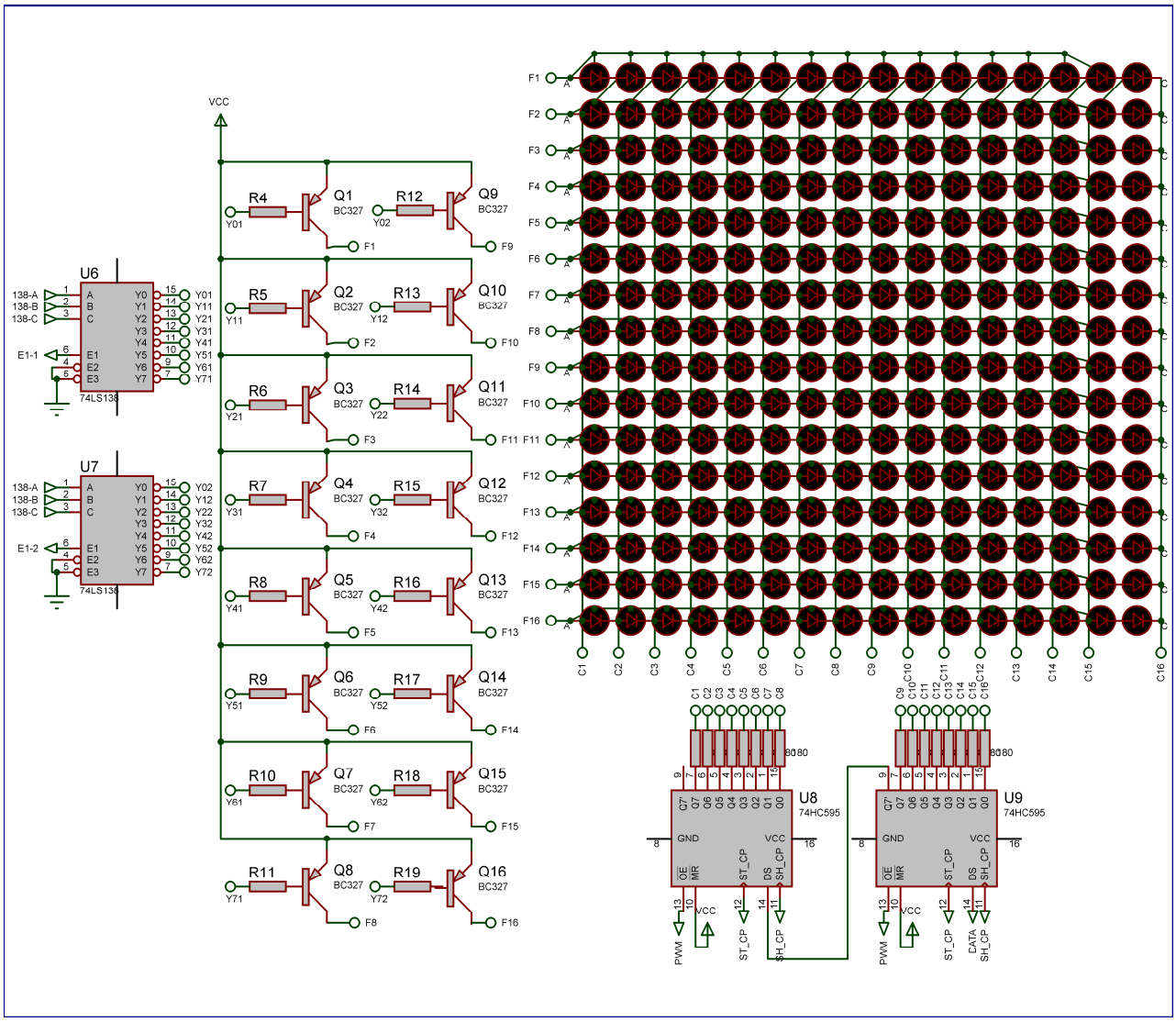


Figura 35. Circuito excitación ledes

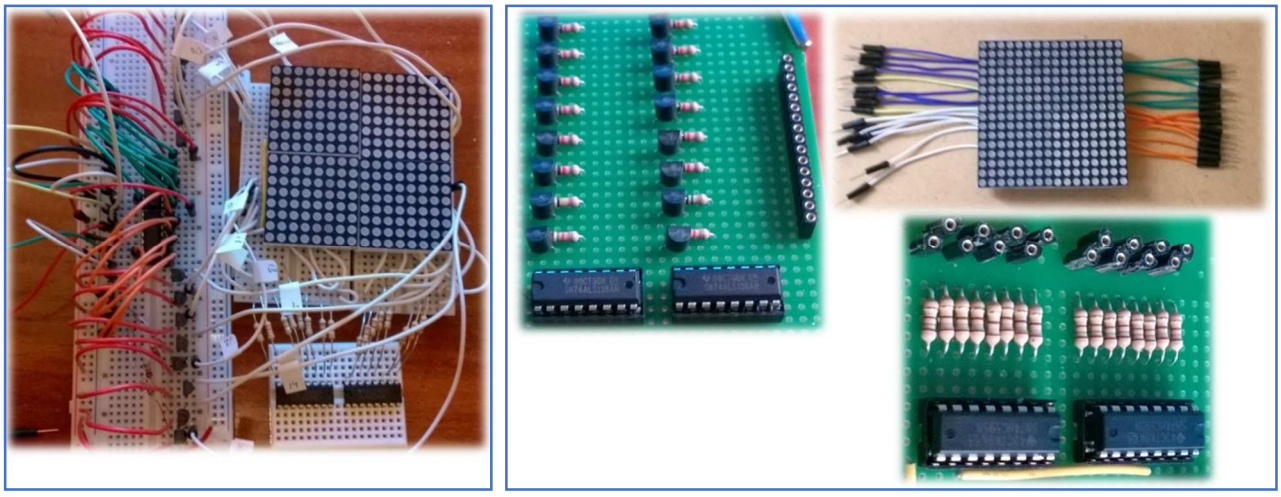


Figura 36. Circuito excitación ledes en protoboard y placa

5. ELECCIÓN DEL PROCESADOR Y PROGRAMA DE CONTROL

5.1. ESTUDIO DE MICROCONTROLADORES

Un microcontrolador es un circuito integrado programable que contiene los componentes necesarios para controlar una tarea determinada. Para ello necesita un programa que dirige el funcionamiento del mismo y una memoria donde se almacena dicho programa. Por lo tanto, seleccionaremos un microprocesador y lo programaremos para que ejecute las tareas requeridas, que en resumen son: leer el sensor de luz, recibir información a través de un dispositivo conectado al puerto serie, y representar en la matriz de ledes esta información, ajustando el brillo a las condiciones externas.

En el mercado existen numerosos fabricantes de microcontroladores, entre los cuales podemos citar a *Altera, Freescale* (antes *Motorola*), *Intel, National Semiconductor, Microchip, NXP o Texas Instruments*. Cada uno de ellos, además, tiene varias familias y líneas de producto. Esto nos da una idea de la cantidad de opciones que tenemos. Para acotar la búsqueda nos centraremos en *Microchip*, debido a que su uso está muy extendido y a que ofrece muchas herramientas y ejemplos para el desarrollo del programa.

Sus microcontroladores PIC pertenecen a la familia del tipo RISC, arquitectura que se basa en instrucciones cortas (admiten entre 33 y 60), por lo que son muy rápidos y eficaces. Son de bajo coste y ofrecen buenas prestaciones, ya que algunos modelos disponen de memoria de gran capacidad (RAM, EEPROM, FLASH) y una gran cantidad de dispositivos periféricos integrados, como módulos de control PWM, osciladores internos, convertidores A/D o módulos de comunicación (USART, SPI, I2C, USB).

La arquitectura del procesador sigue el modelo *Harvard*, donde la memoria de programa o instrucciones es independiente de la de los datos, con tamaños y longitudes de palabra diferentes. Esto permite a la CPU acceder simultáneamente a las dos memorias.

Otro aspecto destacable es la segmentación en la ejecución de las instrucciones, técnica con la que el procesador puede realizar al mismo tiempo la ejecución de la instrucción y la búsqueda del código de la siguiente, por lo que se puede ejecutar la instrucción en un ciclo sin tener que esperar los cuatro necesarios.

Los PIC tienen una arquitectura basada en bancos de registros y todos los elementos del sistema están implementados físicamente como tales. Su tensión de alimentación típica es $5V$ y las corrientes de salida por pin están comprendidas entre 20 y $25mA$.

En el próximo apartado veremos qué requisitos tenemos para nuestro sistema, y en función de ellos, seleccionaremos un PIC concreto de entre todas las opciones que nos ofrece *Microchip*.

5.2. ANÁLISIS DE LOS REQUISITOS DE NUESTRO SISTEMA

Debido al gran número de modelos de *Microchip*, vamos a centrar la búsqueda del más adecuado en las necesidades de nuestro sistema. En los capítulos anteriores hemos visto qué funciones tendrá internamente y qué componentes externos conectaremos. Esto nos lleva a la siguiente lista de requisitos:

- Conversor A/D y módulo PWM: la salida del sensor LDR irá conectada a un pin analógico y necesitamos que el PIC disponga de conversor A/D para tratar la señal y convertirla en digital. Además, el PIC debe tener módulo PWM, ya que gestionaremos el brillo de los ledes con modulación por ancho de pulsos.

- Pines de salida suficientes. Para los drivers de ledes, las 16 filas irán conectadas a dos demultiplexores 138, para los cuales nos hacen falta un total de 5 pines. A nivel de columnas, vamos a utilizar dos registros de desplazamiento 595 en cascada, así que utilizaremos otros 4 pines de salida.
- USART: el canal RS232 irá conectado al PIC a través de un integrado *MAX232* y necesitamos dos pines (TX y RX) para la transmisión y recepción de datos.
- Memoria suficiente: en lugar de utilizar un módulo de memoria externo aprovecharemos las capacidades del PIC para gestionar internamente los datos que se reciben por el canal serie. Esta gestión incluye almacenar la información y transformar el código del carácter que recibimos por su valor correspondiente para representarlo en la matriz. La traducción la haremos en base a mapas que irán almacenados en el PIC.

En la página web de *Microchip* disponemos de un potente buscador por características. Nuestra primera decisión es utilizar un PIC de 8 bits, ya que no necesitamos un tamaño de palabra mayor. Atendiendo a los requisitos anteriores y después de introducirlos en la búsqueda y analizar los resultados, nos decantamos por el **PIC16F877A**. Pertenece a la familia PIC16, de rango medio y tiene muy buenas prestaciones.

Del manual de uso proporcionado por el fabricante, extraemos sus características principales, a partir de las cuales comprobamos que cumple todos nuestros requisitos:

Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
Bytes	# Single Word Instructions						SPI	Master I^2C			
14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Tabla 10. Características PIC16F877A

De cara a que la programación del PIC sea más directa en el próximo apartado, mostramos los pines que utilizaremos, así como su aspecto físico:

Pin	Función
AN0	Entrada analógica de la señal del sensor LDR. Conversor AD.
RD0	Señal DS para el registro de desplazamiento.
RD1	Señal de *latch* para el registro de desplazamiento.
RD2	Señal de reloj para el registro de desplazamiento
RD3	Señal para seleccionar las primeras 8 filas.
RD4	Señal para seleccionar las últimas 8 filas.
RC2	Pulso PWM para *reset* del registro de desplazamiento.
RB0-RB2	Pines de salida para las filas.
RC6	Pin para transmisión de datos RS232 (TX).
RC7	Pin para recepción de datos RS232 (RX).
OSC1-2	Oscilador de cristal externo.
MCLR	Circuito de *reset*.

