UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Grado en Ingeniería Electrónica Industrial y Automática

Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

RESUMEN

En este proyecto se exponen y analizan las distintas partes que conforman la aplicación. Dicha aplicación consiste en una matriz led tridimensional compuesta por 512 leds organizados en forma de cubo, la cual será controlada por la placa TMS320F28027 de la familia C2000 de Texas Instruments. Para que la matriz pueda ser manejada por el microcontrolador será necesario diseñar los diferentes circuitos de adaptación para otorgarle la capacidad de controlar dicha cantidad de leds. Estos circuitos se basarán en la alimentación del circuito, el control de la conmutación de las diferentes capas del cubo y el manejo de la información hacia la matriz. El código de control que unirá todos estos circuitos y etapas será programado utilizando lenguaje C. Con el fin de que el sistema tenga una buena apariencia estética, se llevará a cabo el diseño de una carcasa donde se albergará toda la electrónica, de modo que quede oculta. Todo esto proporcionará un resultado final en forma de producto.

Todo el proceso será explicado en los documentos proporcionados a continuación, estos son:

1 - Memoria.
2 - Pliego de condiciones.
3 - Presupuesto.
4 - Documentación del código.
5 - Esquemas eléctricos.
6 - Planimetría.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Grado en Ingeniería Electrónica Industrial y Automática

1 - Memoria

Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Objetivo

El objetivo de este documento es mostrar las directrices que han sido seguidas para desarrollar la implementación de una matriz LED tridimensional, concretamente, formada por 8 leds en el eje “X”, 8 leds en el eje “Y” y 8 leds en el eje “Z”. Dicha matriz será controlada mediante el DSC TMS32F28027 de la familia C2000 de Texas Instruments.

Se mostrará cómo se ha llevado a cabo el diseño del circuito electrónico, el diseño de la placa de circuito impreso, la caja en la que se albergará toda la electrónica y el ensamblaje final.

2. Justificación del proyecto

En la actualidad existen diferentes plataformas y fabricantes de microcontroladores, entre los cuales los más conocidos y usados son los kits de experimentación Arduino y los microcontroladores PIC de Microchip. Esto se debe a su precio y su sencillez de uso, así como la cantidad de información existente en la red.

Pero además de estos existen muchas otras plataformas que proporcionan mayor versatilidad que los mencionados anteriormente.

Este proyecto surge como demostración de que con otras placas de experimentación también es posible realizar proyectos de características similares a los que se realiza con los microcontroladores ya mencionados.

La placa utilizada en este caso será el kit de experimentación proporcionado por Texas Instruments basado en el microcontrolador TMS320F28027 de la familia C2000. Ésta se encargará de ejecutar el algoritmo de control diseñado para el encendido de una matriz led tridimensional ejecutando efectos luminosos.

3. Introducción

3.1. ¿Qué son los microcontroladores?

Una pieza clave del proyecto es el microcontrolador. Consiste en un circuito integrado el cual puede ser programado para ejecutar diferentes tareas registradas en su memoria. Se trata de dispositivos similares a los ordenadores personales, ya que disponen de las tres principales unidades funcionales de estos (Unidad central de procesamiento, memoria y periféricos de entrada/salida. Aunque, además, disponen de periféricos que les permiten realizar tareas más específicas, como Timers, periféricos de comunicación (UART, USART, SPI, I2C, CAN, etc), convertidores A/D, etc.

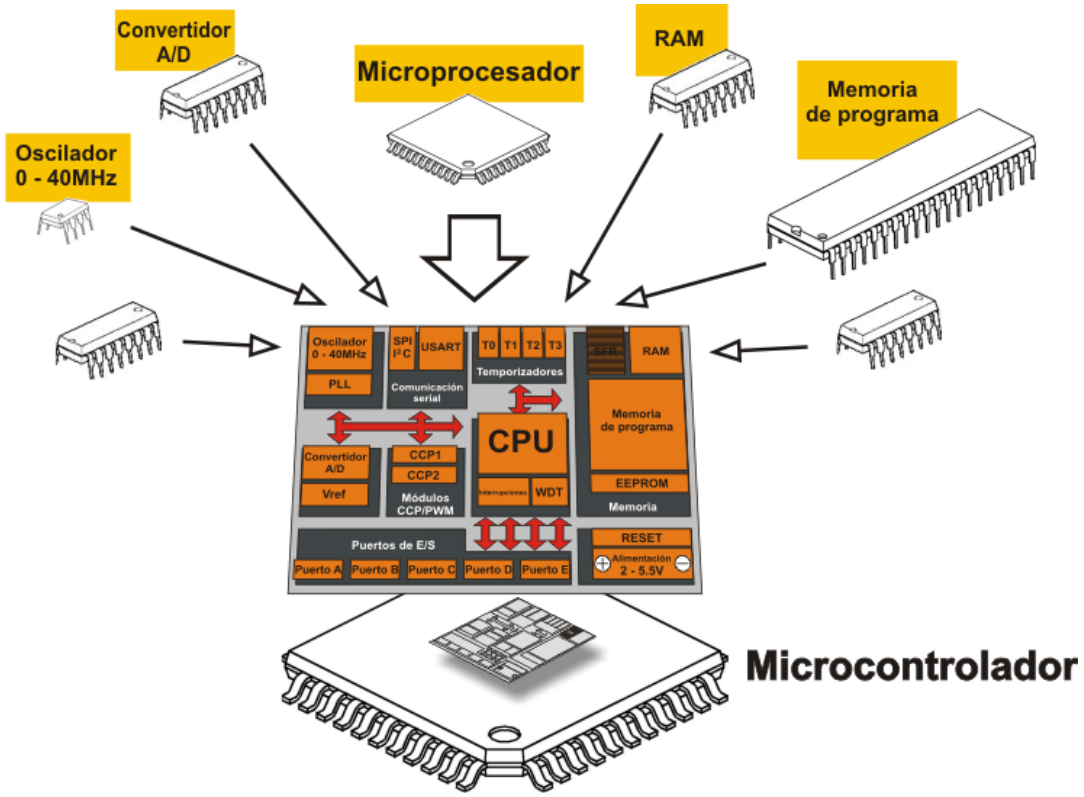


Figura 1: Ejemplo de diagrama de bloques de un microcontrolador, ya que puede diferir entre diferentes modelos.

A diferencia de los ordenadores personales, estos circuitos son utilizados para aplicaciones que requieren menor tamaño de procesamiento que un ordenador personal.

Las características de los microcontroladores son muy variadas, y, dependen de la aplicación en la que vayan a ser usados. En la siguiente tabla se puede observar las diferentes familias que destacan según el número de bits utilizados por su procesador.

Empresa	8 bits	16 bits	32 bits
Atmel	AVR	-	SAM7 SAM3 SAM9 AVR32
Microchip	Familia 10f2xx Familia 12Cxx Familia 16Fxx Familia 18Fxx	PIC24F PIC24H dsPIC33F	PIC32
NXP Semiconductors	80C51	XA	Cortex-M3 Cortex-M0 ARM7 ARM9
Texas Instruments	TMS370	MSP430	C2000 Cortex-M3 TMS570

Tabla 1: Microcontroladores más utilizados clasificados por empresa y número de bits del microprocesador.

3.1.1. Aplicaciones de los microcontroladores

Los microcontroladores están presentes en la mayoría de los dispositivos que son utilizados en la vida diaria (Teléfonos móviles, hornos, despertadores, secadores de pelo, automóviles, etc.), facilitando la realización de diferentes tareas de las personas.

Como se ha dicho anteriormente, cada aplicación dispone de diferentes especificaciones, por lo que usar adecuadamente un tipo de microcontrolador u otro, mejorará las prestaciones de la aplicación, así como reducirá los costes de esta.

Por ejemplo, el control de un electrodoméstico sencillo como una batidora, utilizará un procesador pequeño (4 u 8 bits), mientras que, en un reproductor de música o vídeo, debido a las características de los archivos que ejecutará, requerirá un procesador de 32 o 64 bits.

3.2. ¿Qué es una matriz LED?

Las matrices de leds consisten en circuitos formados por un conjunto de diodos led dispuestos de tal manera que forman pantallas en las que se pueden representar diferentes imágenes o formas gracias a la iluminación adecuada de cada uno de éstos.

Estas matrices o pantallas suelen estar formadas por una gran cantidad de leds, para poder formar letras que se visualicen de forma clara y sencilla, los cuales se controlan por medio de microcontroladores y circuitos intermedios. Dependiendo de la cantidad leds que se usen, el número de entradas/salidas del microcontrolador deberá ser mayor o menor.

Uno de los tamaños que más se suele utilizar para la representación de letreros en 2 dimensiones, es una matriz de 7 filas por 80 columnas, permitiendo escribir unas 14 o 16 letras de 7 “píxeles” de altura. Esto hace un total de 560 leds, los cuales no pueden ser controlados directamente por el microcontrolador. Debido a esto, para llevar a cabo su control se utilizan circuitos intermedios basados en la multiplexación, lo cual permiten encender mayor cantidad de leds con una cantidad razonable de pines del microcontrolador.

Este tipo de dispositivos se utiliza en gran cantidad de situaciones en las que se quiere proporcionar información sobre algún establecimiento, trayecto de un vehículo o simplemente para entretener.

Figura 2: Ejemplo de aplicación de una pantalla led para mostrar el trayecto en el transporte público.

Además de las pantallas en 2 dimensiones, también existen matrices led formando figuras tridimensionales, ya sea por la geometría de la pantalla o por su método de mostrar las imágenes. En esta parte se pueden destacar los cubos de leds o las pantallas giratorias.

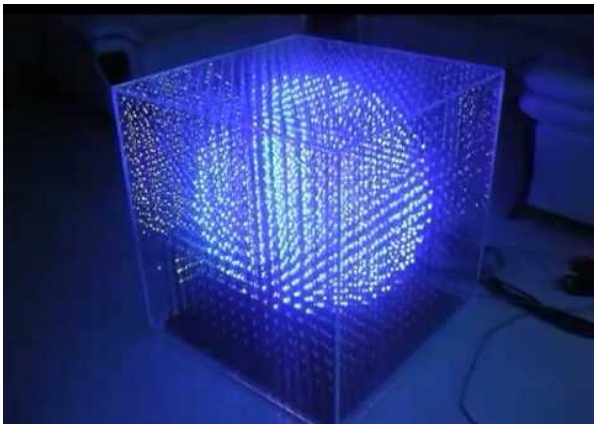


Figura 3: Ejemplo de matriz led tridimensional.

Figura 4: Ejemplo de matriz led giratoria.

3.2.1. Principio de funcionamiento de una matriz tridimensional

En una pantalla de estas características resulta muy difícil que todos los leds se mantengan encendidos constantemente, por eso, para representar los diferentes efectos sobre la matriz, se utiliza el fenómeno conocido como persistencia de la visión.

Este fenómeno demuestra como una imagen permanece en la retina humana una décima de segundo, antes de desaparecer, permitiendo de esta manera que se pueda ver la realidad como una secuencia de imágenes ininterrumpidas.

Este fenómeno se aplica en dicha aplicación mediante el encendido y apagado de los diferentes leds, a una velocidad más rápida de la que el ojo humano pueda percibirlo, de modo que se muestren las imágenes deseadas en movimiento. Es decir, nunca se tendrá encendido al mismo tiempo todos los leds, sino que parpadearán a una velocidad tan alta que permanecerán en la retina del receptor simulando que se encuentran encendidos al mismo tiempo.

4. Descripción general de la aplicación

La aplicación definida consiste en una matriz tridimensional en forma de cubo constituida por 64 leds en cada una de las capas hasta formar 8. Con esto se consigue un cubo formado por 512 leds en total. Dicha matriz se podrá utilizar para visualizar imágenes en 3 dimensiones, pero, debido al tamaño de la pantalla, se tratará de imágenes de baja calidad.

El cubo led estará formado por 8 capas de 64 leds. Cada una de las capas se conecta con la anterior mediante la unión de sus ánodos, y los 64 leds que forman cada capa se unen entre sí mediante los cátodos. Esta configuración es utilizada para reducir el número de salidas digitales necesarias para controlar la luminaria, ya que, conectando cada led por separado se necesitaría un total de 512 salidas.

En la figura 5 y en la figura 6 se puede apreciar cómo se han realizado las conexiones de las capas. Para que resulte más fácil de comprender, se ha utilizado un cubo de dimensiones reducidas.

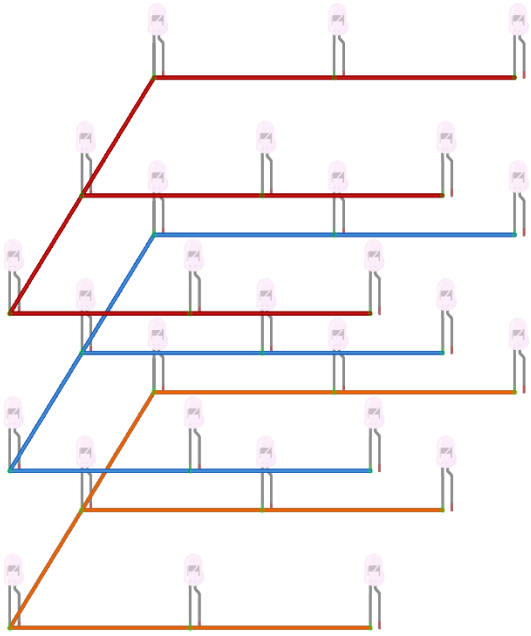


Figura 5: Muestra de cómo se deben encontrar conectados los cátodos. Cada color representa el cableado de una capa.

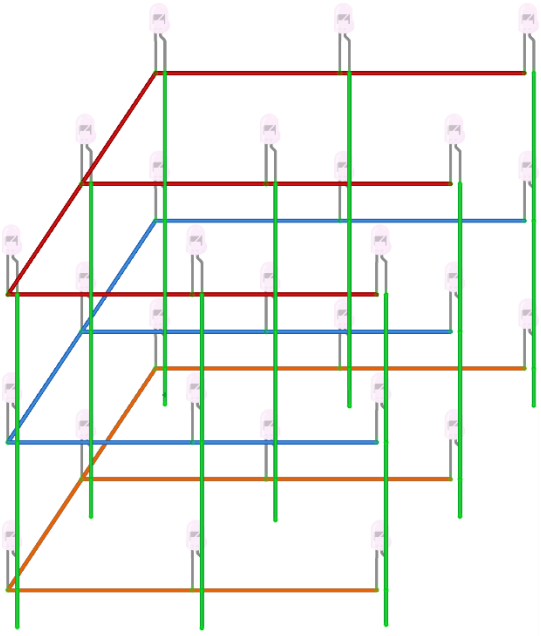


Figura 6: El cableado verde muestra cómo deben ir conectados los ánodos de las diferentes capas.

Para controlar los diodos se necesita controlar tanto el cátodo como el ánodo. En este caso se tienen todos los cátodos de una misma capa conectados entre sí, por lo que bastará con utilizar un único pin por cada capa que dispongamos, en este caso 8 salidas digitales.

De la misma manera, los ánodos de las diferentes capas se encuentran conectados entre sí, en este caso se necesitará un pin digital para controlar cada columna del cubo. Concretamente se necesitará usar un total de 64 salidas digitales.

Teniendo en cuenta las salidas necesarias para manejar las capas y las columnas, se deberían tener un total de 72 salidas. Este número sigue siendo una cantidad excesiva de salidas, por lo que será necesario incorporar un circuito decodificador encargado de manejar todas estas salidas.

De modo que no va a ser posible manejar todos los leds de golpe, el software realizará un control de las salidas basado en la multiplexación y persistencia de la visión.

5. Soluciones alternativas

Antes de la selección de la solución final, existen diferentes alternativas que han sido evaluadas para su incorporación.

5.1. Microcontroladores

Existen multitud de microcontroladores útiles para esta aplicación en el mercado. Como se ha nombrado en apartados anteriores, otra buena solución es la utilización de un Arduino. Este en sí mismo es una placa de experimentación, pero hace referencia a microcontroladores de la familia ATMEL AVR. Otro tipo de microcontroladores que se podría haber utilizado son los PIC de Microchip.

Para esta aplicación cualquier tipo de microcontrolador sería adecuado, siempre y cuando disponga de la cantidad mínima necesaria de pines de entrada/salida.

Estos microcontroladores han sido descartados debido a su expansión en el mercado, ya que se trata de componentes muy utilizados.

5.2. Transistores de conmutación

Como se verá en apartados posteriores, en el circuito, existe una parte dedicada a la conmutación de transistores para el encendido de las diferentes capas. Esta etapa se puede generar mediante transistores bipolares (BJT) o transistores de efecto de campo (MOSFET).

Los transistores BJT son semiconductores controlados por la corriente que circula por su base. Este factor es importante, ya que los microcontroladores no son capaces de entregar grandes cantidades de corriente (suelen entregar un máximo de 20 mA).

Un ejemplo de utilización de transistor bipolar para esta aplicación sería la utilización del transistor 2N2222, un transistor tipo NPN. Se ha descartado la utilización de este componente debido a que no es capaz de soportar la corriente circundante, por lo que se debería añadir una línea idéntica en paralelo de modo que la corriente se divida. Esto provoca un incremento del número de componentes, y, por lo tanto, un mayor tamaño del circuito.

5.3. Registros de desplazamiento

La etapa de multiplexación se podría haber basado en registros de desplazamiento con entrada serie y salida paralelo como puede ser la utilización del integrado 74HC595. Esto ha sido rechazado debido a que resultaba más visible la programación con un registro de desplazamiento paralelo-paralelo.

5.4. Matriz LED

Para la elaboración de la matriz existen diferentes alternativas a considerar:

- Color: existen diodos LED de todos los colores, rojos, azules, amarillos, blancos, etc. Además, también existen LEDs multicolores dependiendo de la corriente que circule por sus pines.
- Tamaño: los diodos LED a considerar son de 3 o 5 mm.
- Forma: la matriz no tiene por qué formarse de manera uniforme, se puede disponer de diferentes tamaños para los diferentes ejes, formando estructuras geométricas como cubos, pirámides o formas más irregulares.

6. Solución adoptada

Como se ha visto en el apartado de soluciones alternativas, existen diferentes soluciones para elaborar el mismo tipo de aplicación. En este caso se explicarán las diferentes soluciones que se han adoptado.

