TFG UOC

TFG Grado de Ingeniería de Tecnologías y Servicios de Telecomunicación

PANEL LED 16x16

Santiago Javier Alvaro Pelado
[Fecha]

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Panel LED informativo 16x16*
Nombre del autor:	*Santiago Javier Álvaro Pelado*
Nombre del consultor/a:	*Carlos Gonzalo Moreno Soriano*
Nombre del PRA:	*Germán Cobo Rodríguez*
Fecha de entrega (mm/aaaa):	*01/2023*
Titulación o programa:	*Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación*
Área del Trabajo Final:	*Diseño de Sistemas Electrónicos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Panel LED, multiplexación, microcontrolador*

Resumen del Trabajo

En la actualidad la información se ha convertido en algo muy importante, a todos nos gusta estar informados de lo que nos rodea en todo momento y de aquellas cosas que pudieran afectar a nuestro día a día.

Para que la información sea útil ha de ser una información actualizada, ya que la información obsoleta no tiene ningún valor. Por estos motivos, las pantallas informativas han tomado gran protagonismo en los últimos años y están siendo instaladas prácticamente en todos lados. Poco a poco están sustituyendo a los tablones informativos clásicos, donde los papeles se acumulaban al disponer de poco espacio y la información suministrada se quedaba obsoleta rápidamente.

La nueva generación de paneles informativos están optimizados para ofrecer información dinámicamente y actualizada en tiempo real y a la vista de todos. Y los podemos encontrar en administraciones públicas para ofrecer información municipal, en fachadas de edificios orientadas para publicidad o comercios ofreciendo ofertas y precios a clientes, en paradas de autobús o estaciones para mostrar el horario de trenes o autobuses en tiempo real o incluso en carreteras y autovías donde se muestra información de tráfico, accidentes, etc.

Además, la gestión de estos paneles puede hacerse a distancia y/o de manera centralizada, por lo que en todos ellos podrías tener la misma información.

Abstract

At present, information has become crucial, everyone likes to be informed of all that surrounds us all the time and all the things that affect us.

Information must be updated to be useful, because of an obsolete information has no value. That's why, in recent years, information screens have taken a leading role and are being installed everywhere. Gradually the classic information boards are being replaced, where hundreds of sheets accumulated due to having limited space and the information provided there quickly became obsolete.

The new generation of information panels are optimized to offer dynamically updated information in real time and situated in places where everybody can see it. We can find them in public administrations to provide municipal information, on building fronts for all of kind of advertising or shops present sales and their product prices to customers, at bus stops or stations to show the transport schedule in real time or even on roads and highways where we can see traffic information, roads status or accidents.

In addition, the management of these panels can be done remotely and/or centrally, so you could have the same information in all of them.

Índice

Lista de figuras 6
Lista de formulas 8
Lista de tablas 8
Glosario 9
1. Introducción 10
1.1. Contexto y justificación del Trabajo 10
1.2. Objetivos del Trabajo 11
1.2.1 Objetivos generales. 11
1.2.2 Objetivos específicos 11
1.3. Enfoque y método seguido 11
1.4. Planificación del Trabajo 12
1.5. Riesgos e incidencias 13
1.6. Breve sumario de productos obtenidos 14
2. Materiales 15
3. Panel LED 16
3.1. Estudio sobre los LEDs 16
3.1.1. Descripción LEDs 16
3.1.2. Funcionamiento 17
3.1.3. Tipos de LED 18
3.1.4. Nuestro panel LED 20
3.2. Driver de control del panel LED 22
3.2.1. Funcionamiento de una pantalla LED 22
3.2.2. Registros de desplazamiento y 74HC595 24
3.2.3. Contador en anillo y CD4017 26
3.2.4. Control de nuestro panel 28
4. Sensor de luz ambiental 33
4.1. Descripción LDR 33
4.2. Funcionamiento 33
4.3. Tipos de LDR 34
4.4. Pros y contras de la utilización de un LDR en nuestro proyecto 34
4.5. Nuestro sensor de luz 34
5. Protocolo MODBUS 38
5.1. Protocolo 38
5.2. PUERTO RS-232 38
5.2.1. El conector 39
5.2.2. Implementación en DB-9 39
5.2.3. Cable de conexión 40
5.2.4. MAX232 40
6. Microcontrolador 42
7. Fuente alimentación 44
8. Programación 46
8.1. LDR y PWM 46
8.2. Modbus 49
8.3. Driver LEDs 52
8.3.1. Vertical 52
8.3.2. Horizontal 52
9. PCBs 57

9.1. Panel LED 57
9.2. Driver LED 58
9.3. Placa controladora 59
9.4.1 Montaje 60
9.4.2 Prototipo V1.0 61
10. Presupuesto 65
11. Conclusiones 66
12. Bibliografía 69
12. ANEXOS 70
12.1 ANEXO I 70

Lista de figuras

FIGURA 1. PANEL LED PARKING 10
FIGURA 2. PANEL LED ADIF 10
FIGURA 3. CALENDARIO DEL PROYECTO 13
FIGURA 4. DIAGRAMA DE GANNT 13
FIGURA 5. DIAGRAMA DE BLOQUES DEL CIRCUITO 16
FIGURA 6. PARTES DE UN LED 16
FIGURA 7. GAMA DE COLORES LED 17
FIGURA 8. CIRCUITO BÁSICO PARA UN LED 17
FIGURA 9. PWM DE UN LED 18
FIGURA 10. CIRCUITO 1 AC PARA UN LED 18
FIGURA 11. CIRCUITO 2 AC PARA UN LED 18
FIGURA 12. VARIEDAD DE FORMAS DE UN LED 19
FIGURA 13. LED SMD 19
FIGURA 14. LED COB 19
FIGURA 15. COMPARACIÓN RAID DE LEDS 20
FIGURA 16. MICROLED 20
FIGURA 17. MATRIZ LED 8X8 20
FIGURA 18. CIRCUITO MATRIZ LED 8X8 21
FIGURA 19. CIRCUITO MATRIZ LED 16X16 21
FIGURA 20. MATRIZ LED 16X16 21
FIGURA 21. LED WS2812 22
FIGURA 22. WS2812 EN SERIE 22
FIGURA 23. DEMULTIPLEXOR 1 ENTRADA 4 SALIDAS 23
FIGURA 24. RESPUESTA VISUAL 23
FIGURA 25. ORDEN ENCENDIDO DE LAS FILAS 23
FIGURA 26. REPRESENTACIÓN DE LA F 24
FIGURA 27. DESPLAZAMIENTO DE REGISTRO 24
FIGURA 28. PINEADO 74HC595 25
FIGURA 29. FLIP-FLOPS INTERNOS DEL 74HC595 25
FIGURA 30. RESPUESTA DEL 74HC595 26
FIGURA 31. CONTADOR EN ANILLO 26
FIGURA 32. TABLA DE VERDAD DEL CONTADOR EN ANILLO 27
FIGURA 33. ESTRUCTURA INTERNA CD4017 27
FIGURA 34. INTEGRADO CD4017 27
FIGURA 35. RESPUESTA CD4017 28
FIGURA 36. DOS CD4017 EN CASCADA 28
FIGURA 37. INTEGRADO UNL2981 29
FIGURA 38. ESQUEMA SINCRONIZADO VERTICAL 29
FIGURA 39. LDR 33
FIGURA 40. CIRCUITOS BÁSICOS PARA LDR 33
FIGURA 41. DIMENSIONES LDR 35
FIGURA 42. GRAFICA RESPUESTA LDR 35
FIGURA 43. CIRCUITO BÁSICO LDR 36
FIGURA 44. SEGUIDOR DE TENSIÓN 36
FIGURA 45. CIRCUITO LDR 36
FIGURA 46. CABLE RS232 38
FIGURA 47. CONECTOR DB9 MACHO Y HEMBRA 39
FIGURA 48. CONEXIÓN RS232 40
FIGURA 49. INTEGRADO MAX232 40

FIGURA 50. MX232 41
FIGURA 51. CIRCUITO MAX232 41
FIGURA 52. CIRCUITO PROTOBOARD MAX232 42
FIGURA 53. MICROCONTROLADOR PIC16F1615 43
FIGURA 54. CIRCUITO MICROCONTROLADOR 43
FIGURA 55. LM2596 44
FIGURA 56. CIRCUITO LM2596 5.0 45
FIGURA 57. DISEÑO CIRCUITO FUENTE DE ALIMENTACIÓN 45
FIGURA 58. CÓDIGO LECTOR VOLTAJE EN EL LDR 47
FIGURA 59. SIMULACIÓN CIRCUITO LDR Y SALIDA PWM AL MÍNIMO 48
FIGURA 60. SIMULACIÓN CIRCUITO LDR Y SALIDA PWM CASI AL MÁXIMO 48
FIGURA 61. CÓDIGO DE AJUSTE DE VALOR MÁXIMO EN EL LDR 48
FIGURA 62. SIMULACIÓN CIRCUITO LDR Y SALIDA PWM AL MÁXIMO 49
FIGURA 63. CÓDIGO DE VARIABLES PARA LA LIBRERÍA MODBUS 49
FIGURA 64. CÓDIGO QUE LEE LA ENTRADA DE DATOS 50
FIGURA 65. CÓDIGO PARA LEER LA ENTRADA DE DATOS Y RESPONDE 51
FIGURA 66. FUNCIONES LATCH() Y RESET() 52
FIGURA 67. FUNCIONES CLOCK() Y LATCH() 52
FIGURA 68. FUNCIÓN PARA MANDAR DATOS AL 74HC595 52
FIGURA 69. CÓDIGO DE COMPROBACIÓN SALIDAS 74HC595 53
FIGURA 70. SIMULACIÓN 74HC595 53
FIGURA 71. VALORES PARA REPRESENTAR LA LETRA F 54
FIGURA 72. VALORES PARA REPRESENTAR LA LETRA A Y B 54
FIGURA 73. TABLA ASCII 54
FIGURA 74. PCB DEL PANEL LED (CAPA SUPERIOR E INFERIOR) 57
FIGURA 75. PCB PANEL LED, COLOCACIÓN COMPONENTES 57
FIGURA 76. REPRESENTACIÓN 3D PANEL LED 57
FIGURA 77. PCB DEL DRIVER LED (CAPA SUPERIOR E INFERIOR) 58
FIGURA 78. COLOCACIÓN COMPONENTES DRIVER LED 58
FIGURA 79. REPRESENTACIÓN 3D PANEL DRIVER LED 58
FIGURA 80. PCB DEL MICROCONTROLADOR (CAPA SUPERIOR E INFERIOR) 59
FIGURA 81. COLOCACIÓN DE LOS COMPONENTES PCB MICROCONTROLADOR 59
FIGURA 82. REPRESENTACIÓN 3D PCB MICROCONTROLADOR 59
FIGURA 83. ENSAMBLADO DE LAS PCB (LED Y DRIVER) 60
FIGURA 84. RESULTADO FINAL (FRONTAL Y TRASERO) 60
FIGURA 85. CONEXIONADO CINCO PANELES LED 60
FIGURA 86. RESULTADO FINAL CINCO PANELES LED 61
FIGURA 87. PLACAS PCB (MICROCONTROLADOR, DRIVER Y PANEL LED) 61
FIGURA 88. COMPONENTES Y PCB 61
FIGURA 89. PROTOTIPO YA CON LOS COMPONENTES SOLDADOS 62
FIGURA 90. VISTA POSTERIOR DEL PROTOTIPO ENSAMBLADO 62
FIGURA 91. VISTA FRONTAL DEL PROTOTIPO ENSAMBLADO 62
FIGURA 92. CONEXIÓN PICK3 CON EL MICROCONTROLADOR 62
FIGURA 93. PINEADO PICK3 CON PIC16F1615 62
FIGURA 94. SOFTWARE MPLAB V6.05 63
FIGURA 95. COMPROBACIÓN MATRIZ LED 63
FIGURA 96. ERROR EN LAS ETIQUETAS DE KICAD 64
FIGURA 97. ARREGLO DE LA PCB DEL DRIVER LED 64
FIGURA 98. PROTOTIPO V1.0 FUNCIONANDO 64
FIGURA 99. DIAGRAMA DE GANTT FINAL 66

Lista de formulas

ECUACIÓN 1. ECUACIÓN PARA LIMITAR LA CORRIENTE PARA UN LED ..... **¡ERROR! MARCADOR NO DEFINIDO.**
ECUACIÓN 2. DIVISOR DE TENSIÓN ..... **¡ERROR! MARCADOR NO DEFINIDO.**
ECUACIÓN 3. CONSUMO ELÉCTRICO CINCO PANELES ..... **¡ERROR! MARCADOR NO DEFINIDO.**
ECUACIÓN 4. PERIODO DEL PWM ..... **¡ERROR! MARCADOR NO DEFINIDO.**
ECUACIÓN 5. FRECUENCIA DEL PWM ..... **¡ERROR! MARCADOR NO DEFINIDO.**
ECUACIÓN 6. FRECUENCIA OBTENIDA EN LA SALIDA PWM ..... **¡ERROR! MARCADOR NO DEFINIDO.**

Lista de tablas

TABLA 1. MATERIALES DE FABRICACIÓN LED 17
TABLA 2. CARACTERÍSTICAS MATRIZ LED 1088AS 20
TABLA 3. COMPARATIVA INTEGRADOS DRIVERS PARA LEDS 24
TABLA 4. CARACTERÍSTICAS ELÉCTRICAS UDN2981 29
TABLA 5. CARACTERÍSTICAS LDR GL5528 35
TABLA 6. VALORES LDR (RESISTENCIA E INTENSIDAD LUMÍNICA) 35
TABLA 7. PINES CONECTOR DB9 RS232 39
TABLA 8. COMPARATIVA DRIVERS RS-232 40
TABLA 9. CONSUMO ELÉCTRICO COMPONENTES 44

Glosario

LED: light-emitting diode o diodo emisor de luz.

RS232: Recommended Standard 232 o Estándar Recomendado 232.

PCB: Printed Circuit Board o placa de circuito impreso.

PN: estructura fundamental de los componentes electrónicos llamados semiconductores

GND: ground, punto de referencia de tensiones en un circuito, normalmente 0 V.

LDR: light-dependent resistor o fotorresistencia

LUX: Unidad de medida para el nivel de luz en el Sistema Internacional.

TCP/IP: Transmission Control Protocol/Internet Protocol o Protocolo de control de transmisión/Protocolo de Internet

UDP: Protocolo de control de transmisión/Protocolo de Internet

RS-485: Norma técnica que especifica las características eléctricas de un sistema de comunicación serie.

NZR-L: No Zero Return, codificación de una señal digital de no retorno a cero

DB-25: Conector eléctrico D-sub de 25 pines. Normalmente para conectar computadoras a periféricos.

DB-9: Conector eléctrico D-sub (D-subminiature) de 9 pines.

DTE: Data termination equipment, equipo de terminación de datos de una red responsable de solicitar y/o enviar información activamente.

DCE: Data Communication Equipment, equipo conectado al DTE mediante una línea RS-232

RX: abreviatura de recepción en telecomunicaciones.

TX: abreviatura de transmisor en telecomunicaciones.

TTL: Transistor-Transistor Logic, circuito que utiliza el transistor como elemento principal donde los niveles de tensión suelen ser 5 V

USART: Universal Synchronous and Asynchronous serial Receiver and Transmitter, protocolo de dos hilos para el intercambio de datos serie.

PWM: Pulse-width modulation, modulacion por ancho de pulsos de una señal.

1. Introducción

1.1. Contexto y justificación del Trabajo

En la actualidad la información se ha convertido en algo muy importante y debido a esto nos gusta estar informados en todo momento. Por ese motivo, las pantallas informativas han tomado gran protagonismo en los últimos años y están presentes en casi todos lados.

Una de las grandes ventajas de las pantallas informativas es que en ellas puedes mostrar cantidad de información y mantenerla actualizada constantemente, ya que una información obsoleta no tiene ningún valor y es algo que ocurría constantemente con el uso de soportes más tradicionales, pero con el uso de las pantallas informativas se puede ofrecer información en tiempo real.

Figura 1. Panel LED parking

Los principales lugares donde podemos encontrar este tipo de pantallas suelen ser en administraciones públicas para ofrecer información municipal, en fachadas de edificios orientadas para publicidad o comercios ofreciendo ofertas y precios a clientes, o incluso en paradas de autobús o estaciones para mostrar el horario de trenes o autobuses.

Existen multitud de tipos de pantallas, pero la tecnología más extendida son las pantallas de LED, estas constan de una matriz de diodos LED controlados por un microcontrolador que los enciende y los apaga según qué información tenga que mostrar.

Figura 2. Panel LED ADIF

Podemos encontrar multitud de tamaños e incluso pantallas modulares y totalmente escalables para crear el tamaño adecuado según las necesidades. Además, la tecnología LED tiene ciertas características que la hacen muy atractiva, como, por ejemplo, la durabilidad, esta tecnología tiene vida útil muy larga y requiere poco mantenimiento. También el LED tiene muy buena visibilidad, incluso en ambientes muy iluminados, y esto es debido a que los LED emiten luz por sí solos. Por último, las pantallas informativas son dispositivos que estarán en funcionamiento durante mucho tiempo, en algunos lugares prácticamente estarán en funcionamiento 24/7 y por eso la preocupación por un bajo consumo eléctrico es muy importante y en ese aspecto los LEDs son muy eficientes.

1.2. Objetivos del Trabajo

1.2.1 Objetivos generales.

El objetivo principal de este proyecto es la creación de un prototipo de una pantalla LEDs, de un tamaño de 16x16 LEDs. Que, mediante una entrada de datos RS232 sea capaz de recibir datos y mostrarlos. El dispositivo también será capaz de conocer la luminosidad exterior y ajustar la intensidad de los LEDs para que siempre sean visibles.

1.2.2 Objetivos específicos

- Diseño de un circuito analógico que adapte la señal analógica proporcionada por un sensor de luminosidad a una de las entradas del microcontrolador, para que el microcontrolador sea capaz de consultar la luminosidad exterior y así regular la intensidad de los LEDs.

- Diseño de un circuito que adapte los niveles de señal de un puerto RS232 a los niveles específicos de nuestro microcontrolador para poder leer los datos que entren desde este puerto.

- Diseño de un circuito para el control del panel LED, que sea capaz de excitar los LEDs y controlar una matriz de tamaño 16x16. El circuito debe cumplir con el objetivo de utilizar el mínimo número posible de entradas/salidas de nuestro microcontrolador.

- Selección de un microcontrolador de acuerdo a nuestras necesitadas y que sea capaz de cumplir todas las tareas.

- Implementar un firmware que sea capaz de leer los datos de entrada por el puerto RS232, y mostrar la información por la matriz LED, siempre contando con la luminosidad exterior, ajustando la intensidad de la pantalla.

- Por último, diseñar una fuente de alimentación para dar salida a los voltajes necesarios para nuestro proyecto. Siempre teniendo en cuenta el consumo de estos.

1.3. Enfoque y método seguido

El alcance de este proyecto no se limitará a la parte teórica, sino que se diseñará y fabricará un prototipo básico que nos acerque lo más posible al dispositivo de consumo para poder estudiar y analizar los problemas que podamos encontrar.

El trabajo se dividirá en varias etapas,

- La primera de ellas se centrará en la investigación de los paneles informativos, se estudiará el funcionamiento de los tipos de LEDs y las clases que existen en el mercado. También investigaremos como se realiza el control de una matriz de LEDs

y los interfaces de comunicación entre los drivers y el microcontrolador y la interface por donde llegarán datos al panel. Se tendrá que investigar los microcontroladores más adecuados para este proyecto teniendo en cuenta las especificaciones de este, y por último las fuentes de alimentación.

- En la segunda etapa se elegirán los componentes necesarios y más adecuados para nuestro propósito, los LEDs, microcontroladores y demás componentes.

- En la tercera etapa se procederá a diseñar los circuitos necesarios para que funcione el microcontrolador. Diseñaremos el circuito driver para el control de la matriz de LED, el circuito analógico de control de luminosidad, la interface de datos y por último la fuente de alimentación.

- En esta etapa se diseñará una protoboard y se creará un firmware básico para mostrar información en los LEDs. Se procederá a hacer pruebas y correcciones.

- Y se finalizará realizando un diseño del *layout* definitivo en PCB.

1.4. Planificación del Trabajo

El comienzo del trabajo será el 29/9/2022, contaremos con 11 días laborables para presentar la PEC1 donde, principalmente realizaremos un pequeño briefing del proyecto, aquí definiremos el proyecto, el alcance de este, las fases en las que lo dividiremos y la planificación. Contando que se dediquen 4 horas diarias de media, obtendremos unas 44 horas de trabajo.

El día 18/10/2022 comenzaremos con PEC2 del proyecto en la que dedicaremos la primera, segunda y tercera etapa, anteriormente descritas. Tendremos unos 19 laborables días para realizar todo, por eso dedicaremos una media de 4 horas diarias que hacen 76 horas de trabajo.

El día 15/11/2022 entraremos en la PEC3 y contaremos con 24 días laborables, unas 96 horas de trabajo, aquí desarrollaremos la cuarta etapa, el montaje en protoboard, desarrollo del firmware, aquí es donde posiblemente nos encontremos multitud de problemas no contemplados.

El día 20/12/22 comenzamos la última entrega, la PEC4, contamos con 23 días laborables, que serían unas 92 horas de trabajo. Aquí realizaremos la quinta y última etapa del proyecto y redactaremos la memoria, que hasta ahora habrá sido un simple borrador, para poder realizar la entrega el 23/01/2022.

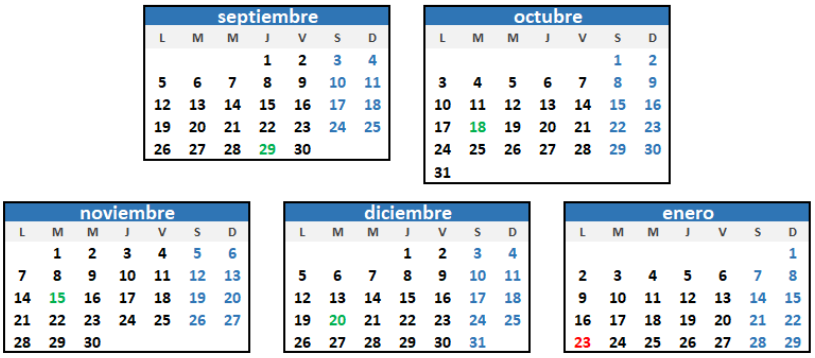


Figura 3. Calendario del proyecto

A continuación, vemos en el siguiente diagrama de Gannt donde hemos dividido el trabajo en cuatro entregas y en cada una de ellas comenzamos con un par de días de corrección de errores de la anterior entrega y unos días al final para poder redactarlo todo.

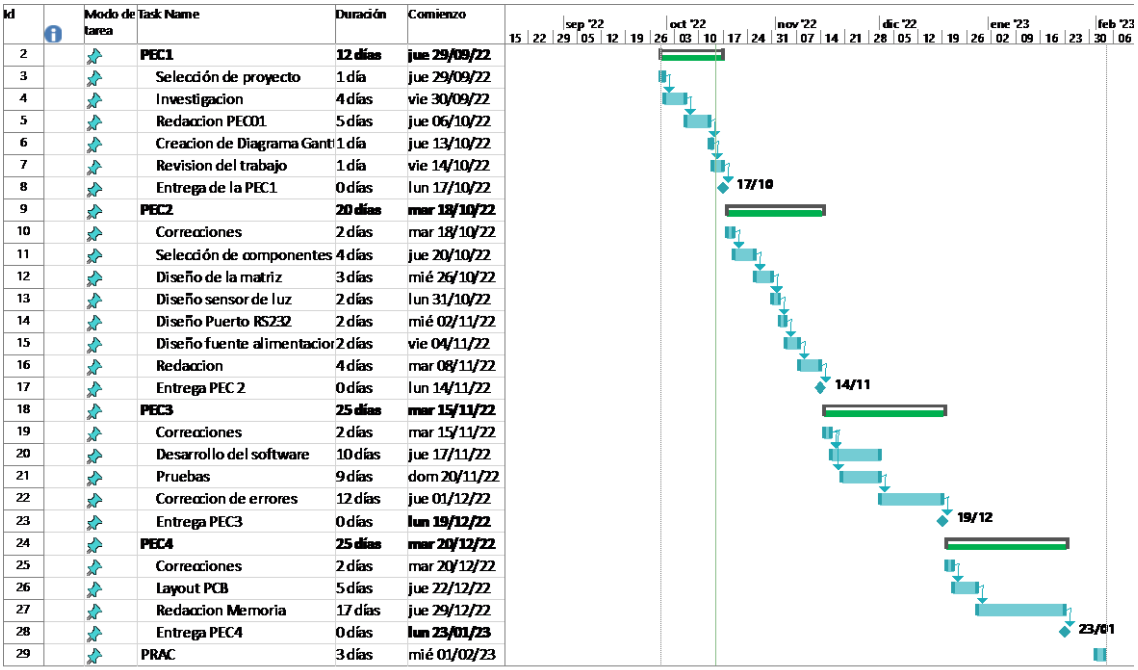


Figura 4. Diagrama de Gannt

1.5. Riesgos e incidencias

Durante la realización del proyecto nos podemos enfrentar a diferentes riesgos como, por ejemplo:

- Que no se cumpla con el cronograma, es posible que las tareas del proyecto requieran más tiempo y más desarrollo del esperado. Es quizá el riesgo con más probabilidad y para evitarlo debemos tener una buena definición del alcance del proyecto, también una buena planificación, sobreestimando los tiempos asignados a cada tarea y no ajustar demasiado los plazos. En el caso que se cumpla se realizará una revisión del cronograma cada cierto tiempo para ir realizando acciones correctoras.