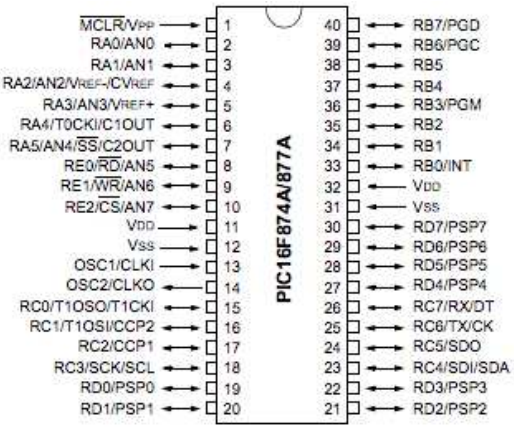


Figura 37. Relación de pines PIC16F877A

El esquema completo del bloque en *Proteus* y el montaje en *protoboard* y placa son los siguientes:

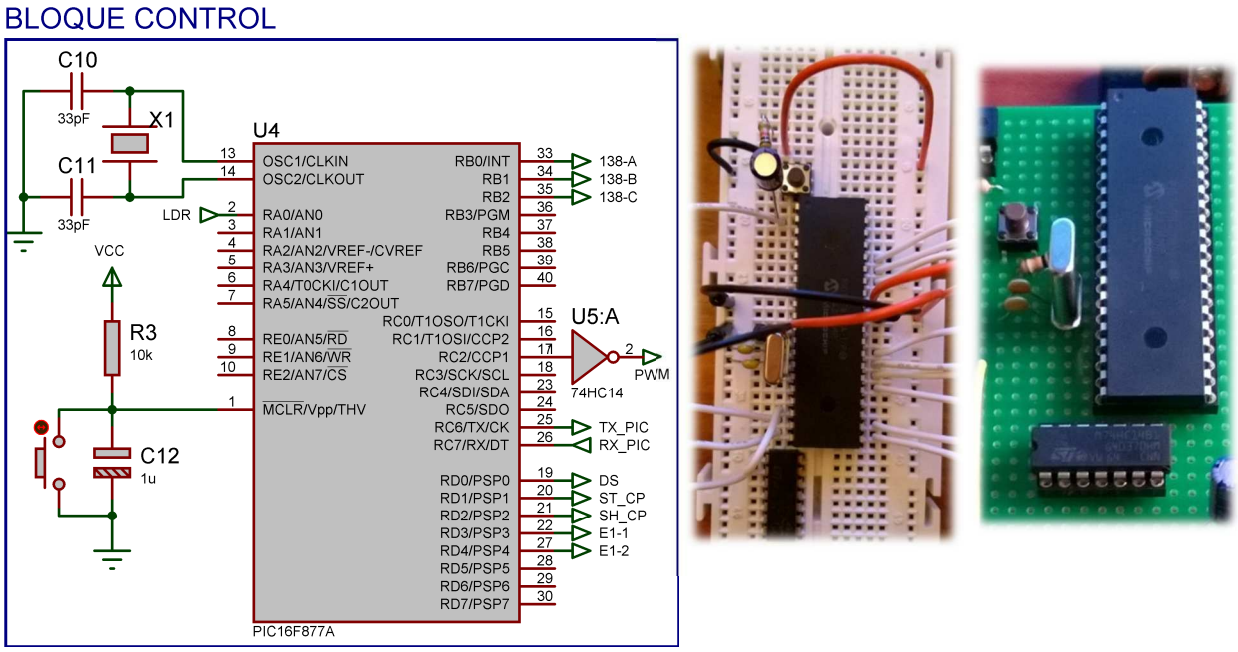


Figura 38. Esquema conexión PIC16F877A en Proteus, protoboard y placa

Se utiliza un cristal externo de $20MHz$ que nos dará bastante precisión. Por su parte, los capacitadores aumentarán la estabilidad y protección del sistema. Sus valores, así como el circuito de *reset,* se han tomado de la hoja de especificaciones de *Microchip*.

5.3. PROGRAMACIÓN DEL MICROCONTROLADOR

El lenguaje de programación que utilizaremos para el PIC es C, por ser un lenguaje que conocemos y para el que existe mucha documentación y ejemplos proporcionados por el fabricante. Nos ayudaremos del compilador CCS y haremos las simulaciones en *Proteus*.

Ya hemos mencionado previamente las tareas que ejecutará el PIC. Para abordar su programación vamos a dividir esta sección en partes, y explicaremos el programa principal y cada una de las tres tareas. Además, el código estará comentado para mayor referencia:

- **Cabecera del programa**: en la cabecera indicamos el modelo de PIC que vamos a usar, las librerías externas, los *fuses*, la frecuencia del reloj y definimos los puertos que vamos a utilizar.

```
#include <16F877A.h>                    // PIC que vamos a utilizar.
#device adc=10                          // Configuramos conversor AD a 10 bits.
#fuses NOWDT, HS, PUT, NOPROTECT        // Fuses:
                                        // NOWDT: WatchDog desactivado (aplicación no comercial).
                                        // HS: activamos cristal de alta frecuencia.
                                        // PUT: Power Up Timer activado para asegurar un arranque estable.
                                        // NOPROTECT: protección contra la lectura de la Flash desactivada.
#use delay(clock=20000000)              // Frecuencia a 20Mhz.
// Configuración de pines del PIC
#define DS PIN_D0                       // Pin conectado a DS del 595.
#define ST_CP PIN_D1                    // Pin conectado al latch del 595.
#define SH_CP PIN_D2                    // Pin conectado al Clock del 595.
#define E11 PIN_D3                      // Pin conectado al primer demultiplexor de las filas.
#define E12 PIN_D4                      // Pin conectado al segundo demultiplexor de las filas.
#define PWM PIN_C2                      // Pin conectado al enable del 595. Pulso PWM.
// Inclusión de librerías
#include <string.h>                     // Librería String.h
```

- **Programa para leer el sensor LDR y adaptar la luminosidad de los ledes**:

(Nota: todos los detalles sobre el bloque LDR se encuentran en el apartado "Estudio del sensor de luz").

Algoritmo: esta parte del programa leerá de una entrada analógica el valor de voltaje que arroja el sensor LDR, lo convertirá en términos de luz y en función del resultado hará que los ledes estén encendidos más o menos tiempo, de modo que brillarán más o menos según cuál sea el nivel de luz ambiental.

Para realizar estas tareas utilizaremos dos módulos del PIC:

- El **módulo AD** para convertir el valor de la entrada a su correspondiente valor digital. Su resolución es de 10 bits, por lo que el valor mínimo que podemos obtener es 0 y el máximo 1023.

El fabricante nos indica que el módulo tiene ocho entradas análogas, que son multiplexadas dentro de un circuito de muestreo y retención. La salida del multiplexor es la entrada al convertidor, que genera el resultado por medio de aproximaciones sucesivas. Este será un valor entre 0 y 1023, y nos indicará el nivel de luminosidad ambiente. Como hemos visto, a mayor luminosidad, mayor será el brillo de los ledes.

Para utilizarlo, en primer lugar indicamos la resolución del ADC en la cabecera:

```
#device adc=10                    // Configuramos conversor AD a 10 bits.
```

Dentro del programa principal señalamos qué pin recibe la señal analógica, qué reloj utilizará el módulo y qué canal:

```
setup_adc_ports(AN0);             // Configuramos RA0 como pin Analógico
setup_adc(adc_clock_internal);    // Indicamos el reloj RC
set_adc_channel(0);               // El canal del ADC será el pin 0
```

Finalmente, solo queda hacer la lectura de la conversión y guardarla en una variable:

```
luminosidad = read_adc();              // Valor de la conversión de la señal del LDR
voltaje = (5.0/1023.0)*(luminosidad);  // Valor del voltaje
```

- El **módulo PWM**, para controlar el tiempo de encendido de los ledes. Este módulo genera una onda cuadrada con una frecuencia dada (generalmente bastante alta) y el ciclo de trabajo de la señal irá cambiando en función del resultado de la conversión del AD. PWM (o modulación por ancho de pulsos) consiste en modificar a una señal digital de frecuencia constante el tiempo en que la señal está en alto sin variar la frecuencia. Este tiempo en alto es lo que llamamos *duty*, durante el cual la señal estará a 1.

El *duty cycle* del PWM tiene una resolución de hasta 10 bits, por lo tanto, podemos establecer una correspondencia directa entre el resultado del AD (nivel de luminosidad ambiente) y el *duty*. Si la luminosidad es 1023 (valor máximo), el *duty* será del 100% (y el led estará encendido todo el tiempo que dure el ciclo). En cambio, si la luminosidad es de 512 (valor medio), el *duty* será del 50% y el led se apagará pasada la mitad del ciclo.

La señal PWM irá conectada al pin de *enable* del registro de desplazamiento 595. Ya que este pin del *latch* está activo a nivel bajo, a la salida del PIC incluimos un integrado que invierte la señal, de modo que cuando la señal PWM sea 1 al pin de *enable* le llegará un 0 y el registro funcionará con normalidad. Cuando sea 0, al pin le llegará un 1 y el *latch* pondrá a 0 todos los valores del registro, de modo que los ledes se apagarán. La puerta inversora que vamos a utilizar es un integrado genérico, el *74HC14*.

Por su parte, el parámetro fundamental del PWM es la frecuencia de modulación, cuyo inverso es el periodo. La frecuencia es programable en función de varias variables:

- La frecuencia del oscilador principal, F_{osc}, que hemos definido a $20MHz$.
- El *pre-scaler* (PRE) o divisor previo del TMR2, que es el *timer* que utiliza el PWM como base de tiempo y que puede tomar los valores 1:1, 1:4 o 1:16.
- El registro PR2, de 8 bits, asociado al *timer* TMR2.

La frecuencia y el periodo PWM se calculan como:

$$F_{PWM} = F_{OSC}/[4 \cdot PRE \cdot (PR2 + 1)] \qquad T_{PWM} = [(PR2 + 1) \cdot 4 \cdot PRE] \cdot F_{OSC}$$

Ecuación 5. Cálculo de la frecuencia y el periodo PWM

Nos interesa que la frecuencia del PWM sea alta y tener la máxima resolución del ciclo de trabajo, que son 10 bits. Por lo tanto, y teniendo en cuenta que trabajamos a $20MHz$, establecemos el valor de PRE en 1 y el de PR2 en 255, de forma que la frecuencia del PWM será:

$$F_{PWM} = \frac{20.000}{[4 \cdot 1 \cdot (255 + 1)]} = 19{,}53KHz$$

Ecuación 6. Frecuencia PWM sistema

Esta es la frecuencia máxima que podemos obtener con estos valores. Si quisiéramos tener una frecuencia mayor tendríamos que bajar el PR2 y con ello, dispondríamos de una menor resolución para el ciclo de trabajo. De hacerlo así, podríamos llegar a una frecuencia de PWM de $78KHz$.

Una vez definidos estos valores, solo nos queda habilitar el módulo CCP del PIC para que funcione en modo PWM, e indicar al programa los parámetros establecidos:

```
setup_ccp1(CCP_PWM);                // Configura el módulo CCP1 como PWM
setup_timer_2(T2_DIV_BY_1,127,1);   // Configuracion de Timer 2 para establecer frec. PWM
```

Puertos utilizados: AN0 (entrada analógica), RC2 (salida digital, PWM).

Programación: el programa completo del sensor de luminosidad es:

```
int16 luminosidad;                  // Valor de la conversión del ADC
float voltaje, duty_cycle;          // Tensión de entrada en AN0 y variable para el duty

setup_ccp1(CCP_PWM);                // Configura el módulo CCP1 como PWM
setup_timer_2(T2_DIV_BY_1,127,1);   // Configuracion de Timer 2 para establecer frec. PWM

setup_adc_ports(AN0);               // Configuramos RA0 como pin Analógico
setup_adc(adc_clock_internal);      // Indicamos el reloj RC
set_adc_channel(0);                 // El canal del ADC será el pin 0
delay_ms(50);                       // Tras el encendido del sistema o Reset

while(TRUE) {
    // Leemos la conversión y obtenemos la tensión Vout
    delay_us(20);
    luminosidad = read_adc();               // Valor de la conversión de la señal del LDR
    voltaje = (5.0/1023.0)*(luminosidad);   // Valor del voltaje
    set_pwm1_duty(luminosidad);             // Establecemos el tiempo en que el pulso está a nivel alto en cada ciclo.
    duty_cycle = (luminosidad/1023.0)*100;  // Cálculo del % de ciclo solo para simulación
    // Escribimos la información en el TX
    printf("ADC:%4ld  Volt:%4.1f Duty Cycle:%4.2f%%\n\r",luminosidad,voltaje,duty_cycle);
    delay_ms(1);                            // Periodo de muestreo: T=0,001s
```

Simulación: estamos simulando cómo responde el sensor a los cambios de luminosidad, cuál es el valor de voltaje que se lee desde el LDR, cuál es su conversión a valor de 10 bits y cómo es el pulso de salida que llega al pin de *enable* del registro. El terminal nos va devolviendo las mediciones y el osciloscopio ilustra el pulso de salida del pin C2, conectado al registro. Captamos tres situaciones: oscuridad, luminosidad media y plena luminosidad, y para cada una de ellas podemos apreciar el ancho del pulso de salida:

Nota: para la simulación eliminamos la puerta inversora a la salida del PWM para así poder ver en el osciloscopio el pulso en alto.