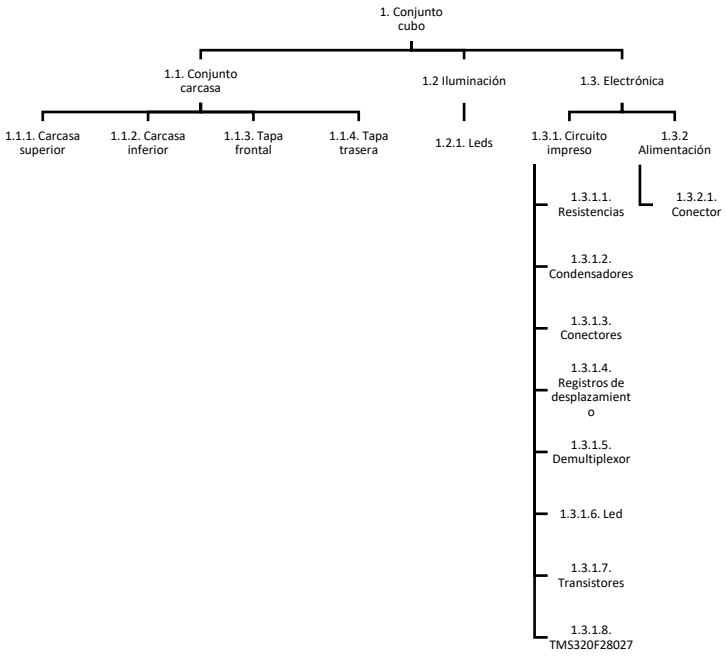


Figura 7: Organigrama técnico

En el organigrama técnico de la figura 7 se puede observar a rasgos generales la solución que se implementará. El sistema se dividirá en 3 subsistemas principales:

- 1.1. Conjunto carcasa: se ha diseñado una carcasa modular constituida por 4 piezas diferentes con el fin de facilitar el acceso a su interior.
- 1.2. Iluminación: Formada por los leds dando forma de cubo a la matriz a controlar.
- 1.3. Electrónica: a su vez lo constituyen 2 subsistemas:
    - 1.3.1. Circuito impreso: formado por todos los componentes que harán funcionar la aplicación.
    - 1.3.2. Alimentación: esta parte consta de un conector a través del cual se alimentará el circuito mediante una fuente de alimentación de 5 V capaz de proporcionar un mínimo de 1,5 A.

La selección de componentes se ha realizado tomando como eje de elección el kit de experimentación LaunchXL-F28027. Como éste solo dispone de un máximo de 22 entradas/salidas digitales. Por esta razón se hace imposible controlar el número de salidas comentado en el apartado anterior.

Para ello, el circuito contará con una etapa encargada de controlar todo mediante software (el microcontrolador), una etapa multiplexora encargada de manejar los 64 ánodos de los leds, una etapa decodificadora que reducirá el número de entradas totales a una cantidad de pines manejables por el microcontrolador y, finalmente, una etapa encargada de gestionar los cátodos de los leds para el encendido de las diferentes capas.

6.1. Hardware. Diagrama de bloques

En la figura 8 se puede ver una representación simplificada de cómo interactúan las diferentes etapas del circuito, antes descritas, entre sí.

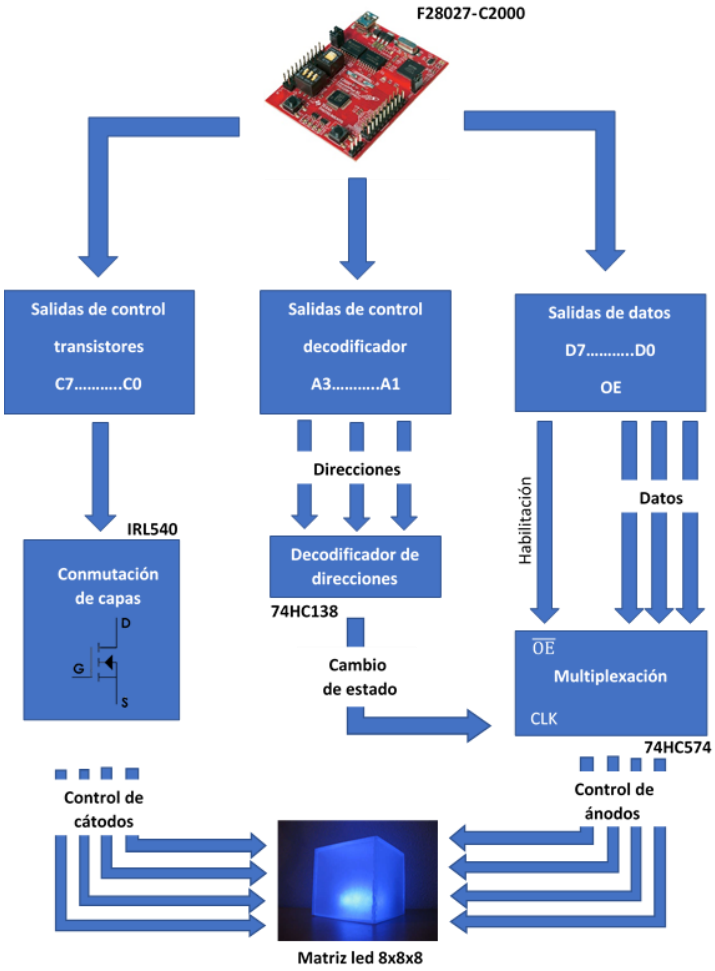


Figura 8: Diagrama de bloques funcional

6.1.1. Matriz LED

Atendiendo a las soluciones alternativas antes mostradas, se ha decidido otorgar a la matriz LED las siguientes características:

- Color: se utilizarán LEDs de color azul debido a su tensión umbral.
- Tamaño de los LED: para evitar un aspecto sobrecargado se utilizarán LEDs de 3 mm.
- Tamaño de la matriz: se usará una matriz en forma de cubo con 8 LEDs por eje, es decir, un total de 512 LEDs formando una matriz cúbica.

6.1.2. Salidas F28027-C2000

Como ya se ha mencionado con anterioridad, el microcontrolador es el encargado de controlar todo el sistema, para ello utilizará sus diferentes pines de entradas y salidas. En este caso en particular, solo se utilizarán dichos pines configurados como salidas, ya que no se han utilizado entradas para modificar el comportamiento.

En la tabla 2 se puede observar la relación existente entre la salida de datos (D0-D7) y las salidas utilizadas del microcontrolador. Estas salidas serán las encargadas de controlar la coordenada X de la matriz de leds.

Nombre	F28027	Descripción
D0	GPIO 0	Coordenada X en la posición 0
D1	GPIO 1	Coordenada X en la posición 1
D2	GPIO 2	Coordenada X en la posición 2
D3	GPIO 3	Coordenada X en la posición 3
D4	GPIO 4	Coordenada X en la posición 4
D5	GPIO 5	Coordenada X en la posición 5
D6	GPIO 6	Coordenada X en la posición 6
D7	GPIO 7	Coordenada X en la posición 7
OE	GPIO 34	Habilitación de los 74HCT574 Se habilita con lógica negada

Tabla 2: Descripción de las salidas de datos del F28027

Para controlar las direcciones del decodificador se utilizan 3 salidas digitales del F28027, las cuales corresponden con los pines A3, A2 y A1 del integrado 74HCT138. En la tabla 2 se expresa la relación entre ambos componentes.

Nombre	F28027	Descripción
A1	GPIO 16	Se encargan de seleccionar las diferentes salidas del decodificador
A2	GPIO 17	
A3	GPIO 18	

Tabla 3: Descripción de las salidas de control del decodificador 74HCT138

Finalmente se encuentran las salidas encargadas de gestionar la conmutación de las diferentes capas que componen el cubo, es decir, cada salida corresponderá directamente con la coordenada Z de la matriz de leds. En la tabla 4 se puede ver detallada esta relación.

Nombre	F28027	Descripción
C0	AIO 6	Coordenada Z en la posición 0
C1	GPIO 28	Coordenada Z en la posición 1
C2	GPIO 29	Coordenada Z en la posición 2
C3	AIO 12	Coordenada Z en la posición 3
C4	AIO 4	Coordenada Z en la posición 4
C5	AIO 14	Coordenada Z en la posición 5
C6	AIO 2	Coordenada Z en la posición 6
C7	AIO 10	Coordenada Z en la posición 7

Tabla 4: Descripción de las salidas de control de los transistores encargados de manejar las capas

6.1.3. Multiplexación y decodificador de direcciones

En este apartado se abordan los bloques de multiplexación y decodificación al mismo tiempo, ya que son dos bloques van relacionados directamente entre sí.

Como se ha visto el diagrama funcional, el microcontrolador es el encargado de controlar los datos que se le introducen a la etapa de multiplexación, así como su habilitación. Pero es el decodificador quien le proporciona la señal la señal necesaria para que cambie de estado.

6.1.3.1. Selección de componentes

6.1.3.1.1. 74HCT574

Para la implementación de la etapa de multiplexación se han elegido los integrados 74HCT574, los cuales consisten en registros de desplazamiento que cambian de estado con los pulsos positivos de la señal de reloj.

Estos integrados se encuentran disponibles en ambas tecnologías (TTL y CMOS). Como el microcontrolador utiliza niveles lógicos entre 0 y 3.3 V, para nivel bajo y alto, respectivamente, se han elegido los modelos de la familia TTL, ya que otorgan un mayor rango de trabajo. En la figura 9, se puede observar como los niveles TTL son más adecuados para trabajar con el F28027.

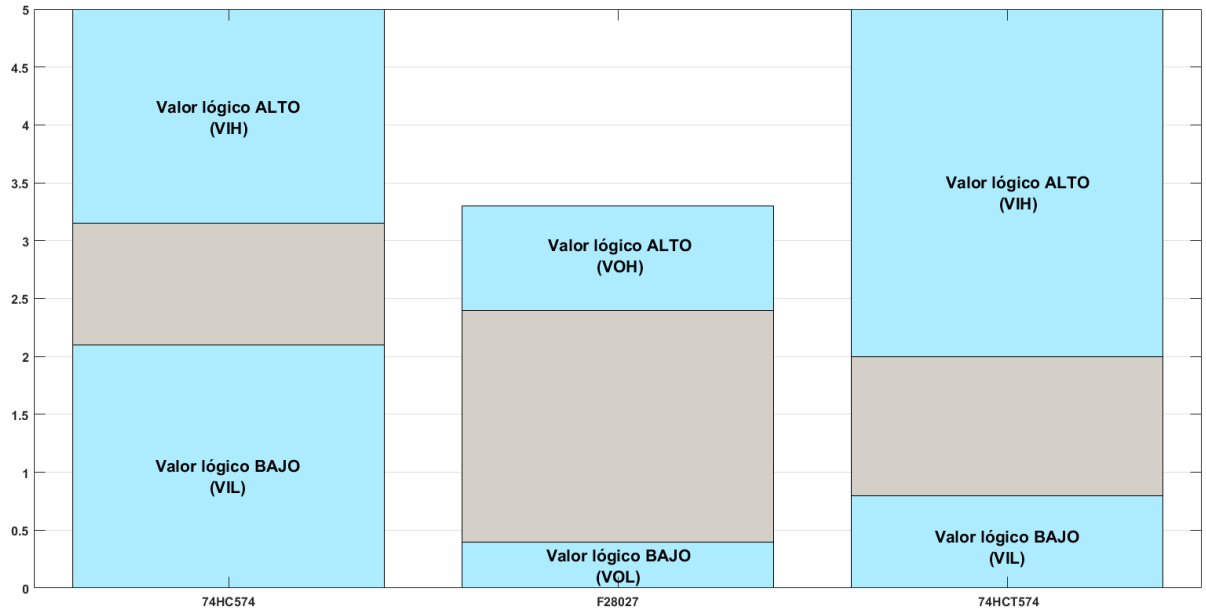


Figura 9: Comparación entre los valores lógicos de las familias TTL Y CMOS con la salida del launchpad F28027. A la izquierda se puede observar la familia CMOS y a la derecha la familia TTL. La columna central hace referencia a los valores de salida del microcontrolador.

Como se ha podido demostrar con la figura 4, los valores de salida del microcontrolador se encuentran completamente dentro del rango de detección de la familia TTL. En cambio, si comparamos con la familia CMOS se podrían dar casos erráticos debido a malas detecciones de un nivel lógico alto.

6.1.3.1.2. 74HCT138

Se trata de un decodificador de 8 bits, el cual selecciona una de sus 8 salidas de acuerdo con la combinación que se disponga a su entrada. De la misma manera que en el apartado anterior, la tecnología TTL confiere mejores rangos de trabajo.

6.1.3.1.3. Resistencias

Las resistencias utilizadas son de un valor de 100 Ω. Este valor se obtiene teniendo en cuenta la corriente máxima que puede suministrar el integrado 74HCT574 y la corriente a la que se quiere limitar el diodo LED.

Como el integrado alimentará el diodo con 5 V y la corriente a la que se quiere limitar el LED es de 20 mA, con una tensión de codo de 2.2 V, aplicando la ley de ohm se tiene que:

$$V = IR \ (1)$$

$$5 - 2.8 = 20R \ (2)$$

$$\boldsymbol{R = 110\ \Omega}\ \boldsymbol{(3)}$$

Con la expresión (3) se puede deducir que las resistencias a utilizar serán de 110 Ω. Al no tratarse de un valor normalizado, se escogerá la resistencia normalizada inmediatamente inferior, la cual es de 100 Ω.

La etapa de multiplexación está constituida por 8 circuitos integrados (74HCT574), cada uno corresponde con una coordenada Y de la matriz tridimensional. Éstos son los encargados decir que leds se deben encender gracias a las señales D0-D7 proporcionadas por el microcontrolador (Coordenada X). Todos ellos utilizan las mismas señales de entrada, de este modo se reduce de 64 a 9 (D0-D7 más OE) el número de salidas digitales utilizadas.

Los 74HCT574 trabajan mediante una señal de reloj (CLK). Por cada pulso de subida que reciben, los datos que tienen en las patillas de entrada pasan a la salida. En este punto es donde el decodificador ejerce su función.

Las señales CLK de los 75HCT574 se encuentran conectadas a las patillas de salida del 74HCT138 (Señales Y0-Y7). Este es el encargado de generar los flancos de subida para que se produzca el cambio de estado antes mencionado.

En la figura 9 se puede observar más en detalle cómo es la interacción entre ambas etapas.

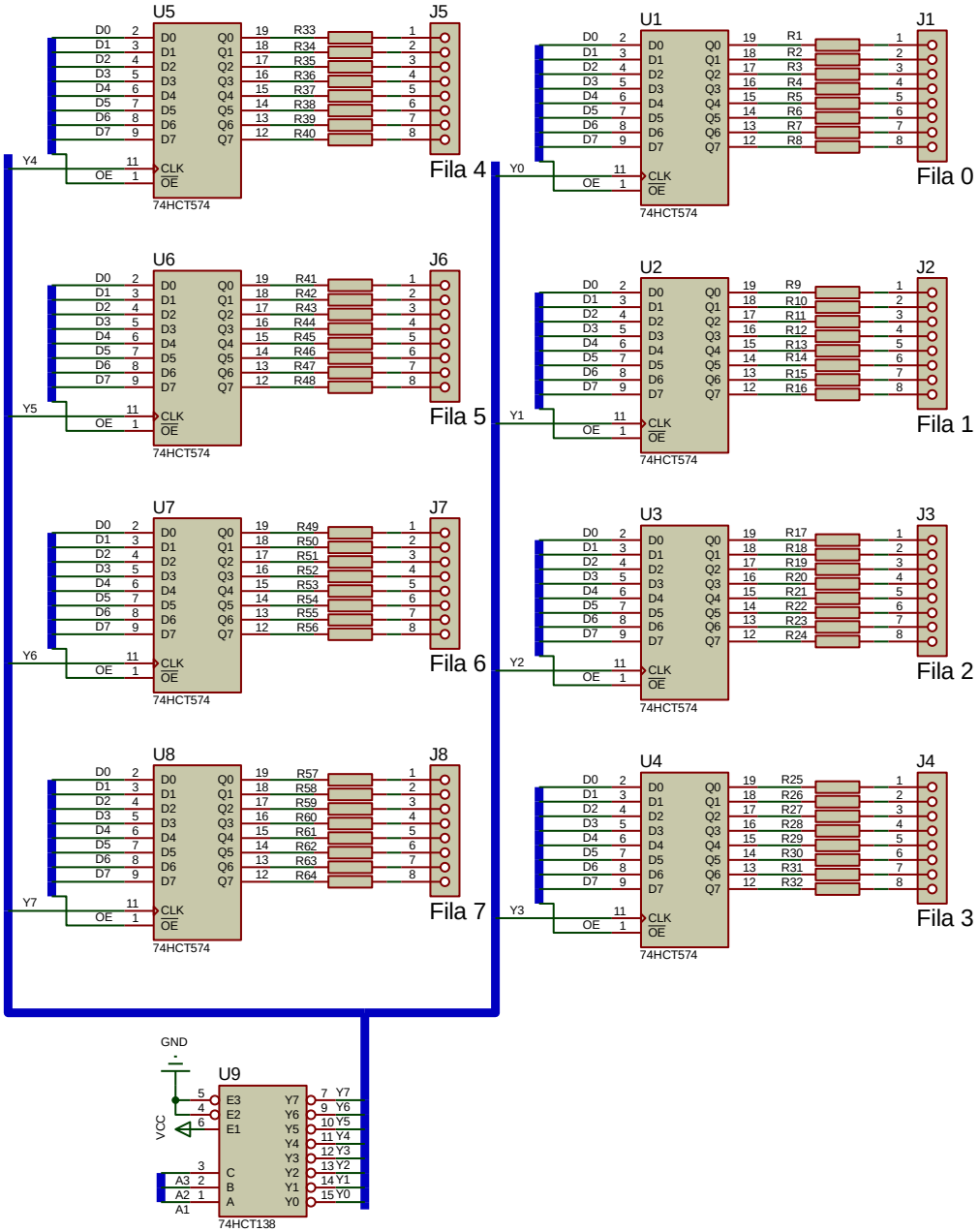


Figura 10: Esquema detallado de las conexiones entre el decodificador y la etapa de multiplexación

Mediante el software se controlan las entradas del decodificador siguiendo la tabla de verdad que se puede observar en la figura 11.

Control			Input			Output							
$\overline{E1}$	$\overline{E2}$	E3	A2	A1	A0	$\overline{Y7}$	$\overline{Y6}$	$\overline{Y5}$	$\overline{Y4}$	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	–	–	–	–	–	–	–	–	–	–	–
X	X	L	–	–	–	–	–	–	–	–	–	–	–
L	L	H	–	–	–	–	–	–	–	–	–	–	–
–	–	–	L	L	L	H	H	H	H	H	H	H	L
–	–	–	L	L	H	H	H	H	H	H	H	L	H
–	–	–	L	H	L	H	H	H	H	H	L	H	H
–	–	–	L	H	H	H	H	H	H	L	H	H	H
–	–	–	H	L	L	H	H	H	L	H	H	H	H
–	–	–	H	L	H	H	H	L	H	H	H	H	H
–	–	–	H	H	L	H	L	H	H	H	H	H	H
–	–	–	H	H	H	L	H	H	H	H	H	H	H

Figura 11: Tabla de verdad del integrado 74HCT138

A pesar de que se verá más en detalle en los apartados siguientes, en la siguiente imagen se puede ver como el microcontrolador pasa la información a las líneas de datos y, además, como se maneja la tabla de verdad del 74HCT138.

```
//Actualización de la información en el BUS de datos (D0-D7)
for(i=0;i<8;i++)
{
    //Pasa la información del buffer al bus de datos
    GpioDataRegs.GPADAT.all=cubo[capa_actual][i];
    //Decodificación de las señales A3-A1
    GpioDataRegs.GPADAT.all=(GpioDataRegs.GPADAT.all & 0x000000FF)|((0x07 & ((Uint32)i+1))<<16);
}
```

Figura 12:Código que maneja los datos y el decodificador

6.1.4. Conmutación de capas

Esta etapa se encarga de encender la capa que se desea. Para ello utiliza una serie de transistores MOSFET trabajando entre región óhmica y corte, de modo que, cuando el microcontrolador saque un nivel alto en la patilla de la capa deseada, los cátodos de esta capa pasen a estar a masa (GND), haciendo que se enciendan los leds.