- Distorsión del alcance, donde empieza el proyecto está claro, pero podemos llegar a distorsionar donde terminar, hasta donde está definido el alcance de este. Este tipo de riesgos tienen una probabilidad media y para evitar que ocurra es muy importante definir muy bien el alcance de este.

- Falta de recursos, el proyecto se llevará a cabo con componentes comunes y fáciles de encontrar, pero puede que ocurra que algún componente tardemos en conseguirlo o sea imposible. Por ese motivo un habrá que estudiar el mercado y los tiempos de suministro a la hora de seleccionar los componentes.

1.6. Breve sumario de productos obtenidos

Al final del proyecto obtendremos los siguientes productos:

- Memoria, que contendrá:
    - Funcionamiento del panel
    - Diseños electrónicos en CAD
    - Programación del microcontrolador
- Panel LED 16x16 que consta de:
    - PCB de panel LED
    - PCB con los drivers del panel
    - PCB con el microcontrolador

2. Materiales

Para poder llevar a cabo este proyecto será necesario una serie de recursos:

Hardware,

- Portátil con un procesador Intel i7 con 16 Gb de memoria RAM y 512Gb de disco duro, para poder realizar todo el desarrollo, simulaciones e investigación.
- También contaremos con un programador para microcontroladores MPLAB PICKIT 3 para cargar nuestro programa.

Software,

- Todo el proyecto será planificado y escrito en Microsoft Office 16 (Word, Excel y Project).
- Utilizaremos Fritzing 0.9.1b.64 para los dibujos de los montajes en protoboard
- Para el diseño de la PCB utilizaremos Kicad 6.0
- Utilizaremos PIC C Compiler V5.049 para programar el microcontrolador
- Utilizaremos Proteous 8 para hacer simulaciones

3. Panel LED

Para una mejor comprensión del diseño del panel LED vemos que este se divide en seis bloques:

- Matriz LED 16x16
- Driver de control de la matriz
- Circuito de adaptación de señal para el sensor de luz
- Circuito de adaptación de señal para el canal RS232
- Circuito para el funcionamiento de un microcontrolador
- Fuente de alimentación

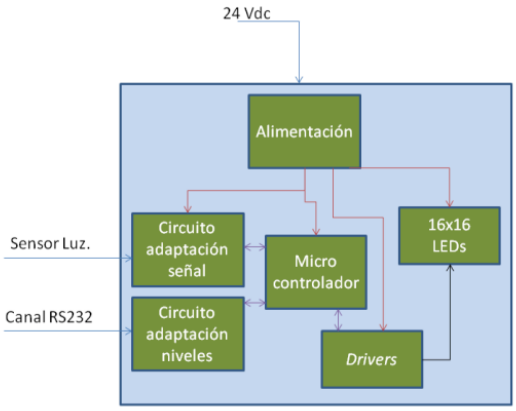


Figura 5. Diagrama de bloques del circuito

3.1. Estudio sobre los LEDs

3.1.1. Descripción LEDs

Un diodo LED no es más que un componente electrónico, más específicamente un semiconductor, con la capacidad de emitir luz. Un diodo LED tiene dos patillas y permite la circulación de la corriente en un solo sentido. Cuando la corriente circula en el sentido correcto, se dice que está polarizado, es cuando entonces emite luz.

El diodo LED está formado por una capsula plástica que contiene un semiconductor y dos patillas, una larga o ánodo (+) y una corta o cátodo (-).

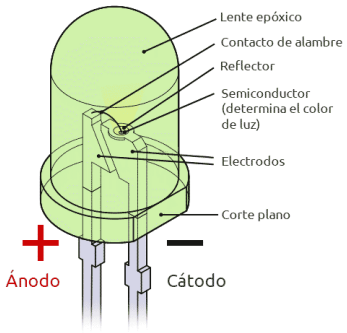


Figura 6. Partes de un LED

3.1.2. Funcionamiento

El diodo está compuesto por una unión PN y cuando esta es recorrida por una corriente es capaz de emitir fotones luminosos gracias al fenómeno de la electroluminiscencia. Los electrones de la región N cruzan la barrera de potencial y ocupan los huecos de la región P tal y como lo hace un diodo normal. Excepto que en este tipo de semiconductores no solo se libera calor sino también se elimina el exceso de energía en forma de luz.

Figura 7. Gama de colores LED

Esta luz, dependiendo de la aleación con la que esté fabricado dará diferentes longitudes de onda, dando lugar a diferentes colores.

Material	**Color**
Arseniuro de galio (GaAs)	Infrarrojo
Arseniuro de galio y aluminio (AlGaAs)	Rojo e infrarrojo
Arseniuro fosfuro de galio (GaAsP)	Rojo, anaranjado y amarillo
Fosfuro de galio (GaP)	Verde
Nitruro de galio (GaN)	Verde
Seleniuro de zinc (ZnSe)	Azul
Nitruro de galio e indio (InGaN	Azul
Carburo de silicio (SiC)	Azul
Diamante (C)	Ultravioleta

Tabla 1. Materiales de fabricación LED

Un LED puede ser excitado mediante una corriente continua, por impulsos (PWM) o por corriente alterna.

- Si queremos utilizar corriente continua el circuito más básico sería el siguiente, donde la resistencia limitaría la cantidad de corriente que circula por él mediante la formula:

$$R = \frac{V_{in} - V_{led}}{I_{led}} \quad (1)$$

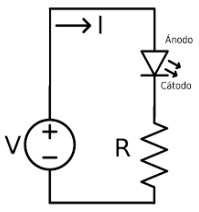


Figura 8. Circuito básico para un LED

- Si es por impulsos o PWM (Pulse Width Modulation), suele ser utilizado para regular la intensidad del LED. Una señal cuadrada es utilizada para activar el LED, y dependiendo del ciclo de trabajo se iluminará más o menos.

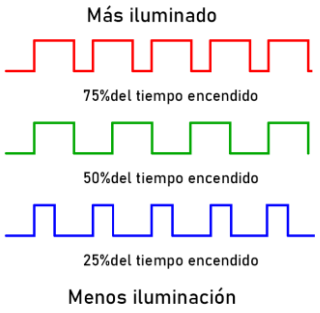


Figura 9. PWM de un LED

- Si lo que queremos es usar corriente alterna, debemos añadir un circuito protector para que la corriente inversa no lo destruya. De esta manera solamente iluminará el LED en los ciclos positivos de la señal. O bien se podría filtrar la señal mediante un puente de diodos.

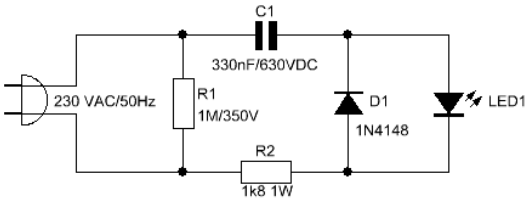


Figura 10. Circuito 1 AC para un LED

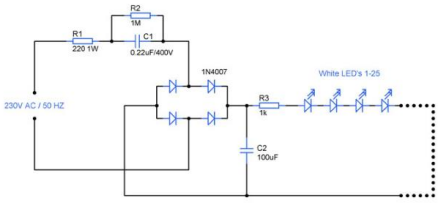


Figura 11. Circuito 2 AC para un LED

3.1.3. Tipos de LED

En el mercado podemos encontrar gran variedad de encapsulados, redondos, cuadrados, SMD de diferentes tamaños, pero existen cuatro grandes tipos:

- LED DIP (Dual In-Line Package), son los LEDs tradicionales formados por un encapsulado de plástico y dos o tres (RGB LEDs) patillas.

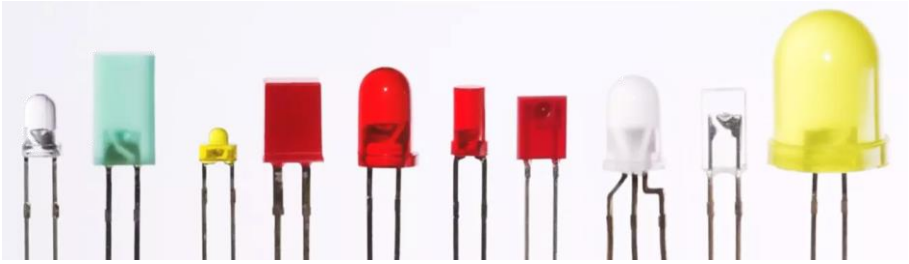


Figura 12. Variedad de formas de un LED

- LED SMD (Surface Mounted Diode), son los LEDs que van soldados sobre la superficie de una placa, suelen ser bastante más eficientes que los DIP y los podemos encontrar en cantidad de tamaños diferentes (3014, 2835 o 1206 entre otros). El desarrollo de estos LEDs permitió la integración de 3 LEDs en el mismo chip logrando gamas de colores de hasta 16 millones de colores diferentes. También trajo consigo la creación de las tiras de LED y downlight, por el contrario, generan bastante calor.

Figura 13. LED SMD

- LED COB (Chip on Board) se basa en la inclusión de varios diodos dentro del mismo encapsulado, reduciendo el consumo y aumentando la luminosidad.

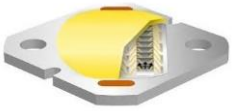


Figura 14. LED COB

Aquí vemos la diferencia entre un raid de LEDs, LEDs SMD y un COB

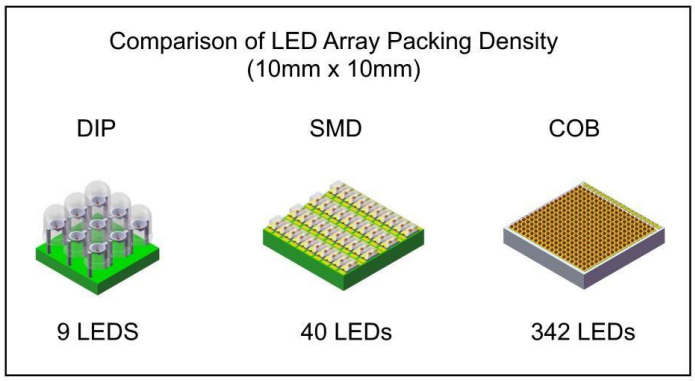


Figura 15. Comparación RAID de LEDs

- MicroLED, se emplean una serie de LEDs microscópicos para definir un pixel, más populares para la creación de pantallas planas de LED. Mejoran la eficiencia, la latencia y el contraste.

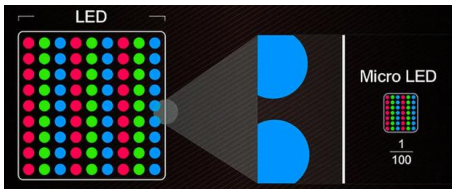


Figura 16. MicroLED

3.1.4. Nuestro panel LED

Como hemos visto existen cantidad de tipos de LED pero para el proceso de prototipado elegiremos una matriz LED de 8x8 de LEDs rojos bastante estándar, que es fácil de encontrar y muy económico, en el mercado existen ya paneles de 16x16 pero son bastante más caros que lo que nos costaría juntar cuatro paneles de 8x8. Más adelante, se podría optar por otro tipo de LED de alta luminosidad y mayor eficiencia.

Elegí cuatro matrices 8x8 modelo 1088AS y sus principales características son:

Figura 17. Matriz LED 8x8

Color	rojo, longitud de onda 625 ~ 630nm
Dimensiones	32mm x 32mm x 8.0mm
Tipo	Filas Cátodo Columnas ánodo
Corriente	20mA
Voltaje	2,1V

Tabla 2. Características matriz led 1088AS

Este es pinout de la matriz y su conexionado interno de los LEDs.

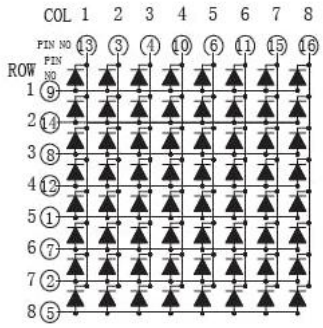


Figura 18. Circuito matriz LED 8x8

En nuestro proyecto diseñaremos un panel de 16x16, por lo que necesitamos juntar cuatro paneles de 8x8 y así formar una matriz 16x16, en total contaremos con 256 LEDs. Para juntar los paneles bastaría con unir columnas y filas.

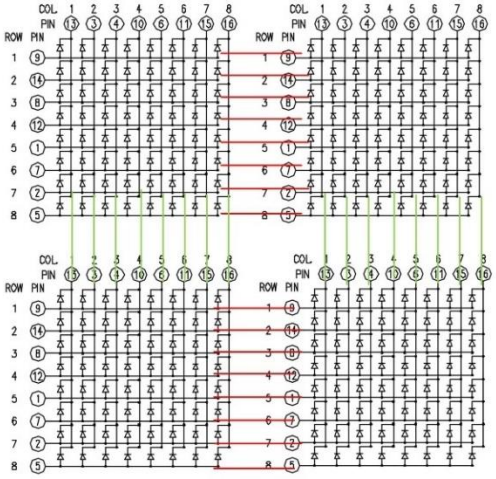


Figura 19. Circuito matriz LED 16x16

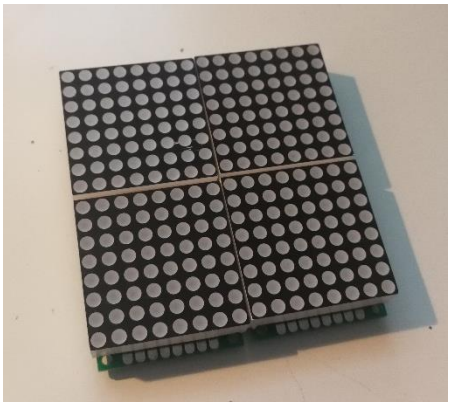


Figura 20. Matriz LED 16x16

3.2. Driver de control del panel LED

3.2.1. Funcionamiento de una pantalla LED

Encender y apagar un solo LED es algo muy sencillo, pero en una pantalla puede haber cientos e incluso miles de LEDs formando una gran matriz, lo que lo complica un poco, además de conectar todos y cada uno de los LEDs directamente a un microcontrolador es imposible. Por ese motivo, hay que diseñar un driver de control que hace uso de algunas estrategias que ayudan en esta labor.

Por ejemplo, en el mercado existe un tipo de LED RGB que lleva un integrado WS2812 que es capaz, mediante un control SISO (serial in serial out) de 24 bits, producir hasta 256 valores de intensidad para cada color, 8 bits por color G, 8 bits por color R y 8 bits por color B lo que suman 16777216 diferentes colores.

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Según su datasheet estos LEDs se disponen en cascada conectando el primero al microcontrolador y los datos van pasando por todos los LEDs, desde el primero, que se queda con los 24 primeros bits y deja pasar el resto hacia los siguientes LEDs, hasta que se le manda un RESET.

Figura 21. LED WS2812

La principal ventaja sería que mediante tres pines (GND, Vcc y Data) podemos controlar cientos de LEDs pero si en algún momento quisiéramos encenderlos todos en blanco al mismo tiempo, el consumo eléctrico se dispararía, por ejemplo. Un solo LED a máxima intensidad según su datasheet consume 60mA (20mA por color) por lo que un panel de 256 LEDs, como requiere nuestro proyecto, obtendríamos un consumo total de $256 \cdot 60mA = 15{,}360\ A$, se trata de mucha corriente y queda descartado ya que uno de los objetivos del proyecto es el rendimiento energético.

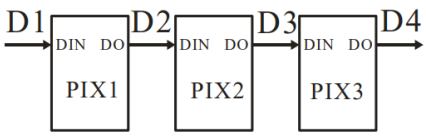


Figura 22. WS2812 en serie

Otra forma de poder controlar cientos de LEDs y que cuenta con una mejor eficiencia energética seria la multiplexación. La multiplexación de una señal no es más que el envío de varias señales, tanto analógicas como digitales, desde varias entradas a una sola entrada. De esta manera, realizando una multiplexación por división de tiempo somos capaces de que todas las señales puedan usar un mismo recurso.

Para este proyecto haremos uso de un demultiplexor, que hace todo lo contrario. Enviar desde unos pocos de pines de salida de nuestro microcontrolador hacia cientos de LEDs para poderlos controlar todos ellos al mismo tiempo.

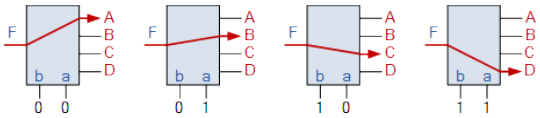


Figura 23. Demultiplexor 1 entrada 4 salidas

Por otro lado, debemos conocer un fenómeno visual que afecta a la vista humana, la persistencia retiniana. Que dice que una imagen permanece en la retina de un humano una décima de segundo hasta que va desapareciendo. Por lo tanto, si podemos encender y apagar los LEDs a una velocidad superior de 60 Hz el ojo humano lo verá como el sí LED estuviera encendido todo el tiempo.

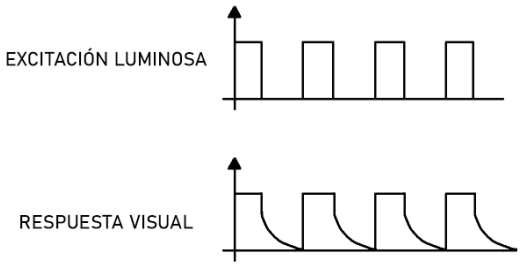


Figura 24. Respuesta visual

Nuestro panel LED tendrá 16 filas por 16 columnas, lo que haremos será agrupar las filas de 16 e ir encendiéndolas y apagándolas en orden descendente, desde la primera fila hasta la última. Para que dé la sensación que están todos los LEDs encendidos. Aunque estos estén solamente 1/16 parte del tiempo, esto, por supuesto afectará a la luminosidad del panel.

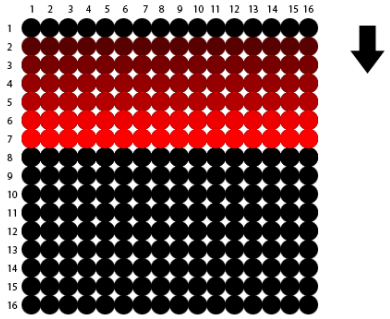


Figura 25. Orden encendido de las filas

Si lo que queremos es mostrar algún dibujo o letra tendremos que sincronizar el barrido vertical con los encendidos individuales de los LEDs de los grupos de dieciséis. Por ejemplo, de la primera y segunda fila no encenderemos ningún LED, de la tercera, cuarta y quinta encenderemos los LEDs desde el cuarto hasta el décimo. Luego de la sexta, séptima y octava fila solo los LEDs cuatro, cinco y seis, etc. Así podremos dibujar cualquier letra.

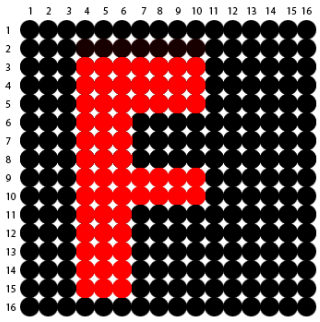


Figura 26. Representación de la F

3.2.2. Registros de desplazamiento y 74HC595

Para este fin haremos uso de un recurso muy utilizado en electrónica, un *shift register* o registro de desplazamiento. Bastaría con disponer una serie de flip-flops en cascada, donde la salida del anterior está conectado a la entrada de la siguiente puerta. De tal manera que los datos que introducimos en el primer flip-flop son pasados al siguiente al ritmo de una entrada de reloj de esta manera una entrada de datos en serie se convertiría en una salida de datos en paralelo.

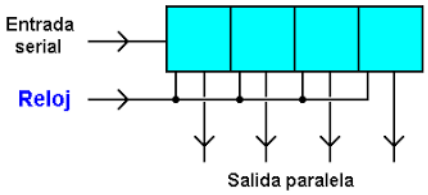


Figura 27. Desplazamiento de registro

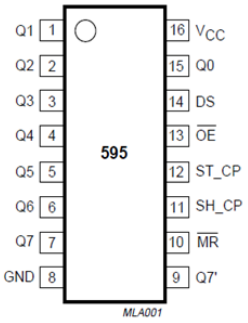
En el mercado existen varios integrados que realizan esta función, algunos de ellos son específicos para *display* LEDs de siete segmentos, hasta algunos que manejan hasta 16 LEDs como puede ser el MAX7219, HT16K33, SX1509, 74HC595.

Integrado	**MAX7219**	**HT16K33**	**SX1509**	**74HC595**
Descripción	Control Serial display led 8 digitos.	Controlador LED 16 dígitos de 8 segmentos	Expansor I/O 16 canales	Registro de desplazamiento 8bits Serie a paralelo
Encapsulado	DIP 24 pines	SOP 28 pines	QFN-UT 20 pines	DIP 16 pines
Precio	11,61 € (RS components)	1,026 € (RS component)	6,50 € (bricogeek)	0,09 € (Aliexpress)

Tabla 3. Comparativa integrados drivers para LEDs

Vemos que el integrado HT16K33 es un driver LED que maneja hasta *displays* de 16x8, por lo que con dos de estos tendríamos todo lo necesario para nuestro proyecto, pero lo descartamos ya que solamente lo encontramos disponible en encapsulado SOP y para un prototipo es más manejables los encapsulados DIP. El MAX7219 y el SX1509 también quedan descartados por precio ya que podemos encontrar el 74HC595 por tan solo 0,09 € en encapsulado DIP 16 pines. Este integrado es muy popular debido a su versatilidad, es de fácil acceso, económico y con muchísima información.

Este integrado viene definido en su datasheet como “serial a paralelo de 8-bits con registros de desplazamiento con latches 3 estados”. Viene en un encapsulado tipo DIP de 16 pines, donde vemos sus ocho salidas en paralelo Q0-Q7, también posee una salida Q’7 que es utilizada para conectar más integrados en serie y así poder ampliar las salidas. El SH_CP (shift register Clock input) sería la entrada de reloj que marca la velocidad de entrada de datos a través de DS, ST_CP marcaría los tiempos de salida de los datos y por último tenemos OE (Output enable) y MR (Master reset) que son entradas inversas que habilitan la salida o resetean los estados los flip-flops internos.



PINNING

PIN	SYMBOL	DESCRIPTION
1	Q1	parallel data output
2	Q2	parallel data output
3	Q3	parallel data output
4	Q4	parallel data output
5	Q5	parallel data output
6	Q6	parallel data output
7	Q7	parallel data output
8	GND	ground (0 V)
9	Q7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_CP	shift register clock input
12	ST_CP	storage register clock input
13	$\overline{OE}$	output enable (active LOW)
14	DS	serial data input
15	Q0	parallel data output
16	V_{CC}	positive supply voltage

Figura 28. Pineado 74HC595

Internamente esta sería la disposición de los flip-flops:

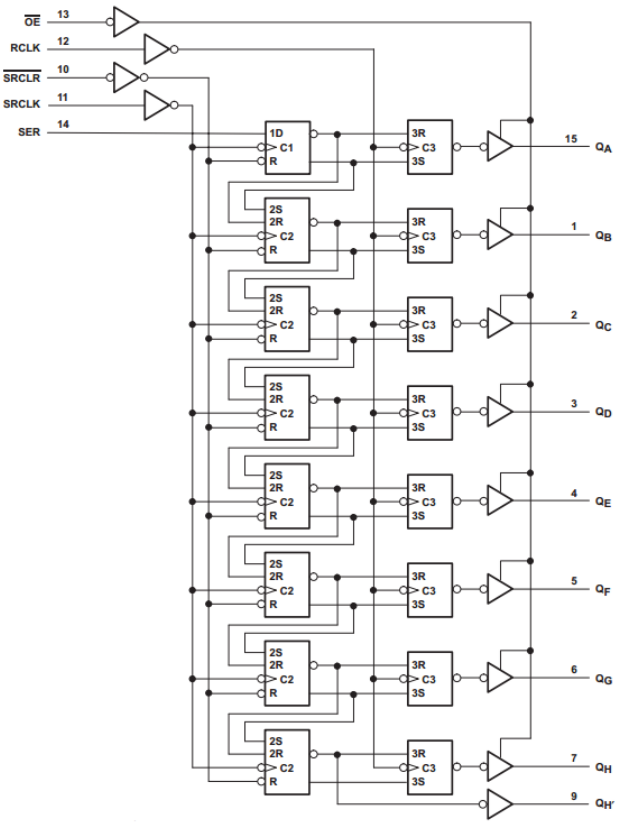


Figura 29. Flip-flops internos del 74HC595

Vamos a ver como funcionaria nuestra 74HC595, en la entrada SHCP introduciremos una señal de reloj. Esta señal marcará la velocidad a la que los datos se vayan desplazando desde el primer registro hasta el último.