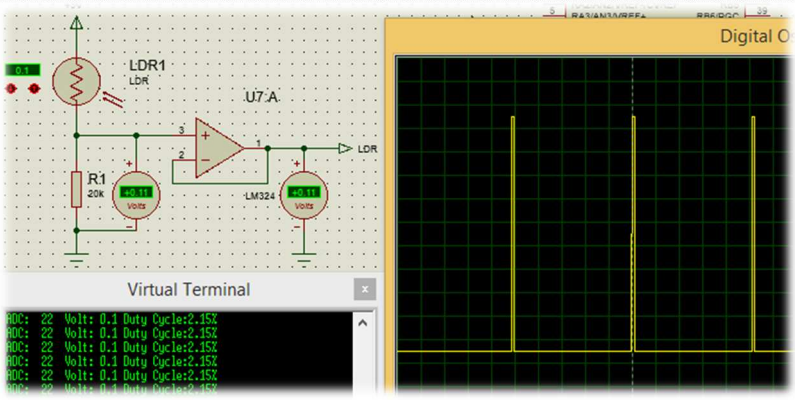
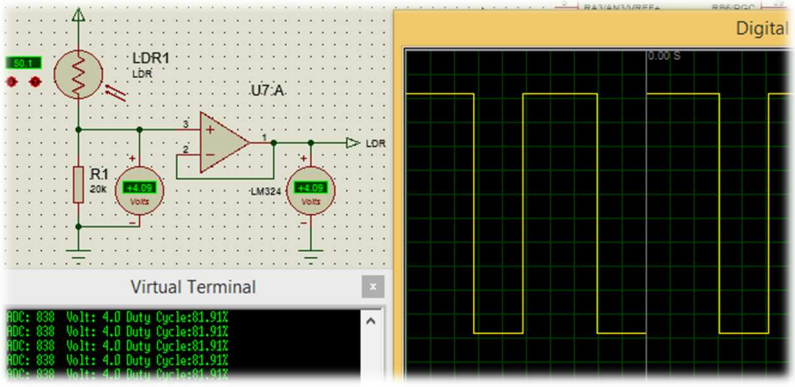
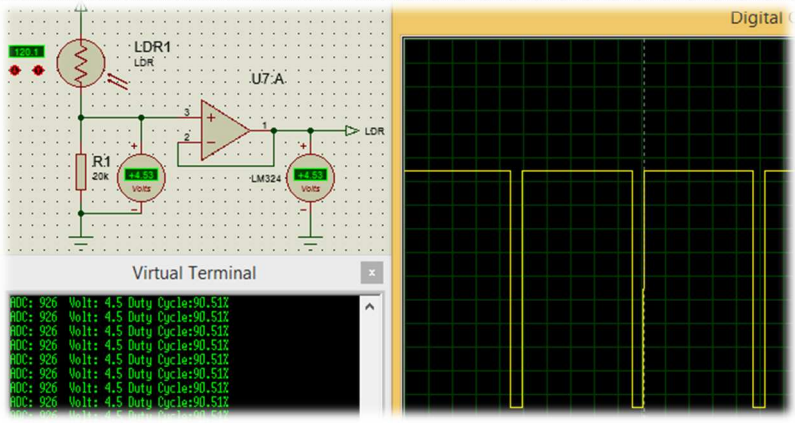
Escenario 1: oscuridad. $0{,}1\ lux$. Señal analógica: $0{,}1V$. ADC (luminosidad): 22. *Duty cycle*: $2{,}15\%$.	
Escenario 2: luz media. $50{,}1\ lux$. Señal analógica: $4{,}0V$. ADC (luminosidad): 838. *Duty cycle*: $81{,}91\%$.	
Escenario 3: luz intensa. $120{,}1\ lux$. Señal analógica: $4{,}53V$. ADC (luminosidad): 926. *Duty cycle*: $90{,}51\%$.	

Figura 39. Simulación sensor luminosidad en distintos escenarios

- **Programa para escuchar y comunicarse con el puerto serie**.

(Nota: los detalles sobre el bloque de conexión RS232 se encuentran en el apartado "Adaptación de la señal RS232").

La entrada de canal serie por la que se recibe la información que mostrará el panel implementa el protocolo MODBUS. Sin ánimo de entrar en detalles profundos sobre sus fundamentos, debemos señalar que es un protocolo de comunicación serie para la transmisión de información entre equipos conectados a un mismo bus, donde hay un maestro (*master*) y uno o varios (hasta 247) esclavos (*slaves*). El maestro envía peticiones a los esclavos, que responden o realizan las acciones requeridas. En nuestro entorno, el equipo que envía la información actúa de maestro y el PIC de esclavo.

Las peticiones y respuestas se encapsulan en tramas MODBUS, con una estructura definida. Nosotros vamos a centrarnos en la modalidad MODBUS-RTU, ya que es más eficiente que la alternativa ASCII.

MODBUS implementa una serie de funciones de lectura y escritura, y para dar respuesta a nuestras necesidades, solo nos interesa la función 10, FUNC_WRITE_MULTIPLE_REGISTERS. Esta función escribe en los registros oportunos la información que se envía. En nuestro caso, no necesitamos escribir en los registros, sino obtener el contenido que llega en hexadecimal y guardarlo en un vector. Nuestro programa, cuando reciba una trama, dirigida a él o a todos los esclavos (*broadcast*), comprobará si se corresponde con esta función, y de no ser así, no la atenderá. En caso de serlo, tratará los datos recibidos y los irá introduciendo en un vector, que posteriormente será enviado al panel para su representación.

La trama MODBUS RTU tiene esta estructura:

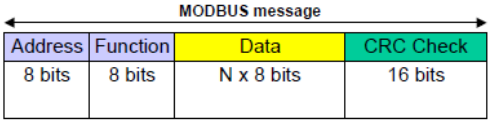


Figura 40. Estructura trama MODBUS

Donde "*Address*" es la dirección del esclavo al que se dirige la trama (o 0 si se envía a todos los equipos), "*Function*" es el código de la petición, "*Data*" son los campos con la información y "*CRC*" se utiliza para la comprobación de errores. Las librerías MODBUS disponibles con el compilador CCS incluyen funciones para verificar este campo al recibir la trama, y serán las que usemos para su control.

En la función que vamos a implementar, el maestro nos enviará los caracteres dentro del campo "*Data*". Cada carácter estará formado por 2 bytes. El tratamiento que haremos de estos datos consistirá en introducirlos en un vector.

La trama de **petición** que recibiremos será de la siguiente forma:

Id Esclavo	**Código Función**	**Datos**										**CRC**	
		Dirección 1^{er} registro		**Nº registros a escribir**		**Contador de bytes**		**Valor registro 1**		**Valor registro N**			
1 byte	1byte	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB

Tabla 11. Trama de petición MODBUS

Por último, tendremos que enviar la **respuesta** al maestro siguiendo las convenciones MODBUS, confirmando que se han tratado los datos, o que ha habido algún error. En caso de que la petición vaya dirigida a todos los esclavos (*broadcast*) no habrá que enviar respuesta.

Datos						**CRC**	
Id Esclavo	**Código**	**Dirección 1^{er} registro**		**Nº registros escritos**			
1 byte	1byte	MSB	LSB	MSB	LSB	MSB	LSB

Tabla 12. Trama de respuesta MODBUS

Algoritmo: el PIC está constantemente a la espera de recibir datos por el canal R232. Cuando hay información nueva recibe la trama, comprueba que es correcta, verifica que va dirigida a él o al *broadcast*, y que es una petición de la función 10, y si se cumple todo lo anterior, almacena los datos oportunos en un vector, y envía la respuesta al maestro, ya sea de error, en caso de que lo haya, o de confirmación.

Puertos utilizados: RC6 (transmisión) y RC7 (recepción).

Programación: en la cabecera configuramos todos los parámetros MODBUS, incluimos el archivo con la librería MODBUS, que contiene todas las funciones necesarias, e indicamos que usaremos RS232. Los comentarios del código incluyen todos los detalles:

- Configuración de los parámetros MODBUS:

```
// Configuración MODBUS
#define MODBUS_TYPE MODBUS_TYPE_SLAVE             // Indicamos que somos dispositivo esclavo
#define MODBUS_SERIAL_TYPE MODBUS_RTU             // Modo MODBUS-RTU
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64           // Tamaño del búfer de recepción
#define MODBUS_SERIAL_BAUD 9600                   // Velocidad 9600
#define MODBUS_SERIAL_TX_PIN PIN_C6               // Pin de transmisión de datos
#define MODBUS_SERIAL_RX_PIN PIN_C7               // Pin de recepción de datos
#define MODBUS_ADDRESS 0x01                       // Identificador de esclavo
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA   // Habilitamos interrupciones UART
#include <libreria_modbus.c>                      // Librería de funciones MODBUS
```

- Configuramos el canal RS232 con los datos pertinentes:

```
// Configuración canal RS232
#use rs232(baud=MODBUS_SERIAL_BAUD,xmit=MODBUS_SERIAL_TX_PIN,rcv=MODBUS_SERIAL_RX_PIN,
                                    parity=N,stream=MODBUS_SERIAL,bits=8,stop=2,errors)
```

- En el programa principal, dentro del "*Main*", iniciamos el módulo MODBUS:

```
modbus_init();                    // Iniciamos MODBUS
```

- Después entramos en un bucle infinito con un `while(TRUE)`, y nos quedamos a la espera de recibir datos por el bus, en cuyo caso comprobamos si se recibe una petición dirigida a nosotros o a todos los esclavos, y si es la función 10 (si es así se guardan en el vector los datos a representar). Se envía respuesta al maestro y se llama a las funciones que gestionarán el *array* de texto para mostrarlo en el *display*.

```
while(!modbus_kbhit());       // Si se reciben datos por el bus
delay_us(50);                 // Tiempo de espera para estabilización

// Comprueba si la petición es para nuestro dispositivo o para todos (broadcast)
if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)
{
   switch(modbus_rx.func) // Comprueba el código de la función
   {
      case FUNC_WRITE_MULTIPLE_REGISTERS:
         if(modbus_rx.data[0] || modbus_rx.data[2] ||
            modbus_rx.data[1] >= 8 || modbus_rx.data[3]+modbus_rx.data[1] > 8)
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
         else
         {
            int i,j;
            // Guardamos los datos de la trama en el vector TECLADO
            for(i=0,j=5; i < modbus_rx.data[4]/2; ++i,j+=2)
               TECLADO[i] = make16(modbus_rx.data[j],modbus_rx.data[j+1]);
            // Enviamos respuesta al maestro
            modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
                        make16(modbus_rx.data[0],modbus_rx.data[1]),
                        make16(modbus_rx.data[2],modbus_rx.data[3]));
            event_count++;
            // Llamada a funciones para representar el texto recibido en el panel
            prepararTexto();
            escribirTexto();
         }
         break;
      default:    // Se lanza excepción si la función no es la 10
         modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
```

En este punto entra en funcionamiento el algoritmo de gestión de filas y columnas que veremos más adelante.