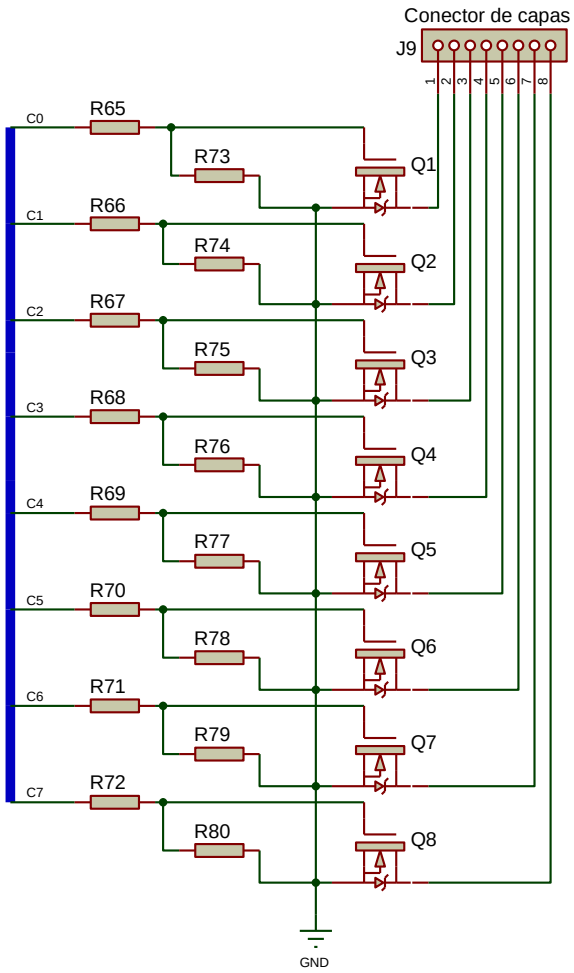


Figura 13: Esquema detallado del encendido de las capas

Los transistores MOSFET son componentes controlados por tensión, por lo que no demanda corrientes elevadas a los pines de salida del microcontrolador, debido a su gran impedancia en su terminal de puerta (“G”).

Se colocan resistencias de pull-down con el fin de garantizar que la puerta del transistor se encuentra a un nivel lógico bajo, cuando la patilla del microcontrolador está sacando un “0”.

6.1.4.1. Selección de transistor MOSFET

Para que los MOSFET trabajen entre los modos nombrados en el apartado anterior, se debe cumplir:

1. Corte:

$$V_{GS} < V_{GS(th)} \ (4)$$

2. Región óhmica:

$$\begin{cases} V_{GS} > V_{GS(th)} \\ V_{GS} - V_{GS(th)} \geq V_{DS} \geq 0 \end{cases} (5)$$

El transistor elegido para llevar a cabo esta función ha sido el IRL540N, el cual una tensión umbral máxima ($V_{GS(th)}$) de 2 V, y es capaz de soportar una corriente máxima de 28 A.

Para conocer si estas condiciones se cumplen, a continuación, se van a realizar los cálculos basados en un modelo de lo que sería una capa conectada al terminal drenador ("D") del MOSFET (Figura 13). Para realizar dichos cálculos se tiene que:

- Cada capa está formada por 64 LEDs.
- Consumo de cada LED =20 mA, tensión forward = 2.2 V.
- Resistencias asociadas a cada LED = 100 Ω.

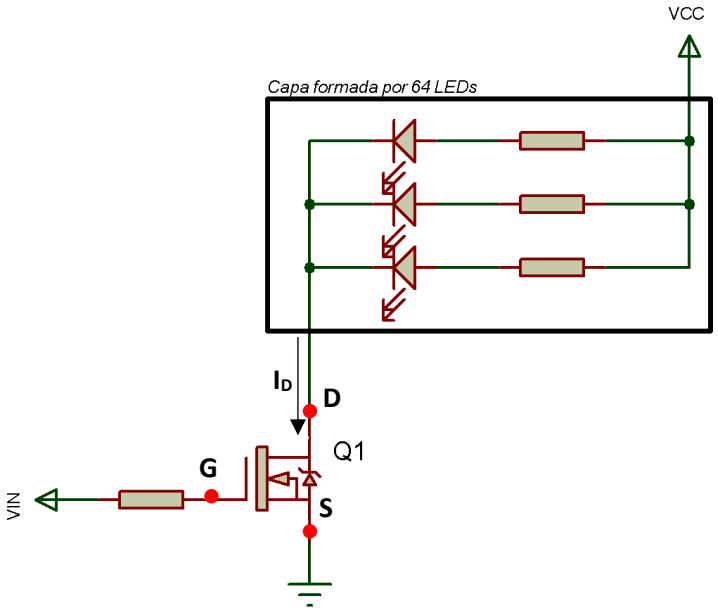


Figura 14: Esquema equivalente a la conexión de una capa al MOSFET

Al tratarse de 64 LEDs conectados en paralelo con un consumo de 20 mA cada uno, significa que por el drenador del MOSFET existirá una corriente máxima de 1280 mA; por lo tanto, la corriente I_D que deberá soportar el MOSFET será mínimo de 1.3 A.

La tensión V_D se calcula a partir de la caída de tensión de los diodos y la resistencia limitadora de corriente asociada, de modo que queda lo siguiente:

$$V_D = 5\,V - (20\,mA \cdot 100\,\Omega) - 2.8\,V \quad (6)$$

$$V_D = 0.2\,V \quad (7)$$

De las hojas de características del MOSFET se puede extraer el valor de la tensión $V_{GS(th)}$, siendo esta de entre 1 y 2 V. Por lo tanto, cuando la tensión VIN valga 0 V, el transistor se encontrará en modo corte, ya que se cumple la ecuación (4).

Para conocer si el transistor entrará en la zona óhmica cuando la tensión VIN valga 3.3 V (Tensión de salida del microcontrolador para un nivel lógico alto), se deberán cumplir las expresiones de la ecuación (5).

$$V_{GS} = V_G - V_S = 3.3 - 0 = 3.3\,V \quad (8)$$

$$V_{DS} = V_D - V_S = 0.2 - 0 = 0.2\,V \quad (9)$$

Por lo tanto, según las especificaciones del modelo seleccionado, y aplicando la expresión (5):

$$\begin{cases} 3.3\,V > 2\,V \\ 3.3 - 2\,V \geq 0.2\,V \geq 0 \end{cases} \quad (10)$$

Dado que se cumple la expresión (2), el transistor será capaz de entrar a trabajar en la zona óhmica. Pero a pesar de eso, se debe comprobar si es capaz de hacer circular la corriente demandada por la capa. Para ello, se puede extraer de la hoja de características la gráfica que relaciona estas magnitudes.

En la figura 14 se puede observar cómo, con los datos obtenidos, el transistor es capaz de hacer circular hasta un total de 2 A, valor suficiente para manejar los 1.3 A que utiliza cada capa de la aplicación.

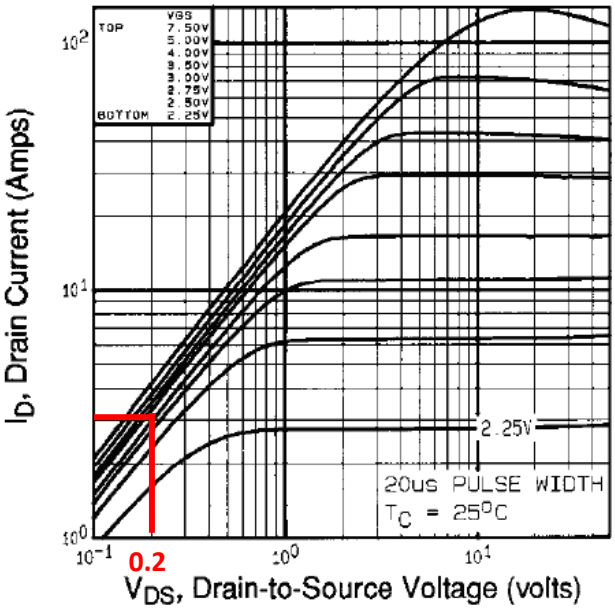


Figura 15: Curva de corriente del transistor MOSFET IRL540N

6.1.4.2. Cálculos térmicos

Al tratarse de una aplicación en la que los transistores se encuentran en continua conmutación (Proceso en el que se desprende mucha energía), conviene determinar si será necesario o no la utilización de un disipador para refrigerar dichos componentes.

Para conocer dicha necesidad, se aplica la ley de Ohm térmica:

$$P_s = \frac{T_j - T_a}{R_{ja}} \ (11)$$

Donde Ps es la potencia disipada; Tj es la temperatura de unión; Ta es la temperatura ambiente y Tja es la temperatura de unión-ambiente (Figura 16).

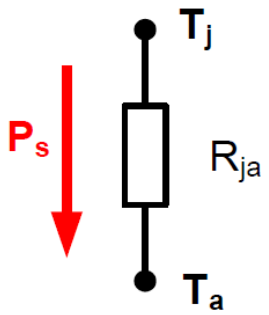


Figura 16: Representación gráfica de la ley de Ohm térmica

Siguiendo la información mostrada en el datasheet (T_j= 175 °C; T_a= 30 °C y R_{ja}= 62 °C/W) y aplicando en la ecuación (11):

$$P_s = \frac{175 - 30}{62} = 2.34\ W \ (12)$$

La potencia máxima que puede disipar cada transistor, sin necesidad de disipador, es de 2.34 W. De acuerdo con esto, se debe calcular la potencia real disipada por cada MOSFET en la aplicación.

La figura 17 es la representación de las formas de onda generadas por la tensión V_{DS}, la corriente I_D y la potencia disipada por el transistor. Como se puede observar, en los periodos de conmutación es cuando más potencia se disipa, pero también se disipa una pequeña parte durante el tiempo en el que el transistor está activo. De esta forma la potencia real disipada será la suma de la potencia durante la conmutación y la potencia generada durante la conducción, es decir, se calculará el área encerrada por la curva PMOSFET.

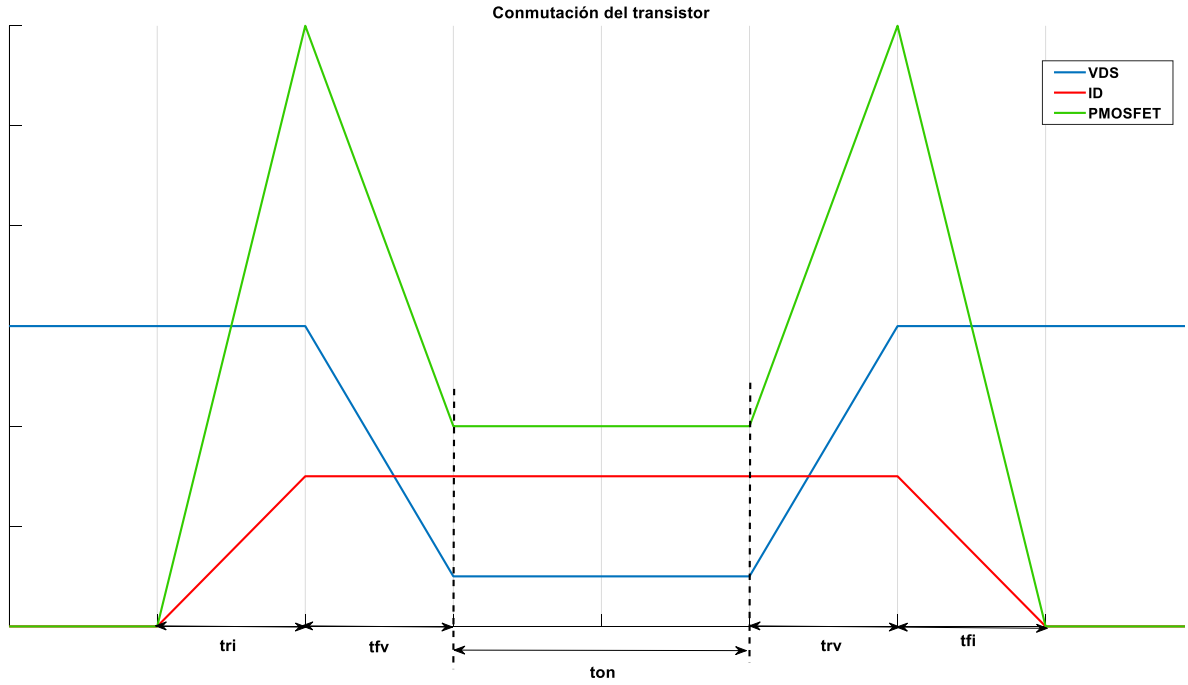


Figura 17: Formas de onda del transistor MOSFET

Teniendo en cuenta que la aplicación está programada para conmutar a una velocidad aproximada de 6 kkHz y que tri=trv=tr y tfi=tfv=tf, siendo tr y tf los tiempos de subida y de bajada respectivamente, extraídos de las hojas de características, se puede calcular lo siguiente:

$$P_{Conm} = \frac{V_{CC}\, I_D}{T}\left[t_r + t_f\right] (13)$$

$$P_{Conm} = 5 \cdot 1.3 \cdot 6000[170 \cdot 10^{-9} + 80 \cdot 10^{-9}] (14)$$

$$P_{Conm} = 3\, mW\ (12)$$

Para calcular la potencia disipada durante la conducción se deben de tener en cuenta el tiempo durante el que el transistor se encuentra activo (t_{on}), la tensión drenador-surtidor durante la conducción ($V_{DS(on)}$), la frecuencia de conmutación y la corriente que circulará (I_D).

$$t_{on} \approx \delta \cdot T\ (15)$$

Siendo el ciclo de trabajo aproximadamente de un 12 %, hallado de forma experimental.

$$t_{on} \approx \frac{0.12}{6000} \approx 20\, \mu s (16)$$

De la misma manera se calcula la tensión $V_{DS(on)}$:

$$V_{DS(on)} = I_D R_{DS(on)} (17)$$

Al trabajar en la región óhmica, la resistencia $R_{DS(on)}$ ha de ser calculada de la siguiente forma:

$$R_{DS(on)} = \frac{1}{k\left(V_{GS} - V_{GS(th)}\right)}\ (18)$$

Siendo k una constante que puede ser determinada utilizando los calores correspondientes a un punto específico de la curva mediante la fórmula:

$$k = \frac{I_D}{\left(V_{GS} - V_{GS(th)}\right)^2} = \frac{1.3}{(3.3-2)^2} = 0.7692 \ (19)$$

Aplicando la expresión (19) sobre la expresión (18), se obtiene la resistencia del transistor:

$$R_{DS(on)} = \frac{1}{0.7692(3.3-2)} = 1\ \Omega \ (20)$$

Y, por lo tanto, aplicando sobre la expresión (17):

$$V_{DS(on)} = \ 1.3 \cdot 1 = 1.3\, V (21)$$

De esta manera, ya se tienen todos los datos para calcular la potencia disipada durante la conducción:

$$P_{Cond} = \frac{1}{T}\left[V_{DS(on)} I_D t_{on}\right] = 6000 \cdot 1.3 \cdot 1.3 \cdot 20 \cdot 10^{-6} = 202.8\, mW \ (22)$$

Finalmente, se obtiene que la potencia total disipada por el transistor durante su operación:

$$P_T = P_{Conm} + P_{Cond} = 3 + 202.8 = 205.8\, mW \ (23)$$

Si se compara con la expresión (9), se puede apreciar que no es necesario colocar ningún disipador a los transistores, ya que la potencia real disipada es mucho menor al límite del transistor.

$$P_S \gg P_T \rightarrow 2.34\, W \gg 205.9\, mW \ (24)$$

Este hecho era de esperar ya que no se trabaja con grandes tensiones ni corrientes en esta aplicación.

6.2. Software. Diagramas de flujo

La aplicación software básicamente se desarrolla a través de tres bloques principales: el programa principal, la interrupción del Timer 0 y el código de los efectos. Como se verá en apartados posteriores, el código se compone de más archivos además de los ya comentados, de manera que se facilite su uso.

Los archivos por los que está compuesto el código completo son:

Nombre del archivo	Extensión	Descripción
Version_1.1	.c	Contiene el programa principal del programa
Init_Func	.c	Contiene las funciones para la inicialización de los periféricos e interrupciones del Launchpad
Init_Func	.h	Contiene las cabeceras de las funciones de inicialización
Draw_Func	.c	Contiene las funciones básicas para llevar a cabo el encendido de los leds
Draw_Func	.h	Contiene las cabeceras de las funciones de dibujo
Effect_Func	.c	Contiene las funciones que desarrollan los diferentes efectos
Effect_Func	.h	Contiene las cabeceras de las funciones de dibujo
GLOBAL_DEFS	.h	Contiene las definiciones a nivel global de proyecto
GLOBAL_VARS	.h	Contiene las variables globales a nivel de proyecto.

Tabla 5:Descripción de los archivos que componen el código

Dentro del fichero “GLOBAL_VARS.h” existen dos variables importantes:

- cubo [8][8]: se trata de una matriz que albergará la información de encendido de todo el cubo entero. Es manejada como un buffer de datos al que se accede para almacenar la información o leer su estado.
- capa_actual: consiste en un contador para llevar la cuenta de la capa en la que se está trabajando.

6.2.1. Programa principal

Este programa es el encargado de llevar a cabo todas las acciones necesarias para la inicialización de las variables, interrupciones, periféricos y salidas necesarias. En la figura 18 se puede observar cómo se desarrollará el flujo de datos de este programa.

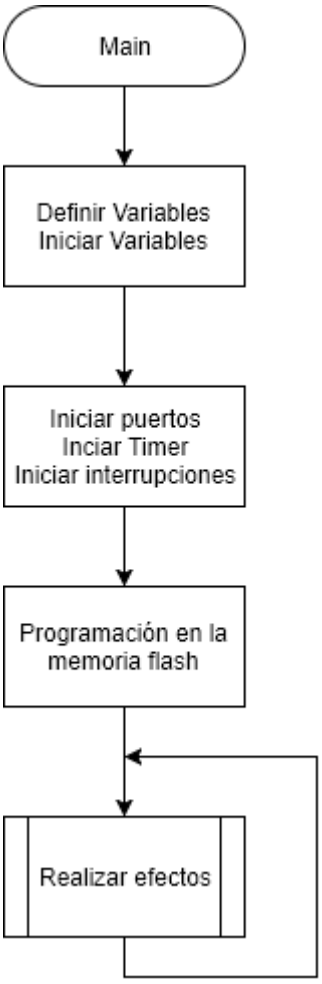


Figura 18: Diagrama de flujo del programa principal

Como se ha podido ver, el programa principal consta de un bucle sin fin el que se ejecutarán los diferentes efectos según la secuencia programada.