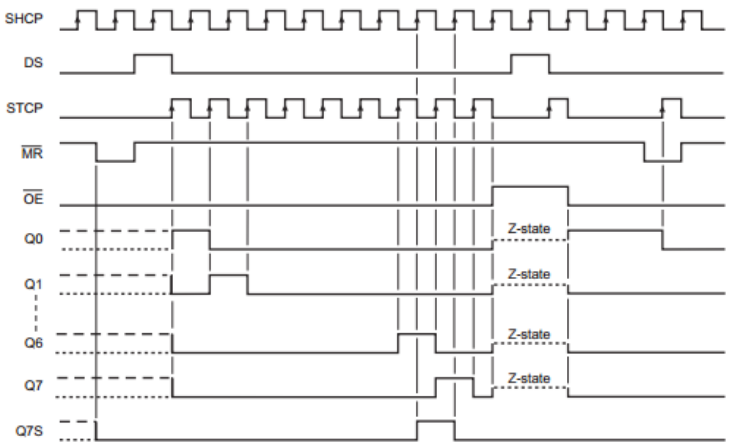


Figura 30. Respuesta del 74HC595

Cuando todos los registros tengan el valor deseado simplemente con dar un pulso a la entrada STCP, las salidas desde Q0-Q7 cambiaran de valor, y pasarán al valor que tengan los registros en ese momento. Todo ello ocurrirá si MR se encuentra en estado HIGH y OE en LOW.

3.2.3. Contador en anillo y CD4017

Un contador en anillo no es más que un circuito contador compuesto por flip-flops en serie, conectados a un registro de desplazamiento y la última salida del flip-flop conectado a la entrada del primero formando una estructura en forma de anillo. De esta manera una vez que acabe de contar volverá al comienzo.

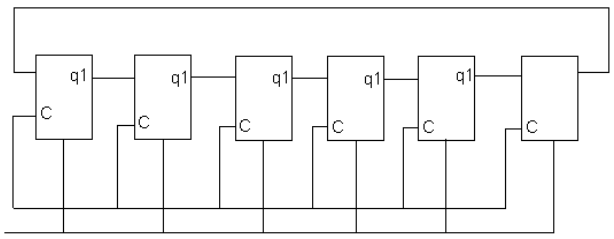


Figura 31. Contador en anillo

0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

Figura 32. Tabla de verdad del contador en anillo

Un integrado con esta configuración sería el CD4017, que se trata de un contador decimal capaz de contar desde uno hasta diez, al ritmo de una entrada de reloj.

Como decíamos, está compuesto por un cinco flip-flops dispuestos en serie:

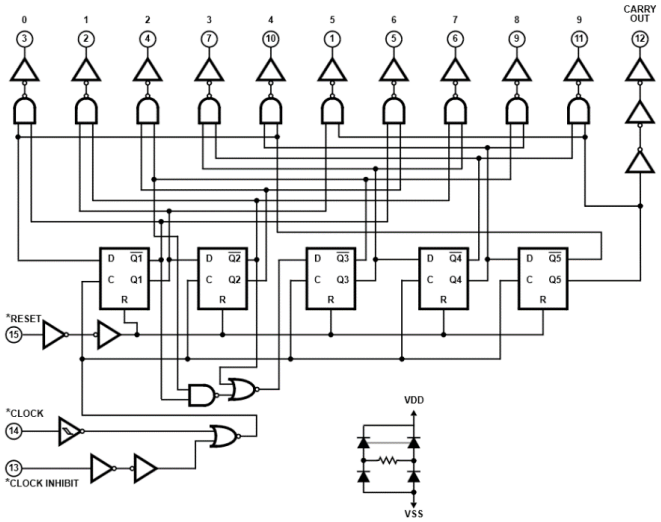


Figura 33. Estructura interna CD4017

Aquí vemos la disposición de los pines:

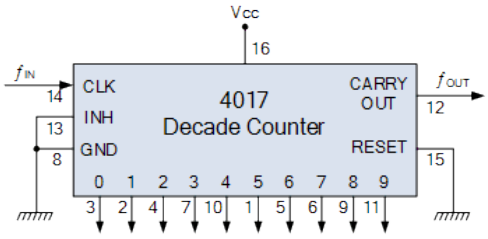


Figura 34. Integrado CD4017

Y por último la respuesta en sus salidas al ritmo de los ciclos de reloj:

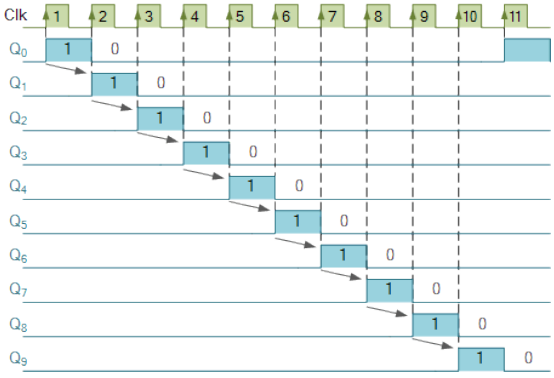


Figura 35. Respuesta CD4017

3.2.4. Control de nuestro panel

Para controlar nuestro panel LED necesitaremos dos 74HC595 para controlar el encendido de los LEDs en grupos de 16 y para el barrido vertical implementaremos un circuito contador en anillo con dos CD4017.

9.4.1.1 Barrido vertical

Sencillamente lo que realizará será ir encendiendo las filas de LED en orden, desde la primera fila hasta la última y luego volver a empezar. De esta manera controlaremos el barrido vertical de manera automática y no tendremos que implementar nada de programación en el microcontrolador, solamente sincronizar el comienzo de la secuencia.

Para esto utilizaremos dos CD4017 conectados en cascada y sincronizado con el STCP de los 74HC595.

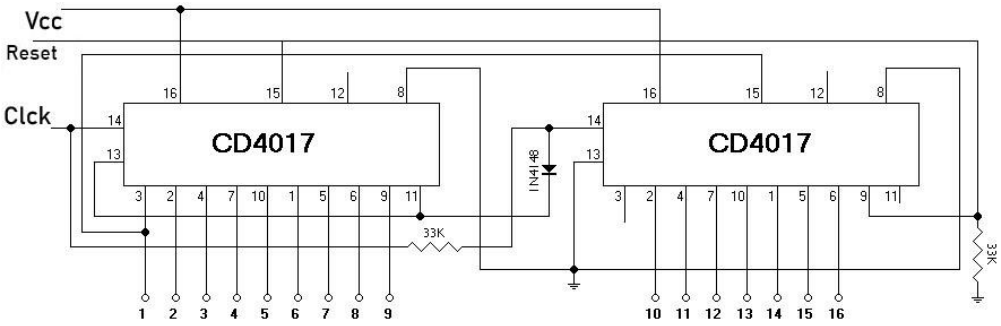


Figura 36. Dos CD4017 en cascada

Como se puede dar el caso que necesitemos los 16 LEDs encendidos al mismo tiempo cada salida debería poder entregar una corriente de 320mA (20mA por LED). Vemos en su

datasheet que no llega a entregar tanta corriente, por lo que utilizaremos el driver UDN2981 para entregar dicha corriente. El UDN2981, según su datasheet es un driver de 8 canales capaz de entregar, en ciertas circunstancias hasta 500mA por salida. Por lo tanto, es suficiente para el diseño de nuestro panel.

Characteristic	Symbol	Notes	Rating	Units
Output Voltage Range	V_{CE}		5 to 50	V
Input Voltage	V_{IN}	UDN2981	25	V
		A2982, UDN2982	20	V
Output Current	I_{OUT}		–500	mA
Package Power Dissipation	P_D	See graph	–	–
Operating Ambient Temperature	T_A	Range E	–40 to 85	°C
		Range S	–20 to 85	°C
Maximum Junction Temperature	T_J(max)		150	°C
Storage Temperature	T_{stg}		–55 to 150	°C

Tabla 4. Características eléctricas UDN2981

Aquí vemos la configuración del integrado.

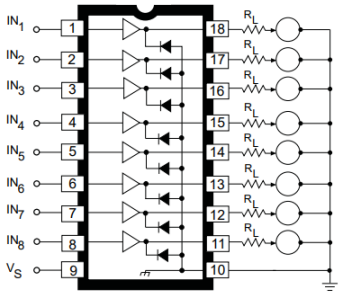


Figura 37. Integrado UNL2981

Y conectaremos las salidas de los CD4017 a las entradas de dos UDN2981, de la siguiente manera.

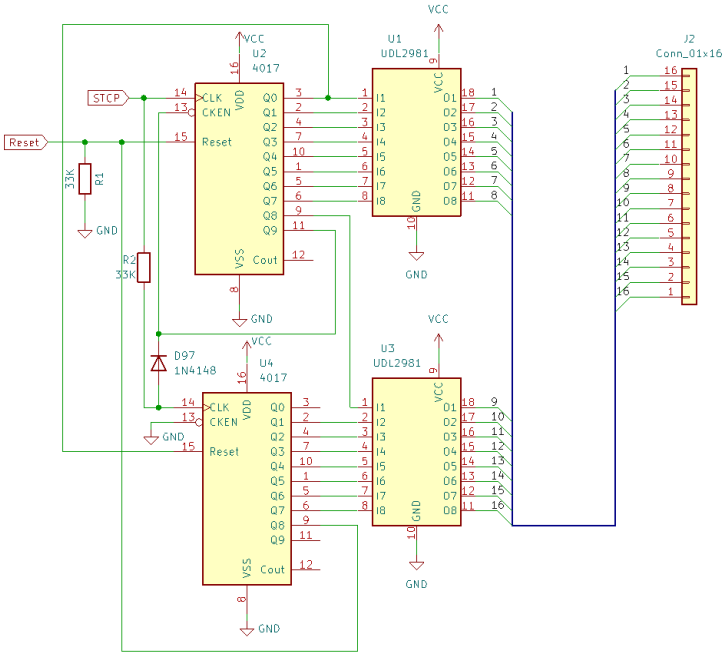
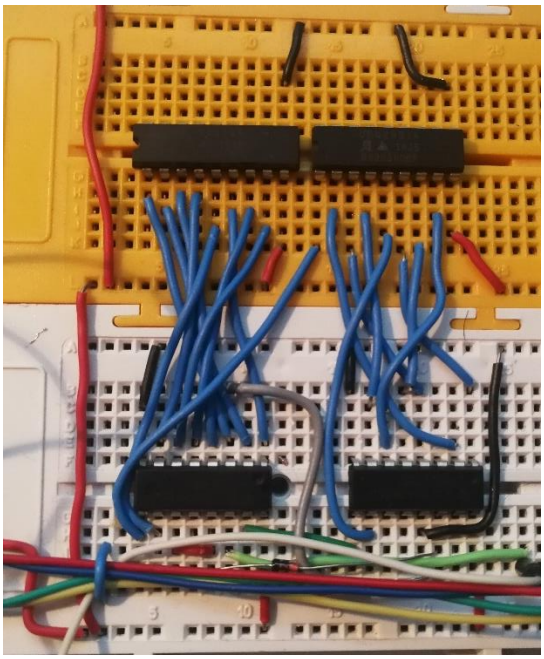
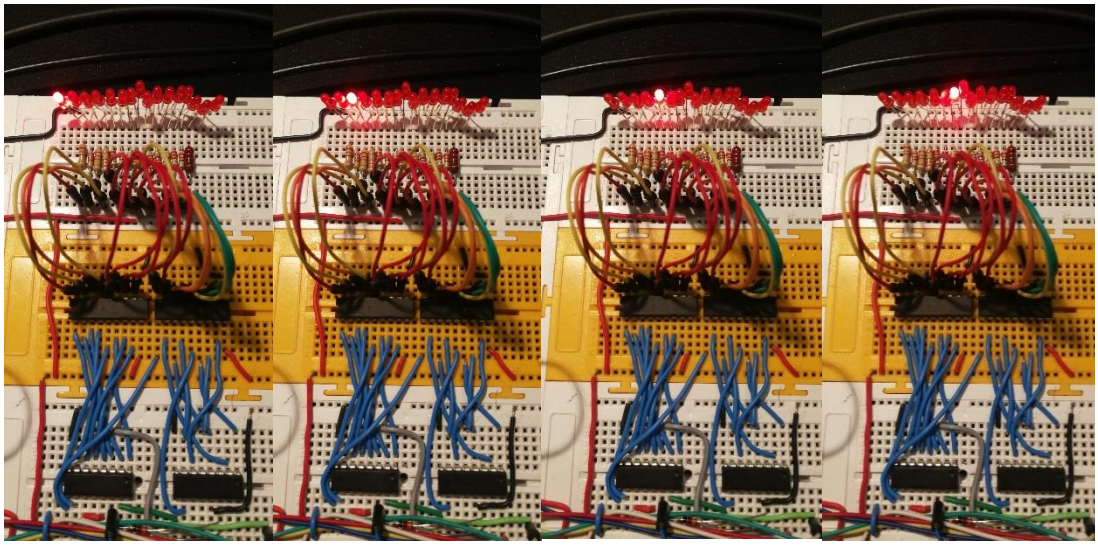


Figura 38. Esquema sincronizado vertical

Montamos el circuito en la protoboard.



Conectamos en la entrada de STCP un pulso de reloj y vemos que se van encendiendo los 16 LEDs consecutivamente, una vez llegado al último vuelve al comienzo. Si ponemos la entrada RST en estado alto la cuenta comienza de nuevo. Así realizaremos el barrido vertical del panel LED.

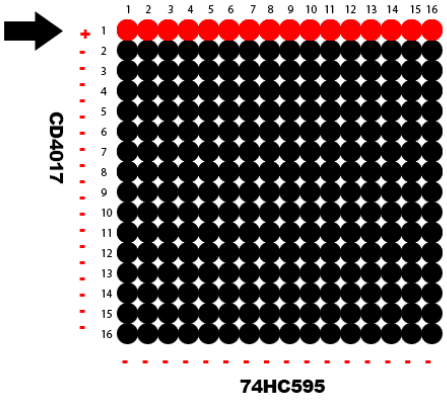


9.4.1.2 Barrido horizontal

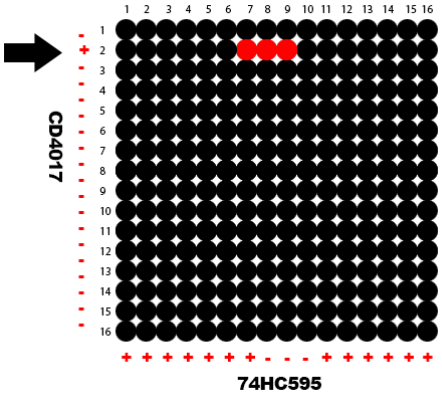
Para el control de encendido de las columnas utilizaremos dos integrados 74HC595 en cascada, donde los LED de la matriz estarán conectados a los integrados por su cátodo mediante una resistencia para limitar la corriente que circula por cada LED.

De tal manera que necesitaremos poner las salidas de los 74HC595 en estado bajo para que los LEDs se iluminen o en estado alto si queremos que estos se apaguen.

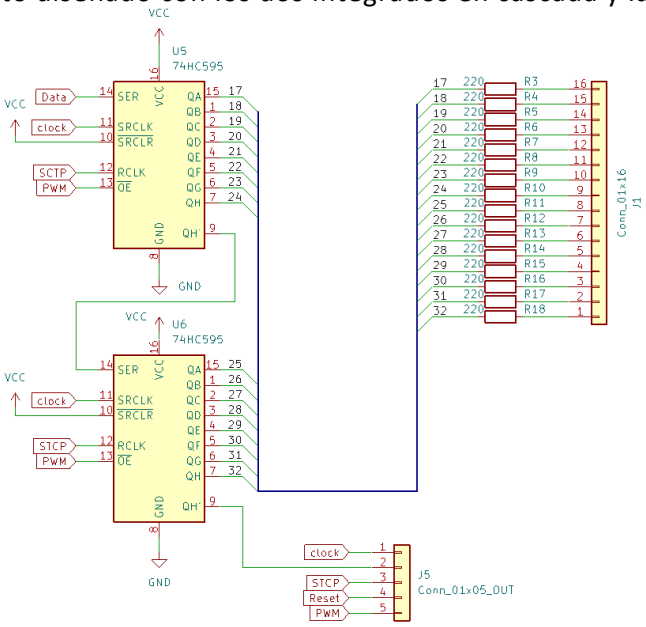
Aquí vemos como en la primera columna todas las salidas de los *shift registers* están en estado bajo, por lo que se enciende toda la fila.



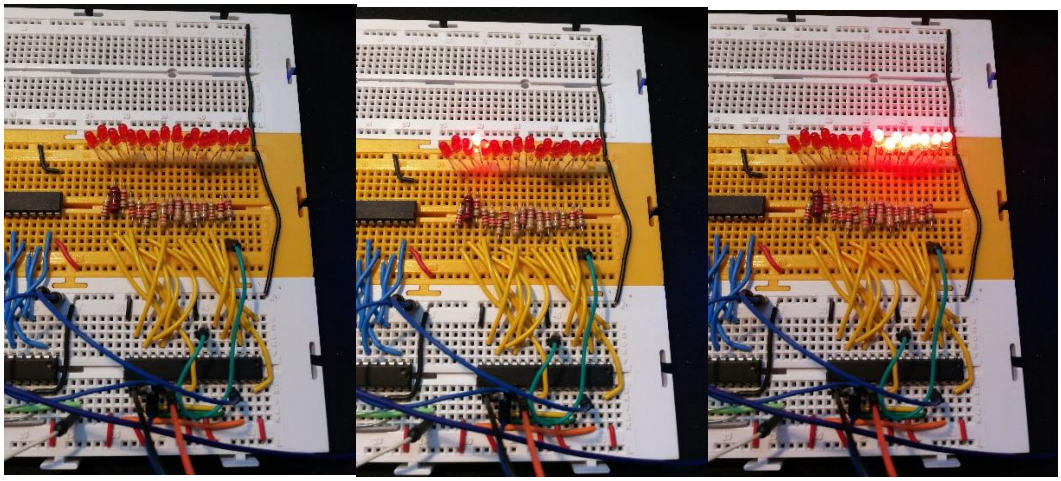
Por ejemplo, en otra fila ponemos en estado bajo las salidas 7,8,9 y el resto en estado alto, por eso solo se encienden los leds 7,8,9 de la segunda fila.



Este seria el circuito diseñado con los dos integrados en cascada y las resistencias.



Montamos el circuito y comprobamos, vemos que podemos controlar los LED, en la primera foto vemos todos los LEDs apagados, en la segunda solamente el LED 4 y la tercera foto los últimos ocho LEDs.



4. Sensor de luz ambiental

Los LEDs del panel se iluminarán más o menos según la cantidad de luz ambiental que le rodee. Para ello tenemos que dotar al sistema de un sensor de luz. Utilizaré un LDR que, mediante un circuito adaptador analógico envíe una señal al microcontrolador para que este regule la intensidad en los LEDs. Este dispositivo es muy versátil y la electrónica necesaria para hacerlo funcionar es muy simple.

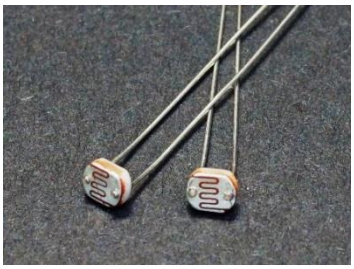


Figura 39. LDR

4.1. Descripción LDR

LDR proviene de Light Dependent Resistor o fotorresistor. Este dispositivo no es más que una resistencia que varía su valor dependiendo de la cantidad de luz que recibe. Normalmente su valor varia inversamente, cuanto mayor luz menor será su resistencia, llegando a valores desde varios MΩ hasta unos pocos de ohmios.

4.2. Funcionamiento

El funcionamiento de una LDR se basa en la utilización de un semiconductor como el sulfuro de cadmio que cuando le llega luz este absorbe fotones dando a los electrones la energía suficiente para saltar la banda de conducción, disminuyendo la resistencia entre sus terminales.

Al tratarse de un dispositivo que varía su resistencia podemos medir este valor de diferentes formas, por ejemplo, con el siguiente circuito y según como se conecte obtendremos un resultado u otro.

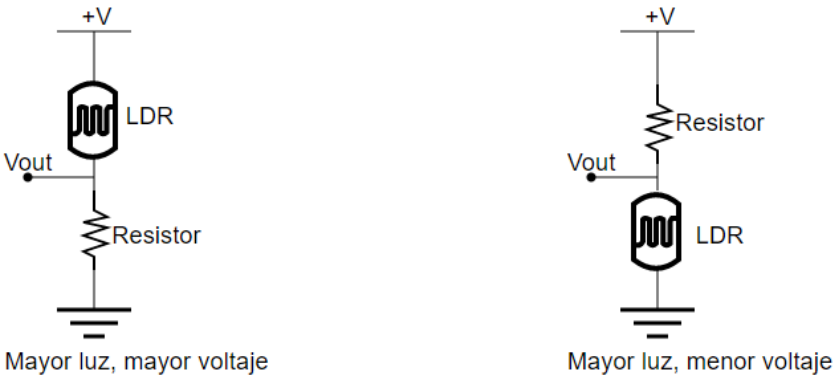


Figura 40. Circuitos básicos para LDR

1- A mayor luz, mayor voltaje, con esta configuración, si el LDR está conectado directamente al nodo positivo, cuando se excite la resistencia, esta descenderá su resistencia provocando que en el voltaje de salida obtengamos cada vez mayor tensión en Vout.

2- A mayor luz, menor voltaje, en esta otra configuración obtenemos lo contrario, cuanto mayor sea la luz incidente en la fotorresistencia un menor voltaje obtendremos en Vout ya que la célula está directamente conectada al nodo de GND

4.3. Tipos de LDR

En el mercado podemos encontrar de varios tipos, y podríamos clasificaciones según el material del que están construidos o según la respuesta a la luz:

- Según material del que están fabricadas:

  - Sulfuro de cadmio: los fotorresistores construidos con sulfuro de cadmio son muy sensibles a todo tipo de radiación luminosa dentro del espectro visible.

  - Sulfuro de plomo: Sin embargo, las que están construidas con este otro material son solamente sensibles a la luz infrarroja.

- Según su respuesta:

  -Respuesta lineal: A este tipo de LDR son más conocidos como fotodiodos y suelen polarizarse de modo inverso.

  -Respuesta no lineal, son la clase de LDRs más comunes y no tienen polaridad.

También encontramos sensores con diferentes valores, aunque los más estándar suelen ser cercanos al MegaOhm en oscuridad y expuestas a la luz del día valores de unos 50/100 Ohm. Podemos encontrar que algunos LDR aguantan hasta unos 600V y de 50mW hasta 1W de potencia.

4.4. Pros y contras de la utilización de un LDR en nuestro proyecto

- Muy sensible
- Fácil empleo.
- Bajo costo.
- No hay potencial de unión.
- Alta relación resistencia luz-oscuridad

- Respuesta espectral estrecha, aunque para nuestra aplicación es más que suficiente
- Tiene cierto retardo (en el orden de 1ms) y no responde rápido a los cambios de luz, poco uso en aplicaciones que tenga cambios de luz a gran velocidad.

4.5. Nuestro sensor de luz

Para el proyecto encontré diferentes LDR, todos ellos muy parecidos solamente variaba de su resistencia, opté por un LDR de 8~20 KΩ a 10Lux (GL5528), resulta ser bastante estándar y fácil de encontrar.