Simulación: para la simulación de la comunicación MODBUS entre el equipo maestro y el PIC hemos creado un par de puertos virtuales con *Virtual Serial Port Driver 7.2*. El puerto COM4 será nuestro PIC, y en *Proteus* lo definimos como tal. El puerto COM5 lo utilizará el maestro. El envío de tramas lo haremos con el software *Multi Way V10*. El programa calcula el CRC y monta la trama completa. En el terminal vemos cómo se envía una petición al esclavo 01 correspondiente a la función 10, con los bytes de datos y el CRC, y el PIC responde con el mensaje de respuesta oportuno:

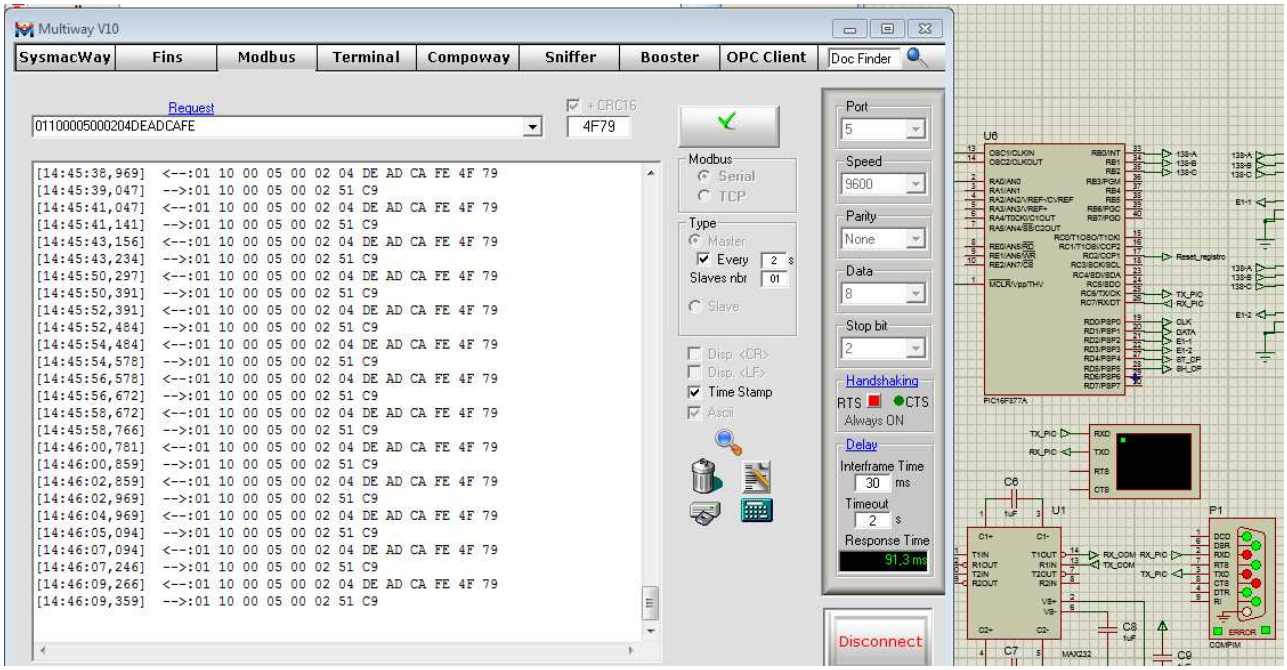


Figura 41. Simulación recepción y respuesta tramas MODBUS.

- **Programa para manejar los drivers de los ledes y enviar el dato a las filas y a las columnas**:

Algoritmo: el grueso del sistema es la gestión de filas y columnas y la representación de los datos en la matriz. Se ha intentado elaborar un algoritmo lo más sencillo posible, que cumpliera con efectividad los requisitos.

La gestión del encendido se resume de esta manera: después de recibir una trama correcta, el programa tiene guardada en un *array* de caracteres la información que debe representarse. Toma cada uno de estos caracteres y busca su correspondencia en un mapa de bits, que es lo que se enviará al registro de desplazamiento. El mapa incluye todos los caracteres básicos y extendidos (con el fin de poder representar cualquier carácter, tal y como indican los requisitos del proyecto), por lo que tiene un total de 255 posiciones.

A continuación podemos ver un extracto del mapa:

```
const char MAPA[255][7] ={
   0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,    //        Space 32
   0xFB,0xFB,0xFB,0xFB,0xFB,0xFF,0xFB,    //        !
   0xF5,0xF5,0xF5,0xFF,0xFF,0xFF,0xFF,    //        "
   0xF5,0xF5,0xE0,0xF5,0xE0,0xF5,0xF5,    //        #
```

Como vemos, cada carácter se codifica a partir de 7 valores, cada uno de los cuales representa en binario la información de una fila. Los ceros y los unos indican qué columnas deben estar encendidas o apagadas. Para entender mejor el funcionamiento, vamos a explicar con ejemplos cómo se procesaría la letra "A".

En el mapa de bits la letra A tiene asociados los siguientes valores hexadecimales:

```
0xFB,0xF5,0xEE,0xEE,0xE0,0xEE,0xEE,    //          A
```

Cada valor es la información de una fila, tomados de izquierda a derecha, siendo el primer valor la primera fila y el séptimo valor, la séptima. Si lo convertimos a binario tendremos los ceros y los unos. Como nuestra matriz es de ánodo común y un led enciende cuando recibe un 1 por la fila y un 0 por la columna, los ceros del valor nos dirán qué columnas estarán encendidas cuando se active la fila.

Hex	Binario	F/C	1	2	3	4	5	6	7	8
0xFB	11111011	1	1	1	1	1	1	0	1	1
0xF5	11110101	2	1	1	1	1	0	1	0	1
0xEE	11101110	3	1	1	1	0	1	1	1	0
0xEE	11101110	4	1	1	1	0	1	1	1	0
0xE0	11100000	5	1	1	1	0	0	0	0	0
0xEE	11101110	6	1	1	1	0	1	1	1	0
0xEE	11101110	7	1	1	1	0	1	1	1	0

Figura 42. Ejemplo representación letra A

Esta es la representación del carácter simplificado, que ocupa 7 filas y un máximo de 5 columnas con ledes encendidos, el resto de las columnas estarán apagadas. Para utilizar al completo el panel, que tiene 16 filas y 16 columnas, procedemos de la siguiente manera:

- Concatenaremos varios caracteres para que se vean uno detrás de otro, representando solo las 5 últimas columnas de cada uno, además de una en blanco como espacio, por lo que se pueden ver hasta 3 caracteres simultáneamente (aunque no completos). En el programa esto lo conseguimos enviando al registro solo los valores a partir del tercer dígito binario:

```
for(registro=0;registro<8;registro++) {          // Llenamos el registro bit a bit
   OUTPUT_BIT(DS,shift_left(&CARACTER,1,0));     // Enviamos los valores al registro
      if(registro>=2) {                          // Solo enviamos a partir del tercer bit
         OUTPUT_HIGH(SH_CP);                     // Metemos bit por bit en la memoria intermedia
         OUTPUT_LOW(SH_CP);
         } // if registro
   } // for registro
```

- Para que el carácter ocupe todas las filas del panel y no solo 7, expandimos cada uno haciendo que cada fila de la representación simplificada encienda dos filas del panel. Por lo tanto, cuando en el registro hayamos introducido los valores correspondientes al primer valor hexadecimal del mapa, activaremos las filas 1 y 2 del panel; cuando tengamos el segundo valor, las filas 3 y 4, y así sucesivamente. De esta forma, como vemos en la imagen, estamos expandiendo el carácter. Dejamos en blanco las columnas 15 y 16, ya que cuando el carácter sea una letra minúscula tendrá varias filas vacías en la parte alta, y de esta forma, compensamos visualmente ese efecto:

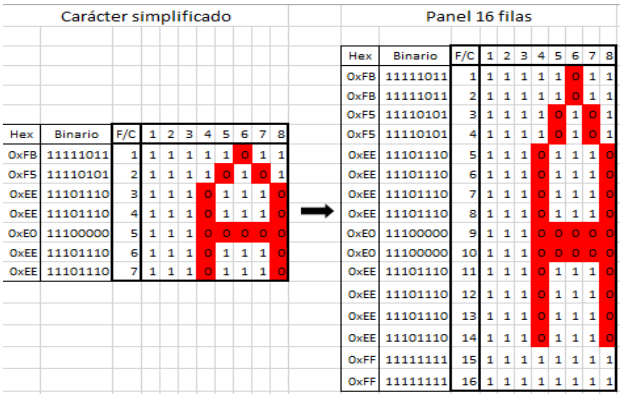


Figura 43. Detalle expansión caracteres

Una vez entendida la codificación de los caracteres, procedemos a explicar la gestión de filas y columnas.

Gestión de los registros *latch* para representar las columnas:

- El programa hace un recorrido por los 7 valores de representación del *array* MAPA:

```
for(posicion=0;posicion<8;) {                    // Bucle por las 7 posiciones del array MAPA
```

- Para cada uno de ellos hace un bucle por los caracteres del *array*, tomándolos de tres en tres:

```
for(i=elemento;i<elemento+3;i++) {               // Bucle para cada caracter que representamos
```

- Busca cada carácter en el MAPA e introduce los valores en el registro bit a bit haciendo un desplazamiento a la izquierda. A partir del tercer dígito envía el pulso de reloj (así eliminamos las columnas en blanco) y además introduce una última fila vacía. Una vez se han metido todos los valores, la señal del *latch* los envía a las salidas.

```
CARACTER=MAPA[TEXTO[i]-' '][posicion];          // Buscamos el caracter en el mapa de bits
if(posicion==7)CARACTER=0xFF;                   // Añadimos una última fila en blanco
for(registro=0;registro<8;registro++) {         // Llenamos el registro bit a bit
   OUTPUT_BIT(DS,shift_left(&CARACTER,1,0));    // Enviamos los valores al registro
      if(registro>=2) {                         // Solo enviamos a partir del tercer bit
         OUTPUT_HIGH(SH_CP);                    // Metemos bit por bit en la memoria intermedia
         OUTPUT_LOW(SH_CP);
         } // if registro
   } // for registro
   OUTPUT_HIGH(ST_CP);                          // Señal de latch para pasar los bits a las salidas
   OUTPUT_LOW(ST_CP);
   } // for i
```

Por ejemplo, si los tres caracteres que vamos a mostrar son “A0H”, esta tabla nos indica cómo serán las salidas del *latch* al finalizar cada iteración de j:

		A		0		H		Salidas del latch															
Iteración j	j	Hex	Binario	Hex	Binario	Hex	Binario	F/C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1ª iteración	0	0xFB	1011	0xF1	10001	0xEE	01110	1	1	0	1	1	1	1	0	0	0	1	1	0	1	1	1	0
2ª iteración	1	0xF5	0101	0xEE	01110	0xEE	01110	2	0	1	0	1	1	0	1	1	1	0	1	0	1	1	1	0
3ª iteración	2	0xEE	1110	0xEC	01100	0xE0	01110	3	1	1	1	0	1	0	1	1	0	0	1	0	1	1	1	0
4ª iteración	3	0xEE	1110	0xEA	01010	0xEE	00000	4	1	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0
5ª iteración	4	0xE0	0000	0xE6	00110	0xEE	01110	5	0	0	0	0	1	0	0	1	1	0	1	0	1	1	1	0
6ª iteración	5	0xEE	1110	0xEE	01110	0xF5	01110	6	1	1	1	0	1	0	1	1	1	0	1	0	1	1	1	0
7ª iteración	6	0xEE	1110	0xF1	10001	0xFB	01110	7	1	1	1	0	1	1	0	0	0	1	1	0	1	1	1	0
8ª iteración	7	0xFF	1111	0xFF	11111	0xFF	11111	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figura 44. Ejemplo estados salidas latch

Gestión del encendido de las filas:

Cuando en los registros se han metido ya los valores de una fila de los caracteres pertinentes, entra en juego la gestión de los decodificadores. Utilizamos una variable, $deco$, que se irá incrementando de 0 a 255 (es de 8 bits). Cuando el programa llegue al punto de querer activar una fila, comprobaremos si $deco$ ha llegado a 8 o no. Si no ha llegado, debemos activar el primer decodificador (correspondiente a las 8 primeras filas) y si no, el segundo. Activamos y desactivamos esperando $2ms$ y enviamos al decodificador los bits A-B-C. Estos bits son los 3 últimos de la variable $deco$ y nos servirán para decidir cuál es la fila que se activará (a nivel bajo). Cada dos veces que hagamos este proceso habremos recorrido las 16 filas, lo cual controlamos con la variable $completo$, que hace que $posición$ se incremente solo cuando ya se han activado todas las

filas y entre de nuevo en el bucle para representar los siguientes 3 caracteres. De esta forma, en cada iteración de $posición$ encendemos dos filas y conseguimos que las letras se "alarguen" y ocupen todo el panel.

```
// Gestión de las filas
if(deco&8)                                  // "deco" es variable de 0 a 255. En este rango los 3
                                            // últimos bits en binario son 8
OUTPUT_HIGH(E12);else OUTPUT_HIGH(E11);     // Activa el decodificador que corresponda
delay_ms(2);                                // Espera 2ms
OUTPUT_LOW(E11);OUTPUT_LOW(E12);            // Desactiva todas las filas
OUTPUT_B(deco+=1);                          // Envía los 3 últimos bits de deco a las salidas A-B-C
                                            // de los 138 para la selección de fila
if(completo)posicion++;                     // "completo" puede valer 0 ó 1. Incrementa posicion cada 16 filas
completo=!completo;
} // for posicion
```

Gestión del efecto de movimiento:

Solo nos queda simular el efecto de movimiento. En lugar de trabajar con *buffers* o con vectores auxiliares, vamos a utilizar dos variables, m y n, que se van incrementando a diferente ritmo. Cada vez que se termina de representar un grupo de 3 caracteres, m se incrementa y hace que n se incremente hasta un total de 5 veces, momento en el cual el valor de $elemento$ (posición en el *array* del carácter que estamos representando) se incrementa y hace que el vector avance. Si en la primera iteración representamos los caracteres 1, 2 y 3 (cuando $elemento = 0$ e i itera en estos 3 valores), al terminarla $elemento$ será igual a 1, por lo que i recorrerá los caracteres 2, 3 y 4 del vector, y así sucesivamente. De este modo, estamos representando cada carácter un total de tres veces, pero en cada una de ellas se verá en una posición diferente en la matriz. En concreto, y por los valores que hemos dado a m y n, desplazado 8 columnas a la izquierda. Por la velocidad de refresco del programa la sensación es de movimiento y de texto que se desliza, a modo de *scroll* horizontal, en lugar de apariciones y desapariciones de caracteres.

```
// Incremento de variables para producir el efecto de movimiento
m+=1;
if(m==2){m=0;n+=1;};                // Cuando m es 1 lo ponemos a 0 e incrementamos n
if(n==5){n=0;elemento+=1;}          // Cuando n llega a 8 lo ponemos a 0 e incrementamos d
if(elemento>(length))elemento=0;    // Cuando llegamos al final de los caracteres, empezamos de nuevo.
```

Programación: el código correspondiente a esta parte del programa es:

```
while(TRUE) {
    for(posicion=0;posicion<8;) {                   // Bucle por las 7 posiciones del array MAPA
      for(i=elemento;i<elemento+3;i++) {             // Bucle para cada caracter que representamos
         CARACTER=MAPA[TEXTO[i]-' '][posicion];     // Buscamos el caracter en el mapa de bits
         if(posicion==7)CARACTER=0xFF;              // Añadimos una última fila en blanco
         for(registro=0;registro<8;registro++) {    // Llenamos el registro bit a bit
            OUTPUT_BIT(DS,shift_left(&CARACTER,1,0));// Enviamos los valores al registro
               if(registro>=2) {                    // Solo enviamos a partir del tercer bit
                  OUTPUT_HIGH(SH_CP);               // Metemos bit por bit en la memoria intermedia
                  OUTPUT_LOW(SH_CP);
                  } // if registro
            } // for registro
            OUTPUT_HIGH(ST_CP);                     // Señal de latch para pasar los bits a las salidas
            OUTPUT_LOW(ST_CP);
            } // for i
```

```
    // Gestión de las filas
    if(deco&8)                                  // "deco" es variable de 0 a 255. En este rango los 3
                                                // últimos bits en binario son 8
    OUTPUT_HIGH(E12);else OUTPUT_HIGH(E11);     // Activa el decodificador que corresponda
    delay_ms(2);                                // Espera 2ms
    OUTPUT_LOW(E11);OUTPUT_LOW(E12);            // Desactiva todas las filas
    OUTPUT_B(deco+=1);                          // Envía los 3 últimos bits de deco a las salidas A-B-C
                                                // de los 138 para la selección de fila

    if(completo)posicion++;                     // "completo" puede valer 0 ó 1. Incrementa posicion cada 16 filas
    completo=!completo;
    } // for posicion
    // Incremento de variables para producir el efecto de movimiento
    m+=1;
    if(m==2){m=0;n+=1;};              // Cuando m es 1 lo ponemos a 0 e incrementamos n
    if(n==5){n=0;elemento+=1;}        // Cuando n llega a 8 lo ponemos a 0 e incrementamos d
    if(elemento>(length))elemento=0;  // Cuando llegamos al final de los caracteres, empezamos de nuevo.
} //while
```

Simulación: las imágenes que vemos en la siguiente página muestran los distintos *frames* que se pueden observar si introducimos por teclado el texto: ***"Hola!"***. Aunque son imágenes estáticas y no permiten apreciar el movimiento, sí dan una idea del modo en que funciona el panel.

Nota: en la simulación en *Proteus* hemos sustituido el conjunto decodificador 138 + los transistores por el decodificador 238, que tiene el mismo funcionamiento que el 138 pero con salida activa a nivel alto. El motivo es que *Proteus* no simula correctamente el comportamiento de los transistores, por lo que necesitamos eliminarlos de la simulación y utilizar un selector de fila a nivel alto en su lugar.

Asimismo, para una simulación más intuitiva, la versión del programa ofrece un terminal virtual por el que se escribe directamente el texto que se quiere representar, de modo que no hay que enviar tramas MODBUS sino texto ASCII.

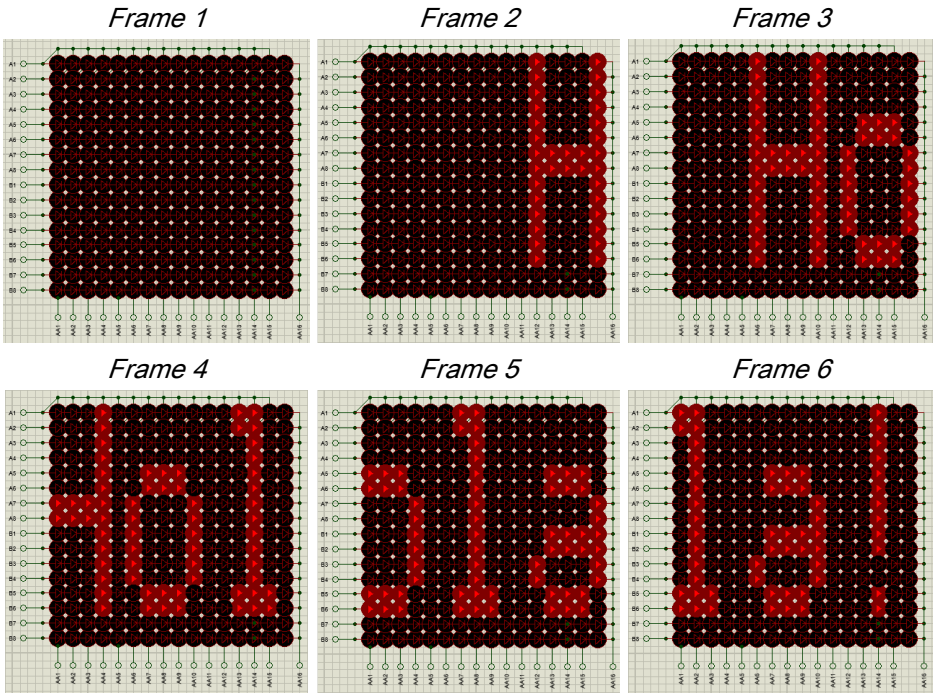


Figura 45. Simulación escritura de texto en el panel

6. DISEÑO DE LA FUENTE DE ALIMENTACIÓN

6.1. ESTUDIO DE LA FUENTE DE ALIMENTACIÓN

El equipo recibe una señal de $24V$ de continua, que procede de una fuente ya existente. Para diseñar el circuito más adecuado vamos a repasar las características de voltaje y consumo de todos los elementos del sistema que debemos alimentar. Extraemos esta información de las hojas de producto del fabricante y tomamos los valores de operación recomendados:

Componente	Voltaje	Corriente
PIC16F877A	+5V	25mA (pines) – 300mA (Máx. Vss)
74HC595	+4.5V a +5.5V	50mA
74HC138	+4.5V a +5.5V	8mA
MAX232	+5V	0,01uA
LM324	+3V a +32V	20mA
BC327	+5V	16x20mA=320mA
	Total	***<698mA***

Tabla 13. Voltaje y corriente componentes

El sistema en su conjunto se ha concebido con el menor número de componentes posible, con el fin de optimizar el diseño, abaratar costes y reducir el riesgo de errores de cableado. Además, uno de los requisitos del proyecto es que el equipo sea lo más eficiente posible, y de ahí que los elementos se hayan seleccionado respetando el criterio del bajo consumo.

Como vemos en la tabla anterior, todos los componentes se alimentan, o pueden alimentar, con una tensión de $5V$. Esto, sin duda, facilita el diseño de la fuente, puesto que no necesitamos obtener diferentes niveles de voltaje, sino uno de $5V$. Además, en el peor de los escenarios la corriente que demanda el sistema no superaría los $700mA$.

El bloque, por lo tanto, proporcionará una tensión regulada y contendrá los circuitos que aseguren una alimentación correcta de todos los bloques del sistema. Adicionalmente, debemos asegurarnos también de que cuenta con las protecciones básicas contra cortocircuitos, posibles excesos de voltaje y con protección ante una polarización inversa del sistema.

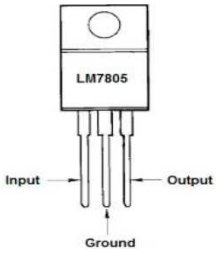


Figura 46. LM7805

Al valorar las opciones de diseño se exploró la posibilidad de desarrollar una fuente de alimentación estabilizada de $5V$ a través de un regulador de voltaje fijo, en concreto el *LM7805*. Su implementación era sencilla, puesto que consistía en conectar el positivo de la fuente externa ($24V$) a su pin *Input*, obteniendo a su salida (pin *Output*) los $5V$ requeridos. El exceso de energía se convertiría en calor. El inconveniente de este circuito es que el exceso de energía es demasiado amplio ($19V$), y aunque en teoría y atendiendo a la hoja de producto el voltaje de entrada aceptado es de $7V$ a $25V$, al hacer los cálculos de los valores de disipación en función de la corriente, podemos ver que (tomando $I = 700mA$ y $V = 19V$):

$$\boxed{P = 0{,}70 \cdot 19 = 13{,}3W}$$

Ecuación 7. Cálculo potencia circuito

En el *datasheet* nos indican que la temperatura máxima de operación es de $150℃$, con una disipación de $23℃$ por vatio. A $25℃$, el resultado superaría los $300℃$, de modo que estaría fuera del rango de operación. Aunque se utilizara un muy buen disipador el circuito no sería óptimo, por lo que recurrimos a otras opciones.

Se podría utilizar un regulador entre los $24V$ y los $5V$ para que la etapa reductora no fuera tan drástica. Sin embargo, la energía disipada seguiría siendo la misma, solo que el calor se repartiría entre dos reguladores.

Por todo ello, finalmente llegamos a otra solución, utilizar el regulador *LM2575* de *Texas Instruments*. Es un regulador *step-down* que nos entrega una salida máxima de corriente de $1A$, y se puede alimentar con un voltaje máximo de entrada de $40V$ sin que disipe el exceso ($19V$ en nuestro caso) mediante calor.