Para que a la hora de programar fuera más sencillo reconocer las diferentes partes del código, se han utilizado diferentes archivos para la creación de las funciones de inicialización, definición de variables e inicialización de variables globales. Alguno de estos archivos también se utiliza en para programar la interrupción o los efectos.

Una parte para tener en cuenta a la hora de desarrollar esta parte del código es la programación en la memoria flash. Si no se realiza esta acción, una vez se desconecte el cable USB o se reinicie el Launchpad, éste desaparecerá y dejará de ejecutarse.

6.2.2.Interrupción del Timer 0

Esta puede ser considerada la parte más importante del código. Es la encargada de llevar a cabo el fenómeno de multiplexación. Se encarga de ejecutarse a una frecuencia aproximada de 5,26 kHz, tiempo suficientemente rápido para que el ojo humano retenga la imagen, dando la sensación de que los leds se encuentran encendidos de forma estática, cuando en realidad se encuentran en continuo parpadeo.

Para conocer las acciones desarrolladas por la interrupción se puede observar el diagrama de flujo representado en la figura 19.

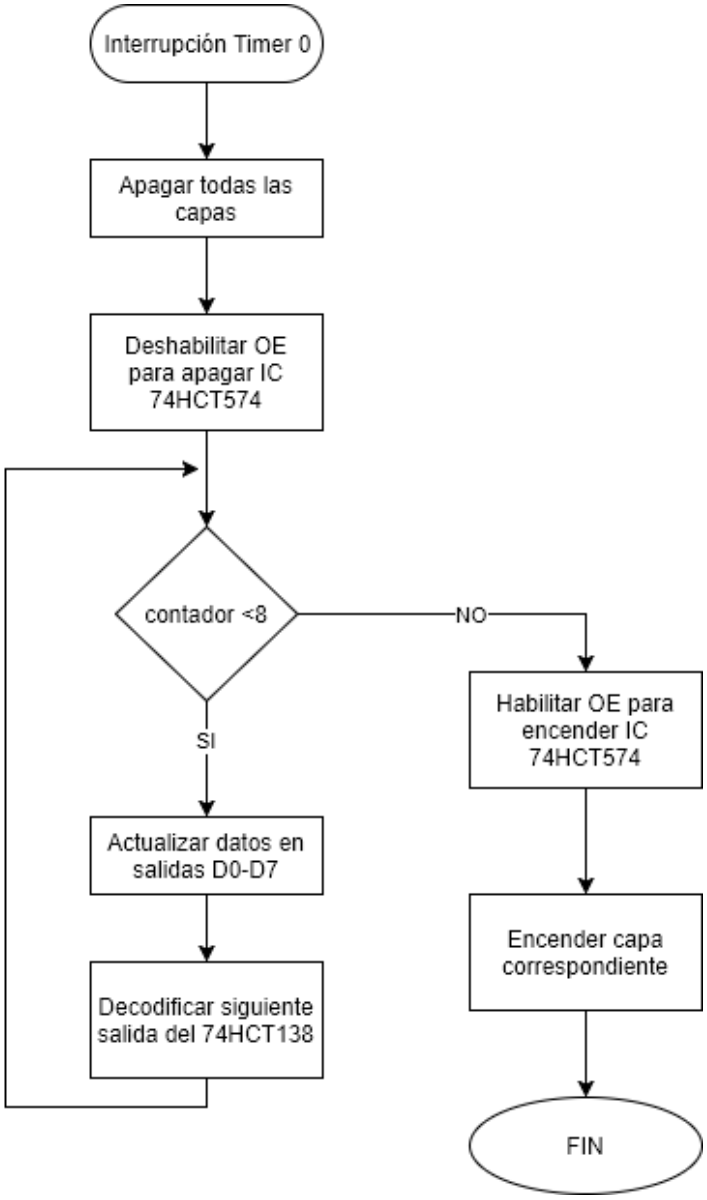


Figura 19: Diagrama de flujo de la interrupción del Timer 0

Para poder entender cómo se produce el encendido de las capas y la actualización de la información en las salidas D0-D7, y cómo la información se almacena en los 74HCT574 de las diferentes filas, se pueden ver los diagramas de tiempo en las figuras 20 y 21.

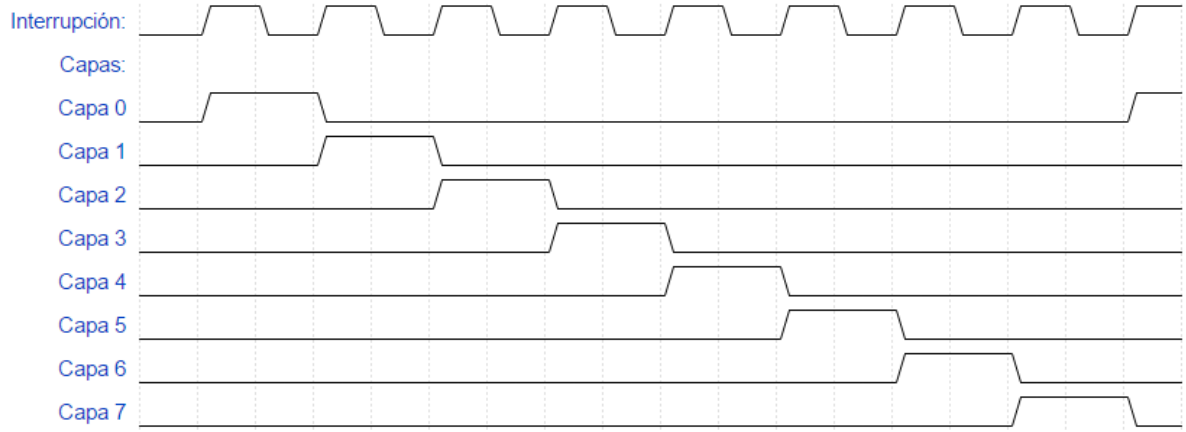


Figura 20: Diagrama de tiempo del encendido de las capas

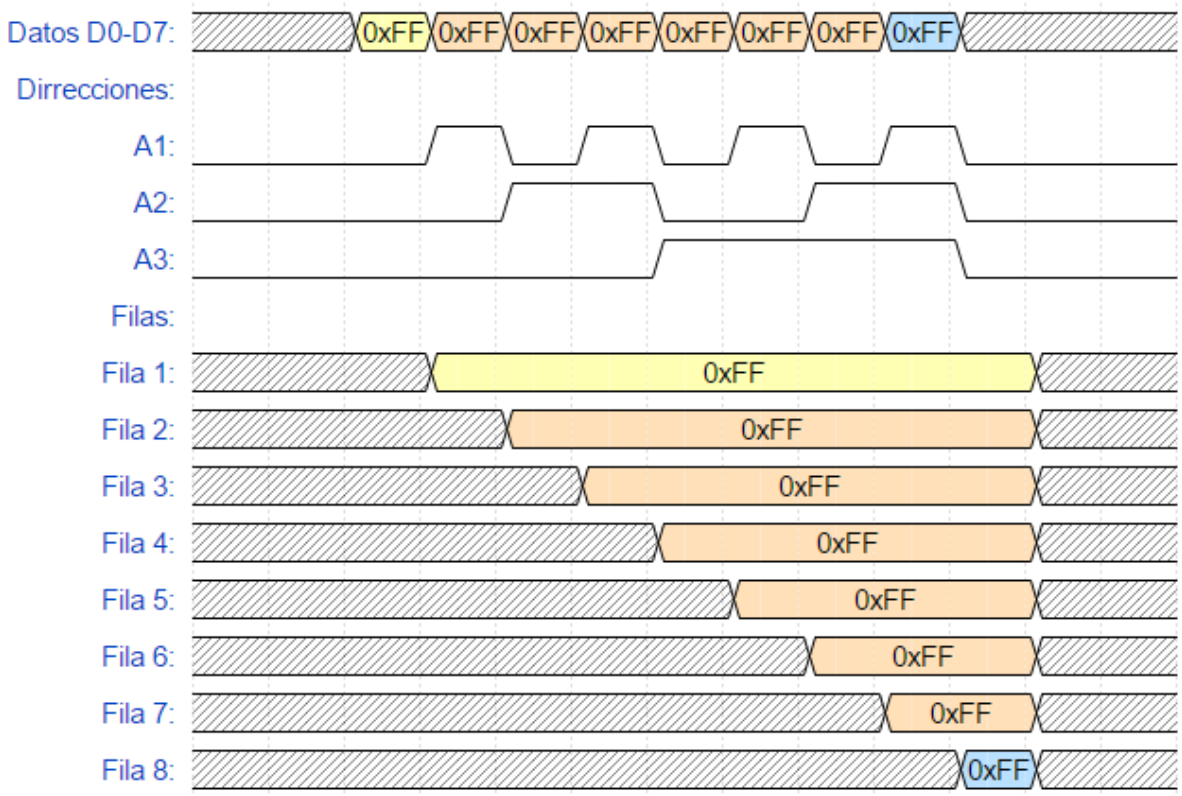


Figura 21: Diagrama de tiempos de la transmisión de datos a los correspondientes 74HCT574

La principal diferencia entre ambos consiste en la base de tiempos. Mientras cada capa se enciende en una nueva ejecución de la interrupción, los datos de encendido se pasan todos durante una única ejecución.

En el diagrama de la figura 16 se muestra un ejemplo para el encendido de toda una capa, los datos serán transmitidos a la variable creada como buffer de datos. Esta variable estará organizada para

albergar la información de las diferentes capas y filas del cubo, de manera que se pueda acceder a ella fácilmente.

Una vez, la información ya se encuentra presente, el programa decodificará la posición a la que esta información va dirigida. Para entenderlo mejor, en la figura 11 se ha mostrado como se ha gestionado este proceso.

Como se puede ver en el diagrama de tiempos de la figura 20, existe un ciclo de retraso entre la decodificación de la dirección y el almacenamiento de la información entre las diferentes filas. Esto es debido a que los integrados 74HCT574 actualizan la información en cada pulso positivo, además, el 74HCT138 activa sus salidas a un nivel lógico bajo.

6.2.3.Programa de efectos

La galería de efectos se basa en la actualización del buffer de datos para el encendido sucesivo de los correspondientes leds, provocando, de esta manera, diferentes animaciones sobre la matriz de leds.

En la figura 22 se puede observar como actuarán los diferentes efectos según el diagrama de flujo base para este programa.

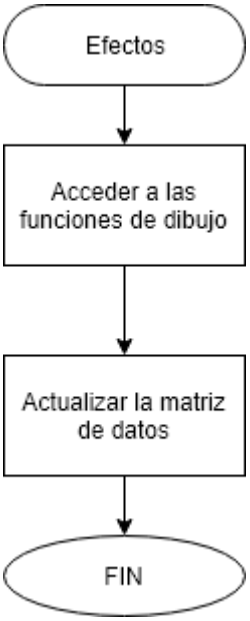


Figura 22:Diagrama de flujo genérico para los diferentes efectos

Para que sea más fácil la programación se ha decidido crear una librería donde se recojan las funciones básicas de dibujo, como puede ser el encendido de un único led, de un plano, invertir la posición de un led, etc.

7. Diseño y fabricación

El proceso de diseño se puede dividir en dos:

- Diseño de la placa de circuito impreso: Circuito con acabado profesional que garantiza un correcto funcionamiento del circuito.
- Diseño de la caja: Consiste en un mueble impreso en 3D que albergará el circuito y el cableado necesario, de modo que se vea un aspecto aseado y bonito.

7.1. Diseño de la placa de circuito impreso

El proceso de diseño de la placa de circuito impreso se ha llevado a cabo a partir del esquemático diseñado. El programa utilizado para su realización ha sido ARES Proteus ®.

La placa se basa en un diseño de doble capa, en la que el encapsulado de los componentes es de tipo through-hole (o de agujero pasante), y, además, en cada una de las capas se dispone un plano de señal (Figura 23).

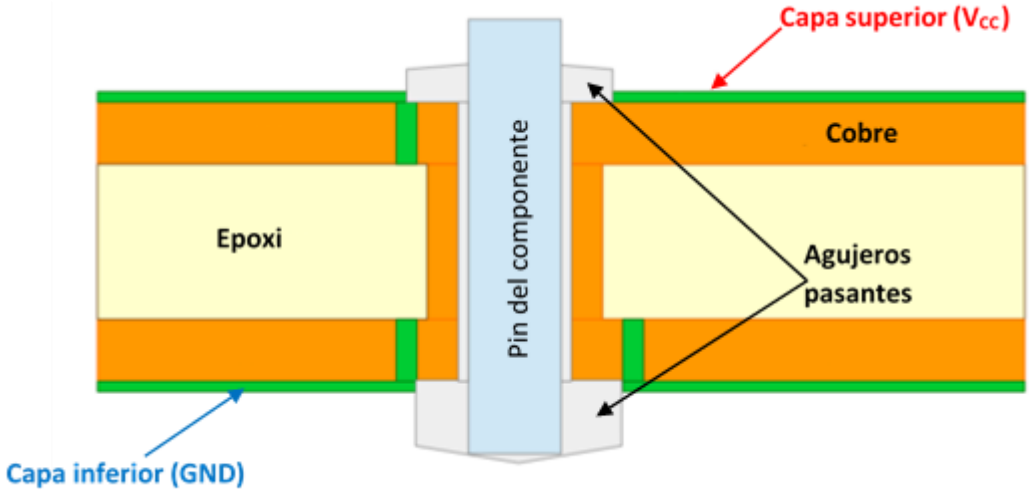


Figura 23: Representación gráfica de las características del circuito

El diseño se ha basado en las características que se muestran en la tabla 6:

Característica	Descripción	
Dimensiones	140 x 126 mm	
Número de capas	2	TOP – Plano de alimentación
		BOTTOM – Plano de masa
Espesor de la PCB	1.6 mm	
Ancho de pista general	0.254 mm	
Ancho de pista de mayor amperaje	1.27 mm	
Tamaño de vía	0.381 mm	
Separación mínima entre elementos	0.254 mm	
Agujeros de fijación	4 x 3 mm	

Tabla 6: Parámetros de diseño del circuito impreso

El resultado del diseño se puede contemplar en la simulación 3D realizada con el mismo programa de diseño. En las figuras 24 y 25 se representa dicho modelo visto desde diferentes puntos.

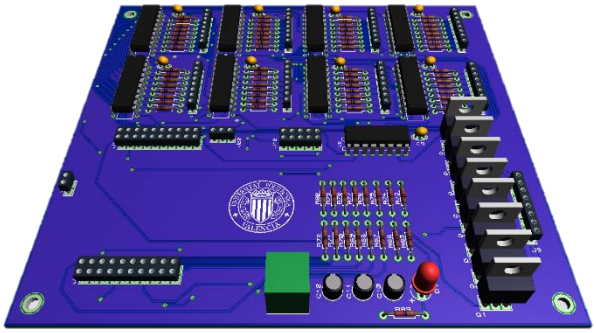


Figura 24: Modelo 3D de la capa superior del circuito

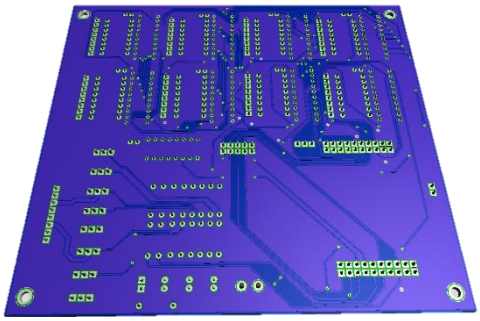


Figura 25: Modelo 3D de la capa inferior del circuito

7.2. Diseño de la carcasa

La caja se ha diseñado mediante el programa SolidWorks® con el fin de obtener un modelo 3D para más tarde poderlo enviar a la impresora 3D para imprimir.

El diseño de la caja se ha realizado en último lugar, tras la fabricación de la placa de circuito impreso, con el fin de poder adaptarla a las necesidades de tamaño de este.

La caja se ha realizado de forma modular de modo que resulte más fácil acceder a su interior en caso de necesidad de reparación o conexión.

Está constituida por un total de 4 piezas:

- Tapa trasera, placa lisa que podría ser constituida por cualquier tipo de diseño.
- Tapa frontal, por donde se realizará el conexionado de los cables de alimentación.
- Carcasa inferior, dónde se hallará fijada la placa de circuito impreso.
- Carcasa superior, dónde se encontrarán los orificios a través de los cuales pasarán los pines de los LED para su conexión con el circuito impreso.