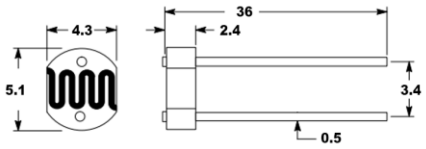


Figura 41. Dimensiones LDR

Resistencia a 10 Lux	8~20 KΩ
Resistencia en oscuridad 0 Lux	1 MΩ (min)
Valor Gamma 100-10 Lux	0.7
Potencia	100mW
Máximo voltaje	150 V
Respuesta espectral pico	540nm
Temperatura de trabajo	-30 ~+70 ºC

Tabla 5. Características LDR GL5528

Mediante una aplicación para móvil que simula un luxómetro podemos experimentamos con el LDR y simular las condiciones de luz en las típicas situaciones donde se podría encontrar nuestro panel. Estos son los valores estándar de lux en diferentes lugares.

1 lux	Luna llena
100 lux	Pasillo en una zona de paso
300 lux	Sala de reuniones
500 lux	Oficina bien iluminada
600 lux	Salida o puesta de sol
1000 lux	Iluminación estándar de un estudio de televisión
32.000 lux	Luz solar en un día medio (mín.)
100.000 lux	Luz solar en un día medio (máx.)

Vamos probando con diferentes intensidades de luz y obtenemos diferentes valores de resistencia de nuestro LDR. Obtenemos la siguiente tabla en escala logarítmica que relaciona la intensidad de luz recibida con la resistencia del LDR.

LUX	LDR Ω
25000	100 Ω
5000	350 Ω
1000	850 Ω
500	1,3 KΩ
400	1,78 KΩ
100	4 KΩ
50	6,5 KΩ
10	21KΩ

Tabla 6. Valores LDR (Resistencia e intensidad lumínica)

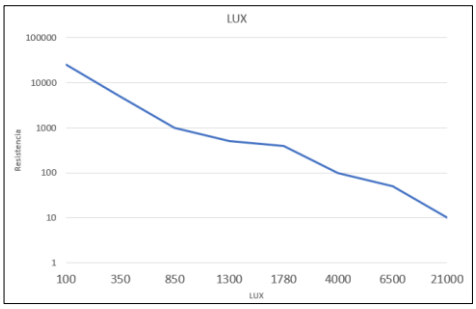


Figura 42. Grafica respuesta LDR

Para nuestro proyecto, montamos el siguiente circuito y donde el voltaje de salida vendrá dado por un divisor de corriente que viene determinado por la siguiente expresión:

$$V_{OUT} = V_{IN} \frac{LDR}{R + LDR} \tag{2}$$

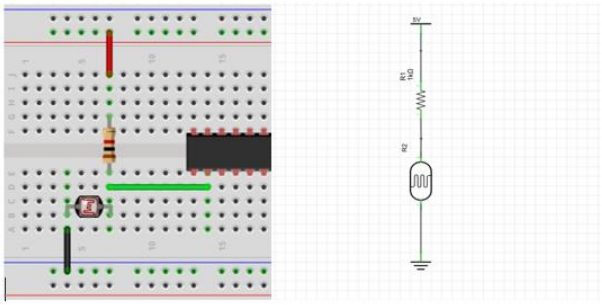


Figura 43. Circuito básico LDR

Este circuito es bastante simple y aunque es efectivo tiene sus problemas, como por ejemplo que no contempla las impedancias de entrada del microcontrolador. Por ese motivo podemos incorporar un amplificador operacional con una configuración de seguidor de tensión o buffer, para que ofrezca una baja impedancia de salida.

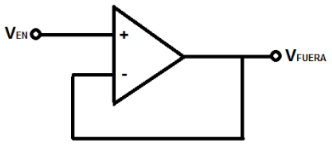


Figura 44. Seguidor de tensión

Utilizaremos un LM741, que se trata de un amplificador operacional de propósito general que podemos alimentar mediante una tensión de 5V. Entonces el circuito quedaría de la siguiente manera.

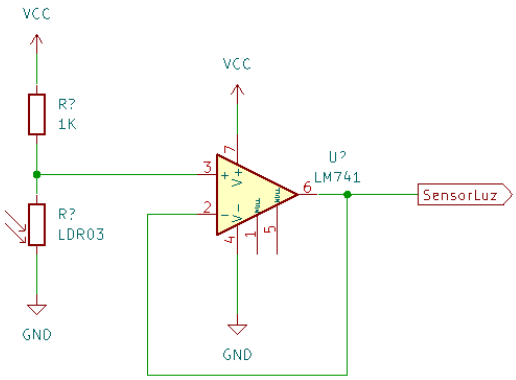
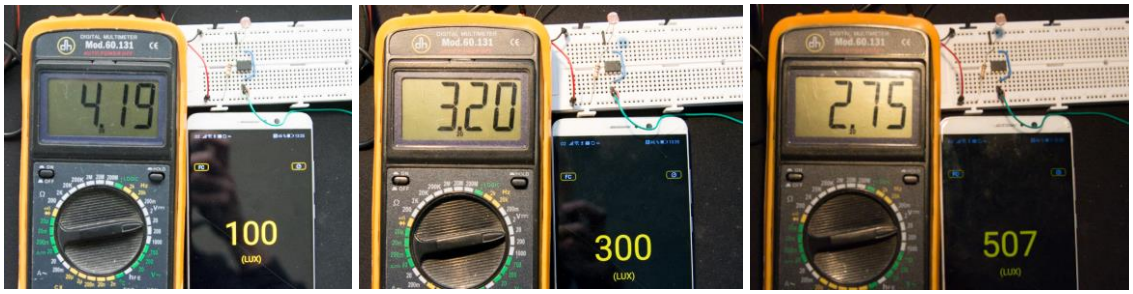


Figura 45. Circuito LDR

Montamos el circuito y comprobamos la salida del seguidor de tensión:



Obtenemos los siguientes resultados que corresponden con los cálculos del divisor de tensión

LUX	LDR Ω	Voltaje
25000	100 Ω	0,419 V
5000	350 Ω	1,5 V
1000	850 Ω	2,25 V
500	1,3 KΩ	2,7 V
400	1,78 KΩ	3,2 V
100	4 KΩ	4,19 V
50	6,5 KΩ	4,4
10	21KΩ	4,8

5. Protocolo MODBUS

5.1. Protocolo

Se trata de un protocolo serie de comunicaciones abierto. Fue desarrollado por Modicon (actualmente Schneider Electric) a finales de los años 70 para uso industrial en el control de sistemas de automatización.

Este protocolo está ampliamente aceptado debido a su fácil uso, su confiabilidad y sobre todo que es abierto. Lo que significa que su uso está libre de derechos no hay que pagar por su instalación y uso.

Normalmente suele utilizarse para transmitir señales desde dispositivos de instrumentación y control hacia un controlador principal, se basa en una arquitectura esclavo/maestro, donde el dispositivo que solicita información se llama maestro y el que suministra la información es el esclavo. Al comienzo se creó para ser utilizado sobre una capa serial, aunque más adelante se crearon implementaciones sobre TCP/IP y UDP.

Como MODBUS no especifica la capa física puede ser utilizado en diferentes estándares, como RS-232, RS-485 o Ethernet TCP/IP.

Para este proyecto implementaremos una entrada serie RS-232 en nuestro panel LED para transmitir datos hacia este desde un ordenador.

5.2. PUERTO RS-232

RS-232 recommended Standard 232 o estándar recomendado 232, designa una norma para el intercambio de datos entre un equipo terminal y un equipo de comunicación de datos. Este estándar define especificaciones mecánicas, eléctricas, funcionales y de procedimientos típicos de un protocolo orientado al enlace físico punto a punto empleando un intercambio en modo serie de datos binarios.

Figura 46. Cable RS232

En este tipo de comunicación se distinguen dos tipos de dispositivos un equipo terminal de datos DTE (Data Terminal Equipment) y un equipo de comunicación de datos DCE (Data Communications Equipment).

Las especificaciones eléctricas del puerto serial se encuentran en el estándar RS232 de EIA (Electronics Industry Association) donde define los niveles de voltaje y el tipo de señal a transmitir:

- Se utiliza codificación NZR-L, es decir el cero lógico se codifica con un pulso positivo y uno lógico se codifica con un pulso negativo y no se retorna 0 con bits iguales consecutivos, con unos rangos de tensión permitidos de entre 3 y 15 V y de entre -3 y -15 V. La tensión nominal es de 12 V. Y la tensión máxima de 25 V.

- De los 25 pines sólo, cuatro son utilizados para datos. El resto son de control, temporización, tierra y pruebas. La especificación eléctrica para estos circuitos es igual que para los datos, considerando el estado ON equivalente al cero lógico y OFF al uno lógico.
- La tasa de bits máxima que se recomienda en la norma para la distancia máxima de 15 metros es de 20 kbps. Esta velocidad se puede aumentar si se disminuye la distancia de conexión. Ahora bien, existen aplicaciones que se salen de las especificaciones del estándar que llegan a velocidades de hasta 116 kbps.

5.2.1. El conector

Existen 2 implementaciones funcionales de la EIA-232 en función del conector y número de conductores utilizado. Por ello el conector normalmente empleado en los interfaces RS-232 es un conector DB-25, aunque es normal encontrar la versión de 9 pines DB-9 de forma más difundida, este es el conector que usaremos en nuestro proyecto ahorrando espacio ya que solo utilizaremos 4 pines. El estándar define que el conector hembra se situará en los DCE y el macho en el DTE. Aunque es fácil encontrar excepciones.

5.2.2. Implementación en DB-9

Se ha desarrollado una versión más sencilla de la normal EIA-232 utilizando un conector DB-9, con 9 pines. Al igual que para el conector DB-25, el conector para el DTE (ordenador) debe ser macho.

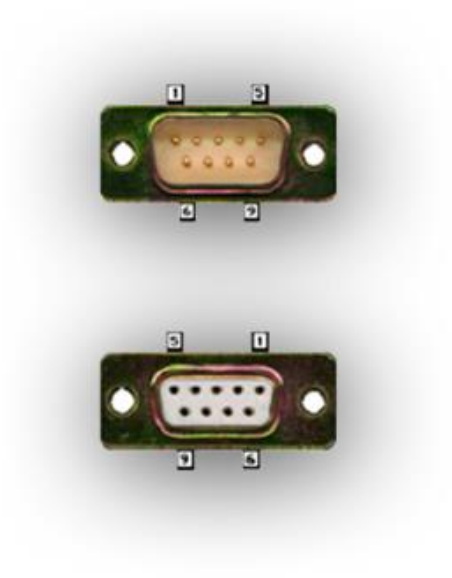


Figura 47. Conector DB9 macho y hembra

No. pin	Nombre	Función	Dirección
1	DCD	Data Carrier Detect -Detección de portadora	IN
2	RX	Recepción de datos	IN
3	TX	Transmisión de datos	OUT
4	DTR	Data Terminal Ready -DTE listo	OUT
5	SGND	GND	-
6	DSR	Data Set Ready -DCE listo	IN
7	RTS	Request to send -Petición para enviar	OUT
8	CTS	Clear to send -Listo para enviar	IN
9	RI	Ring Indicator -Indicador llamada entrante	IN

Tabla 7. Pines conector DB9 RS232

5.2.3. Cable de conexión

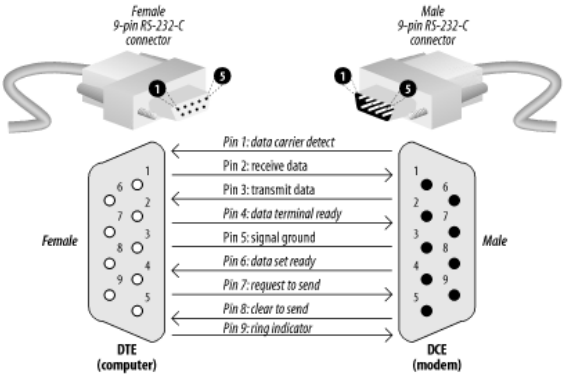


Figura 48. Conexión RS232

El RS232 usa niveles lógicos incompatibles con los TTL, con lo que requerimos un convertidor de niveles lógicos para conectar nuestro sistema con nuestro microcontrolador a otro que maneje RS232 como puede ser un ordenador.

Para este fin hemos encontrado varios tipos de integrados como pueden ser ST232, ICL232, MAX202 o MAX232.

Integrado	ST232	ICL232	MAX202	MAX232
Fabricante	ST	Renesas	Texas Instrument	
Numero drivers	Dual drive	Dual drive	Dual drive	Dual drive
Tipo	SOP/TSSOP 8 pines	SOIC W	PDIP 16 pines	PDIP 16 pines
Precio	1,2 € (RS-COMPONENTS)	3,981 €(RS-COMPONENTS)	4,5 € (RS-COMPONENTS)	0,42 € (RS-COMPONENTS)

Tabla 8. Comparativa drivers RS-232

Tras analizar precio y accesibilidad, nos quedamos con el MAX232 que se trata de un driver muy asequible bastante popular por lo que podemos encontrarlo con facilidad, también el MAX202 que se trata del mismo driver, pero con unos condensadores necesarios ya integrados en él, pero el precio es bastante superior y nos cuesta menos añadir esos condensadores a nuestro circuito.

5.2.4. MAX232

El MAX232 es un circuito integrado de Maxim que con tan solo unos condensadores convierte las señales de un puerto serie RS-232 a señales compatibles con los niveles TTL de circuitos lógicos. Este integrado sirve como interfaz de transmisión y recepción para las señales RX, TX, CTS y RTS. Tiene salidas para manejar niveles de voltaje del RS-232 (aprox. ± 7.5 V) que las produce a partir de un voltaje de alimentación de + 5 V utilizando multiplicadores de voltaje internos con la adición de condensadores externos. Esto es de mucha utilidad para la implementación de puertos serie RS-232 en dispositivos que tengan una alimentación simple de + 5 V.

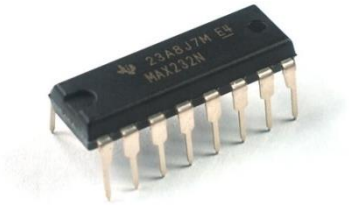


Figura 49. Integrado MAX232

Aquí vemos información obtenida del datasheet, el patillaje del integrado y el circuito necesario para su implementación.

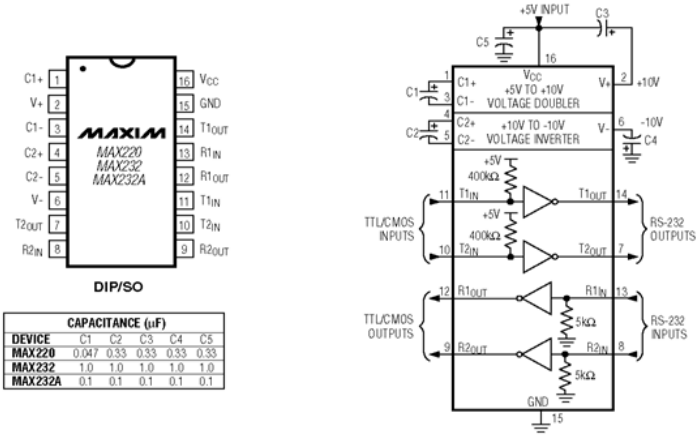


Figura 50. MX232

El funcionamiento es bastante sencillo, las entradas de recepción de RS-232 (las cuales pueden llegar a ± 25 V), se convierten al nivel estándar de 5 V de la lógica TTL. Estos receptores tienen un umbral típico de 1.3 V, y una histéresis de 0.5 V.

Aquí vemos el esquema del circuito que utilizaremos para el montaje del integrado. Los pines de TX y RX que van directamente conectados a nuestro microcontrolador

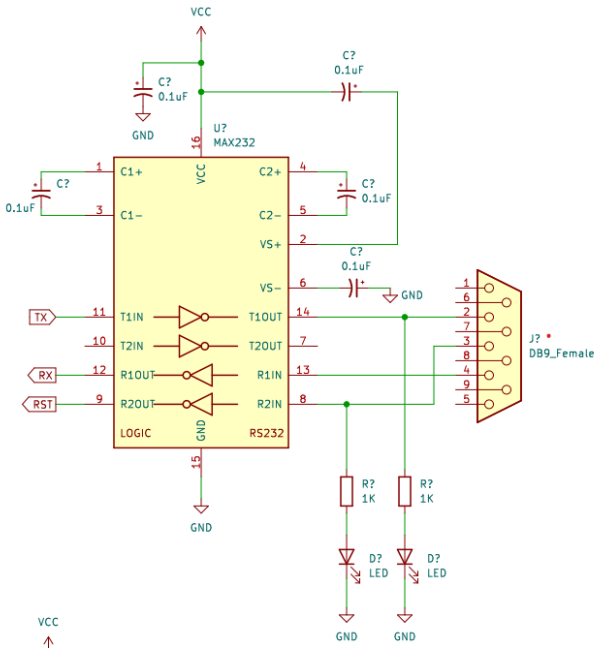


Figura 51. Circuito MAX232

Probaremos el circuito, así quedaría el circuito montado en la protoboard.

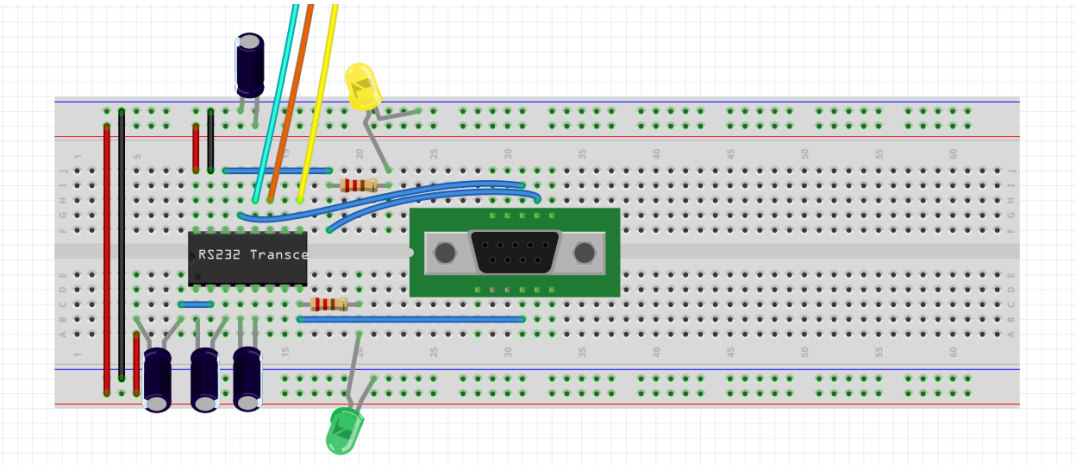


Figura 52. Circuito protoboard MAX232

6. Microcontrolador

Un microcontrolador es un circuito integrado programable, con la capacidad de ejecutar tareas que previamente hayan sido grabadas en su memoria. En el mercado existen distintas marcas que ofrecen microcontroladores con diferentes características para diferentes propósitos como pueden ser Microchip Technology Inc, Texas Instrument o Atmel Corporation entre otras.

Para nuestro proyecto nos decantaremos por un microcontrolador de la marca Microchip, ya que es un fabricante bastante extendido con cantidad de información disponible. Dentro de este fabricante existen diferentes familias de microchips y elegiremos uno depende de nuestras necesidades. Nuestro microcontrolador debe ser capaz de recibir información a través del puerto RS-232 por lo que debe tener entrada USART, también debe ser capaz de disponer de un conversor A/D para conectar el sensor de luz y una salida PWM para controlar la luminosidad de los LEDs y, debe ser capaz de manejar nuestro panel LED que consume un total de 5 pines digitales y por último poseer memoria suficiente para toda nuestra programación.

Por lo que haciendo un resumen de especificaciones mínimas

USART	1
A/D	1
PWM	1
LED	5
Total de pines	9

De todas las familias que ofrece este fabricante nos iremos a un microcontrolador de la gama de 8-bit MCUs, que es más que suficiente, para ello he optado por el PIC16F1615 que cumple con los requisitos mínimos anteriormente descritos.

Este son los pines del PIC16F1615 extraído de su datasheet.

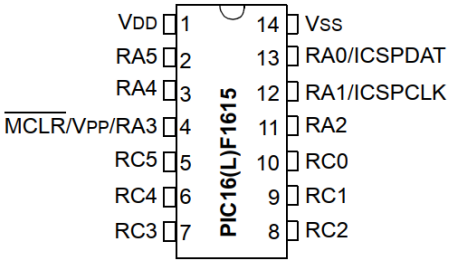


Figura 53. Microcontrolador PIC16F1615

Y este sería el circuito mínimo necesario a implementar para el funcionamiento de este.

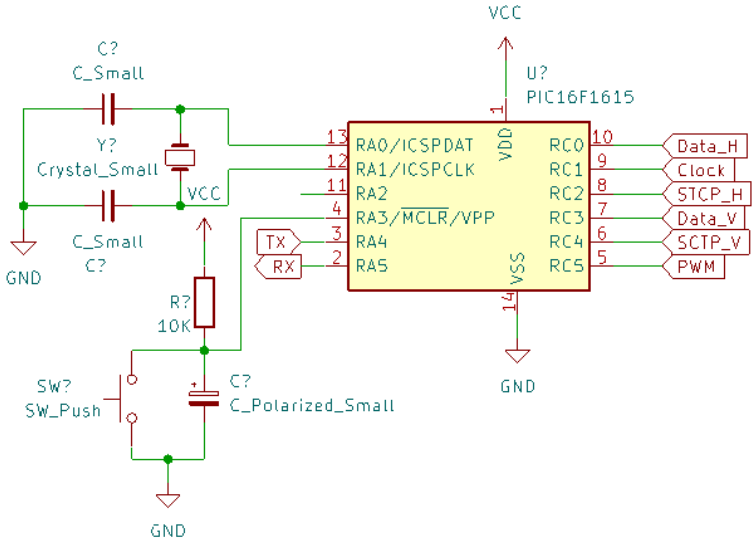


Figura 54. Circuito microcontrolador

7. Fuente alimentación

Para el buen dimensionamiento y diseño de la fuente de alimentación debemos tener en cuenta tanto la tensión de trabajo de los componentes como el máximo consumo de estos en un momento dado. Para ello consultamos las especificaciones eléctricas en los correspondientes datasheets.

Componente	**Voltaje**	**Corriente**
Microcontrolador PIC16F1615	+5V	250mA
74HC595 x4	+5V	80mA
UDN2984 x2	+5V	20mA
LM741	+5V	2,8mA
MAX232	+5V	10mA
Panel LED (x16 LEDs)	+5V	320mA
TOTAL	+5V	683 mA

Tabla 9. Consumo eléctrico componentes

Vemos que todos los componentes se les puede alimentar con el mismo voltaje de 5V, esto hará que nuestra fuente de alimentación sea bastante sencilla ya que no debemos obtener diferentes voltajes y el consumo total de sistema para un solo panel será 683mA.

Teniendo en cuenta que deberemos trabajar para dar una corriente de 683 mA y una tensión de salida de 5 V podemos optar por diferentes reguladores de tensión. En principio podríamos utilizar el LM7805, que se trata de un regulador de voltaje capaz de entregar 5 V y hasta 1 A. Este regulador posee protección por sobrecalentamiento y cortocircuito, lo que lo hace bastante robusto y seguro, pero vemos que necesitamos reducir la tensión desde 24 V hasta 5 V lo que nos obliga a disipar mucha potencia $0{,}683\text{A} \cdot 19\text{V} = 12{,}977\ \text{W}$, cercana a su máxima, aunque funcionaria tendríamos que montar un disipador. Si el voltaje de entrada fuera menor sería la mejor opción, pero también existen otras alternativas como reguladores de tensión conmutados, como el LM2596, LM2672N, LM2575T, LMZ14201 o L5970D. Todos ellos ofrecen la capacidad de reducir la tensión desde 24 V hasta 5 V con una eficiencia mayor del 90%, ofreciendo hasta 1 A de salida y con apenas componentes externos.

De todos ellos optare por el LM2596 en la versión de 5V de salida fija. Este dispositivo permite un voltaje de entrada hasta 40 V y ofrece una salida de hasta 3 A de corriente. Elegiré el encapsulado TO-220 que es el que mejor se adapta para este proyecto. Mirando en el datasheet vemos que para montar el circuito simplemente necesitamos cuatro componentes más.