La hoja de producto nos da datos concretos sobre el dispositivo:

- Voltaje fijo de salida de $5V$.
- Voltaje de entrada de $7V$ a $40V$.
- Corriente de salida de $1A$.
- Protección contra exceso de corriente.
- No requiere disipador (disipación limitada internamente).
- Opera a temperaturas de $-40℃$ a $125℃$.

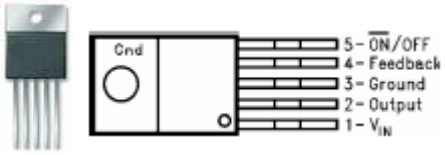


Figura 47. Mecánica y pines LM2575

El *LM2575* necesita otros componentes pasivos para su funcionamiento y son una inductancia y un diodo *Schottky*. La propia hoja de producto nos indica cómo debe ser el circuito y los valores de los componentes:

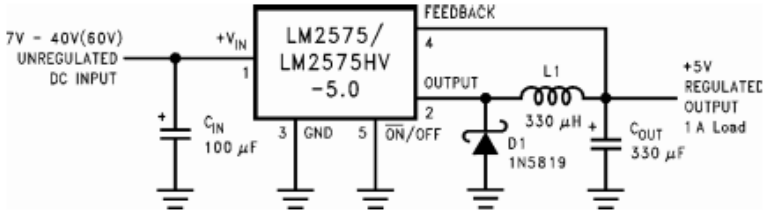


Figura 48. Circuito LM2575

Para proteger el circuito y garantizar su estabilidad, seguiremos las indicaciones del fabricante. Los componentes de la fuente y sus valores los estudiaremos en el siguiente apartado.

6.2. DISEÑO DEL CIRCUITO DE LA FUENTE DE ALIMENTACIÓN

El elemento principal del circuito de la fuente es el regulador *LM2575*. Su pin de V_{IN} va conectado al polo positivo de la fuente de $24V$ de continua que nos viene dada. Los pines 3 y 5 (GND y ON/OFF) van conectados a masa, y su pin de $Output$ será la salida con los $5V$ que buscamos.

El resto de componentes son necesarios para proteger el regulador y asegurar la estabilidad de la señal, y se han incluido siguiendo las indicaciones de la hoja de producto.

El circuito completo lo podemos ver en esta imagen:

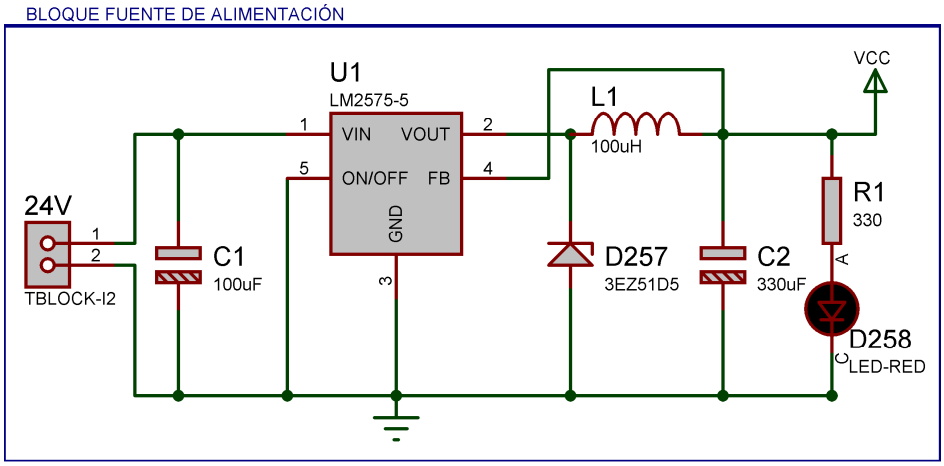


Figura 49. Circuito fuente de alimentación

En la simulación en Proteus observamos cómo a partir de una entrada de $24V$ obtenemos una salida de $5V$:

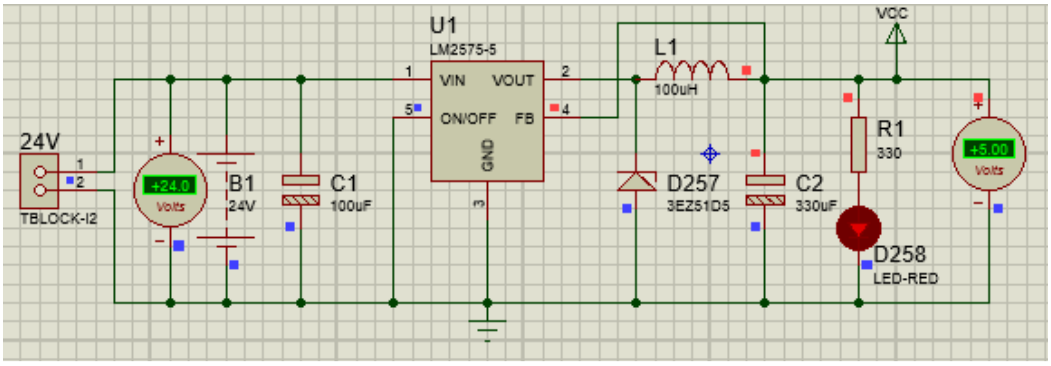


Figura 50. Simulación fuente de alimentación.

Notas:

- Los componentes externos (condensadores electrolíticos C1 y C2, bobina L1 y diodo *Schottky* D258) se incluyen a partir de las instrucciones del fabricante, y sus valores son los correspondientes al modelo de regulador en cuestión para el voltaje de salida que precisamos.
- La batería de 24V es una fuente de $24V$ de corriente continua. En *Proteus* la simulamos como batería con un bloque de dos pines que utilizaremos en la PCB para conectar la fuente. Esta debe ser de entre $7V$ *y* $40V$.
- LM2575: en *Proteus* no existía la librería pero se ha obtenido a través de Internet. Para la placa PCB se utiliza su encapsulado TO-220.
- Los componentes externos deben colocarse lo más cerca posible del regulador para evitar picos de voltaje debidos a la fluctuación de corriente y a la inductancia del cableado.

A la salida del circuito tendremos los $5V$ que necesitamos para alimentar los distintos bloques. En el siguiente apartado, "Diseño de la placa PCB", tendremos en detalle la disposición de los componentes y la conexión a V_{CC} y GND de cada uno de los módulos.

7. DISEÑO DE LA PLACA PCB

7.1. ESTUDIO DE LOS COMPONENTES DE LA PLACA

Con el equipo ya diseñado, nos corresponde crear la placa de circuito impreso que haga físicamente real el proyecto. Hemos utilizado *Ares* de *Proteus*, ya que todos los esquemas los hemos desarrollado con *Isis*.

La primera decisión ha sido si utilizar varias placas y distribuir los bloques entre ellas o si colocar el equipo al completo sobre una única. Se ha optado por hacer el diseño sobre dos placas, no porque las dimensiones del equipo lo requieran, sino porque el panel estará expuesto para que se vea la información, y no sería estético que la electrónica fuera visible. Además, así protegemos los componentes.

De este modo, las placas para el prototipo serán:

1) **PCB1**, con los ledes y el sensor LDR. El sensor debe estar en la superficie para poder captar la luz y lo colocaremos lo más lejos posible de la matriz para que la luz de los ledes no condicione las medidas y con un separador de plástico. La conexión con la placa inferior se hará a través de tiras de pines. En *Proteus* utilizamos un modelo genérico de conector que en realidad serían pines de conexión.
2) **PCB2**, con en el resto de los componentes electrónicos, correspondientes a los demás bloques.

Las dos placas se pueden posteriormente atornillar por las esquinas para que queden como un único elemento. Dejaremos unos $5cms$ entre ellas para que los componentes inferiores tengan espacio suficiente.

Por su parte, se han tomado las siguientes consideraciones:

- Cada placa mide $12{,}7cm$ de largo por $15cm$ de ancho.
- El grosor de las pistas es de 30th, considerando una corriente máxima de $1A$.
- Para medir la distancia entre componentes se han seguido las indicaciones de las hojas de producto. Los ledes son SMD, tal y como vimos en la sección 2.3 y los demás componentes son *Though Hole*.
- Entre las pistas y el borde de la placa dejamos al menos $3mm$.
- Parte de las pistas se han trazado manualmente puesto que las herramientas automáticas tienen errores. Para los ledes, al ser SMD, se han trazado tanto en la cara superior como en la inferior (a través de agujeros), y para el resto de componentes siempre que ha sido posible se han trazado en la inferior.
- La distancia de separación entre pistas paralelas es uniforme.
- Asumimos que la fuente externa de $24V$ tiene un interruptor, por lo que no se ha incluido en la PCB.
- Se añade plano de masa a ambas placas.

En el dispositivo real se han utilizado placas perforadas para reducir el tiempo necesario en la perforación e insolación. Al no encontrarse unas del mismo tamaño que las del prototipo, se ha hecho una ligera redistribución de los componentes en la placa inferior, y se han trazado las pistas en consecuencia. Respecto a la placa superior, al ser 4 matrices, se han soldado los pines directamente a los cables que irán conectados a la placa inferior. Veremos los planos de la PCB inferior del dispositivo real en “ANEXO III: Construcción del dispositivo”.

7.2. DISEÑO DE LA PLACA PCB

En las dos siguientes páginas mostramos las placas del prototipo en su vista frontal y en la de pistas:

PCB 1: Panel de ledes y sensor LDR:

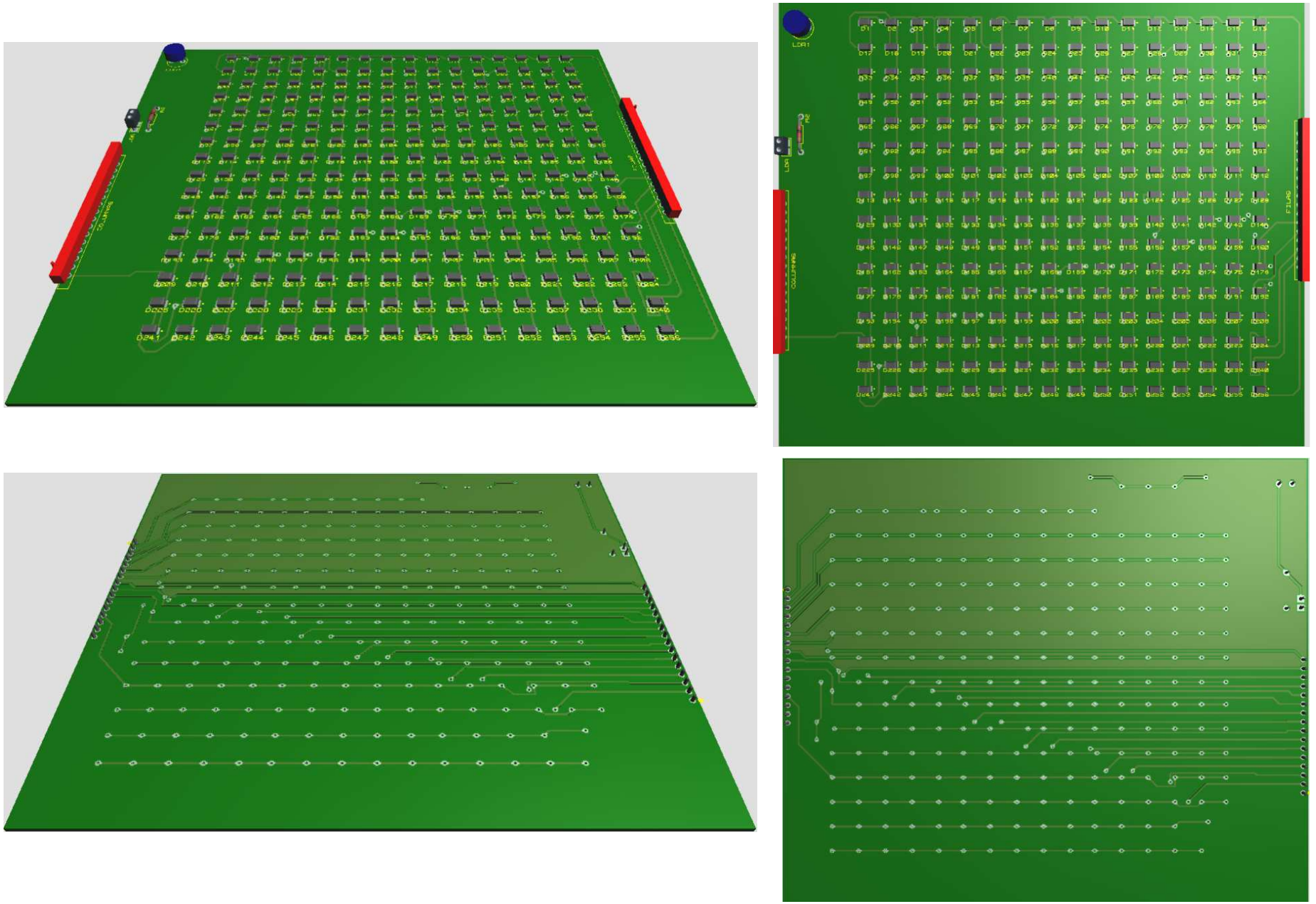


Figura 51. PCB 1 - Caras componentes y pistas. Vistas superior y frontal

PCB 2: Electrónica interna:

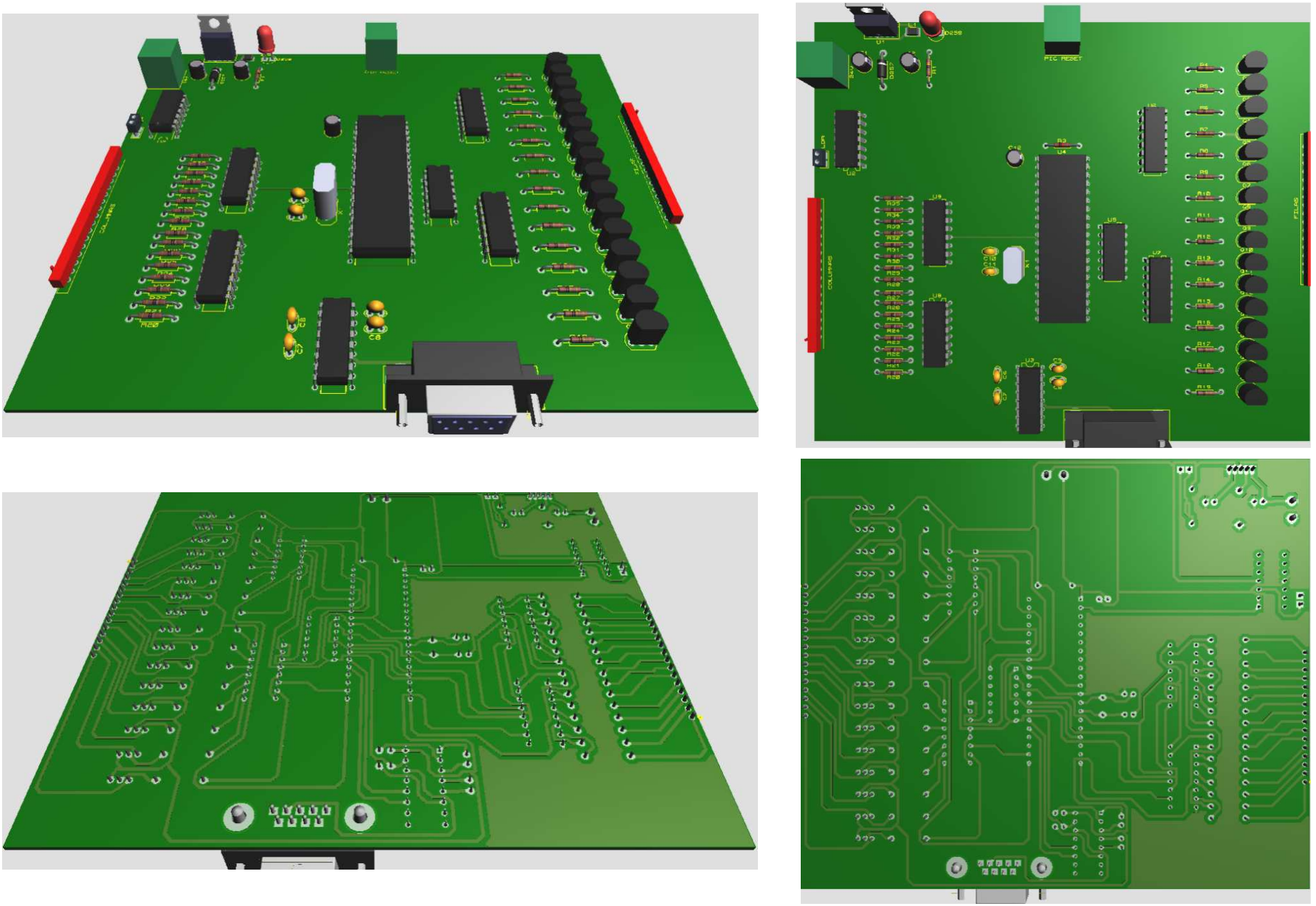


Figura 52. PCB 2 - Caras componentes y pistas. Vistas superior y frontal

8. CONCLUSIONES Y AMPLIACIONES

Terminado el proyecto, es momento de determinar si hemos cumplido los objetivos marcados y en qué medida. Para evaluarlo podemos hacer un repaso de cuáles eran los requisitos del sistema:

1- Matriz de 16x16 de leds rojos capaz de mostrar cualquier información: así está definido nuestro panel y el código implementado nos asegura que se pueda representar cualquier dato.
2- Equipo eficiente que minimice el consumo eléctrico: se han escogido leds de consumo medio en detrimento de los de alta potencia. Para las columnas, el multiplexado garantiza que solo puede haber encendido un led por columna en cada momento. Para las filas, se utiliza una etapa de potencia a través de transistores para controlar el consumo. A través de la modulación por ancho de pulsos (PWM) controlamos el tiempo de encendido de los leds.
3- Comunicación con dispositivo externo a través de RS232 y protocolo MODBUS: el equipo que envía la información se conecta a través de RS232 y el programa de control implementa el protocolo MODBUS.
4- Luminosidad de los leds adaptada a la luz externa: el sensor LDR envía la señal de nivel de luz externa al microprocesador, y el tiempo de encendido de los leds varía en función de esta.
5- Alimentación de $24V$ de continua: se ha diseñado una fuente de alimentación regulada para obtener $5V$ a partir de los $24V$ dados.
6- Programa de control que continuamente monitoriza la señal del sensor de luz, controla el encendido y apagado de los leds y está pendiente del canal RS232 por si llega un nuevo mensaje.

Por otra parte, el panel que hemos planteado se ha diseñado teniendo en cuenta futuras ampliaciones. Esto queda patente en las decisiones tomadas respecto a su diseño y programación:

- El PIC seleccionado (16F877A) tiene unas prestaciones muy superiores a las necesarias para cumplir con la funcionalidad requerida en el proyecto. Su elección no ha sido trivial, sino que se ha tenido en cuenta que el mismo esquema y la misma concepción pueden ser adaptados a nuevos requerimientos. Si quisiéramos ampliar la matriz en vertical, y que tuviera 64 filas, por ejemplo, disponemos de pines para añadir más decodificadores. Si necesitáramos más memoria, tiene RAM suficiente.
- El programa de gestión de filas y columnas también se ha desarrollado pensando en una fácil adaptación a estos requisitos. Hemos visto cómo la variable "*deco*" va activando los grupos de filas cada 8 iteraciones. El programa está preparado para poder representar los caracteres en paneles con más filas.
- El sistema admite mejoras dependiendo de la aplicación real que se quiera dar al dispositivo. Por ejemplo, el sensor de luminosidad empleado, el LDR, es un genérico cuya precisión es suficiente para este proyecto, pero que podría ser sustituido por un sensor de alta precisión, o incluso un sensor solar, en caso de que el equipo tuviera un fin más profesional. De igual modo, todos los integrados que hemos seleccionado para la gestión del encendido (decodificadores y registros de desplazamiento) pueden ser reemplazados por circuitos de *drivers* diferentes. Si nuestra matriz fuera mucho más larga, tendríamos que revisar el planteamiento puesto que esto afectaría sin duda a la luminosidad de los leds. Actualmente cada uno enciende una dieciseisava parte del tiempo (sin tener en cuenta la reducción del ciclo de trabajo

en función de la luz ambiente), por lo que para otras necesidades habría que revisar la etapa de potencia que estamos entregando al sistema.

- Una de las ampliaciones que se había contemplado en un inicio era el montaje físico del dispositivo y que al final se ha podido ir haciendo a medida que avanzaba el proyecto. Para poder terminarlo a tiempo se han utilizado las matrices de 8x8 y la placa es perforada, si bien se han seguido todos los esquemas mostrados en la memoria. Aunque hemos ido viendo los bloques por partes, en los anexos se puede ver el diseño final. Para el futuro queda el montaje sobre PCB virgen y la inclusión de ledes individuales. Por otro lado, el programador del PIC no ha llegado a tiempo y no se ha podido programar el equipo, por lo que esta también será labor para el futuro, además de probar distintos algoritmos de representación.

Solo nos queda comentar las conclusiones principales que hemos extraído del proyecto. En líneas generales ha sido un trabajo exigente con unos requerimientos de tiempo y esfuerzo altos teniendo en cuenta que se dispone de poco más de cuatro meses para su elaboración. A pesar de ello, una correcta planificación ha sido clave para poder completarlo en los plazos establecidos. Se considera que el planteamiento del TFC es muy profundo, puesto que se aborda el desarrollo de un prototipo de principio a fin, desde la parte más teórica, pasando por la simulación de los circuitos, hasta llegar al diseño de la PCB. Tener que elegir y justificar la selección de componentes también ha sido clave para darle una dimensión más real. A pesar de todos los contratiempos y dificultades las impresiones finales son muy positivas. El entrar en contacto con programas utilizados a nivel profesional, o el desarrollo de la PCB, entre otros, son aspectos a los que no nos hubiéramos enfrentado de no ser por un proyecto de estas características.

Pero sin duda la conclusión final con la que nos quedamos es que el trabajo ha resultado infinitamente motivador, puesto que de otro modo no se hubiera encontrado el tiempo ni la energía para querer construirlo. El montaje no ha resultado sencillo, ya que además de las dificultades inherentes al proyecto, se añaden las siguientes: encontrar los componentes necesarios a tiempo (hasta mediados de noviembre que se terminó de diseñar el circuito a nivel conceptual no se pudo iniciar la fase de compra), soldar la placa, teniendo en cuenta la escasa experiencia previa, y el no saber en ocasiones cómo solucionar los errores. Cuando se disponga del programador del PIC y una vez programado, el panel de ledes muestre el mensaje "*Enhorabuena, lo has conseguido*" se confirmará que el duro trabajo no ha sido en balde.

A pesar de la frustración de no poder completarlo, este paso ha sido como encajar la última pieza de un puzle y será el primero de un largo camino en el mundo de la electrónica.

9. BIBLIOGRAFÍA

Obras de referencia:

[1] Pérez, M.A.; Álvarez J.C; Campo J.C.; Ferrero F.J., "Instrumentación electrónica", Thomson, 2004.

[2] Roselló Canal, M., "Aplicaciones electromagnéticas y electrónicas", UOC, 2010.

[3] Pérez i Navarro, A.; Martínez Carrascal, J.A.; Muñoz Medina, O., "Teoría de circuitos", UOC, 2014.

[4] Martínez Carrascal, J.A.; Bara Iniesta, M., "Electrónica Digital", UOC, 2014.

[5] García Breijo, E., "Compilador CCS y simulador Proteus para Microcontroladores PIC", Alfaomega, 2008.

[6] Forrest, M., Mims, III, "*Electronics learning lab, Workbook 1: Basic electronics, transistors and integrated circuits*", Radioshack Corp, 2000.