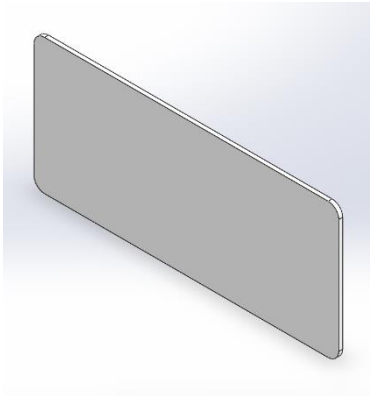


Figura 26: Modelo 3D de la tapa frontal

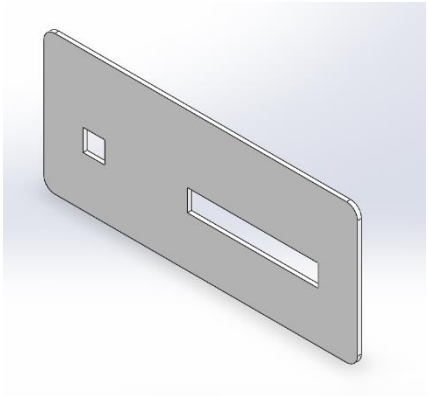


Figura 27: Modelo 3D de la tapa trasera

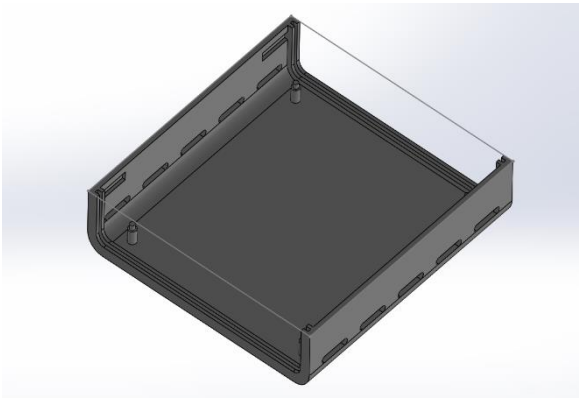


Figura 28: Modelo 3D de la carcasa inferior

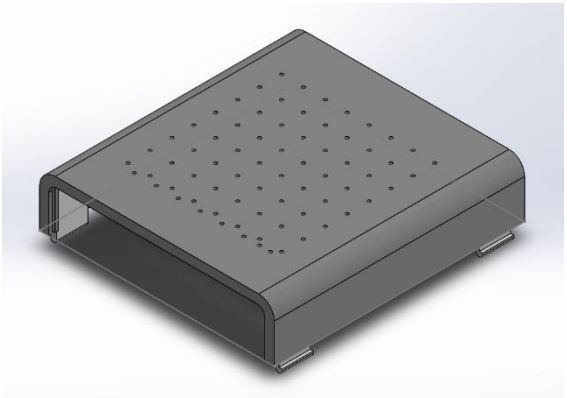


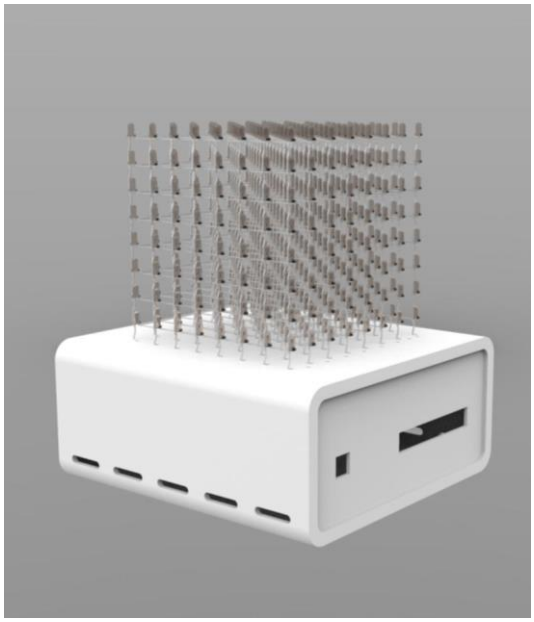
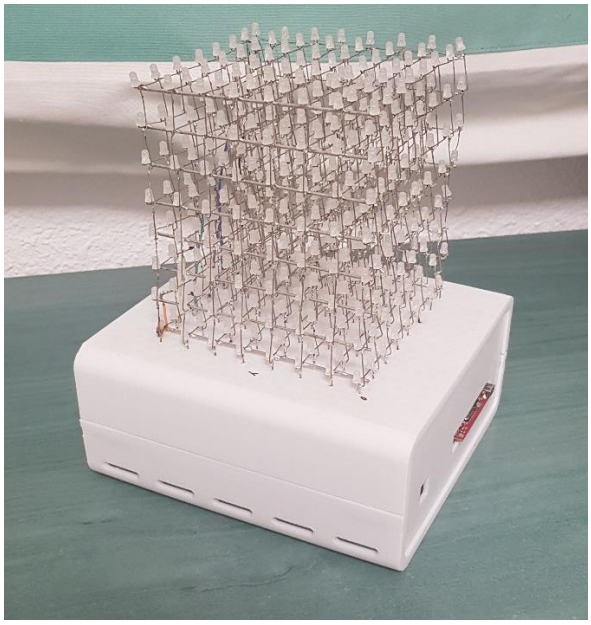
Figura 29: Modelo 3D de la carcasa superior

7.3. Montaje final

Para llevar a cabo el montaje final se han seguido los siguientes pasos:

1. Elaboración del cubo de leds.
2. Impresión 3D de las diferentes partes que constituyen la caja.
3. Inserción del cubo sobre la pieza superior de la caja a través de los agujeros.
4. Soldadura de los diferentes cables que llevan la información a los leds.
5. Conexionado del resto de cables al circuito impreso.
6. Ensamblaje del resto de piezas de la caja

En las figuras 30 y 31 se puede comparar el modelo elaborado en el ordenador con el modelo real que se ha fabricado.

Modelo 3D	Modelo real
 *Figura 30: Modelado 3D*	 *Figura 31: Modelo llevado a la realidad*

8. Posibles mejoras de la aplicación

Este tipo de aplicación es un dispositivo que permite añadirle cualquier tipo de mejora o variación en su sistema.

Las mejoras se podrían diferenciar en:

- Implementación Software:
  - Incorporación de nuevos efectos.
- Implementación Hardware y Software:
  - Uso de diodos LED RGB.
  - Control mediante Bluetooth o infrarrojos de la secuencia de los efectos. Esto implicaría una implementación hardware y software.
  - Control de los efectos de acuerdo con un sonido recibido, es decir, que la secuencia de efectos se adapte a un ritmo recibido mediante un micrófono.
- Implementación física:
  - Aumento de las dimensiones del cubo mediante el alargamiento de las patillas de los diodos.
  - Reducción del tamaño de la placa de circuito impreso gracias a la utilización de componentes SMD en lugar de componentes through-hole.

Estas son algunas de las ideas que podrían ser llevadas a cabo para otorgar mayores prestaciones a la aplicación.

9. Conclusiones

Las matrices de LEDs no son productos innovadores ya que se llevan usando desde hace mucho tiempo para mostrar información, como por ejemplo ofertas de tiendas, información horaria del transporte público, etc.

Estas pueden ser de distintos tamaños, y, dependiendo de las dimensiones que se le quiera otorgar a la matriz, dependerá directamente en el número de componentes que se utilizará, ya que, el número de LEDs a controlar será mayor o menor. Cuanto mayor es el tamaño de la matriz, más complejo es circuito de control, así como su algoritmo de control.

El circuito electrónico no se restringe a los componentes utilizados en este proyecto, cada diseñador puede elegir unos u otros dependiendo de las necesidades o los gustos. La experiencia es un factor clave en el diseño de circuitos, resultando más fácil o más difícil su elaboración dependiendo de ésta, ya que en el mercado existe una gran variedad de componentes con características similares. Según los componentes usados cambiará la forma del circuito, ya que no es lo mismo, por ejemplo, la utilización de registros de desplazamiento serie-paralelo que la utilización de registros paralelo-paralelo. De igual modo sucede con el uso de transistores BJT o MOSFET, ya que poseen características diferentes, pero pueden ser empleados para realizar la misma función.

A la hora de realizar el diseño de la placa de circuito impreso, también influye la selección que se haya realizado de componentes, debido a que todos no disponen de los mismos encapsulados, que pueden facilitar o dificultar el diseño de éste. Se debe prestar especial atención al tipo de conectores que se quiere utilizar porque se pueden cometer fallos que los conectores hembras y machos no encajen entre sí a pesar de ser el mismo tipo de conector, esto es debido a que el mismo conector se puede encontrar en diferentes tamaños.

Problemas similares aparecen cuando se decide elaborar un modelo 3D para llevarlo a la realidad. A la hora diseñar se tienen que tener en cuenta posibles tolerancias de fabricación. Tras la impresión 3D de la caja se detectó que su montaje no era factible debido a que no se habían tenido en cuenta dichas tolerancias, por lo tanto, no se podía llevar a cabo su ensamblaje final. Este tipo de sucesos repercute tanto en tiempos de elaboración como en los costes del producto final.

Como sumario de las conclusiones se puede decir:

1. La cantidad de leds a utilizar influirá directamente en el número de componentes y, por tanto, en la complejidad del circuito.
2. La experiencia es un factor clave a la hora de llevar a cabo diseño de circuitos.
3. Los procesos de fabricación no son procesos perfectos, se debe tener en cuenta que existen tolerancias durante éstos.

10. Bibliografía

1. Thomas L. Floyd. Fundamentos de sistemas digitales. 9ª edición Pearson Prentice Hall.
2. Robert L. Boylestad, Louis Nashelsky. Teoría de circuitos y dispositivos electrónicos. 10ª edición Pearson.
3. Temario Tecnología electrónica. El transistor bipolar (BJT). Universidad Politécnica de Valencia 2013.
4. Temario Tecnología electrónica. El transistor de efecto de campo. Universidad Politécnica de Valencia 2013.
5. Temario Electrónica de potencia. Disipación térmica. Universidad Politécnica de Valencia 2014.
6. Temario Sistema Digitales Aplicados. Ingeniería del software. Universidad Politécnica de Valencia 2015.
7. Temario Sistema Digitales Aplicados. Flujogramas. Universidad Politécnica de Valencia 2015.
8. Temario Sistema Digitales Aplicados. Trabajo final de la asignatura. Universidad Politécnica de Valencia 2015.
9. LAUNCHXL-F28027 C2000 Piccolo LaunchPad Experimenter Kit User's Guide. Texas Instruments. Revisión enero 2014.
10. TMS320F2802x Piccolo Microcontrollers Datasheet. Texas Instruments. Revisión junio 2016.
11. TMS320F2802x Piccolo System Control and Interrupts Reference Guide. Texas Instruments. Revisión febrero 2013.
12. 74HCT138 Datasheet. Diodes Incorporated. Revisión junio 2013.
13. 74HCT574 Datasheet. Nexperia. Revisión marzo 2016.
14. IRL540 Datasheet. Vishay. Revisión marzo 2011.
15. Getting Started. Doxygen Manual. Versión en línea (https://www.stack.nl/~dimitri/doxygen/manual/starting.html).

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Grado en Ingeniería Electrónica Industrial y Automática

2 – Pliego de condiciones

Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

ÍNDICE

1. Alcance del pliego

El objeto de este documento es fijar las condiciones técnicas mínimas que debe cumplir la aplicación descrita. El ámbito de aplicación de este documento se extenderá tanto a los sistemas electrónicos como mecánicos que conforman el producto final. Si se justifica adecuadamente, se podrán llevar a cabo soluciones alternativas a este documento, debido al desarrollo de nuevas tecnologías o a la imposición debido a nuevas normativas, siempre y cuando no impliquen una disminución de los requisitos mínimos especificados en el mismo, ni empeoramiento de su funcionamiento y/o utilidad.

2. Materiales

2.1. Materiales para la elaboración del circuito

2.1.1. Microcontrolador

Se utilizará la placa de experimentación o LaunchPad basado en el microcontrolador TMS320F28027 de Texas Instruments. Éste deberá poseer tantos pines de entrada/salida como sean requeridos en la aplicación, y, en la medida de lo posible, mantener pines libres para la implementación de futuras mejoras.

2.1.2. Resistencias

Todas las resistencias utilizadas serán de valor fijo con una tolerancia máxima admisible del 5%, de modo que se evite en la medida de lo posible la introducción de ruido intrínseco al circuito.

Deberán estar formadas por película metálica, y poseer la capacidad de disipar todo el calor generado durante la operación del producto. Debido a que no se trata de una aplicación de alta potencia, se recomienda la utilización de resistencias de 1/4 W de poder de disipación.

Los valores de éstas serán especificados de acuerdo con las necesidades del circuito implementado.

2.1.3. Diodos LED

Se utilizarán diodos de 3 mm de diámetro con el fin de evitar que un efecto visual/estético de apelotonamiento, dando sensación de mayor compactación y fácil reconocimiento de los efectos.

Los colores de estos es una característica que queda abierta a la selección del desarrollador del producto. La única característica para tener en cuenta es que todos los LEDs deben ser del mismo color.

Además, se incorporará un LED que evalúe el estado del circuito, es decir, si el circuito está encendido o apagado.

2.1.4. Circuitos integrados

Los circuitos integrados utilizados deberán permitir el manejo de los LEDs a diferentes tensiones de salida del microcontrolador, en caso de que en futuras aplicaciones se decida cambiar este componente. Por lo tanto, los integrados deberán ser capaces de trabajar con tensiones de 3.3 y 5 V, niveles lógicos más comunes utilizados por los microcontroladores del mercado.

Para el manejo del encendido de los LEDs se utilizarán registros de desplazamiento con un total de 8 salidas, de acuerdo con las dimensiones del cubo.

Para la gestión del encendido de los registros de desplazamiento se utilizará un decodificador con un total de salidas igual al total de registros de desplazamiento utilizados.

Los circuitos integrados deberán estar dispuestos en zócalos que permitan su fácil reemplazo en caso de fallo del componente.

2.1.5. Transistores

Los transistores seleccionados deberán ser capaces de soportar las corrientes generadas por el circuito, y permitir la conmutación a la frecuencia seleccionada sin limitaciones.

2.1.6. Condensadores

Se deberán utilizar condensadores para la disminución del rizado de tensión que se pueda producir en la alimentación de los diferentes componentes activos, así como a la tensión de entrada del circuito.

2.1.7. Conectores

Se empleará el mínimo número de conectores para la implementación del circuito. Así como, se deberán encontrar conectores que permitan la conexión de nuevos componentes a los pines de entrada del microcontrolador.

2.1.8. Alimentación

Para la alimentación del circuito se deberá utilizar una fuente de alimentación de 5 V, así como debe disponer de un interruptor que permita su conexión y desconexión.

2.1.9. Circuito integrado

El circuito integrado deberá disponer de las menores dimensiones posibles mediante la utilización de componentes through-hole a doble capa.

Su fabricación se llevará a cabo por medio de una empresa especializada, la cual especificará las características mínimas que ha de disponer el diseño para su correcta elaboración.

2.2. Materiales para la elaboración del aspecto estético

2.2.1. Plástico

La caja que albergará el circuito estará realizada mediante impresión 3D utilizando un plástico ABS.

Las dimensiones serán las mínimas requeridas para albergar el circuito impreso y el cableado sin que existan interferencias que puedan provocar fallos de funcionamiento.

La caja deberá respetar el diseño realizado por ordenador cumpliendo con las mínimas tolerancias posibles que pueda ofrecer la impresora.

3. Ejecución

3.1. Programación del software

La programación se realizará en lenguaje de alto nivel, en lenguaje C, mediante las herramientas proporcionadas por Texas Instruments para tal fin.

Con el fin de enviar el código a la placa de experimentación, se deberá disponer de un ordenador con el programa Code Composer Studio (CCS) instalado. Esta herramienta es gratuita y es proporcionada por Texas Instruments en su página web.

Se abrirá un proyecto nuevo donde se encontrará el código completo de la aplicación. Se conectará la placa de experimentación mediante un cable USB al ordenador y se probará que ambos dispositivos están comunicando. Una vez la comunicación es establecida, se procederá a descargar el programa en la memoria flash del microcontrolador, de manera que este se guarde de forma permanente, incluso cuando se produzca un apagado del mismo.

3.2. Ensamblaje

3.2.1. Matriz tridimensional

Los 512 LEDs se dispondrán formando un cubo de 8 LEDs por eje. Su unión se hará uniendo los ánodos de toda la capa y los cátodos de las diferentes capas mediante soldadura con estaño. Cada capa se formará por separado, con un total de 8 capas, las cuales se unirán para dar lugar a tal forma.

3.2.2. Circuito impreso

La placa de circuito impreso se diseñará mediante un programa CAD destinado a este propósito, se generarán los archivos necesarios para que puedan ser enviados a una empresa especializada en la fabricación de circuitos impresos.

Una vez la placa haya sido fabricada, se montarán todos los componentes sobre ella de acuerdo con el esquema eléctrico diseñado y se procederá a su soldadura. El proceso de soldadura se realizará mediante la aplicación de estaño sobre los terminales de los componentes, fundiéndolo sobre éstos mediante un soldador tipo lápiz o pistola de aire caliente, permitiendo que el estaño se enfríe sobre el terminal, quedando, de este modo, los componentes fijos sobre la placa.

3.2.3. Carcasa

La caja se llevará a cabo mediante la técnica de impresión 3D. En primer lugar, se diseñará sobre un programa CAD, generando los archivos necesarios para su interpretación por el programa compatible con la impresora 3D.

La caja se diseñará de forma modular de modo que, si alguna parte de ella sale defectuosa durante la impresión, ésta pueda ser reemplazada de la manera más sencilla posible y utilizando la menor cantidad de material posible.

Una vez la impresión ha finalizado, se lijarán, imprimarán y pintarán las piezas de modo que quede un aspecto profesional.

Cuando la pintura se haya secado, se montarán las piezas de acuerdo con el plano.

3.2.4. Ensamblaje del conjunto

Se dispondrá el cubo sobre la capa superior de la caja permitiendo que los cables queden en el interior de ésta. Sobre el interior de la capa inferior, se colocará el circuito impreso, sobre los puntos destinados para tal cometido, y se atornillará utilizando tornillos M3.

Se llevará a cabo el conexionado de todo el cableado en sus correspondientes puntos de conexión, quedando finalizado el proceso de ensamblaje del cubo.

4. Prueba de servicio

Para realizar la validación del producto, se llevarán a cabo pruebas que demuestren su correcto funcionamiento y cumplan con las especificaciones.

Una vez diseñado el circuito, se realizará un prototipo para evaluar el funcionamiento del mismo, se comprobará que todos los componentes seleccionados interactúan entre sí tal y como se esperaba. Una vez el prototipo se ha validado, se llevará a cabo la fabricación del circuito impreso.

Respecto al aspecto estético, se evaluarán las piezas de la caja impresas, realizando las mediciones pertinentes y comparándolas con el plano. También se evaluará su fragilidad con el fin de que se trate de un producto con la mayor durabilidad posible.

Una vez los componentes han sido validados por separado, y el producto final haya sido ensamblado, se comprobará que el montaje está realizado de manera correcta, y los efectos establecidos se comportan tal y como se desea.

En caso de que alguna de las pruebas que se le realicen tenga un resultado no satisfactorio, se llevarán a cabo las medidas necesarias para su corrección.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Grado en Ingeniería Electrónica Industrial y Automática

3 - Presupuesto

Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

ÍNDICE

ÍNDICE DE TABLAS

1. Objetivo

En el presente documento se va a detallar el presupuesto relacionado con el desarrollo y fabricación unitario del proyecto, es decir, el precio total que costará llevar a cabo la implementación de una unidad de dicha aplicación.

Dentro de éste se incluirán todos los apartados necesarios para explicar y mostrar de una manera clara y precisa el desarrollo del presupuesto.

Los precios de los componentes se han escogido de acuerdo con los precios establecidos por el suministrador de éstos. Así como los precios de los operarios de mano de obra han sido escogidos en base a una estimación.

2. Cuadro de precios elementales

A continuación, se detallará la base de precios utilizada para llevar a cabo el cálculo del precio final unitario del producto. Este conjunto de precios se divide en: Precio de los materiales, precio de la mano de obra de los operarios que intervienen durante el proceso de fabricación y la cantidad dedicada para solventar posibles imprevistos.

Cuadro de precios elementales	
Descripción	**Precio**
Materiales	
TMS32F28027	18,15
Resistencia 100 Ω 1 %	0,046 €
Resistencia 10 kΩ 1 %	0,061 €
Condensador cerámico 0,1 uF 10 %	0,140 €
Condensador de aluminio 10 uF 20 %	0,200 €
Condensador de aluminio 100 uF 20 %	0,246 €
Condensador de aluminio 1000 uF 20 %	0,446 €
Circuito integrado 74HCT574	0,530 €
Circuito integrado 74HCT138	0,560 €
Transistor de efecto de campo IRL540N	0,930 €
Diodo LED	0,062 €
Conector macho 2 pines 2,54 mm	0,054 €
Conector macho 8 pines 2,54 mm	0,187 €
Conector macho 10 pines 2,54 mm	0,376 €
Conector hembra 3 pines 2,54 mm	0,461 €
Conector hembra 20 pines 2,54 mm	0,930 €
Conector 2 terminales	0,496 €
Mano de obra	
Graduado en ingeniería	12,000 €/h
Técnico de taller	7,000 €/h
Medios auxiliares	
Medios auxiliares	5%

Tabla 7:Cuadro de precios elementales utilizados para el cálculo del presupuesto

3. Cuadro de precios descompuestos

3.1. Fabricación del Circuito Impreso

En la tabla 2 se puede observar el coste relacionado con la fabricación de la placa de circuito impreso. Para ello se tienen en cuenta los materiales necesarios, así como las horas estimadas de los operarios que intervienen en este proceso.