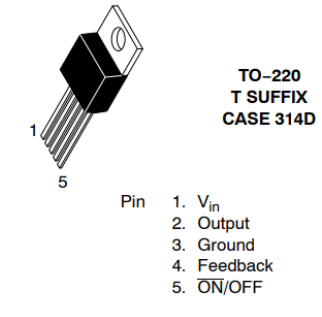


Figura 55. LM2596

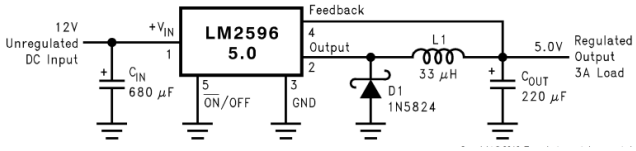


Figura 56. Circuito LM2596 5.0

Diseñanos el circuto en Kicad

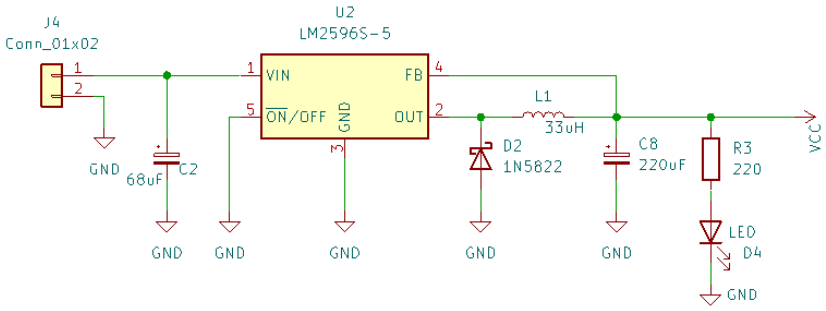
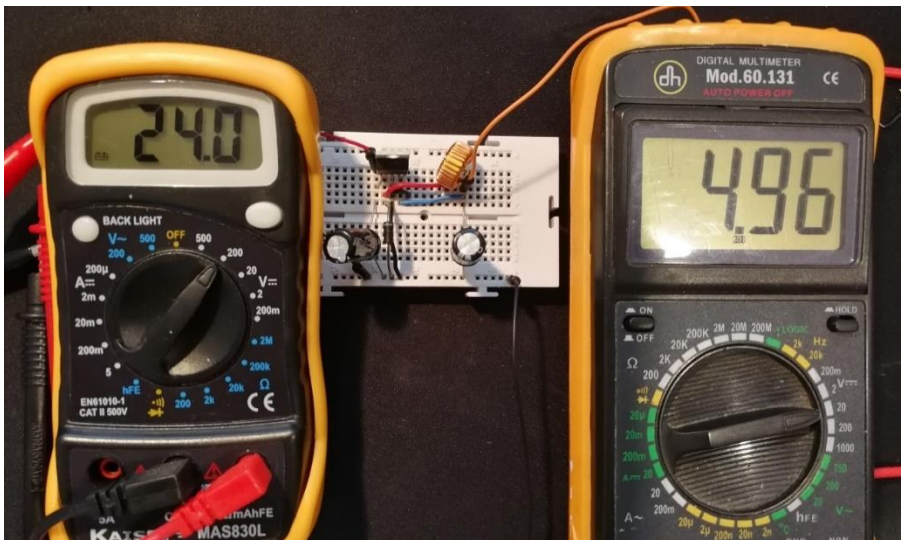


Figura 57. Diseño circuito fuente de alimentación

Montamos el circuito y vemos que es capaz de reducir la tensión desde 24 V a 5 V.



Vemos en su datasheet que es capaz de soportar hasta 3 A, por lo que podríamos dar corriente a más paneles. El primer panel que contiene el microcontrolador consumiría en total 683mA y el resto de paneles que solamente llevan el driver y los LEDs consumirían 420 mA por lo que podríamos soportar hasta 5 paneles más, podríamos crear un panel de 16x96 LEDs con la misma fuente de alimentación.

$$683mA + (420mA \cdot 5) = 2{,}783\ A \qquad (3)$$

8. Programación

8.1. LDR y PWM

Para programar el microcontrolador nos ayudaremos del compilador CCS C Compiler Version 5.049 y realizaremos algunas simulaciones en Proteus 8.13.

Lo primero que hacemos es declarar la configuración del microcontrolador, incluimos la librería especifica de nuestro microcontrolador 16F1615.h. También debemos declarar los fuses que necesitemos, con HS declaramos que utilizaremos un cristal de cuarzo de alta velocidad, exactamente usaremos uno de 16 MHz. También desactivaremos el watchdog. Declaramos la velocidad de trabajo del microcontrolador. Los pines del puerto C como salida. Para poder realizar un seguimiento de la programación habilitaremos el puerto USART a 9600 baudios y según el datasheet la podemos configurar la salida en C5.

```
#include <16F1615.h>
#device ADC=10

#FUSES HS
#FUSES PUT                      //Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOWDT

#use delay(crystal=16000000)    //Configuramos el microcontrolador a una velocidad de 20MHz
#use FIXED_IO( C_outputs=PIN_C5, PIN_C4,PIN_C3,PIN_C2,PIN_C1,PIN_C0 )
// Seleccinamos los pines del puerto C como salida
#pin_select U1RX=PIN_A4    //Seleccionamos el pin para la recepción del puerto UART
#pin_select U1TX=PIN_C5    //Seleccionamos el pin para el envío del puerto UART
#use rs232(UART1,baud = 9600)  //Configuramos el puerto UART
#pin_select CCP1OUT=PIN_C3         // Seleccionamos el PIN C3 como salida del CCP1
```

Comenzaremos programando la entrada del conversor ADC para la entrada del LDR, lo tenemos declarado en la entrada A2 como analógica. Crearemos las variables ldrIn y volts, que utilizaremos para guardar la información de la entrada ADC y en esta segunda variable la usaremos para convertir a voltios y que coincide con la señal real. Y seleccionaremos el canal 2 para poder adquirir los datos de este pin.

Por otro lado, tenemos que configurar el módulo CCP (Capture/Compare/PWM) del microcontrolador para usar PWM, según el datasheet la salida del CCP1 debe estar en el pin C4 que declararemos como PWM_OUT. Para conocer el periodo de nuestro pulso utilizaremos el Timer2 del microcontrolador. Recordamos que el objetivo del pulso PWM es regular la intensidad de los LEDs, por lo tanto, con frecuencias superiores a 50Hz bastaría ya que el ojo humano no apreciaría el parpadeo, aunque se ha demostrado que en señales muy luminosas habría que aumentar la frecuencia.

En el datasheet en el apartado PWM Period vemos que el periodo del pulso viene definido por la siguiente ecuación:

$$Periodo\ PWM = (PR2 + 1) \cdot 4 \cdot \frac{1}{Fosc} \cdot TMR2PrescalerValue \quad (4)$$

Si lo pasamos a frecuencia tenemos:

$$Frecuencia\ PWM = \frac{Fosc}{(PR2+1)\cdot 4\cdot TMR2PrescalerValue} = \frac{16MHz}{(249+1)\cdot 4\cdot 16} \quad (5)$$

Y vemos que, con los siguientes datos, obtenemos 1kHz de pulso, suficiente para nuestro propósito.

$$\frac{16MHz}{(249+1)\cdot 4\cdot 16} = 1kHz \quad (6)$$

Ya en nuestro bucle adquirimos los datos del pin A2 y lo almacenamos en la variable *ldrIn* que más adelante utilizaremos para ajustar el duty clycle de nuestro pulso PWM. Con el valor del ldrIn lo multiplicamos por el resultado de dividir 5 voltios entre 1023 valores de resolución que nos da el DAC a 10 bits, y obtenemos el voltaje de entrada en nuestro microcontrolador. Imprimimos estos valores por la salida serial para ver la simulación.

```
#include <TFG.h>

#include <string.h>
#include <stdio.h>


#define LDR_IN PIN_A2          // LDR in analogico
#define RSTX PIN_A4            // TX
#define RSRX PIN_C5            // RX
#define RESET_V PIN_C4         // Reset Vertical
#define PWM_OUT PIN_C3         // PWM out
#define STCP PIN_C2            // Latch (Storage Register)
#define DS PIN_C1              // Serial Data
#define SHCP PIN_C0            // Clock (Shift Register)




void main()
{
      int16 ldrIn;                // variable para leer la entrada del LDR
      float volts;                // variable para convertir a voltios y comprobar

      output_low(PWM_OUT);                //Ponemos el pin de salida del PWM en nivel bajo
      setup_ccp1(CCP_PWM);                //Configuramos el módulo del micro CCP1
      setup_timer_2(T2_DIV_BY_16,249,1); //200 us overflow, 8,0 ms interrupt
      setup_adc_ports(sAN2);              //Configuramos la entrada 2 como entrada analogica
      setup_adc(ADC_CLOCK_INTERNAL);      //ajustamos el adc
      set_adc_channel(2);         //Seleccionamos el canal 2 para poder adquirir datos de él
      delay_ms(100);


   while(TRUE)
   {

      ldrIn = read_adc();         //leemos la entrada analogica LDR
      volts = (5.0/1023.0)*ldrIn; //div. 5v entre 1023 (10bits) y multipli x el valor y obtener V
      printf("Lectura ADC = %ld | Voltaje = %f \n\r",ldrIn, volts); //imprimimos el valor
      delay_ms(100);
      set_pwm1_duty(ldrIn);               //ajustamos la salida PWM con el dato leido del LDR in
   }
 }
```

Figura 58. Código lector voltaje en el LDR

Podemos observar en la simulación que al obtener un valor cercano a 0 voltios (situación de oscuridad para nuestro circuito) el duty cycle de nuestro pulso se ajusta hacia el mínimo.

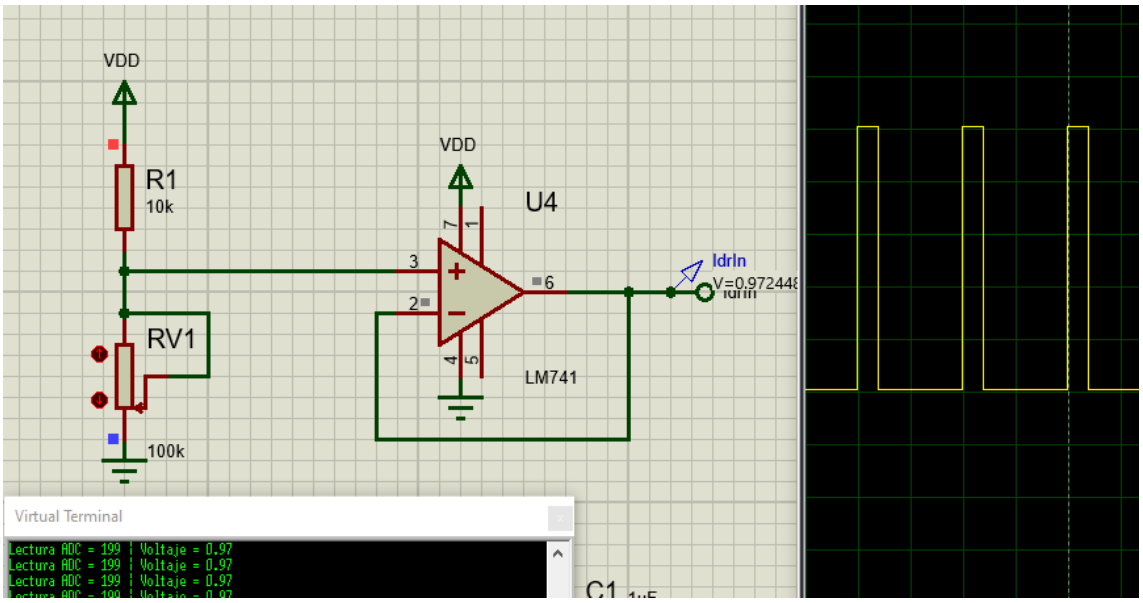


Figura 59. Simulación circuito LDR y salida PWM al mínimo

Si el valor de entrada se acerca al máximo que nos da unos 4 voltios, situación de máxima luz en nuestro circuito, el duty cycle está cerca del máximo.

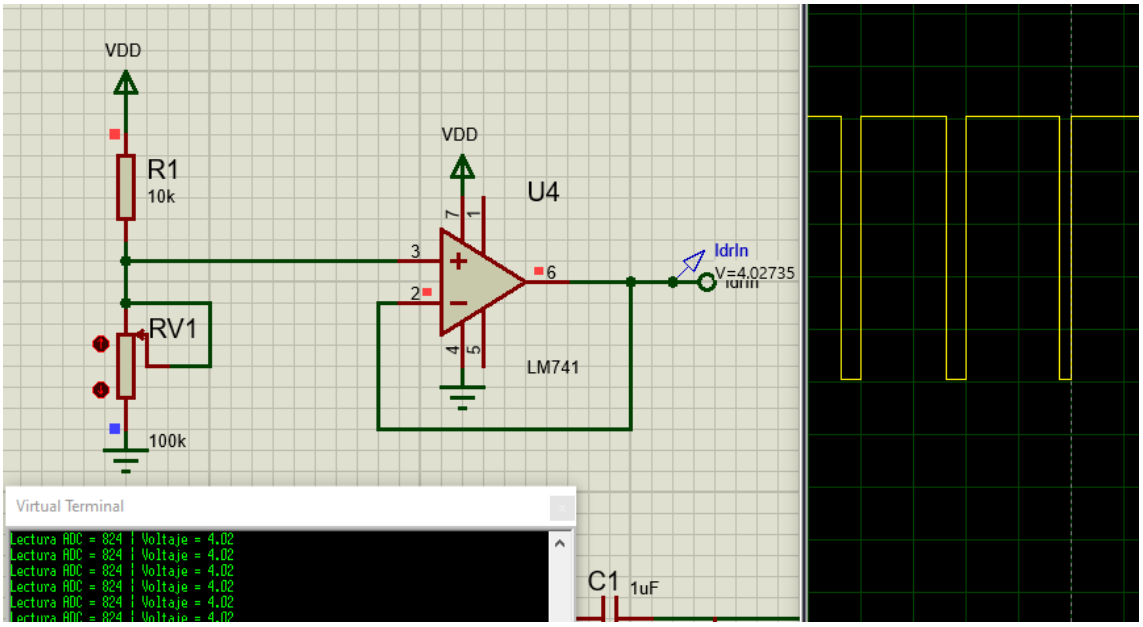


Figura 60. Simulación circuito LDR y salida PWM casi al máximo

Para ajustar un máximo incluiremos unas correcciones en el código

```
duty=(ldrIn/LDR_MAX_VALUE)*1023;  //Ajustamos el valor a un máximo
if (duty > 1023) duty = 1023;     //en el caso de obtener algún valor mayor ajustamos el máximo
tope del duty
```

Figura 61. Código de ajuste de valor máximo en el LDR

Dividimos el valor ldrIn entre un máximo preestablecido (`LDR_MAX_VALUE=800`) obteniendo un factor de conversión y multiplicamos por 1023. Si el factor de multiplicación es mayor de 1, obtendremos valores mayores de 1023, por eso lo limitaremos a 1023. Así obtendremos un ciclo de 100% en valores por encima de 800 o 4 voltios en la entrada analógica.

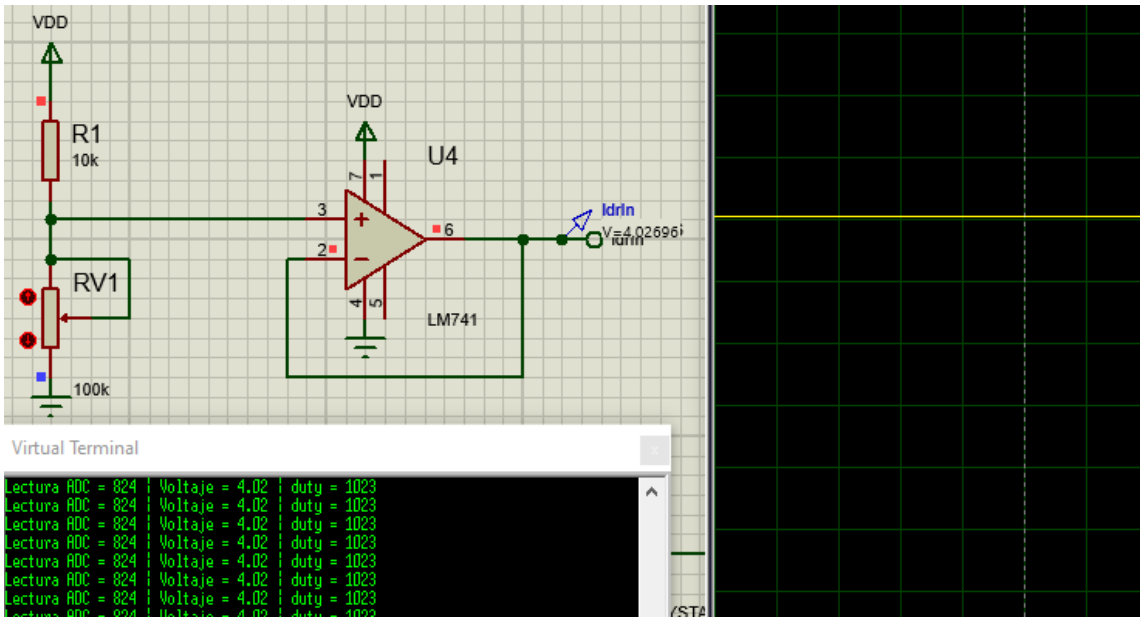


Figura 62. Simulación circuito LDR y salida PWM al máximo

Continuaremos con la programación del modbus

8.2. Modbus

Ahora toca implementar el protocolo modbus, para ello hacemos uso de la librería modbus.c

Para ello definimos ciertas variables como la dirección de nuestro dispositivo, el tipo de protocolo, en nuestro caso RTU, la velocidad de transmisión y los pines a usar. Y declaramos un vector donde almacenaremos los datos.

```
#define MODBUS_PROTOCOL                MODBUS_PROTOCOL_SERIAL
#define MODBUS_TYPE                    MODBUS_TYPE_SLAVE
#define MODBUS_SERIAL_INT_SOURCE       MODBUS_INT_RDA
#define MODBUS_SERIAL_TYPE             MODBUS_RTU


#define MODBUS_SERIAL_RX_BUFFER_SIZE  64
#define MODBUS_ADDRESS                0x01     //define la direccion del esclavo
#define MODBUS_TYPE MODBUS_TYPE_SLAVE
#define MODBUS_SERIAL_TYPE MODBUS_RTU         //use MODBUS_ASCII for ASCII mode
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64
#define MODBUS_SERIAL_BAUD             9600
#define MODBUS_SERIAL_RX_PIN           PIN_C5  // Data receive pin
#define MODBUS_SERIAL_TX_PIN           PIN_A5  // Data transmit pin

#use rs232(baud=MODBUS_SERIAL_BAUD,UART1,bits=8,stop=2,parity=N,stream=MODBUS_SERIAL,
errors)

int16 hold_regs[] = {0x8800,0x7700,0x6600,0x5500,0x4400,0x3300,0x2200,0x1100};
```

Figura 63. Código de variables para la librería modbus

Mediante el siguiente código esperamos la entrada de algún dato y seguidamente comprobamos si se trata de una trama dirigida a nosotros, en ese caso vemos de qué función se trata.

```
while(!modbus_kbhit());

      delay_us(50);

      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)  //
comprobamos la direccion o si se trata de un broadcas
      {
         switch(modbus_rx.func)
         {
            case FUNC_READ_COILS:
               // TO DO
               break;
            case FUNC_READ_DISCRETE_INPUT:
               // TO DO
              break;
            case FUNC_READ_HOLDING_REGISTERS:
               // TO DO
               break;
            case FUNC_READ_INPUT_REGISTERS:
               // TO DO
               break;
            case FUNC_WRITE_SINGLE_COIL:
               // TO DO
               break;
            case FUNC_WRITE_SINGLE_REGISTER:
              // TO DO
               break;
            case FUNC_WRITE_MULTIPLE_COILS:
               // TO DO
               break;
            case FUNC_WRITE_MULTIPLE_REGISTERS:
                              // TODO
               break;
            default:    // No soporta la funcion y reporta una excepcion
               modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
         }
      }
```

Figura 64. Código que lee la entrada de datos

De todas las funciones solamente nos quedaremos con la más útil en nuestro caso, que se trata de la función de escribir en múltiples registros. Quedando el código de la siguiente manera

```
while(!modbus_kbhit());

      delay_us(50);

      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)  //
comprobamos la direccion o si se trata de un broadcas
      {
         switch(modbus_rx.func)
         {
            case FUNC_WRITE_MULTIPLE_REGISTERS:
       //comprobamos si existe algún error en los datos
               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;

                   for(i=0,j=5; i < modbus_rx.data[4]/2; ++i,j+=2)     //almacenamos los
datos
                      hold_regs[i] = make16(modbus_rx.data[j],modbus_rx.data[j+1]);

                   modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,     //respondemos
a la llamada como correcto
                                  make16(modbus_rx.data[0],modbus_rx.data[1]),
                                  make16(modbus_rx.data[2],modbus_rx.data[3]));

                   event_count++;

                // Y aquí vendría nuestro código para representar en pantalla los datos


                   break;
             default:     // No soporta la funcion y reporta una excepcion
                   modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
          }
       }
```

Figura 65. Código para leer la entrada de datos y responde

8.3. Driver LEDs

8.3.1. Vertical

Para el sincronismo vertical solamente creamos dos funciones, una que hace reset (*reset();*) en los contadores 4017 que marcará el comienzo del barrido vertical y latch (*latch();*) que desencadena el cambio de línea, este coincidirá con el latch de los 74HC595

```
void latch(){
   output_high(SHCP);
   delay_us(50);
   output_low(SHCP);
   delay_us(50);
   }

void resetv(){
   output_high(RESET_V);
   delay_us(50);
   output_low(RESET_V);
   delay_us(50);
   }
```

Figura 66. Funciones latch() y reset()

8.3.2. Horizontal

Para el sincronismo horizontal creamos también dos funciones, una para el pulso de reloj (*clock();*) que marcará el ritmo de entrada de datos en el registro de desplazamiento y otro que utilizaremos para mostrar los datos por las salidas (*latch();*) esta función será la misma que marque el cambio de linea

```
void clock(){
   output_high(STCP);
   delay_us(50);
   output_low(STCP);
   delay_us(50);
   }

void latch(){
   output_high(SHCP);
   delay_us(50);
   output_low(SHCP);
   delay_us(50);
   }
```

Figura 67. Funciones clock() y latch()

También creamos una función para mandar los datos, a esta función le pasamos un dato de 16 bits y se encarga de darle salida a través de nuestros dos 74HC595

```
void send_data(unsigned int16 data_out)
{
    int i;
    unsigned value;
    for (i=0 ; i<16 ; i++)
    {
        value = (data_out >> i) & (0x01); //seleccionamos individualmente los bits de value
        output_bit(DS, value);
        clock();
    }
       latch(); // mostramos los datos a la vez pasamos de linea
}
```

Figura 68. Función para mandar datos al 74HC595

Para hacer una prueba del funcionamiento de los 74HC595 en Proteus podemos hacer una simulación donde vayamos poniendo en nivel alto una a una las 16 salidas con el código siguiente:

```
send_data(0b1000000000000000);
delay_ms(200);
send_data(0b0100000000000000);
delay_ms(200);
send_data(0b0010000000000000);
delay_ms(200);
send_data(0b0001000000000000);
delay_ms(200);
send_data(0b0000100000000000);
delay_ms(200);
send_data(0b0000010000000000);
delay_ms(200);
send_data(0b0000001000000000);
delay_ms(200);
send_data(0b0000000100000000);
delay_ms(200);
send_data(0b0000000010000000);
delay_ms(200);
send_data(0b0000000001000000);
delay_ms(200);
send_data(0b0000000000100000);
delay_ms(200);
send_data(0b0000000000010000);
delay_ms(200);
send_data(0b0000000000001000);
delay_ms(200);
send_data(0b0000000000000100);
delay_ms(200);
send_data(0b0000000000000010);
delay_ms(200);
send_data(0b0000000000000001);
delay_ms(200);
```

Figura 69. Código de comprobación salidas 74HC595

Podemos observar cómo se van encendiendo las salidas desde el primer registro hasta el último. Aquí vemos solamente tres capturas, pero nos da una idea de cómo funciona.

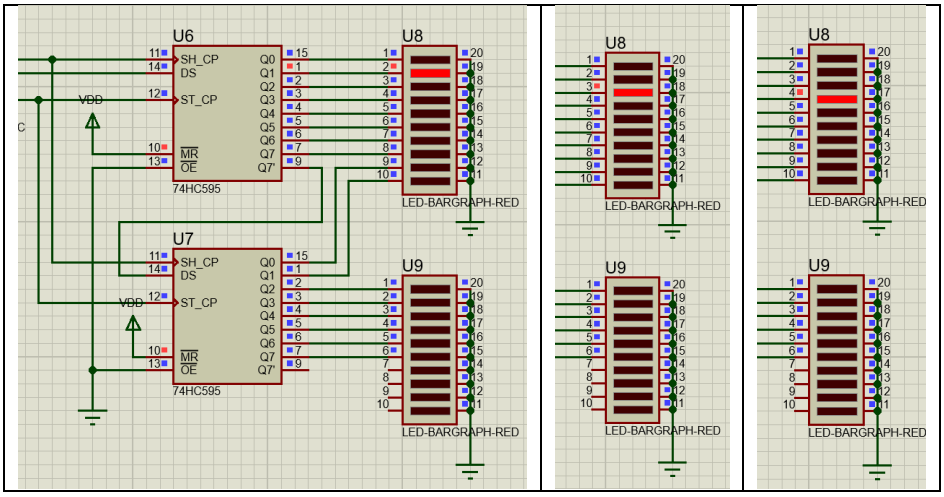


Figura 70. Simulación 74HC595

Ya tenemos control absoluto de nuestro panel LED, controlamos tanto las columnas como las filas. Ahora tenemos que ver cómo representar las letras y símbolos en él. Para esto habrá que diseñarlos primero. Escogeremos la letra F como ejemplo dibujándola sobre una matriz de 16x16. Para obtener una polarización directa de los LEDs que forman la letra y nuestros desplazamientos de registro están conectados en el cátodo (-) de los LEDs, debemos establecer

las salidas de los 74HC595 en estado bajo, coincidiendo con la fila del 4017 que esté en estado alto. Así que asignamos un 0 en los pixeles que queramos que se enciendan y un 1 en los LEDs que queramos que permanezcan apagados.