[7] Forrest, M., Mims, III, "*Electronics learning lab, Workbook 2: Digital logic projects*", Radioshack Corp, 2000.

Hojas de producto:

[8] PIC16F87XA Datasheet. Microchip. http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf

[9] 74HC595: http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf

[10] BC327: http://www.onsemi.com/pub_link/Collateral/BC327-D.PDF

[11] MAX232: http://datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf

[12] LM324: http://www.ti.com/lit/ds/symlink/lm324.pdf

[13] 74HC138: http://www.nxp.com/documents/data_sheet/74HC_HCT138.pdf

[14] LM7805: http://www.ti.com/lit/ds/symlink/lm7805c.pdf

[15] LM2575: http://www.ti.com/lit/ds/symlink/lm2575-n.pdf

[16] LDR GL5516: http://www.espruino.com/datasheets/GL5537.pdf

Sitio web de consulta:

[17] http://www.zootecniadomestica.com/acuarios/practica-del-acuario/iluminacion/luz-led/

[18] http://www.neoteo.com/electronica-basica-diodos-emisores-de-luz-led/

[19] http://nergiza.com/tipos-de-leds-en-bombillas-smd-cob-5050-5630/

[20] http://www.radioelectronica.es/articulos-practicos/110-calculo-circuitos-con-diodos-led

[21] http://www.avagotech.com/docs/AV02-0871EN

[22] http://www.comohacerturobot.com/Taller/taller-sensorluminosidad.htm

[23] http://www.ucontrol.com.ar/revista/0003/ucontrol_revista_0003.pdf

[24] http://www.simplymodbus.ca/

[25] http://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf

[26] http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[27] http://www.tecdigitaldelbajio.com/blog/24-modbus-parte-i-comunicacion-serial.html

10. ANEXOS

10.1. ANEXO I: VALORACIÓN ECONÓMICA DEL PROYECTO

Categoría	Referencia	Modelo	Precio /u	Nº	Total	Proveedor
Capacitador	C1	100uF	0,13 €	1	0,13 €	*Diotronic*
	C2	330uF	0,21 €	1	0,21 €	*Diotronic*
	C6-C9	1uF	0,09 €	4	0,36 €	RS
	C10-C11	33pF	0,34 €	2	0,68 €	RS
	C12	1uF	0,16 €	1	0,16 €	*Diotronic*
Resistencia	R1	330 Ohm	0,02 €	1	0,02 €	RS
	R3	10k Ohm	0,04 €	1	0,04 €	RS
	R20-R35	180 Ohm	0,02 €	16	0,32 €	RS
	R4-R19	1K5 Ohm	0,03 €	16	0,48 €	*Diotronic*
	LDR-1	LDR-GL5516	0,70 €	1	0,70 €	Conectron
Circuitos	U1	LM2575-5	1,36 €	1	2,33 €	Diotronic
Integrados	U3	MAX232	1,00 €	1	1,00 €	RS
	U4	PIC16F877A	4,74 €	1	4,74 €	RS
	U5	74HC14	0,12 €	1	0,12 €	Diotronic
	U6-U7	74HC138	0,62 €	2	1,24 €	RS
	U8-U9	74HC595	0,26 €	2	0,52 €	Diotronic
Transistores	Q1-Q16	BC328	0,05 €	16	0,8 €	Diotronic
LED SMD	D1-D256	LED-RED	0,073 €	256	18,82 €	Farnell
LED	D258	LED-RED	0,13 €	1	0,13 €	*Diotronic*
Diodos	D257	3EZ51D5	0,17 €	1	0,17 €	*Diotronic*
Miscelánea	L1	1mH	0,41 €	1	0,41 €	*Diotronic*
	X1	CRYSTAL 20MHz	0,55 €	1	0,55 €	RS
		Interruptor SPDT	1,19 €	2	2,38 €	RS
		Conector DSUB9	1,37 €	1	1,37 €	RS
		Conectores 16 pines	0,17 €	4	0,68 €	*Diotronic*
			TOTAL		***40,07 €***	

Tabla 14. Valoración económica del proyecto

(*) Solo componentes, sin incluir herramientas ni PCB. Precios válidos en Noviembre de 2014 y sin impuestos.

10.2. ANEXO II: ESQUEMA COMPLETO DEL CIRCUITO

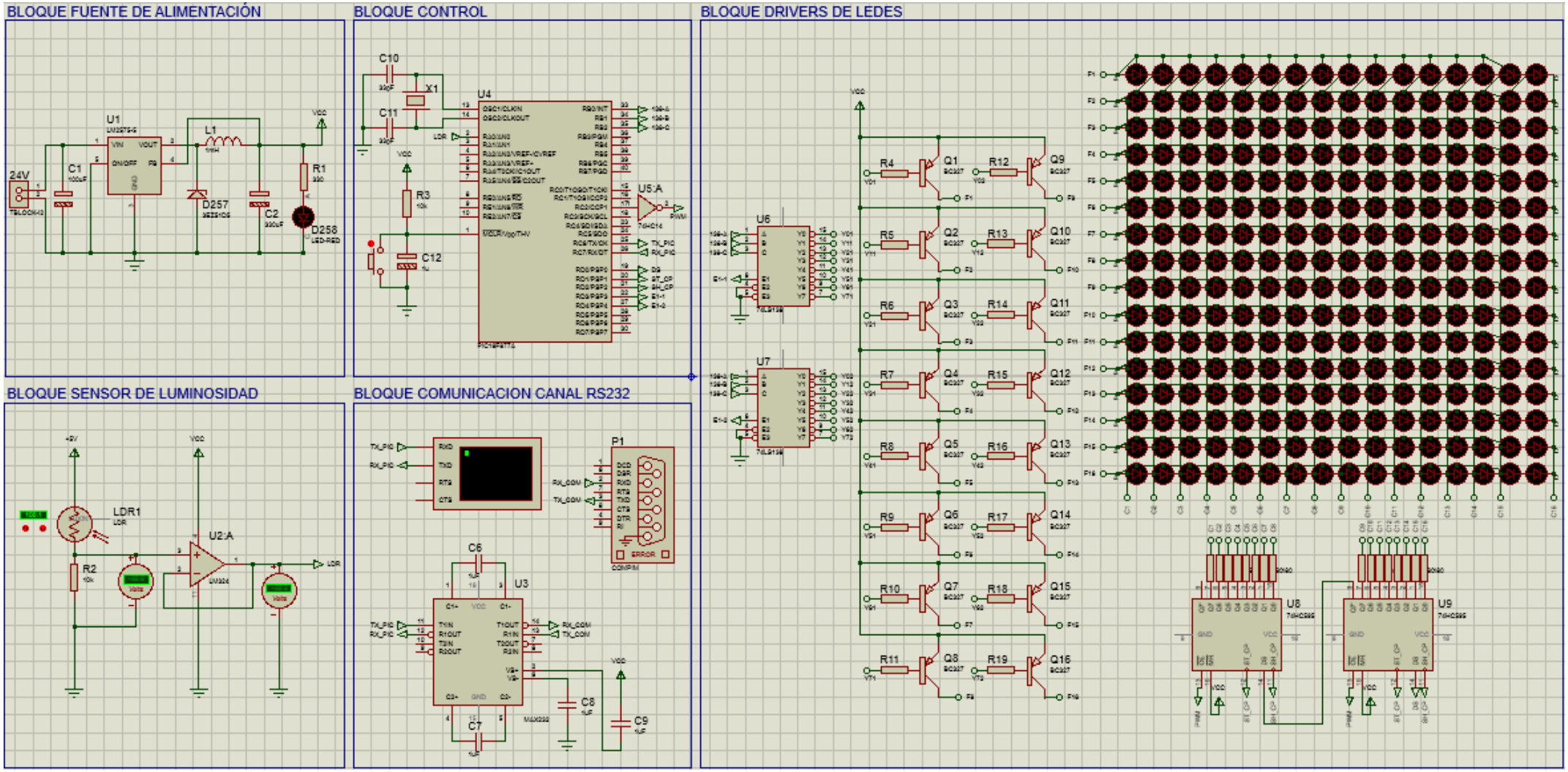


Figura 53. Esquema completo del circuito

10.3. ANEXO III: CONSTRUCCIÓN DEL DISPOSITIVO

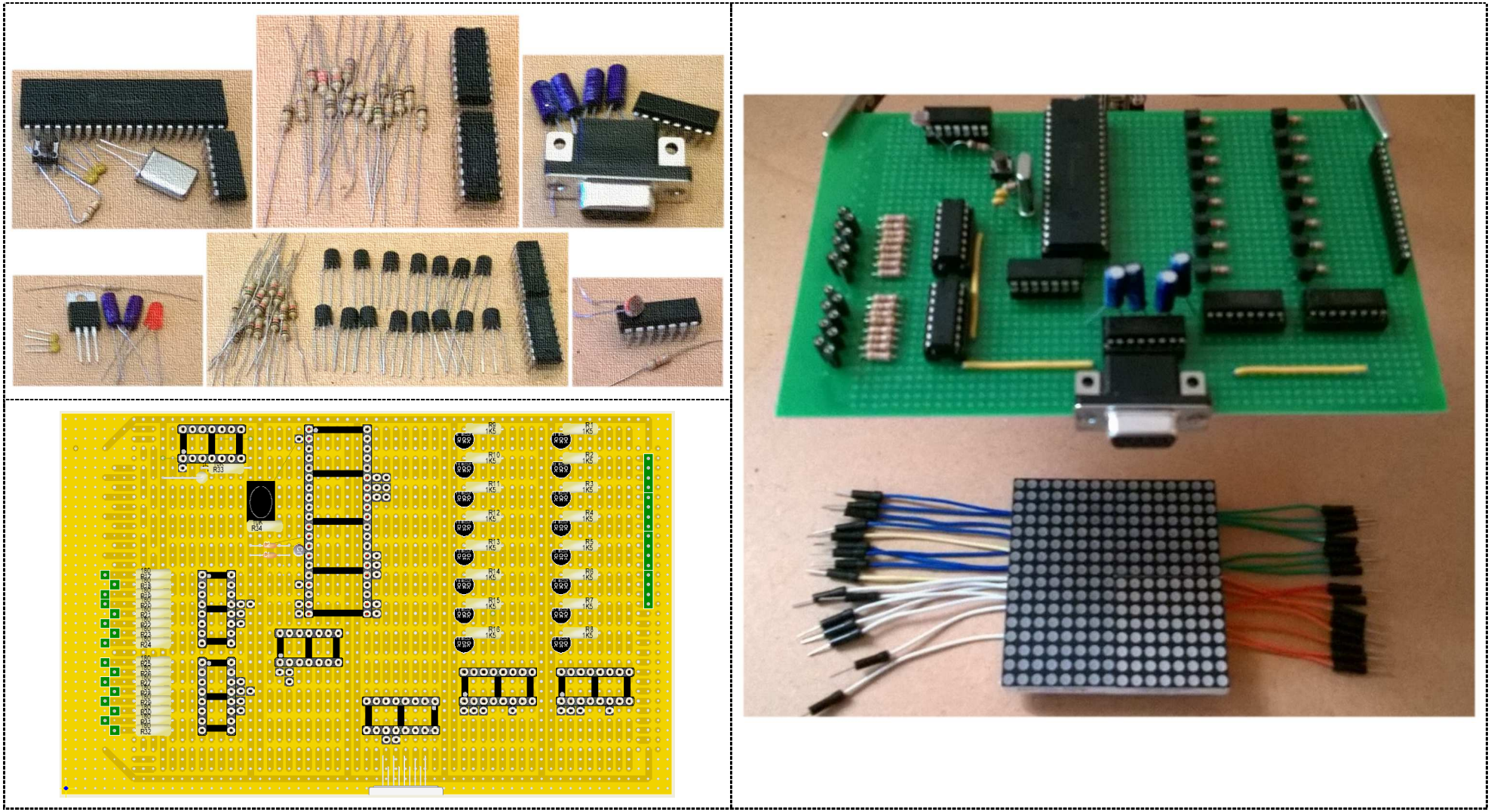


Figura 54. Componentes, PCB y equipo finalizado