Fabricación Circuito Impreso			
Descripción	**Precio**	**Cantidad**	**Parcial**
Materiales			
Productos Industriales			
TMS32F28027	18,15	1	18,150 €
Resistencia 100 Ω 1 %	0,046 €	72	3,312 €
Resistencia 10 kΩ 1 %	0,061 €	8	0,488 €
Condensador cerámico 0,1 uF 10 %	0,140 €	9	1,260 €
Condensador de aluminio 10 uF 20 %	0,200 €	1	0,200 €
Condensador de aluminio 100 uF 20 %	0,246 €	1	0,246 €
Condensador de aluminio 1000 uF 20 %	0,446 €	1	0,446 €
Circuito integrado 74HCT574	0,530 €	8	4,240 €
Circuito integrado 74HCT138	0,560 €	1	0,560 €
Transistor de efecto de campo IRL540N	0,930 €	8	7,440 €
Diodo LED	0,062 €	513	31,806 €
Conector macho 2 pines 2,54 mm	0,054 €	1	0,054 €
Conector macho 8 pines 2,54 mm	0,187 €	9	1,683 €
Conector macho 10 pines 2,54 mm	0,376 €	1	0,376 €
Conector hembra 3 pines 2,54 mm	0,461 €	1	0,461 €
Conector hembra 20 pines 2,54 mm	0,930 €	2	1,860 €
Conector 2 terminales	0,496 €	1	0,496 €
		Subtotal Materiales	73,078 €
Mano de obra			
Mano de obra directa			
Graduado en ingeniería: Diseño	12,000 €/h	10	120,000 €
Técnico de taller: Montaje	7,000 €/h	4	28,000 €
		Subtotal Mano de Obra	148,000 €
Medios auxiliares			
Medios auxiliares	5%	221,078 €	11,054 €
		Coste de Fabricación del Circuito Impreso	**232,132 €**

Tabla 8: Costes de fabricación del circuito impreso.

3.2. Programación

Costes relacionados con la programación llevada a cabo para que la matriz LED luzca diferentes efectos.

Programación			
Descripción	**Precio**	**Cantidad**	**Parcial**
Materiales			
Productos Industriales			
			0,000 €
		Subtotal Materiales	0,000 €
Mano de obra			
Mano de obra directa			
Graduado en ingeniería: Programación	12,000 €	4	48,000 €
		Subtotal Mano de Obra	48,000 €
Medios auxiliares			
Medios auxiliares	5%	48,000 €	2,400 €
		Coste de Programación	**50,400 €**

Tabla 9: Costes relacionados con la programación

3.3. Fabricación Mecánica

En este apartado se detallan los costes que se deberán asumir para la fabricación de la caja que albergará la electrónica, y dará mejor aspecto estético al proyecto.

Fabricación Mecánica			
Descripción	**Precio**	**Cantidad**	**Parcial**
Materiales			
Productos Industriales			
Plástico PLA	16,488	0,3	4,946 €
		Subtotal Materiales	4,946 €
Mano de obra			
Mano de obra directa			
Graduado en ingeniería: Diseño	12,000 €/h	2	24,000 €
Técnico de taller: Impresión y montaje	7,000 €/h	1,5	10,500 €
		Subtotal Mano de Obra	34,500 €
Medios auxiliares			
Medios auxiliares	5%	39,446 €	1,972 €
		Coste de Fabricación Mecánica	**41,419 €**

Tabla 10: Costes relacionados con la fabricación mecánica.

3.4. Ensamblaje del conjunto

A continuación, se detallará los costes relacionados con el montaje final del producto.

Ensamblaje del conjunto			
Descripción	**Precio**	**Cantidad**	**Parcial**
Materiales			
Productos Industriales			
			0,000 €
		Subtotal Materiales	0,000 €
Mano de obra			
Mano de obra directa			
Técnico de taller: Ensamblaje	7,000 €/h	1,5	10,500 €
		Subtotal Mano de Obra	10,500 €
Medios auxiliares			
Medios auxiliares	5%	10,500 €	0,525 €
		Coste del Ensamblaje del conjunto	**11,025 €**

Tabla 11: Costes del ensamblaje final del producto

3.5. Pruebas de servicio

En la siguiente tabla se detallan los costes relacionados con las Pruebas de Servicio.

Pruebas de Servicio			
Descripción	**Precio**	**Cantidad**	**Parcial**
Materiales			
Productos Industriales			
			0,000 €
		Subtotal Materiales	0,000 €
Mano de obra			
Mano de obra directa			
Técnico de taller: Verificación	7,000 €	0,5	3,500 €
		Subtotal Mano de Obra	3,500 €
Medios auxiliares			
Medios auxiliares	5%	3,500 €	0,175 €
		Coste de las Pruebas de Servicio	**3,675 €**

Tabla 12: Costes de las Pruebas de Servicio

4. Resumen del presupuesto

Finalmente se realizará un resumen del presupuesto, en el que se mostrará el coste total de cada una de las partes anteriormente desarrolladas, así como el coste total del desarrollo del producto.

Resumen del presupuesto				
Descripción	***Materiales***	***Mano de obra***	***Medios aux***	***Total***
Fabricación Circuito Impreso	73,078 €	148,000 €	11,054 €	232,132 €
Programación	0,000 €	48,000 €	2,400 €	50,400 €
Fabricación Mecánica	4,946 €	34,500 €	1,972 €	41,419 €
Ensamblaje del conjunto	0,000 €	10,500 €	0,525 €	11,025 €
Pruebas de servicio	0,000 €	3,500 €	0,175 €	3,675 €
TOTAL sin IVA	**78,024 €**	**244,500 €**	**16,126 €**	**338,651 €**

Tabla 13: Resumen del presupuesto con el conjunto de costes totales sin IVA.

El precio remarcado en la tabla 7, se trata del precio final del producto sin I.V.A. A continuación, se muestra el precio final con los impuestos incluidos.

TOTAL con IVA (21 %)	**94,410 €**	**295,845 €**	**19,513 €**	**409,767 €**

Tabla 14: Costes totales con 21 % de IVA.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Grado en Ingeniería Electrónica Industrial y Automática

4 – Documentación del código

Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

ÍNDICE (1)

ÍNDICE (2)

ÍNDICE (3)

1. Documentación de archivos

1.1. Referencia del Archivo Draw_Func.c

Funciones de bajo nivel para gestionar el encendido de los leds.

Este fichero utiliza las siguientes librerías:

- #include <Draw_func.h>
- #include "GLOBAL_VARS.h"
- #include "GLOBAL_DEFS.h"

1.1.1. Funciones

- unsigned char **rango** (int x, int y, int z)

Evalúa si las coordenadas introducidas están dentro de las dimensiones del cubo El valor máximo se encuentra en el archivo **GLOBAL_DEFS.h**.

- void **led_ON** (int x, int y, int z)

Enciende el led que se desea introduciendo sus coordenadas.

- void **led_OFF** (int x, int y, int z)

Apaga el led que se desea introduciendo sus coordenadas.

- unsigned char **led_status** (int x, int y, int z)

Evalúa el estado de un led.

- void **led_cambio** (int x, int y, int z, int estado)

Enciende o apaga un led.

- void **led_toggle** (int x, int y, int z)

Conmuta el estado de un led.

- void **evaluar_coordenadas** (int c1, int c2, int *ac1, int *ac2)

Asegura que las coordenadas x1, y1 y z1 siempre sean menores a las x2, y2 y z2.

- void **planoX** (int x)

Enciende el plano seleccionado de la coordenada X introducida.

- void **planoX_OFF** (int x)

Apaga el plano seleccionado de la coordenada X introducida.

- void **planoY** (int y)

Enciende el plano seleccionado de la coordenada Y introducida.

- void **planoY_OFF** (int y)

Apagar el plano seleccionado de la coordenada Y introducida.

- void **planoZ** (int z)

Enciende el plano seleccionado de la coordenada Z introducida.

- void **planoZ_OFF** (int z)

Apaga el plano seleccionado de la coordenada Z introducida.

- void **plano** (char eje, unsigned char coordenada)

Enciende el plano seleccionado del eje y coordenada introducida.

- void **plano_OFF** (char eje, unsigned char coordenada)

Apaga el plano seleccionado del eje y coordenada introducida.

- void **caja** (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende una caja de las dimensiones solicitadas, encendiendo también su interior.

- void **caja_vacia** (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende una caja de las dimensiones solicitadas con el interior apagado.

- void **caja_aristas** (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende las aristas de una caja de las dimensiones solicitadas.

- char **lineaX** (int inicio, int final)

Enciende una línea en el eje X.

- void **desplazar** (char eje, int direccion)

Desplaza un plano a través de un eje.

- void **encender** (unsigned char matriz)

Enciende el cubo completamente de acuerdo con un patrón.

1.1.2. Descripción detallada

Funciones de bajo nivel para gestionar el encendido de los leds.

Se llevan a cabo diferentes funciones que permiten gestionar de una manera más fácil el encendido de los diferentes leds. Se consigue encender desde un único led hasta realizar funciones de desplazamiento.

1.1.2.1. Documentación de las funciones

1.1.2.1.1. void caja (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende una caja de las dimensiones solicitadas, encendiendo también su interior.

Parámetros:

x1	Coordenada X del origen
y1	Coordenada Y del origen
z1	Coordenada Z del origen
x2	Coordenada X del final
y2	Coordenada Y del final
z2	Coordenada Z del final

Código:

```
299 {
300   int auxY;
301   int auxZ;
302
303   evaluar_coordenadas(x1, x2, &x1, &x2);
304   evaluar_coordenadas(y1, y2, &y1, &y2);
305   evaluar_coordenadas(z1, z2, &z1, &z2);
306
307   for (auxZ=z1;auxZ<=z2;auxZ++)
308   {
309     for (auxY=y1;auxY<=y2;auxY++)
310     {
311       cubo[auxZ][auxY] |= lineaX(x1,x2);
312     }
313   }
314 }
```

1.1.2.1.2. void caja_aristas (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende las aristas de una caja de las dimensiones solicitadas.

Parámetros:

x1	Coordenada X del origen
y1	Coordenada Y del origen
z1	Coordenada Z del origen
x2	Coordenada X del final
y2	Coordenada Y del final
z2	Coordenada Z del final

Código:

```
359 {
360   int auxY;
361   int auxZ;
362
363   evaluar_coordenadas(x1, x2, &x1, &x2);
364   evaluar_coordenadas(y1, y2, &y1, &y2);
365   evaluar_coordenadas(z1, z2, &z1, &z2);
366
367   //Dibuja las lineas del EJE X
368   cubo[z1][y1] = lineaX(x1,x2);
369   cubo[z1][y2] = lineaX(x1,x2);
370   cubo[z2][y1] = lineaX(x1,x2);
371   cubo[z2][y2] = lineaX(x1,x2);
372
373   //Dibuja las lineas del EJE Y
374   for (auxY=y1;auxY<=y2;auxY++)
375   {
376     led_ON(x1,auxY,z1);
377     led_ON(x1,auxY,z2);
378     led_ON(x2,auxY,z1);
379     led_ON(x2,auxY,z2);
380   }
381
382   //Dibuja las lineas del EJE Z
383   for (auxZ=z1;auxZ<=z2;auxZ++)
384   {
385     led_ON(x1,y1,auxZ);
386     led_ON(x1,y2,auxZ);
387     led_ON(x2,y1,auxZ);
388     led_ON(x2,y2,auxZ);
389   } }
```

1.1.2.1.3. void caja_vacia (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende una caja de las dimensiones solicitadas con el interior apagado.

Parámetros:

x1	Coordenada X del origen
y1	Coordenada Y del origen
z1	Coordenada Z del origen
x2	Coordenada X del final
y2	Coordenada Y del final
z2	Coordenada Z del final

Código:

```
326 {
327   int auxY;
328   int auxZ;
329
330   evaluar_coordenadas(x1, x2, &x1, &x2);
331   evaluar_coordenadas(y1, y2, &y1, &y2);
332   evaluar_coordenadas(z1, z2, &z1, &z2);
333
334   for (auxZ=z1;auxZ<=z2;auxZ++)
335   {
336     for (auxY=y1;auxY<=y2;auxY++)
337     {
338       if (auxY == y1 || auxY == y2 || auxZ == z1 || auxZ == z2)
339       {
340         cubo[auxZ][auxY] = lineaX(x1,x2);
341       } else
342       {
343         cubo[auxZ][auxY] |= ((0x01 << x1) | (0x01 << x2));
344       }
345     }
346   }
347 }
```

1.1.2.1.4. void desplazar (char eje, int direccion)

Desplaza un plano a través de un eje.

Parámetros:

eje	Eje en el que se quiere desplazar
direccion	Selección de la dirección, hacia delante o hacia detrás

Código:

```
408 {
 409   int i, x ,y;
 410   int ii, iii;
 411   int estado;
 412
 413   for (i = 0; i < 8; i++)
 414   {
 415      if (direccion == -1)
 416      {
 417         ii = i;
 418      } else
 419      {
 420         ii = (7-i);
 421      }
 422
 423
 424      for (x = 0; x < 8; x++)
 425      {
 426         for (y = 0; y < 8; y++)
 427         {
 428            if (direccion == -1)
 429            {
 430               iii = ii+1;
 431            } else
 432            {
 433               iii = ii-1;
 434            }
 435
 436            if (eje == EJE_Z)
 437            {
 438               estado = led_status(x,y,iii);
 439               led_cambio(x,y,ii,estado);
 440            }
 441
 442            if (eje == EJE_Y)
 443            {
 444               estado = led_status(x,iii,y);
 445               led_cambio(x,ii,y,estado);
 446            }
 447
 448            if (eje == EJE_X)
```

```
449             {
450                 estado = led_status(iii,y,x);
451                 led_cambio(ii,y,x,estado);
452             }
453         }
454     }
455 } }
```

1.1.2.1.5. void encender (unsigned char matriz)

Enciende el cubo completamente de acuerdo a un patrón.

Parámetros:

matriz	Patrón a seguir para el encendido, se trata de una matriz de 512 posiciones

Código:

```
464 {
 465   int z,y;
 466   for (z=0;z<8;z++)
 467   {
 468     for (y=0;y<8;y++)
 469     {
 470       cubo [z][y]= matriz;
 471     }
 472   }
 473 }
```

1.1.2.1.6. void evaluar_coordenadas (int c1, int c2, int * ac1, int * ac2)

Asegura que las coordenadas x1,y1 y z1 siempre sean menores a las x2,y2 y z2.

Se le introduce una coordenada de un vector y la misma coordenada de otro vector. El resultado es asegurar que la coordenada del vector 2 es mayor que la del vector 1.

Parámetros:

c1	Primera coordenada a evaluar
c2	Segunda coordenada a evaluar
**ac1*	Variable auxiliar para realizar el cambio
**ac2*	Variable auxiliar para realizar el cambio

Código:

```
136 {
 137    if(c1>c2)
 138    {
 139       int auxiliar;
 140       auxiliar = c1;
 141       c1=c2;
 142       c2=auxiliar;
 143    }
 144    *ac1=c1;
 145    *ac2=c2; }
```

1.1.2.1.7. void led_cambio (int x, int y, int z, int estado)

Enciende o apaga un led.

Parámetros:

x	Coordenada X
y	Coordenada Y
z	Coordenada Z
estado	Acción a realizar, "1" significa encender,"0" significa apagar

Código:

```
 98 {
 99    if (estado == 1)
100    {
101        led_ON(x,y,z);
102    }
103    else
104    {
105        led_OFF(x,y,z);
106    }
107 }
```

1.1.2.1.8. void led_OFF (int x, int y, int z)

Apaga el led que se desea introduciendo sus coordenadas.

Parámetros:

x	Coordenada X
y	Coordenada Y
z	Coordenada Z

Código:

```
60 {
 61    if(rango(x,y,z)==TRUE)
 62    {
 63        cubo[z][y]&=~(1<<x);
 64    }
 65 }
```

1.1.2.1.9. void led_ON (int x, int y, int z)

Enciende el led que se desea introduciendo sus coordenadas.

Parámetros:

x	Coordenada X
y	Coordenada Y
z	Coordenada Z

Código:

```
46 {
47   if(rango(x,y,z)==TRUE)
48   {
49       cubo[z][y]|=(1<<x);
50   }
51 }
```

1.1.2.1.10. unsigned char led_status (int x, int y, int z)

Evalúa el estado de un led.

Parámetros:

x	Coordenada X
y	Coordenada Y
z	Coordenada Z

Código:

```
74 {
 75   if (rango(x,y,z))
 76   {
 77      if (cubo[z][y] & (1 << x))
 78      {
 79         return TRUE;
 80      } else
 81      {
 82         return FALSE;
 83      }
 84   } else
 85   {
 86      return FALSE;
 87   }
 88 }
```

1.1.2.1.11. void led_toggle (int x, int y, int z)

Conmuta el estado de un led.

Parámetros:

x	Coordenada X
y	Coordenada Y
z	Coordenada Z

Código:

```
116 {
117    if (rango(x, y, z)==TRUE)
118    {
119        cubo[z][y] ^= (1 << x);
120    }
121
122 }
```

1.1.2.1.12. char lineaX (int inicio, int final)

Enciende una línea en el eje X.

Parámetros:

inicio	Coordenada X del origen
final	Coordenada X del final

Código:

```
398 {
399    return ((0xff<<inicio) & ~(0xff<<(final+1)));
400 }
```

1.1.2.1.13. void plano (char eje, unsigned char coordenada)

Enciende el plano seleccionado del eje y coordenada introducida.

Parámetros:

eje	Selección de eje
coordenada	Coordenada del eje seleccionado

Código:

```
249 {
250    switch (eje)
251    {
252       case EJE_X:
253          planoX(coordenada);
254          break;
255
256       case EJE_Y:
257          planoY(coordenada);
258          break;
259
260       case EJE_Z:
261          planoZ(coordenada);
262          break;
263    }
264 }
```

1.1.2.1.14. void plano_OFF (char eje, unsigned char coordenada)

Apaga el plano seleccionado del eje y coordenada introducida.

Parámetros:

eje	Selección de eje
coordenada	Coordenada del eje seleccionado

Código:

```
272 {
 273    switch (eje)
 274    {
 275       case EJE_X:
 276          planoX_OFF(coordenada);
 277          break;
 278
 279       case EJE_Y:
 280          planoY_OFF(coordenada);
 281          break;
 282
 283       case EJE_Z:
 284          planoZ_OFF(coordenada);
 285          break;
 286    }
 287 }
```

1.1.2.1.15. void planoX (int x)

Enciende el plano seleccionado de la coordenada X introducida.

Parámetros:

x	Plano del eje X que se quiere encender

Código:

```
153 {
 154    int z;
 155    int y;
 156    if (x>=0 && x<X)
 157    {
 158       for (z=0;z<Z;z++)
 159       {
 160          for (y=0;y<Y;y++)
 161          {
 162             cubo[z][y] |= (1 << x);
 163          }
 164       }
 165    }
 166 }
```

1.1.2.1.16. void planoX_OFF (int x)

Apaga el plano seleccionado de la coordenada X introducida.