Una vez completado el dibujo, asignamos a cada pixel en vertical un peso diferente, comenzando por el LSB que sea el de arriba, así en binario obtendremos la columna uno como “1111 1111 1111 1111” o 0xFFFF en hexadecimal que es más fácil manejar. La columna ocho “1111 1100 0111 1000” o 0xFC78.

Numero	Peso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
2	2	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
3	4	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
4	8	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
5	16	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
6	32	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
7	64	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
8	128	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
9	256	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
10	512	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
11	1024	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	2048	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	4096	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	8192	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	16384	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
16	32768	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	DEC	65535	65535	0	0	0	0	64632	64632	64632	64632	65528	65528	65535	65535	65535	65535
	HEX	0xFFFF	0xFFFF	0x0	0x0	0x0	0x0	0xFC78	0xFC78	0xFC78	0xFC78	0xFFF8	0xFFF8	0xFFFF	0xFFFF	0xFFFF	0xFFFF

Figura 71. Valores para representar la letra F

Podemos repetir este proceso para todas las letras del abecedario y algunos símbolos. Aquí vemos como quedaría la A y la B.

Numero	Peso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1
2	2	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
3	4	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1
4	8	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
5	16	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
6	32	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
7	64	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
8	128	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
9	256	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
10	512	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
11	1024	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
12	2048	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
13	4096	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
14	8192	1	1	0	0	0	1	1	1	1	0	0	0	0	1	1	1
15	16384	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
16	32768	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	DEC	65535	65535	0	0	0	15996	15996	15996	15996	7224	0	32769	49603	65535	65535	65535
	HEX	0xFFFF	0xFFFF	0x0	0x0	0x0	0x3E7C	0x3E7C	0x3E7C	0x3E7C	0x1C38	0x0	0x8001	0xC1C3	0xFFFF	0xFFFF	0xFFFF

Figura 72. Valores para representar la letra A y B

Vamos a representar los símbolos de la tabla ASCII desde el 32 hasta el 90.

Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sinc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Figura 73. Tabla ASCII

Al final del proceso obtenemos la siguiente tabla que situamos en un vector simboloASCII[];

```
const unsigned long simboloASCII [] = {
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // ESPACIO
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x1000, 0x1000, 0x1000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // !
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFE1, 0xFFE1, 0xFFE1, 0xFFFF, 0xFFFF, 0xFFE1, 0xFFE1, 0xFFE1, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // "
0xFFFF, 0xF1C7, 0xF1C7, 0xF1C7, 0x0,    0x0,    0x0,    0xF1C7, 0xF1C7, 0xF1C7, 0x0,    0x0,    0x0,    0xF1C7, 0xF1C7, 0xFFFF, // #
0xFFFF, 0xFFFF, 0xCFC3, 0x8F81, 0xF01,  0x0,    0x0,    0x1E38, 0x1C78, 0x0,    0x0,    0xF1,   0x81F3, 0x83FF, 0xFFFF, 0xFFFF, // $
0x3FFF, 0x1FFF, 0x8FC3, 0xC799, 0xE399, 0xF1C3, 0xF8FF, 0xFC7F, 0xFE3F, 0xFF1F, 0xC38F, 0x99C7, 0x99E3, 0xC3F1, 0xFFF8, 0xFFFF, // %
0xFFFF, 0xE1FF, 0xC0C7, 0x8003, 0x1C01, 0x3E31, 0x3E79, 0x3E79, 0x1C31, 0x1,    0xC083, 0xC1CF, 0x88FF, 0x1CFF, 0x1FFF, 0xFFFF, // &
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFE1, 0xFFE1, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // '
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC003, 0x8001, 0x8001, 0xFF0,  0x1FF8, 0x1FF8, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // (
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x1FF8, 0x1FF8, 0xFF0,  0x8001, 0x8001, 0xE003, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // )
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFF7, 0xFFD5, 0xFFE3, 0xFF80, 0xFFE3, 0xFFD5, 0xFFF7, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // *
0xFFFF, 0xFE3F, 0xFE3F, 0xFE3F, 0xFE3F, 0xFE3F, 0xC003, 0xC003, 0xC003, 0xFE3F, 0xFE3F, 0xFE3F, 0xFE3F, 0xFE3F, 0xFFFF, 0xFFFF, // +
0xFFFF, 0xFFFF, 0xFFFF, 0x7FFF, 0x3FFF, 0x1FFF, 0x9FFF, 0xDFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // ,
0xFFFF, 0xFFFF, 0xFFFF, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0xFFFF, 0xFFFF, // -
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x3FFF, 0x3FFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // .
0xFFFF, 0xFFFF, 0xFFFF, 0x3FFF, 0xFFF,  0xC3FF, 0xF0FF, 0xFC3F, 0xFF0F, 0xFFC3, 0xFFF0, 0xFFFC, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // /
0xFFFC, 0xFFFC, 0xC000, 0x8000, 0x8000, 0xFF0,  0x1FF8, 0x1FF8, 0x1FF8, 0xFF0,  0x8000, 0x8000, 0xE000, 0xFFFC, 0xFFFC, 0xFFFC, // 0
0xFFFC, 0xFFFC, 0xFFF4, 0xFFF0, 0x3FF0, 0x3FF8, 0x0,    0x0,    0x0,    0x3FFC, 0x3FFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, // 1
0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFE0,  0x7E0,  0x3E0,  0x1F8,  0x10FC, 0x187C, 0x1C38, 0x1E00, 0x1F00, 0x1FC0, 0xFFFC, 0xFFFC, // 2
0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xC7E0, 0x87E0, 0x7E0,  0x1FF8, 0x3E7C, 0x3C3C, 0x1818, 0x0,    0x8180, 0xC3C0, 0xFFFC, 0xFFFC, // 3
0xFFFC, 0xFFFC, 0xFFFC, 0xFE00, 0xFE00, 0xFE00, 0xFE3C, 0xFE3C, 0xFE3C, 0x0,    0x0,    0x0,    0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, // 4
0xFFFC, 0xFFFC, 0xFFFC, 0x1E00, 0x1E00, 0x1E00, 0x1E1C, 0x1E3C, 0x1E3C, 0x8C3C, 0x803C, 0x807C, 0xE0FC, 0xFFFC, 0xFFFC, 0xFFFC, // 5
0xFFFC, 0xFFFC, 0xC07C, 0x801C, 0xC,    0xE04,  0x1F00, 0x1F20, 0x1F30, 0x8E38, 0x803C, 0x807C, 0xE0FC, 0xFFFC, 0xFFFC, 0xFFFC, // 6
0xFFFC, 0xFFFC, 0x1FF8, 0xFF8,  0x7F8,  0x83F8, 0xC1F8, 0xE0F8, 0xF078, 0xF838, 0xFC18, 0xFF00, 0xFF80, 0xFFC0, 0xFFFC, 0xFFFC, // 7
0xFFFC, 0xFFFC, 0xC100, 0x8000, 0x8000, 0xE38,  0x1E7C, 0x1E7C, 0x1E7C, 0xE38,  0x8000, 0x8000, 0xE180, 0xFFFC, 0xFFFC, 0xFFFC, // 8
0xFFFC, 0xFFFC, 0xFF00, 0xFE00, 0xFC00, 0xFC38, 0xFC7C, 0xFC7C, 0xFC7C, 0xFC38, 0x0,    0x0,    0x0,    0xFFFC, 0xFFFC, 0xFFFC, // 9
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC47F, 0xC47F, 0xC47F, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // :
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x7FFF, 0x3FFF, 0x8FF,  0x88FF, 0xC8FF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // ;
0xFFFF, 0xFE7F, 0xFC3F, 0xF99F, 0xF18F, 0xF3CF, 0xE7E7, 0xC7E3, 0xCFF3, 0xCFF3, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // <
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xE31F, 0xE31F, 0xE31F, 0xE31F, 0xE31F, 0xE31F, 0xE31F, 0xE31F, 0xFFFF, 0xFFFF, 0xFFFF, // =
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xCFF3, 0xCFF3, 0xC7E3, 0xE7E7, 0xF7EF, 0xF3CF, 0xF99F, 0xF99F, 0xFC3F, 0xFE7F, 0xFFFF, 0xFFFF, // >
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFF1, 0xFFF0, 0xFFF0, 0x13FC, 0x10FC, 0x1078, 0xFE00, 0xFF01, 0xFFC3, 0xFFFF, 0xFFFF, 0xFFFF, // ?
0xFFFF, 0xFFFF, 0xFE0F, 0xFC07, 0xF9F3, 0xF319, 0xF2E9, 0xF2E9, 0xF2E9, 0xF209, 0xF279, 0xF873, 0xFC07, 0xFE07, 0xFFFF, 0xFFFF, // @
0xFFFF, 0x3,    0x1,    0x0,    0x0,    0xFC78, 0xFC7C, 0xFC7C, 0xFC7C, 0xFC78, 0x0,    0x0,    0x1,    0x3,    0xFFFF, 0xFFFF, // A
0xFFFF, 0x0,    0x0,    0x0,    0x0,    0x3E7C, 0x3E7C, 0x3E7C, 0x3E7C, 0x1C38, 0x800,  0x8000, 0x8081, 0xC1C3, 0xFFFF, 0xFFFF, // B
0xFFFF, 0xFFFF, 0xF00F, 0xC003, 0x8001, 0x8001, 0x7F0,  0xFF8,  0x1FF8, 0x1FF8, 0x1FF8, 0x1FF8, 0x1FF8, 0x1FF8, 0xFFFF, 0xFFFF, // C
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0x1FF8, 0x1FF8, 0x1FF8, 0x1FF8, 0xFF0,  0x8001, 0x8001, 0xE003, 0xF007, 0xFFFF, // D
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0x1E38, 0x1E38, 0x1E38, 0x1E38, 0x1FF8, 0x1FF8, 0x1FF8, 0x1FF8, 0xFFFF, 0xFFFF, // E
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xFC78, 0xFC78, 0xFC78, 0xFC78, 0xFFF8, 0xFFF8, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // F
0xFFFF, 0xFFFF, 0xF00F, 0xC003, 0x8001, 0x8001, 0x7F0,  0xFF8,  0x1E78, 0x1E78, 0x78,   0x78,   0x8078, 0xC0F9, 0xFFFF, 0xFFFF, // G
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xFC7F, 0xFC7F, 0xFC7F, 0xFC7F, 0x0,    0x0,    0x0,    0x0,    0xFFFF, 0xFFFF, // H
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // I
0xFFFF, 0xFFFF, 0xC7FF, 0x87FF, 0x7FF,  0x7FF,  0x3FFF, 0x3FFF, 0x0,    0x0,    0x0,    0x8000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // J
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xF08F, 0xE1C7, 0xC3C3, 0x87E1, 0xFF0,  0x1FF8, 0x3FFC, 0x7FFE, 0xFFFF, 0xFFFF, // K
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0x1FFF, 0x1FFF, 0x1FFF, 0x1FFF, 0x1FFF, 0x1FFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // L
0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xFFE1, 0xFFC3, 0xFF87, 0xFF87, 0xFFC3, 0xFFE1, 0x0,    0x0,    0x0,    0x0,    0xFFFF, // M
0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xFFE1, 0xFF87, 0xFE1F, 0xF87F, 0xE1FF, 0x83FF, 0x0,    0x0,    0x0,    0x0,    0xFFFF, // N
0xFFFF, 0xF00F, 0xC003, 0x8001, 0x87C1, 0xFF0,  0x1FF8, 0x1FF8, 0x1FF8, 0x1FF8, 0xFF0,  0x87C1, 0x8001, 0xC003, 0xE00F, 0xFFFF, // O
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0x0,    0xFE38, 0xFE38, 0xFE38, 0xFE10, 0xFE00, 0xFE00, 0xFF01, 0xFF83, 0xFFFF, 0xFFFF, // P
0xFFFF, 0xF00F, 0xC003, 0x8001, 0x87C1, 0xFF0,  0x1FF8, 0x1FF8, 0x19F8, 0x11F8, 0x1F0,  0x83C1, 0x8001, 0x3,    0xF,    0x1FFF, // Q
0xFFFF, 0xFFFF, 0x0,    0x0,    0x0,    0xF038, 0xF038, 0xE038, 0xC038, 0x8310, 0x700,  0xF81,  0x1FC3, 0x3FFF, 0xFFFF, 0xFFFF, // R
0xFFFF, 0xFFFF, 0xCFC3, 0x8F81, 0xF01,  0xE00,  0x1E38, 0x1E38, 0x1C78, 0x1C78, 0xF0,   0xF1,   0x81F3, 0x83FF, 0xFFFF, 0xFFFF, // S
0xFFFF, 0xFFF8, 0xFFF8, 0xFFF8, 0xFFF8, 0xFFF8, 0x0,    0x0,    0x0,    0xFFF8, 0xFFF8, 0xFFF8, 0xFFF8, 0xFFF8, 0xFFF8, 0xFFFF, // T
0xFFFF, 0xC000, 0x8000, 0x8000, 0x8000, 0x0,    0xFFF,  0x1FFF, 0x1FFF, 0x1FFF, 0xFFF,  0x0,    0x8000, 0x8000, 0xC000, 0xFFFF, // U
0xFFFF, 0xFFF0, 0xFFC0, 0xFF00, 0xFC07, 0xF01F, 0xC03F, 0x1FF,  0x1FF,  0xC03F, 0xF01F, 0xFC07, 0xFF00, 0xFFC0, 0xFFF0, 0xFFFF, // V
0xFA00, 0xF800, 0xE000, 0x81EA, 0x3EA,  0x81EA, 0xE042, 0xF800, 0xF800, 0xE042, 0x81EA, 0x3EA,  0x81EA, 0xE000, 0xF800, 0xFA00, // W
0xFFFF, 0x1FF0, 0xFE1,  0x87C3, 0xC387, 0xE10F, 0xF01F, 0xF83F, 0xF83F, 0xF01F, 0xE10F, 0xC387, 0x87C3, 0xFE1,  0x1FF0, 0xFFFF, // X
0xFFFF, 0xFFF8, 0xFFF0, 0xFFE0, 0xFFC1, 0xFF83, 0x7,    0xF,    0xF,    0x7,    0xFF83, 0xFFC1, 0xFFE0, 0xFFF0, 0xFFF8, 0xFFFF, // Y
0xFFFF, 0x1FF0, 0x1FE0, 0x1FC0, 0x1F80, 0x1F00, 0x1E08, 0x1C18, 0x1838, 0x1078, 0xF8,   0x1F8,  0x3F8,  0x7F8,  0xFF8,  0xFFFF, // Z
};
```

Ahora solamente debemos ver como representar estos datos en nuestro panel LED. Como ejemplo utilizaremos la letra A. Tiene el valor 65 en la tabla ASCII, le restamos 32 que es nuestro primer elemento, obtenemos 33 que multiplicamos por 16 que son los valores por carácter, así obtenemos las posiciones de la letra A, donde el primer elemento está en la posición 528

Ahora quedaría representar el primer bit del 528, 529,530, hasta el 544. Luego el segundo bit de la posición 528, 529,530 hasta la posición 544. Repetimos con las 16 filas.

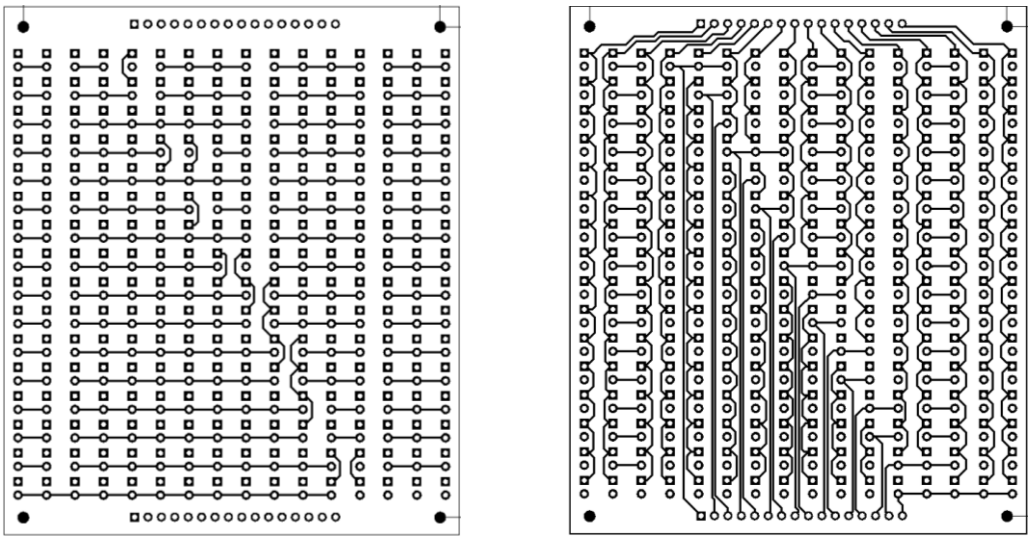
Lo llevamos a código quedando de la siguiente manera:

```
unsigned int letra, direccion;

letra = 65 // posicion de la A en la tabla ASCII
direccion = letra - 32;
direccion = direccion * 16;



// 16 iteracciones para representar la filas
for (y = 0 ; y <  16 ; y++)
    {
      // 16 iteracciones para representar las columnas
       for (x = 0 ; x < direccion + 16 ; x++)
              {
              //seleccionamos individualmente los bits de value
                     value = (simboloASCII[x] data_out >> y) & (0x01);
                     output_bit(DS, value);
                 clock();
              }
              latch(); // mostramos los datos a la vez pasamos de linea
    }
```

9. PCBs

9.1. Panel LED

Una vez diseñados los circuitos los pasamos a Kicad. Aquí podemos ver como quedaría las capas superior e inferior de la placa PCB y la colocación de los LEDs en el panel formando una matriz de 16x16 LEDs.



Vista superior *Vista inferior*

Figura 74. PCB del panel LED (capa superior e inferior)

La placa tendrá una dimensión de 10cm x 8,64cm y tendrá en la parte superior e inferior una tira de pines para conectar con los drivers.

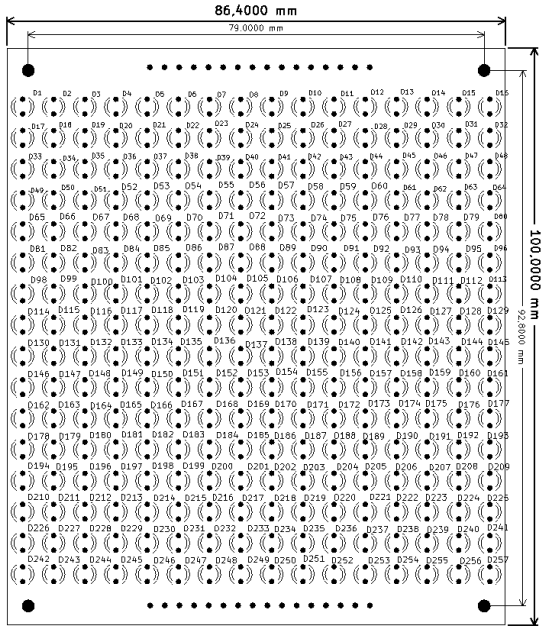


Figura 75. PCB panel LED, colocación componentes

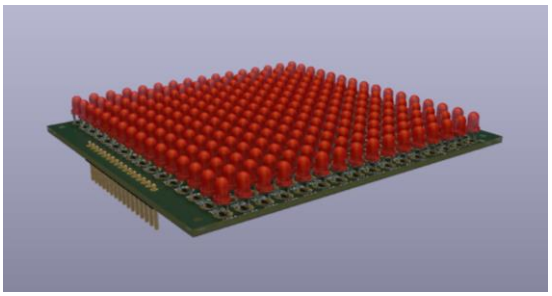
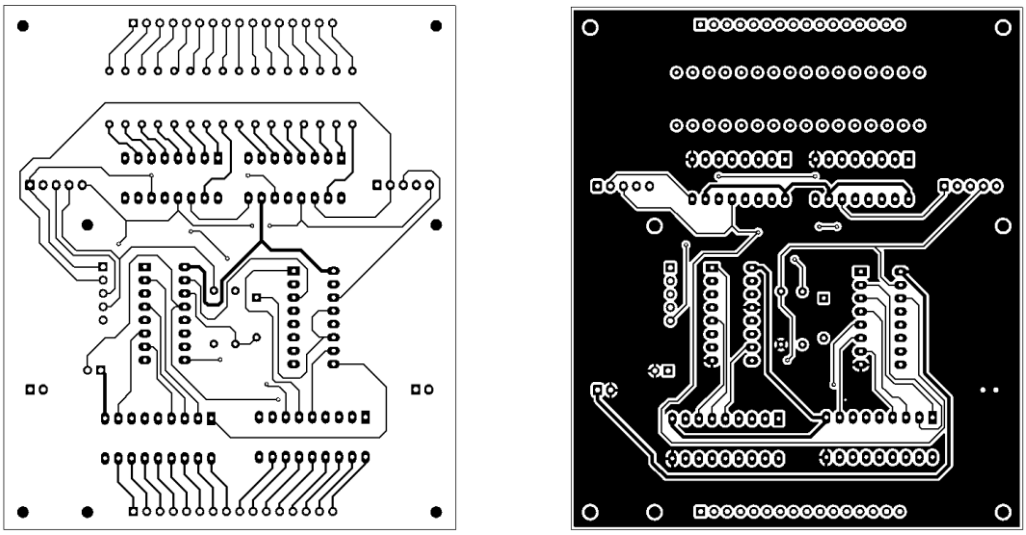


Figura 76. Representación 3D panel LED

9.2. Driver LED

Aquí vemos la placa de control de LEDs, vemos la colocación de los componentes y el conexionado de estos.



Vista superior *Vista inferior*

Figura 77. PCB del driver LED (capa superior e inferior)

Esta placa tendrá la misma dimensión que el panel de LED 10cm x 8,64cm para poder acoplar uno con otro.