Parámetros:

x	Plano del eje X que se quiere apagar

Código:

```
173 {
 174    int z;
 175    int y;
 176    if (x>=0 && x<X)
 177    {
 178       for (z=0;z<Z;z++)
 179       {
 180          for (y=0;y<Y;y++)
 181          {
 182             cubo[z][y] &= ~(1 << x);
 183          }
 184       }
 185    }
 186 }
```

1.1.2.1.17. void planoY (int y)

Enciende el plano seleccionado de la coordenada Y introducida.

Parámetros:

y	Plano del eje Y que se quiere encender

Código:

```
193 {
 194    int z;
 195    if (y>=0 && y<Y)
 196    {
 197       for (z=0;z<Z;z++)
 198          cubo[z][y] = 0xff;
 199    }
 200 }
```

1.1.2.1.18. void planoY_OFF (int y)

Apagar el plano seleccionado de la coordenada Y introducida.

Parámetros:

y	Plano del eje Y que se quiere apagar

Código:

```
207 {
 208    int z;
 209    if (y>=0 && y<Y)
 210    {
 211       for (z=0;z<Z;z++)
 212          cubo[z][y] = 0x00;
 213    }
 214 }
```

1.1.2.1.19. void planoZ (int z)

Enciende el plano seleccionado de la coordenada Z introducida.

Parámetros:

z	Plano del eje Z que se quiere encender

Código:

```
 220 {
 221    int i;
 222    if (z>=0 && z<Z)
 223    {
 224       for (i=0;i<Z;i++)
 225          cubo[z][i] = 0xff;
 226    }
 227 }
```

1.1.2.1.20. void planoZ_OFF (int z)

Apaga el plano seleccionado de la coordenada Z introducida.

Parámetros:

z	Plano del eje Z que se quiere apagar

Código:

```
234 {
235    int i;
236    if (z>=0 && z<Z)
237    {
238       for (i=0;i<Z;i++)
239          cubo[z][i] = 0x00;
240    }
241 }
```

1.1.2.1.21. unsigned char rango (int x, int y, int z)

Evalúa si las coordenadas introducidas están dentro de las dimensiones del cubo El valor máximo se encuentra en el archivo **GLOBAL_DEFS.h**.

Parámetros:

x	Coordenada X
y	Coordenada Y
z	Coordenada Z

Código:

```
28 {
29    if (x>=0 && x<X && y>=0 && y<Y && z>=0 && z<Z)
30    {
31       return TRUE;
32    }
33    else
34    {
35       return FALSE;
36    }
37 }
```

1.2. Referencia del Archivo Draw_func.h

Cabecera de las funciones de dibujo.

1.2.1. Funciones

- unsigned char **rango** (int x, int y, int z)

Evalúa si las coordenadas introducidas están dentro de las dimensiones del cubo El valor máximo se encuentra en el archivo **GLOBAL_DEFS.h**.

- void **evaluar_coordenadas** (int c1, int c2, int *ac1, int *ac2)

Asegura que las coordenadas x1,y1 y z1 siempre sean menores a las x2,y2 y z2.

- void **led_ON** (int x, int y, int z)

Enciende el led que se desea introduciendo sus coordenadas.

- void **led_OFF** (int x, int y, int z)

Apaga el led que se desea introduciendo sus coordenadas.

- unsigned char **led_status** (int x, int y, int z)

Evalua el estado de un led.

- void **led_cambio** (int x, int y, int z, int estado)

Enciende o apaga un led.

- void **led_toggle** (int x, int y, int z)

Conmuta el estado de un led.

- void **planoX** (int x)

Enciende el plano seleccionado de la coordenada X introducida.

- void **planoX_OFF** (int x)

Apaga el plano seleccionado de la coordenada X introducida.

- void **planoY** (int y)

Enciende el plano seleccionado de la coordenada Y introducida.

- void **planoY_OFF** (int y)

Apagar el plano seleccionado de la coordenada Y introducida.

- void **planoZ** (int z)

Enciende el plano seleccionado de la coordenada Z introducida.

- void **planoZ_OFF** (int z)

Apaga el plano seleccionado de la coordenada Z introducida.

- void **plano** (char eje, unsigned char coordenada)

Enciende el plano seleccionado del eje y coordenada introducida.

- void **plano_OFF** (char eje, unsigned char coordenada)

Apaga el plano seleccionado del eje y coordenada introducida.

- void **caja** (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende una caja de las dimensiones solicitadas, encendiendo también su interior.

- void **caja_vacia** (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende una caja de las dimensiones solicitadas con el interior apagado.

- void **caja_aristas** (int x1, int y1, int z1, int x2, int y2, int z2)

Enciende las aristas de una caja de las dimensiones solicitadas.

- char **lineaX** (int inicio, int final)

Enciende una línea en el eje X.

- void **encender** (unsigned char matriz)

Enciende el cubo completamente de acuerdo con un patrón.

- void **desplazar** (char eje, int direccion)

Desplaza un plano a través de un eje.

1.2.2. Descripción detallada

Cabecera de las funciones de dibujo.

1.3. Referencia del Archivo Effect_Func.c

Funciones para la creación de efectos.

Este fichero utiliza las siguientes librerías:

- #include "stdlib.h"
- #include "Effect_Func.h"
- #include <Draw_func.h>
- #include "GLOBAL_DEFS.h"
- #include "DSP28x_Project.h"

1.3.1. Funciones

- void **efecto_lluvia** (int interacciones)

Generar un efecto de lluvia.

1.3.2. Descripción detallada

Funciones para la creación de efectos.

Estas funciones se encargan de llevar a cabo los diferentes efectos utilizando las funciones de dibujo para su programación

---

1.3.2.1. Documentación de las funciones

1.3.2.1.1. void efecto_lluvia (int interacciones)

Generar un efecto de lluvia.

Mediante números aleatorios, generar un efecto parecido al de la lluvia

Parámetros:

interacciones	Número de veces que se quiere repetir

Código:

```
202 {
203    int i, j;
204    int rnd_x;
205    int rnd_y;
206    int rnd_num;
207
208    for (j=0;j<interacciones;j++)
209    {
210       rnd_num = rand()%4;
211
212       for (i=0; i < rnd_num;i++)
213       {
214          rnd_x = rand()%8;
215          rnd_y = rand()%8;
216          led_ON(rnd_x,rnd_y,7);
217       }
218
219       DELAY_US(100000);
220       desplazar(EJE_Z,-1);
221    }
222 }
```

1.4. Referencia del Archivo Effect_Func.h

Cabecera de las funciones de los diferentes efectos.

1.4.1. Funciones

- void **efecto_lluvia** (int interacciones)
  Generar un efecto de lluvia.

1.4.2. Descripción detallada

Cabecera de las funciones de los diferentes efectos.

1.5. Referencia del Archivo GLOBAL_DEFS.h

Definición de variables utilizadas en los diferentes ficheros.

1.5.1.'defines'

- #define **PRG_FLASH** 1

Memoria flash DSP.

- #define **TRUE** 1

Estados booleanos.

- #define **FALSE** 0

Estados booleanos.

- #define **EJE_X** 1

Definición de los ejes del cubo.

- #define **EJE_Y** 2

Definición de los ejes del cubo.

- #define **EJE_Z** 3

Definición de los ejes del cubo.

- #define **X** 8

Dimensiones del cubo.

- #define **Y** 8

Dimensiones del cubo.

- #define **Z** 8

Dimensiones del cubo.

- #define **tiempo** 190

Temporización del Timer 0.

- #define **SYSCLK** 60000000

Frecuencias del sistema.

- #define **LSCLK SYSCLK**/(2*SysCtrlRegs.LOSPCP.bit.LSPCLK)

Frecuencias del sistema.

1.5.2. Descripción detallada

Definición de variables utilizadas en los diferentes ficheros.

Se muestran las definiciones de variables de modo que sea más fácil entender el código, además de facilitar su modificación en caso necesario.

1.5.2.1. Documentación de los 'defines'

1.5.2.1.1. #define EJE_X 1

Definición de los ejes del cubo.

1.5.2.1.2. #define EJE_Y 2

Definición de los ejes del cubo.

1.5.2.1.3. #define EJE_Z 3

Definición de los ejes del cubo.

1.5.2.1.4. #define FALSE 0

Estados booleanos.

1.5.2.1.5. #define LSCLK SYSCLK/(2*SysCtrlRegs.LOSPCP.bit.LSPCLK)

Frecuencias del sistema.

1.5.2.1.6. #define PRG_FLASH 1

Memoria flash DSP.

1.5.2.1.7. #define SYSCLK 60000000

Frecuencias del sistema.

1.5.2.1.8. #define tiempo 190

Temporización del Timer 0.

1.5.2.1.9. #define TRUE 1

Estados booleanos.

1.5.2.1.10. #define X 8

Dimensiones del cubo.

1.5.2.1.11. #define Y 8

Dimensiones del cubo.

1.5.2.1.12. #define Z 8

Dimensiones del cubo.

1.6. Referencia del Archivo GLOBAL_VARS.h

Inicialización de variables globales Variables globales que se utilizan en los diferentes ficheros del proyecto.

1.6.1. Variables

- volatile unsigned char **cubo** [8][8]

Buffer de datos.

- volatile int **capa_actual**

Almacenamiento de la capa actual.

1.6.2. Descripción detallada

Inicialización de variables globales Variables globales que se utilizan en los diferentes ficheros del proyecto.

1.6.2.1. Documentación de las variables

1.6.2.1.1. volatile int capa_actual

Almacenamiento de la capa actual.

1.6.2.1.2. volatile unsigned char cubo[8][8]

Buffer de datos.

1.7. Referencia del Archivo Init_Func.c

Funciones utilizadas para inicializar el sistema.

Este fichero incluye las siguientes librerías:

- #include <Init_Func.h>
- #include "GLOBAL_DEFS.h"
- #include "DSP2802x_Device.h"

1.7.1. Funciones

- void **InitDSP** (void)

Función que inicializa de los periféricos del Launchpad.

- void **Init_Flash** (void)

Inicialización de la memoria FLASH.

- void **WDogDisable** (void)

Desactivación del Watchdog.

- void **Init_GPIO** (void)

Función que inicializa los GPIO.

- void **Init_Timer0** (void)

Función que inicializa el TIMER 0.

- void **Init_Interrupt** (void)

Inicialización de todas las interrupciones necesarias.

- void **capas_OFF** (void)

Función encargada de apagar todas las capas del cubo.

- void **Activacion_OE** (char estado)

Función encargada de seleccionar el estado de la patilla OE de los 74HC574.

1.7.2. Descripción detallada

Funciones utilizadas para inicializar el sistema.

En este archivo se llevan a cabo las funciones para la inicialización de los puertos como salidas, configuración del Watchdog, inicialización del Timer 0 y funciones auxiliares para manejar las salidas del launchpad.

1.7.2.1. Documentación de las funciones

1.7.2.1.1. void Activacion_OE (char estado)

Función encargada de seleccionar el estado de la patilla OE de los 74HC574.

Código:

```
304 {
305   //EL pin OE utiliza lógica negada, por esta razón, se activará con un "0"
306   switch(estado)
307   {
308   case TRUE:
309     GpioDataRegs.GPBDAT.bit.GPIO34=0; //Activación del pin OE
310   break;
311
312   case FALSE:
313      GpioDataRegs.GPBDAT.bit.GPIO34=1;//Desactivación del pin OE
314   }
315
316 }
```

1.7.2.1.2. void capas_OFF (void)

Función encargada de apagar todas las capas del cubo. Se encarga de sacar un "0" por las salidas asignadas para las capas.

Código:

```
274 {
275   //  Capa 0
276   GpioDataRegs.AIOCLEAR.bit.AIO6 = 1;    //Inicializado a "0"
277
278   //  Capa 1
279   GpioDataRegs.GPACLEAR.bit.GPIO28 = 1;    //Inicializado a "0"
280
281   //  Capa 2
282    GpioDataRegs.GPACLEAR.bit.GPIO29 = 1;    //Inicializado a "0"
283
284   //  Capa 3
285    GpioDataRegs.AIOCLEAR.bit.AIO12 = 1;    //Inicializado a "0"
286
287   //  Capa 4
288    GpioDataRegs.AIOCLEAR.bit.AIO4 = 1;    //Inicializado a "0"
289
290   //  Capa 5
291    GpioDataRegs.AIOCLEAR.bit.AIO14 = 1;    //Inicializado a "0"
292
293   //  Capa 6
294    GpioDataRegs.AIOCLEAR.bit.AIO2 = 1;    //Inicializado a "0"
295
296   //  Capa 7
297    GpioDataRegs.AIOCLEAR.bit.AIO10 = 1;    //Inicializado a "0"
298 }
```

1.7.2.1.3. void Init_Flash (void)

Inicialización de la memoria FLASH. Función necesaria para que una vez se cargue el código, este permanezca en la memoria flash.

Código:

```
52 {
53   EALLOW;
54   //Enable Flash Pipeline mode to improve performance
55   //of code executed from Flash.
56   FlashRegs.FOPT.bit.ENPIPE = 1;
57
58   //            CAUTION
59   //Minimum waitstates required for the flash operating
60   //at a given CPU rate must be characterized by TI.
61   //Refer to the datasheet for the latest information.
62
63   //Set the Paged Waitstate for the Flash
64   FlashRegs.FBANKWAIT.bit.PAGEWAIT = 2;
65
66   //Set the Random Waitstate for the Flash
67   FlashRegs.FBANKWAIT.bit.RANDWAIT = 2;
68
69   //Set the Waitstate for the OTP
70   FlashRegs.FOTPWAIT.bit.OTPWAIT = 2;
71
72   //            CAUTION
73   //ONLY THE DEFAULT VALUE FOR THESE 2 REGISTERS SHOULD BE USED
74   FlashRegs.FSTDBYWAIT.bit.STDBYWAIT = 0x01FF;
75   FlashRegs.FACTIVEWAIT.bit.ACTIVEWAIT = 0x01FF;
76   EDIS;
77
78   //Force a pipeline flush to ensure that the write to
79   //the last register configured occurs before returning.
80
81   asm(" RPT #7 || NOP");
82 }
```

1.7.2.1.4. void Init_GPIO (void)

Función que inicializa los GPIO.

En este caso todos los GPIO se programarán como salidas, ya que no existen entradas. Se llevará a cabo la inicialización de los pines usados para:

- Capas
- OE
- Datos
- Direcciones

Código:

```
105 {
106     /*************************************************************************************
107     ****************** Inicialización de las Capas (0 - 7)**************************
*************************************************************************************/
109     EALLOW;
110     // Capa 0
111     GpioCtrlRegs.AIOMUX1.bit.AIO6 = 0;    //Pin configurado como IO Digital
112     GpioCtrlRegs.AIODIR.bit.AIO6 = 1;     //Salida digital
113
114     // Capa 1
115     GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 0;    //Pin configurado como IO Digital
116     GpioCtrlRegs.GPADIR.bit.GPIO28 = 1;     //Salida digital
117
118     // Capa 2
119     GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 0;    //Pin configurado como IO Digital
120     GpioCtrlRegs.GPADIR.bit.GPIO29 = 1;     //Salida digital
121
122     // Capa 3
123      GpioCtrlRegs.AIOMUX1.bit.AIO12 = 0;    //Pin configurado como IO Digital
124      GpioCtrlRegs.AIODIR.bit.AIO12 = 1;     //Salida digital
125
126      // Capa 4
127      GpioCtrlRegs.AIOMUX1.bit.AIO4 = 0;    //Pin configurado como IO Digital
128      GpioCtrlRegs.AIODIR.bit.AIO4 = 1;     //Salida digital
129
130      // Capa 5
131       GpioCtrlRegs.AIOMUX1.bit.AIO14 = 0;    //Pin configurado como IO Digital
132       GpioCtrlRegs.AIODIR.bit.AIO14 = 1;     //Salida digital
133
134       // Capa 6
135       GpioCtrlRegs.AIOMUX1.bit.AIO2 = 0;    //Pin configurado como IO Digital
136       GpioCtrlRegs.AIODIR.bit.AIO2 = 1;     //Salida digital
137
138       // Capa 7
139       GpioCtrlRegs.AIOMUX1.bit.AIO10 = 0;    //Pin configurado como IO Digital
140       GpioCtrlRegs.AIODIR.bit.AIO10 = 1;     //Salida digital
141
142
/*************************************************************************
143       ****************** Inicialización del Output Eneable (OE) ********************
*************************************************************************/
```

```
145         //  Controla la activación de los IC (Utiliza lógica negada)
146         GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;    //Pin configurado como IO Digital
147         GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;     //Salida digital
150
/*************************************************************************
151         ********** Inicialización del bus de datos (D0-D7) **********
*************************************************************************
153         //  D0
154         GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;    //Pin configurado como IO Digital
155         GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;     //Salida digital
156
157         //  D1
158         GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;    //Pin configurado como IO Digital
159         GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;     //Salida digital
160
161         //  D2
162         GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;    //Pin configurado como IO Digital
163         GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;     //Salida digital
164
165         //  D3
166          GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;    //Pin configurado como IO Digital
167          GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;     //Salida digital
168
169          //  D4
170          GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 0;    //Pin configurado como IO Digital
171          GpioCtrlRegs.GPADIR.bit.GPIO4 = 1;     //Salida digital
172
173          //  D5
174          GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;    //Pin configurado como IO Digital
175          GpioCtrlRegs.GPADIR.bit.GPIO5 = 1;     //Salida digital
176
177          //  D6
178          GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;    //Pin configurado como IO Digital
179          GpioCtrlRegs.GPADIR.bit.GPIO6 = 1;     //Salida digital
180
181          //  D7
182          GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 0;    //Pin configurado como IO Digital
183          GpioCtrlRegs.GPADIR.bit.GPIO7 = 1;     //Salida digital
184
185
/*************************************************************************
186         ********** Inicialización de las direcciones (A1-A3) **********
*************************************************************************
188          // A1
189          GpioCtrlRegs.GPAMUX2.bit.GPIO16 = 0;    //Pin configurado como IO Digital
190          GpioCtrlRegs.GPADIR.bit.GPIO16 = 1;     //Salida digital
191
192          // A2
193          GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 0;    //Pin configurado como IO Digital
194          GpioCtrlRegs.GPADIR.bit.GPIO17 = 1;     //Salida digital
195
196          // A3
197          GpioCtrlRegs.GPAMUX2.bit.GPIO18 = 0;    //Pin configurado como IO Digital
198          GpioCtrlRegs.GPADIR.bit.GPIO18 = 1;     //Salida digital
```

```
199
200         EDIS;
201
202 }
```

1.7.2.1.5. void Init_Interrupt (void)

Inicialización de todas las interrupciones necesarias. En este caso solo se utilizará la interrupción relacionada con el Timer 0.