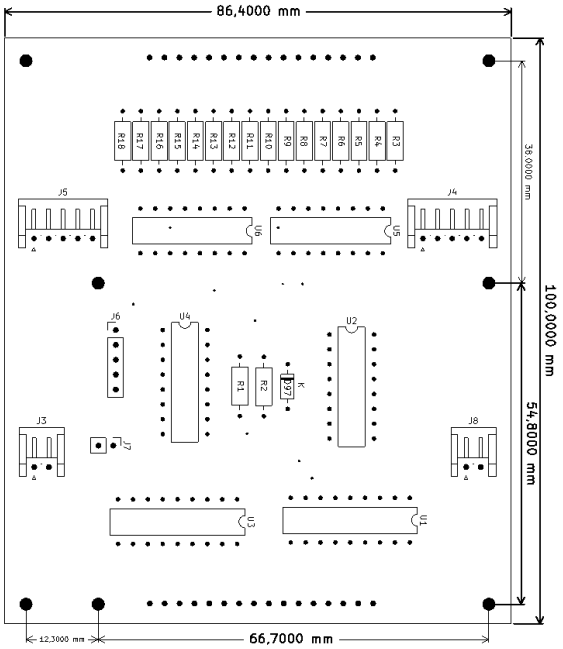


Figura 78. Colocación componentes driver LED

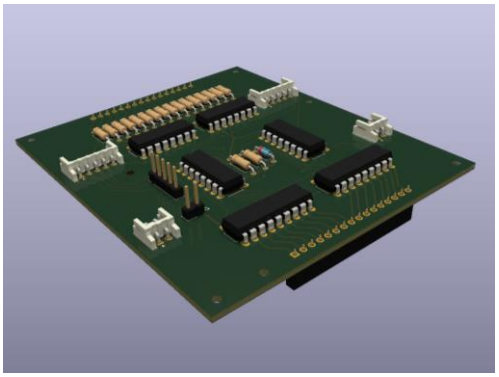
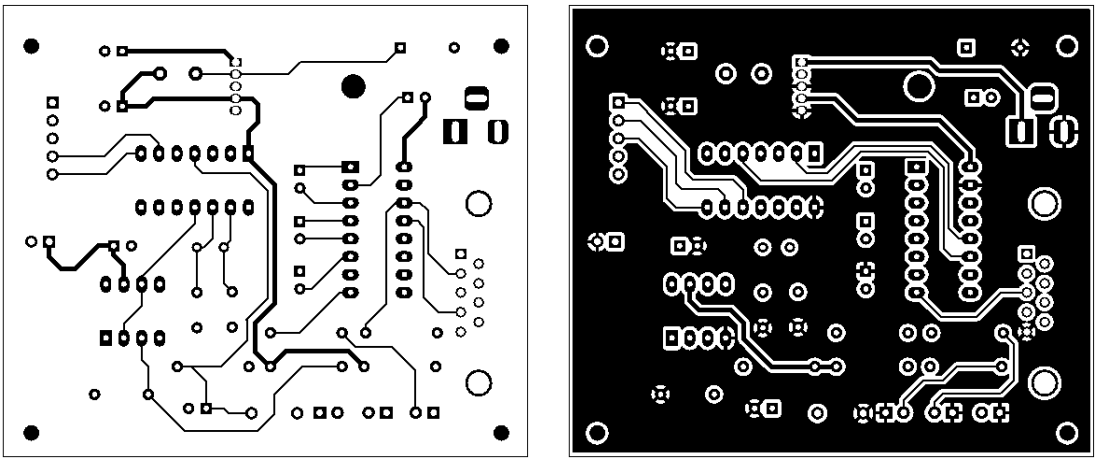


Figura 79. Representación 3D panel driver LED

9.3. Placa controladora

Y la último, la placa seria la placa donde está dispuesto el microcontrolador junto con el resto de periféricos.



Vista superior *Vista inferior*

Figura 80. PCB del microcontrolador (capa superior e inferior)

La placa tendrá un tamaño de 6,39 cm x 7,44 cm con perforaciones para poder sujetarse a la placa de los drivers.

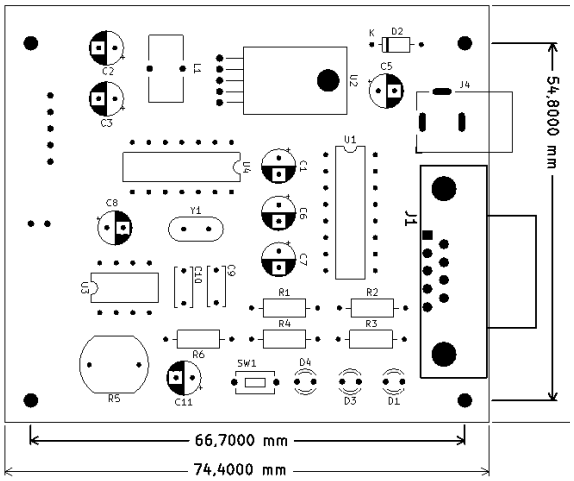


Figura 81. Colocación de los componentes PCB microcontrolador

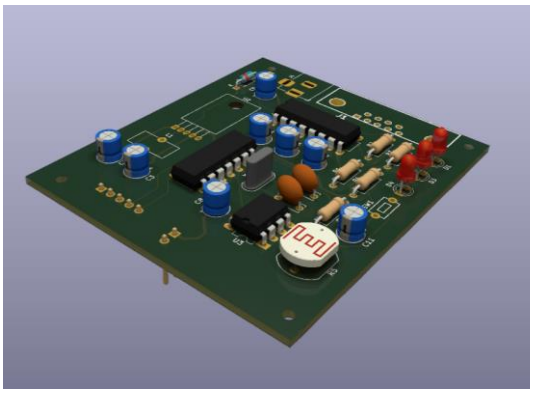


Figura 82. Representación 3D PCB microcontrolador

9.4.1 Montaje

Ambas placas, el panel LED y la controladora quedaran acopladas una con otra de la siguiente manera. Y la placa del microcontrolador la situaremos a detrás.

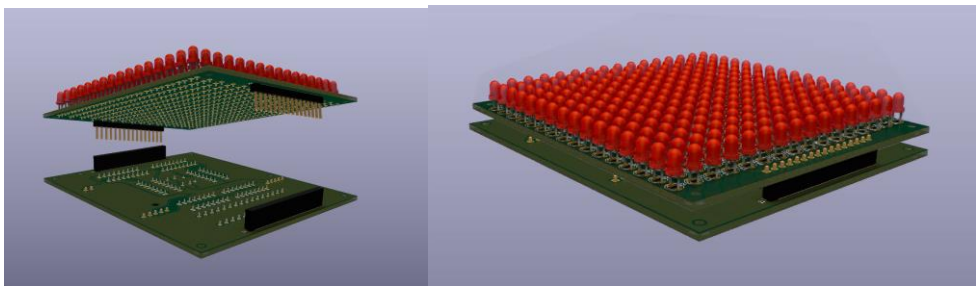


Figura 83. Ensamblado de las PCB (LED y driver)

Todo ensamblado vemos cómo quedará la parte frontal, donde solamente se vera la matriz de LED y por la parte posterior veríamos los drivers y el microcontrolador.

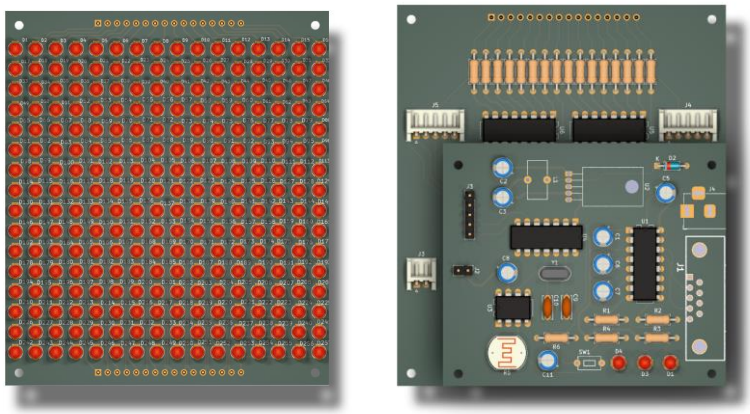


Figura 84. Resultado final (frontal y trasero)

Con ese diseño facilitaríamos la ampliación de la matriz del para poder hacer un panel más largo, de hasta 5 paneles más según la dimensión de nuestra fuente de alimentación.

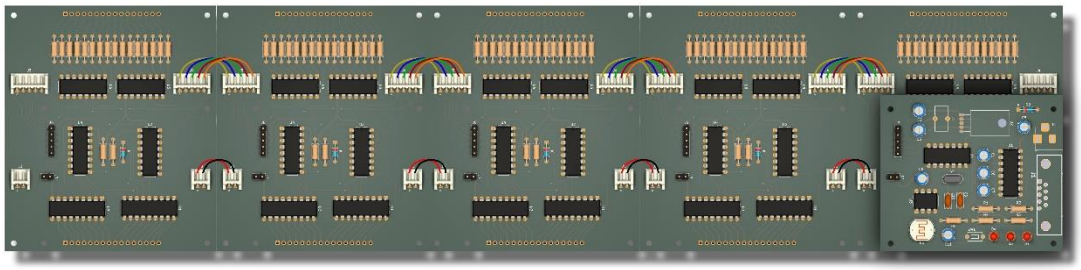


Figura 85. Conexionado cinco paneles LED

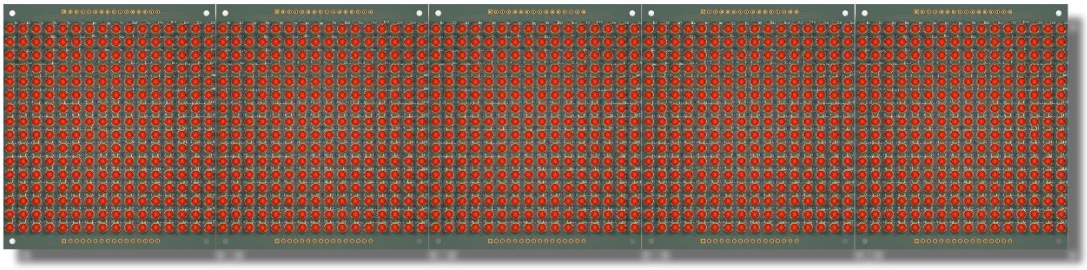


Figura 86. Resultado final cinco paneles LED

9.4.2 Prototipo V1.0

Se mandan fabricar las placas a JLCPCB, llegan en un par de semanas. Se fabricaron en negro para mejorar el efecto de los LEDs. Y se eligieron LEDs verdes con la capsula transparente. El resto de componentes son los mismos que los utilizados en la protoboard.

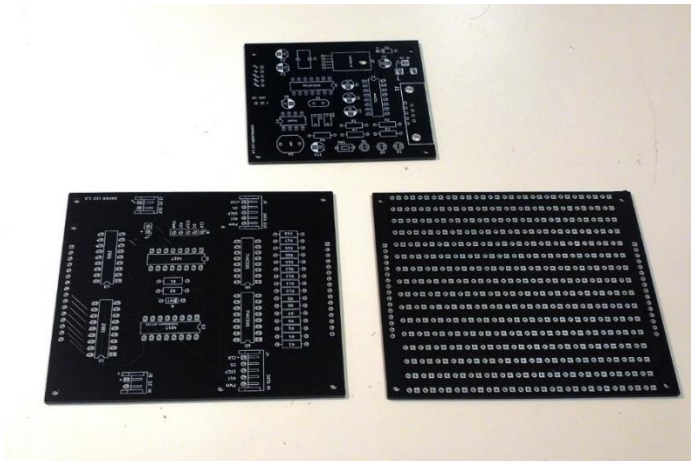


Figura 87. Placas PCB (microcontrolador, driver y panel LED)

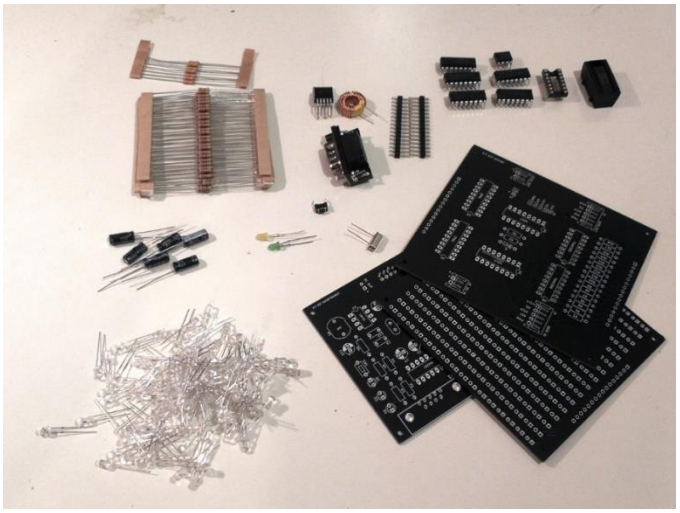


Figura 88. Componentes y PCB

Soldamos todo y este es el resultado obtenido:

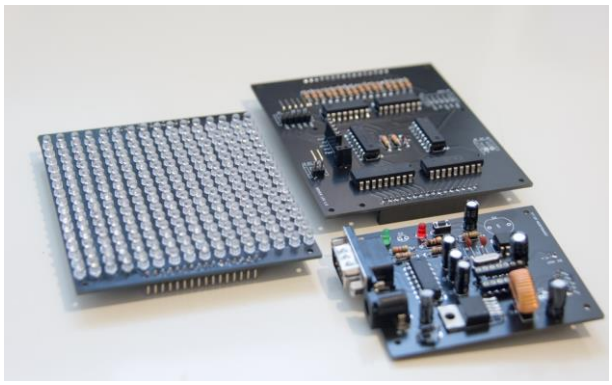


Figura 89. Prototipo ya con los componentes soldados

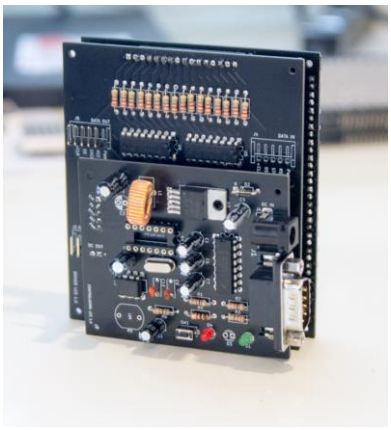


Figura 90. Vista posterior del prototipo ensamblado

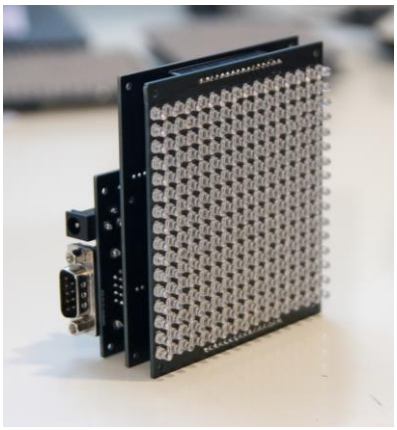


Figura 91. Vista frontal del prototipo ensamblado

Conectamos el programador de MPLAB Pick3 de la siguiente manera a nuestro microcontrolador:

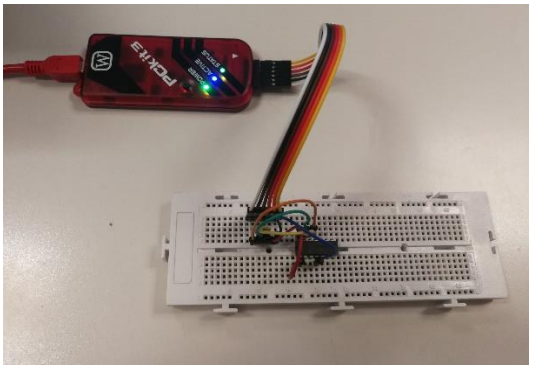


Figura 92. Conexión Pick3 con el microcontrolador

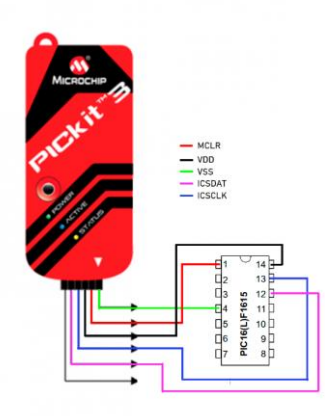


Figura 93. Pineado Pick3 con PIC16F1615

y programamos el microcontrolador con el software MPLAB IPE v6.05

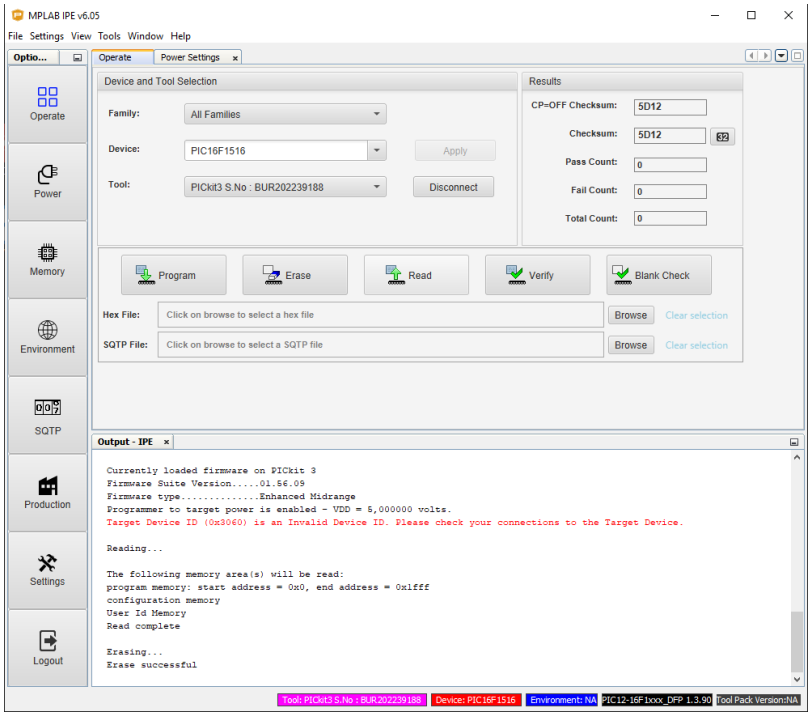


Figura 94. Software MPLAB v6.05

Una vez cargado el software probamos y al parecer hay un algún error en la circuitería de los drivers. Realizamos comprobaciones por partes, primero comprobamos la placa de la matriz de LEDs y funciona.

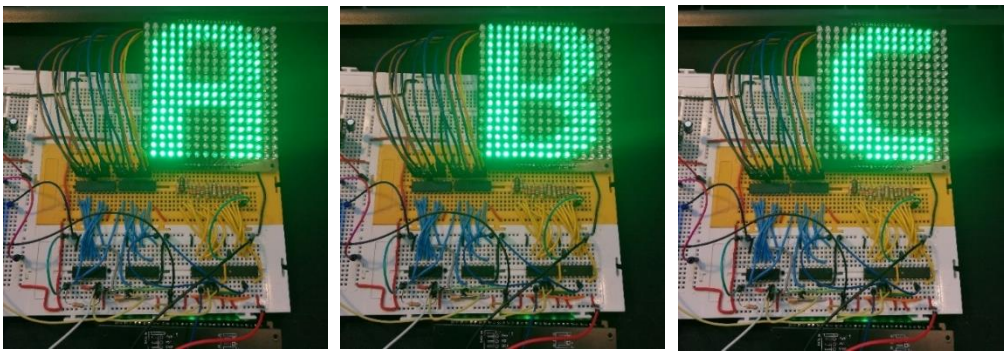


Figura 95. Comprobación matriz LED

Parece que después de un cortocircuito toca reemplazar un integrado 4017. Una vez reemplazado la mitad del panel no responde en el barrido horizontal y resulta ser un fallo a la hora de utilizar etiquetas globales en Kicad.

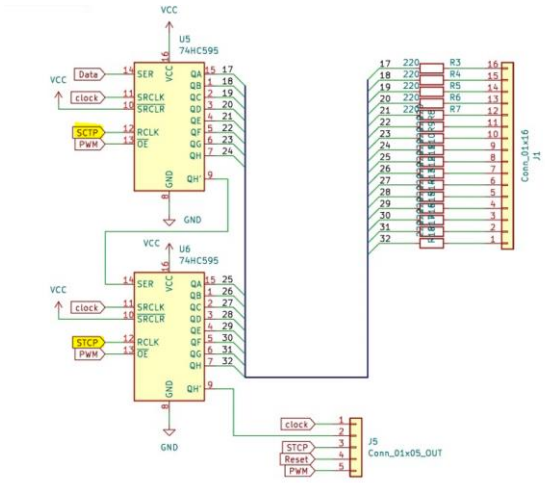


Figura 96. Error en las etiquetas de Kicad

Ambas etiquetas deberían tener el mismo nombre, al ser diferentes, una está como SCTP y la otra STCP, no están conectadas entre sí y por tanto los integrados tampoco. Realizamos un arreglo en la PCB y comprobamos.

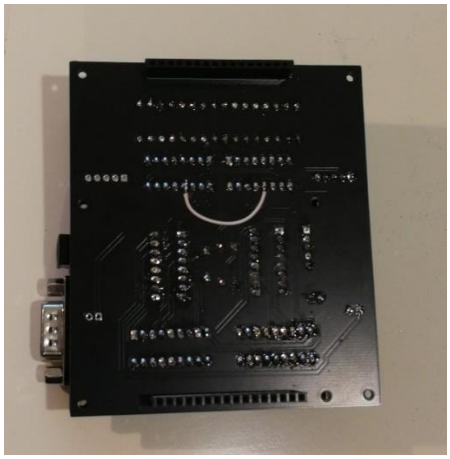


Figura 97. Arreglo de la PCB del driver LED

Figura 98. Prototipo V1.0 funcionando

10. Presupuesto

FABRICACION PLACA PCB

UNID.	DESCRIPCION	PRECIO	P. TOTAL
5	PCB Microcontrolador	2 €	2 €
5	PCB Driver	2 €	2 €
5	PCB Matriz LED	2 €	2 €
1	Gastos de envio	20 €	20 €
		TOTAL	**26 €**

COMPONENTES PLACA MICROPROCESADOR

UNID.	DESCRIPCION	PRECIO	P. TOTAL
1	Microcontrolador PIC16F1615	1,98 €	1,98 €
1	Cristal 16Mhz	2,92 €	2,92 €
2	Condensadores 22pF	0,2 €	0,4 €
1	Condensadores 220uF	0,2 €	0,2 €
4	Condensadores 1uF	0,15 €	0,6 €
1	Condensadores 47uF	0,25 €	0,25 €
5	Resistencia	0,008 €	0,04 €
1	Pulsador	0,035 €	0,035 €
3	LED 3mm	0,023 €	0,069 €
1	Conector DB9 Hembra	0,30 €	0,30 €
1	MAX232	0,92 €	0,92 €
1	LM2596S-5	0,14 €	0,14 €
1	LM741	0,1 €	0,1 €
		TOTAL	**7,95 €**

COMPONENTES PLACA DRIVER

UNID.	DESCRIPCION	PRECIO	P. TOTAL
1	Diodo 1N4148	0,012 €	0,012 €
2	UDN2981	0,45 €	0,9 €
2	4017	0,17 €	0,34 €
2	74HC595	0,23 €	0,46 €
16	Resistencias 220	0,008 €	0,128 €
2	Tira de pines	1,4 €	2,8 €
		TOTAL	**4,64 €**

COMPONENTES PLACA LED

UNID.	DESCRIPCION	PRECIO	P. TOTAL
256	LEDs 3mm	0,023 €	5,88 €
2	Tira de pines	1,4 €	2,8 €
		TOTAL	**8,68 €**

Coste material total de fabricación panel LED **47,27 €**

Todo este coste hace referencia a la parte material del proyecto, como son los componentes y las PCBs. También tendríamos que tener en cuenta los costes de montaje e industrialización. Para ello se podría contar con los mismos fabricantes donde solicité la placa. Ellos te montan los componentes, ya que poseen un repositorio de más de 355.617 componentes en stock. El precio por 5 placas terminadas seria 8€ más del coste total y tardarían 3 días en la fabricación completa (Preparación del pedido, fabricación PCB, colocado de componentes, soldado componentes e inspección final).

En comparación con una fabricación más manual donde tendríamos que contar con la mano de obra de un técnico electrónico junior, que cobra entorno a unos 11,41€ por hora, según la página

web Talent.com (https://es.talent.com/salary?job=t%C3%A9cnico+electronico). Por lo que el montaje de un módulo del dispositivo al completo, las tres PCBs, se tardaría unas 4 horas con pruebas incluidas, unos 34,23€ más por dispositivo.

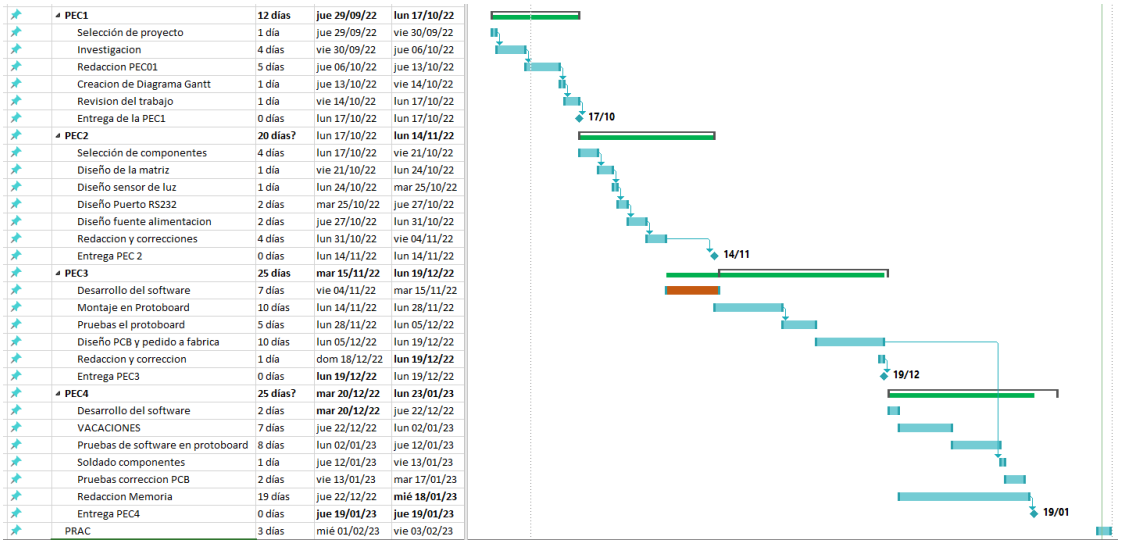
Por lo que si nos quedaríamos con la opción 1 saldría a 47,27€ de materiales, donde comprados al por mayor probablemente reduciríamos costes, más 1,6€ por montaje de modulo (8€/5 placas= 1,6€ por montaje de placa). En total 48,87€ por placa totalmente operativa.

11. Conclusiones

Una vez concluido el proyecto, podemos ver los hitos conseguidos:

- Se diseñó una matriz de 16x16 LEDs y los drivers necesarios para poder controlar dicha matriz. Pudiéndose representar letras, símbolos o cualquier dibujo que queramos en una resolución de 16x16.
- El panel se adapta a la iluminación ambiental gracias a un LDR y un circuito adaptador que manda la información al microcontrolador para una mejor visualización con el mínimo consumo.
- Se diseño una interface de comunicaciones que permita el envío de datos para ser representados en el panel.
- Se diseño una fuente de alimentación con la suficiente potencia que adapte la entrada de 24 V a los 5 V que necesita nuestro circuito.
- Fabricación de una primera versión de un prototipo.

Durante la realización del proyecto hubo que ir replanteando los tiempos. Por lo que el diagrama de Gantt que planteamos al principio sufrió algunos cambios.



	PEC1	**12 días**	**jue 29/09/22**	**lun 17/10/22**
	Selección de proyecto	1 día	jue 29/09/22	vie 30/09/22
	Investigacion	4 días	vie 30/09/22	jue 06/10/22
	Redaccion PEC01	5 días	jue 06/10/22	jue 13/10/22
	Creacion de Diagrama Gantt	1 día	jue 13/10/22	vie 14/10/22
	Revision del trabajo	1 día	vie 14/10/22	lun 17/10/22
	Entrega de la PEC1	0 días	lun 17/10/22	lun 17/10/22
	PEC2	**20 días?**	lun 17/10/22	**lun 14/11/22**
	Selección de componentes	4 días	lun 17/10/22	vie 21/10/22
	Diseño de la matriz	1 día	vie 21/10/22	lun 24/10/22
	Diseño sensor de luz	1 día	lun 24/10/22	mar 25/10/22
	Diseño Puerto RS232	2 días	mar 25/10/22	jue 27/10/22
	Diseño fuente alimentacion	2 días	jue 27/10/22	lun 31/10/22
	Redaccion y correcciones	4 días	lun 31/10/22	vie 04/11/22
	Entrega PEC 2	0 días	lun 14/11/22	lun 14/11/22
	PEC3	**25 días**	**mar 15/11/22**	**lun 19/12/22**
	Desarrollo del software	7 días	vie 04/11/22	mar 15/11/22
	Montaje en Protoboard	10 días	lun 14/11/22	lun 28/11/22
	Pruebas el protoboard	5 días	lun 28/11/22	lun 05/12/22
	Diseño PCB y pedido a fabrica	10 días	lun 05/12/22	lun 19/12/22
	Redaccion y correccion	1 día	dom 18/12/22	**lun 19/12/22**
	Entrega PEC3	0 días	**lun 19/12/22**	lun 19/12/22
	PEC4	**25 días?**	**mar 20/12/22**	**lun 23/01/23**
	Desarrollo del software	2 días	**mar 20/12/22**	jue 22/12/22
	VACACIONES	7 días	jue 22/12/22	lun 02/01/23
	Pruebas de software en protoboard	8 días	lun 02/01/23	jue 12/01/23
	Soldado componentes	1 día	jue 12/01/23	vie 13/01/23
	Pruebas correccion PCB	2 días	vie 13/01/23	mar 17/01/23
	Redaccion Memoria	19 días	jue 22/12/22	**mié 18/01/23**
	Entrega PEC4	0 días	**jue 19/01/23**	**jue 19/01/23**
	PRAC	3 días	mié 01/02/23	vie 03/02/23

Figura 99. Diagrama de Gantt final

La mayor diferencia se produjo a raíz de la entrega de la PEC2, terminé antes de lo esperado y decidí comenzar con tareas destinadas para la PEC3 como el desarrollo del software lo que me permitió dedicar más tiempo al montaje en la protoboard y pruebas de los circuitos. Viendo que

se acercaban Vacaciones y no las había contemplado adelanté el diseño de la PCB para el 5/12 y poder pedirlo fabricar antes de Navidades. Solicité las placas el 19/12 y llegaron el 30/12.

Ya en la PEC4 y viendo que tenía trabajo adelantado reduje las tareas durante las vacaciones de Navidad. También durante toda la duración de la PEC4 llevé en paralelo la redacción de la memoria. Ya con las PCBs las soldé en un día y comencé con las pruebas y finalización de la redacción de la memoria.

Como en todo proyecto apareció algún riesgo no se contemplado al comienzo del proyecto, como por ejemplo de las fechas en las que andábamos y las vacaciones de Navidad. Lo que me llevo a modificar ligeramente las tareas por miedo a no cumplir con fechas de entrega. Adelante ciertas tareas como el diseño en PCB para poder contar con margen suficiente para su fabricación. Y que llegara durante las Navidades.

Otro riesgo que no contemplé fue que hubiera algún error de diseño insalvable que hiciera volver a rediseñar las PCBs y no cumpliera con los plazos de entrega. Pero gracias a las tareas previas de comprobación de los circuitos mediante la protoboard no se materializo ese problema.

Con respecto a los riesgos contemplados al comienzo del proyecto no se materializó ninguno ya que se hizo una selección de componentes bastante estándar y se pudieron adquirir fácilmente. Pero por otra parte hubo que adaptar las tareas para ajustar mejor los tiempos planificados para estas.

Al final, el diseño de nuestro panel ha sido modular de tal manera que pueda ser ampliado, no solo en número de paneles conectados en línea, sino también en funcionalidades para futuras versiones. Por lo tanto, tengo algunas ideas para líneas de trabajo futuras:

- Se le podría dotar al sistema de conexión inalámbrica ya bien sea IEEE 802.11 b/g/n, Bluetooth LE o ambas. Creo que esto daría al sistema lo necesario para poder ser un candidato a dispositivo IoT.
- Otra mejora seria diseñar distintos tipos de fuentes y dibujos, todos estos diseños tendrían que guardarse en la memoria interna del microcontrolador y en caso de no entrar por capacidad, se podría ampliar esta añadiendo una memoria tipo EEPROM o Flash, por ejemplo algunas PCB de desarrollo para el microcontrolador ESP32-C3 tipo (https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/) ya cuentan con una memoria flash de 4MB donde poder almacenar cantidad de diseños.
- En este prototipo solamente se ha considerado la ampliación de los paneles en horizontal, por lo tanto, otra mejora interesante seria poder desarrollar una matriz LED que pudiera unirse modularmente tanto en horizontal como vertical y poder hacer un panel más grande.
- Otra línea de trabajo para la mejora del panel seria cambiar los LEDs monocromáticos por LEDs RGB.
- Y por último me gustaría desarrollar todo esto en componentes tipo SMD para poder miniaturizar toda la electrónica y realizar paneles con leds más juntos y quizá aumentar la resolución de estos, aunque esto implique volver a desarrollar tanto la placa de la matriz LED como la del driver de control.

En conclusión, ha sido un proyecto muy exigente, donde su alcance incluía, no solo la la parte teórica, si no la fabricación de un prototipo. Durante el proceso he encontrado algunos

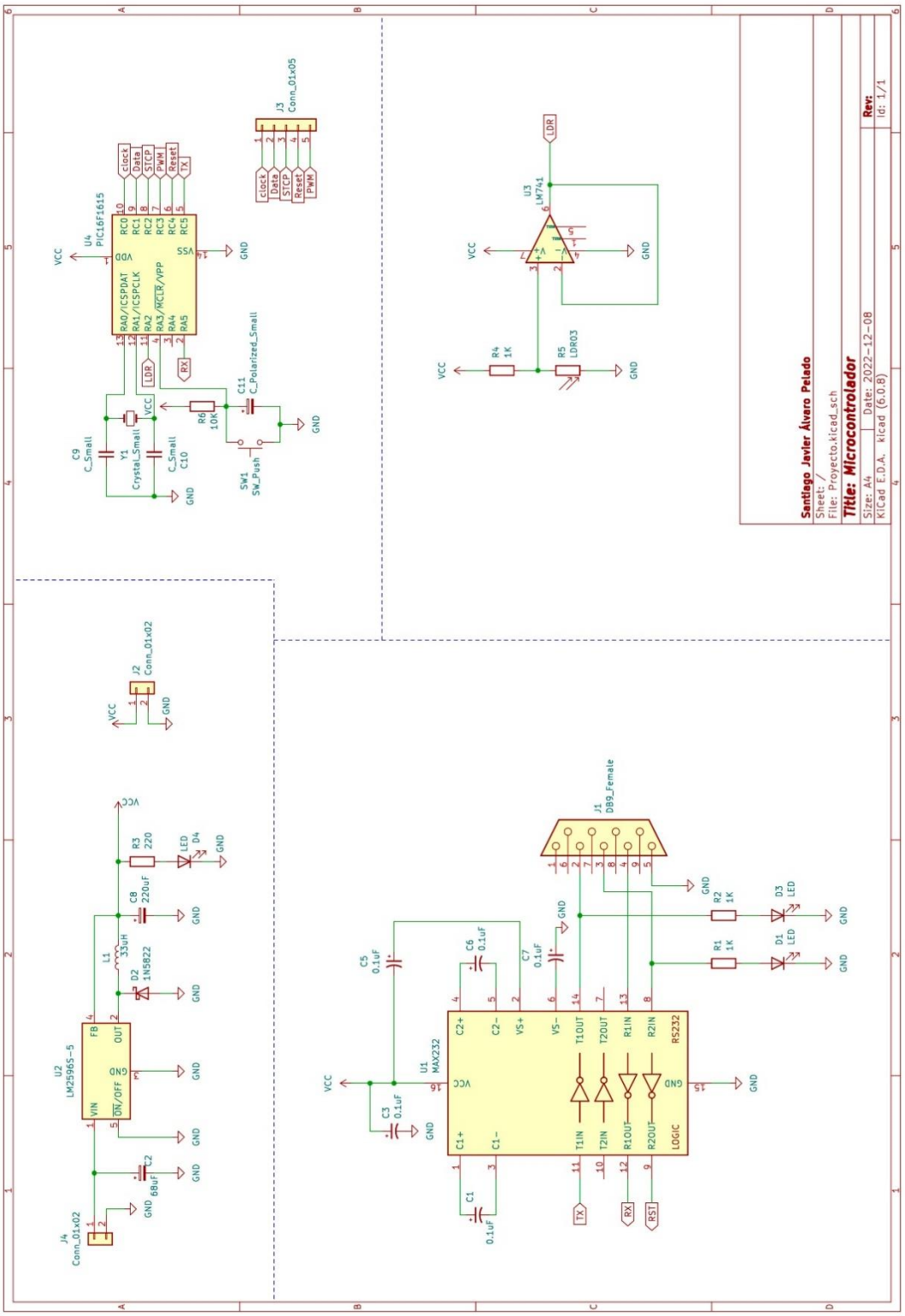
problemas que hubo que solventar y a pesar de eso y con tan solo cuatro meses para la toda investigación, creación de un diseño original, simulación y fabricación, finalmente se ha logrado obtener el panel LED gracias a un trabajo constante y a una buena planificación.

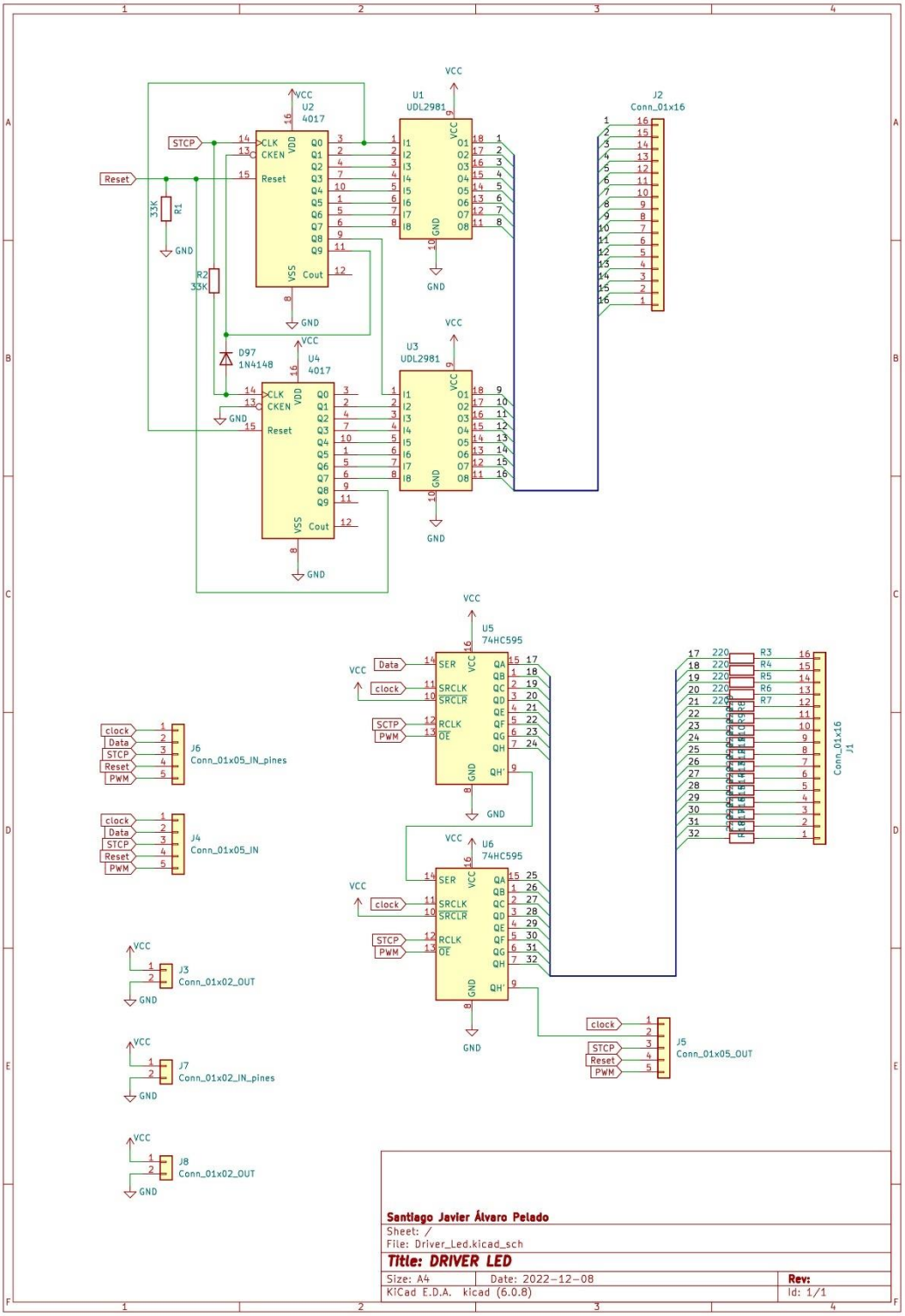
12. Bibliografía

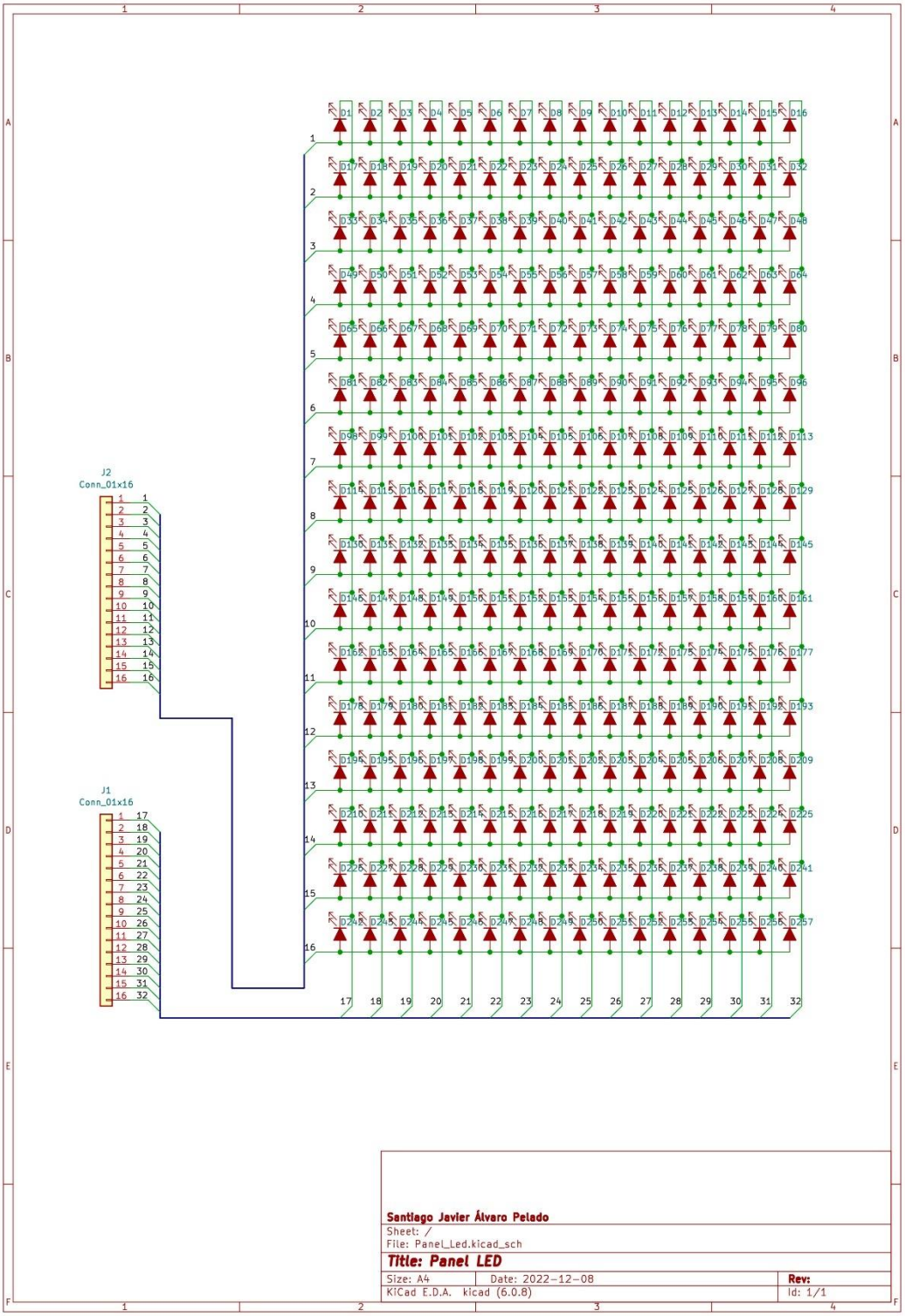
CNET [Consulta 18/10/2022] https://www.cnet.com/tech/home-entertainment/wall-sized-million-dollar-microLED-tvs-point-to-the-future-of-television/
Universidad de Granada [Consulta 18/10/2022] https://www.ugr.es/~juanki/LED.htm
[Consulta 18/10/2022] http://platea.pntic.mec.es/~lmarti2/opto1.htm
Efecto LED [Consulta 19/10/2022] https://www.efectoLED.com/blog/es/tipos-de-chips-LED-que-existen-en-el-mercado/
Silicon Lightworks [Consulta 19/10/2022] https://siliconlightworks.com/resoures/what-are-cob-LEDs
Electronics Tutorials [Consulta 19/10/2022] https://www.electronics-tutorials.ws/combination/comb_3.html
Wikipedia [Consulta 20/10/2022] https://es.wikipedia.org/wiki/Persistencia_de_la_visi%C3%B3n
Best Microcontrollers [Consulta 20/10/2022] https://www.best-microcontroller-projects.com/LED-dot-matrix-display.html
Mecatronicalatam [Consulta 20/10/2022] https://www.mecatronicalatam.com/es/tutoriales/sensores/sensor-de-luz/ldr/
Robots Argentina [Consulta 21/10/2022] https://robots-argentina.com.ar/Sensores_LDR.htm
Wikipedia [Consulta 20/10/2022]
https://es.wikipedia.org/wiki/Fotorresistor#:~:text=%E2%80%8B%20Puede%20tambi%C3%A9n%20ser%20llamado,se%20muestra%20su%20s%C3%ADmbolo%20el%C3%A9ctrico
Learning About Electronic [Consulta 08/11/2022]
http://www.learningaboutelectronics.com/Articulos/Seguidor-de-voltaje.php
Facultad de ciencias Potosí México [Consulta 20/10/2022]
http://galia.fc.uaslp.mx/~cantocar/microcontroladores/SLIDES_8051_PDF/EL_RS232.PDF
Electronic Team, Inc [Consulta 25/10/2022] https://www.eltima.com/es/article/9-pin-serial-port.html
Punto Flotante S.A. [Consulta 25/10/2022] https://www.puntoflotante.net/RS485.htm
Wikipedia [Consulta 24/10/2022] https://es.wikipedia.org/wiki/RS-232
Electrónica Si [Consulta 24/10/2022] https://www.electronicasi.com/wp-content/uploads/2013/05/rs232.pdf
Wikipedia [Consulta 5/11/2022] https://es.wikipedia.org/wiki/Modbus
Modbus.org [Consulta 5/11/2022] https://modbus.org/docs/PI_MBUS_300.pdf
Real Time Automation [Consulta 8/11/2022] https://www.rtautomation.com/technologies/modbus-rtu/
Simply Modbus [Consulta 8/11/2022] http://www.simplymodbus.ca/FAQ.htm
INdustrail Shields [Consulta 8/11/2022] https://www.industrialshields.com/es_ES/blog/blog-industrial-open-source-1/post/libreria-modbus-rtu-master-para-automatizacion-industrial-200
Engineer Ambitiously [Consulta 14/11/2022] https://www.ni.com/es-es/innovations/white-papers/14/the-modbus-protocol-in-depth.html
Wikipedia [Consulta 25/10/2022] https://es.wikipedia.org/wiki/Microcontrolador
Microcontroladores [Consulta 15/10/2022] https://microcontroladoressesv.wordpress.com/empresas-fabricantes-de-microcontroladores/
Mouser Electronics [Consulta 15/10/2022] https://www.mouser.es/c/semiconductors/embedded-processors-controllers/microcontrollers-mcu/8-bit-microcontrollers-mcu/?marcom=188567174&mounting%20style=Through%20Hole&number%20of%20i%2Fos=12%20I%2FO&program%20memory%20size=28%20kB&supply%20voltage%20-%20max=5.5%20V
RS-Online [Consulta 15/10/2022] https://es.rs-online.com/web/c/semiconductores/circuitos-integrados-de-control-de-alimentacion/convertidores-buck/
Geocities [Consulta 08/12/2022]https://www.geocities.ws/it2n/runlit.html
Andreita y Cesar [Consulta 08/12/2022] http://andre-cesar2digital.blogspot.com/2010/09/
https://www.electronics-tutorials.ws/blog/christmas-lights-sequencer-circuit.html
Microcontroller Lab [Consulta 09/01/2023] https://microcontrollerslab.com/ws2812b-rgb-led-pinout-working-interfacing-arduino-applications/
Microchi [Consulta 10/01/2023] https://www.microchip.com/en-us/product/PIC16F1615

12. ANEXOS

12.1 ANEXO I




U1
UDL2981
U2
4017
U3
UDL2981
U4
4017
J2
Conn_01x16
STCP
Reset
R1
33K
R2
33K
D97
1N4148
U5
74HC595
U6
74HC595
Data
clock
SRCLK
SRCLR
RCLK
OE
SER
PWM
R3
R4
R5
R6
R7
J1
Conn_01x16
J6
Conn_01x05_IN_pines
J4
Conn_01x05_IN
J3
Conn_01x02_OUT
J7
Conn_01x02_IN_pines
J8
Conn_01x02_OUT
J5
Conn_01x05_OUT
VCC
GND
Santiago Javier Álvaro Pelado
Sheet: /
File: Driver_Led.kicad_sch
Title: DRIVER LED
Size: A4
Date: 2022-12-08
Rev:
KiCad E.D.A. kicad (6.0.8)
Id: 1/1


J2
Conn_01x16
J1
Conn_01x16
Santiago Javier Álvaro Pelado
Sheet: /
File: Panel_Led.kicad_sch
Title: Panel LED
Size: A4
Date: 2022-12-08
Rev:
KiCad E.D.A. kicad (6.0.8)
Id: 1/1