Código:

```
239 {
240
241   // Enable Peripheral, global Ints and higher priority real-time debug events:
242   IER |= (M_INT1|M_INT2);
243   IER |= (M_INT3|M_INT4);
244
245   IER |= M_INT8;
246   IER |= M_INT9; // Esta deja Bloqueado al HW...
247
248   // Activar Interrupciones Timers-CPU
249   PieCtrlRegs.PIEIER1.bit.INTx7 = 1;  //  T0
251   // Enable the PIE
252   PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
253
254   // Enables PIE to drive a pulse into the CPU
255   PieCtrlRegs.PIEACK.all = 0xFFFF;
256
257   // Enable Interrupts at the CPU level
258   EINT;   // Enable Global interrupt INTM
259   ERTM;   // Enable Global realtime interrupt DBGM
261
262 }
```

1.7.2.1.6. void Init_Timer0 (void)

Función que inicializa el TIMER 0.

Código:

```
209 {
211   EALLOW;
212     SysCtrlRegs.PCLKCR3.bit.CPUTIMER0ENCLK = 1; // CPU Timer-0
213   EDIS;
214
215   // Asignamos los manejadores de interrupción a los timers
216   EALLOW;
217     PieVectTable.TINT0 = &TA0_Cpu_Timer0_isr;
218   EDIS;
219
220   // Se inicializan los periféricos de los temporizadores
221   InitCpuTimers();
222
223   // Se configuran los temporizadores
224   ConfigCpuTimer(&CpuTimer0, 60, tiempo);    //"tiempo" se encuentra definida en GLOBAL_DEFS.h
228   CpuTimer0Regs.TCR.all = 0x4001; //Arrancar Timer 0
231 }
```

1.7.2.1.7. void InitDSP (void)

Función que inicializa de los periféricos del Launchpad.

Se encarga de recopilar las funciones de inicialización:

- WD
- Timer0
- Interrupciones
- GPIOs

Código:

```
34 {
35   DINT;       // Deshabiltar interrupciones de la CPU
36   IER = 0x0000; // Deshabiltar interrupciones de la CPU y borrado de los flags
37   IFR = 0x0000;
38
39   WDogDisable();
40   Init_Timer0 ();
41   Init_Interrupt();
42   Init_GPIO();
43
44 }
```

1.7.2.1.8. void WDogDisable (void)

Desactivación del Watchdog.

Código:

```
88 {
89    EALLOW;
90    SysCtrlRegs.WDCR= 0x0068;
91    EDIS;
92 }
```

1.8. Referencia del Archivo Version_1.1.c

Programa prinicipal del codigo.

Este fichero utiliza las siguientes librerías:

- #include "DSP28x_Project.h"
- #include "GLOBAL_DEFS.h"
- #include "GLOBAL_VARS.h"
- #include <Draw_func.h>
- #include <Effect_Func.h>
- #include <Init_Func.h>

1.8.1. Funciones

- int **main** (void)
- interrupt void **TA0_Cpu_Timer0_isr** (void)
- Interrupcion del Timer 0.

1.8.2. Variables

- volatile unsigned char **cubo** [8][8]

Buffer de datos.

- volatile int **capa_actual** =0

Almacenamiento de la capa actual.

1.8.3. Descripción detallada

Programa principal del código.

En este programa se ejecuta la inicialización de los periféricos del launchapd:

- Entradas salidas
- Memoria Flash
- Timer 0
- Interrupciones

1.8.3.1. Documentación de las funciones

1.8.3.1.1. int main (void)

Código:

```
29          {
30
31
32  InitDSP();
33  //Progamación de la memoria flash
34  #if (PRG_FLASH==1)
35     MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
36     Init_Flash();    // Inicialización de la memoria Flash
37  #endif
38
39  //Bucle infinito
40  for(;;)
41  {
42
43     INTRO();
44
45
46  }
47 }
```

1.8.3.1.2. interrupt void TA0_Cpu_Timer0_isr (void)

Interrupción del Timer 0.

En esta función se lleva a cabo el control completo del encendido del cubo. Se ejecuta a una alta velocidad provocando un efecto de encendido permanente, aunque en realidad se encuentra en continuo parpadeo.

Código:

```
69 {
70
71  int i;//Variable contador
72
73  capas_OFF();//Apagado de todas las capas
74
75  Activacion_OE(FALSE);//Apagado de la patilla OE de los 74HC574
76
77  //Actualización de la información en el BUS de datos (D0-D7)
78  for(i=0;i<8;i++)
79  {
80     GpioDataRegs.GPADAT.all=cubo[capa_actual][i]; //Pasa la información del buffer al bus de datos
81     GpioDataRegs.GPADAT.all=(GpioDataRegs.GPADAT.all & 0x000000FF)|((0x07 &
((Uint32)i+1))<<16);//Decodifica la direccion del correspondiente 74HC574
82
83  }
84
85  Activacion_OE(TRUE); //Habilita los integrados 74HC574
```

```
86
87   //Activación de la capa correspondiente
88   if(capa_actual==0)
89     {
90       GpioDataRegs.AIOSET.bit.AIO6 = 1;    //Capa 0 encendida
91     }
92   else if(capa_actual==1)
93     {
94       GpioDataRegs.GPASET.bit.GPIO28 = 1;  //Capa 1 encendida
95     }
96   else if(capa_actual==2)
97     {
98       GpioDataRegs.GPASET.bit.GPIO29 = 1;  //Capa 2 encendida
99     }
100  else if(capa_actual==3)
101    {
102      GpioDataRegs.AIOSET.bit.AIO12 = 1;  //Capa 3 encendida
103    }
104  else if(capa_actual==4)
105    {
106      GpioDataRegs.AIOSET.bit.AIO4 = 1;   //Capa 4 encendida
107    }
108  else if(capa_actual==5)
109    {
110      GpioDataRegs.AIOSET.bit.AIO14 = 1;  //Capa 5 encendida
111    }
112  else if(capa_actual==6)
113    {
114      GpioDataRegs.AIOSET.bit.AIO2 = 1;  //Capa 6 encendida
115    }
116  else if(capa_actual==7)
117    {
118      GpioDataRegs.AIOSET.bit.AIO10 = 1;  //Capa 7 encendida
119    }
121  capa_actual++;    //Incrementar contador de capa
122
123  //Reinicio de contador de capa
124  if(capa_actual==8)
125  {
126    capa_actual=0;
127  }
130  //Reconocimiento de la interrupción, permitiendo recibir más interrupciones del grupo 1
131  PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
132
133 }
```

1.8.3.2. Documentación de las variables

1.8.3.2.1. volatile int capa_actual =0

Almacenamiento de la capa actual.

1.8.3.2.2. volatile unsigned char cubo[8][8]

Buffer de datos.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Grado en Ingeniería Electrónica Industrial y Automática

5 – Esquemas eléctricos

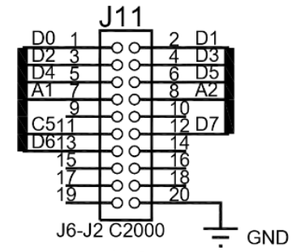
Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

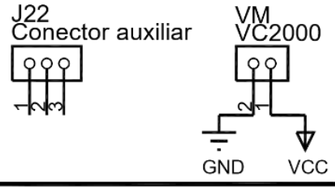
ÍNDICE

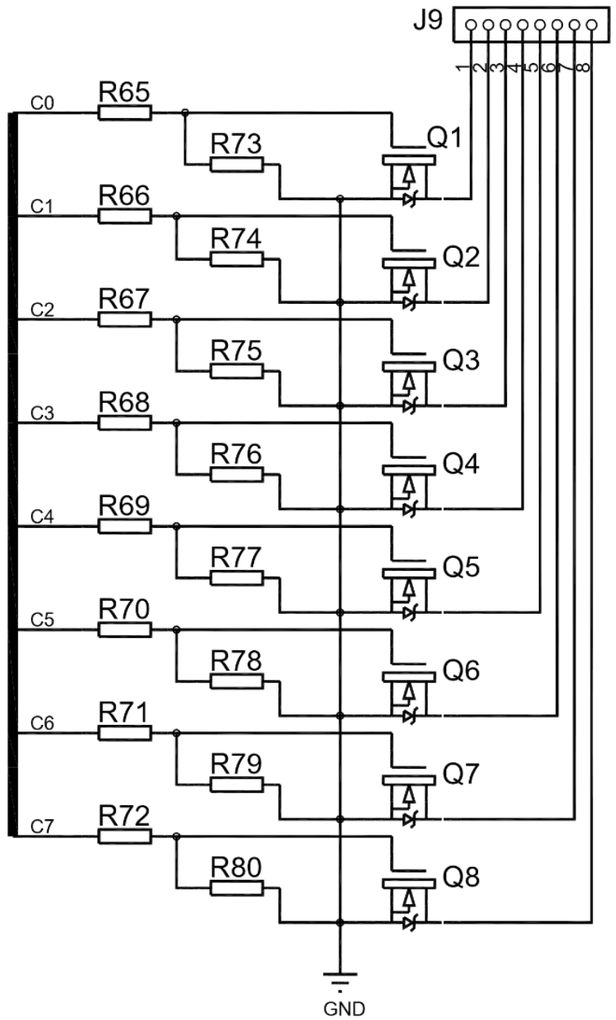
Conectores C2000

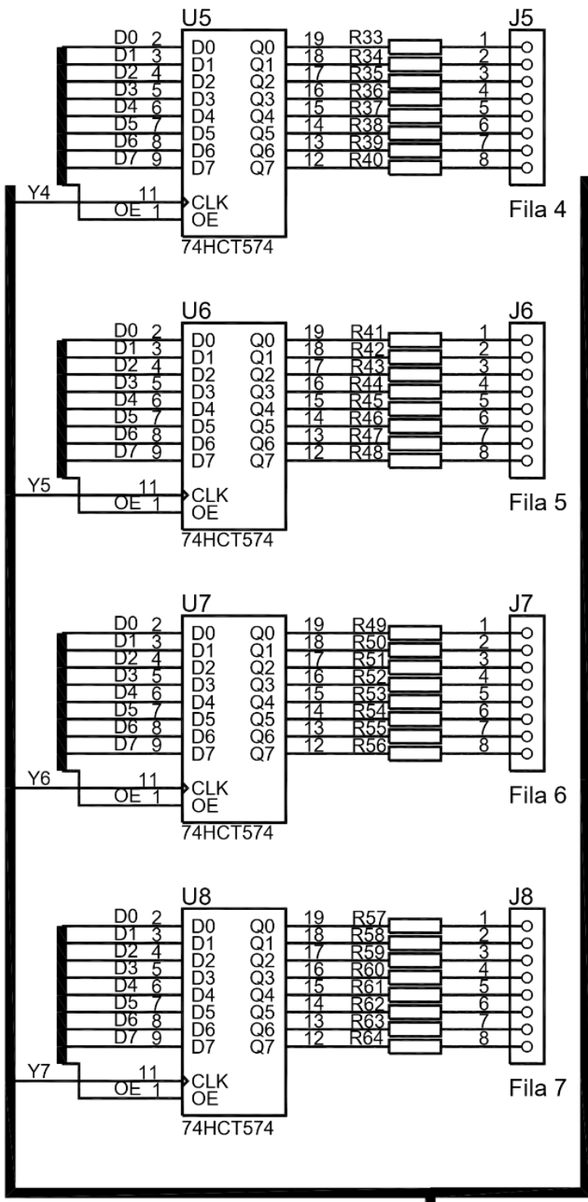
Nota 1

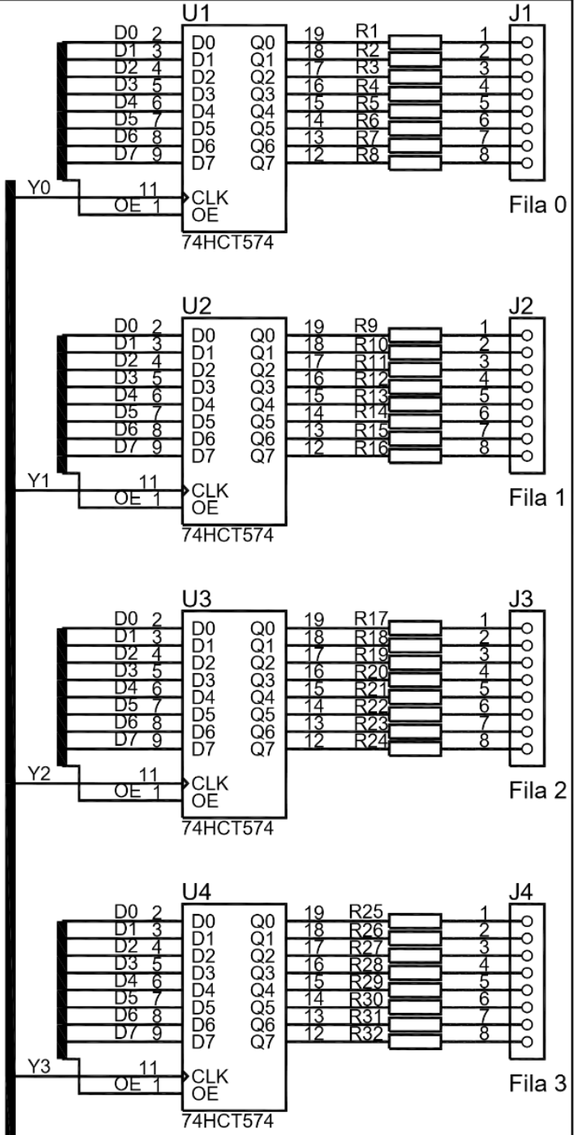


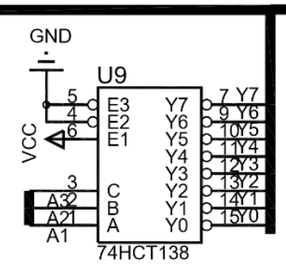
J22 Conector auxiliar



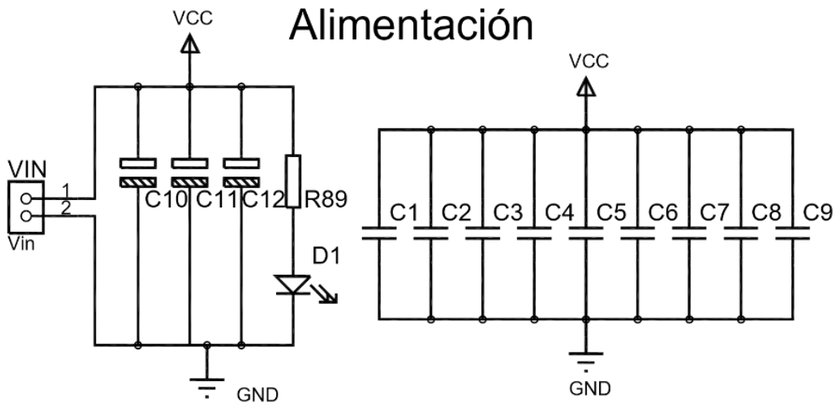








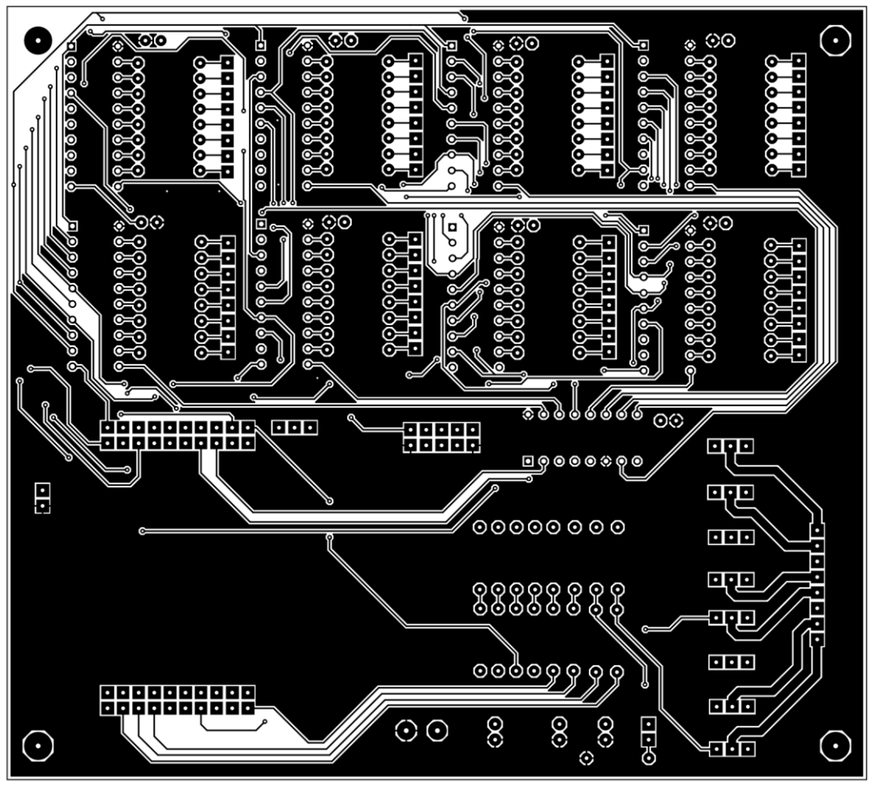
Alimentación



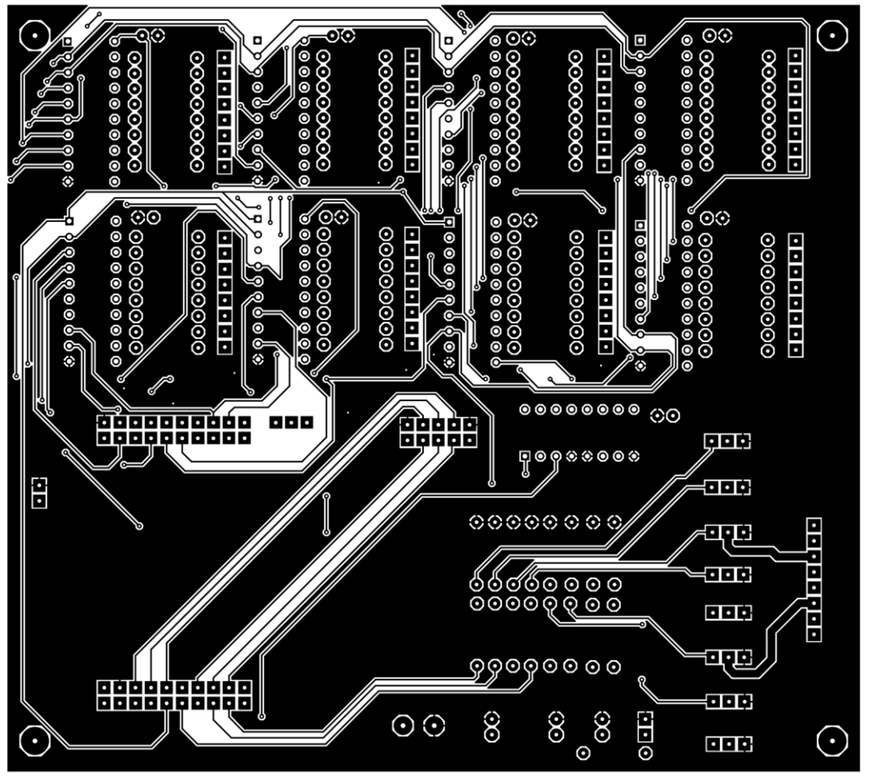
Nota 1: La conexión de los conectores C2000 coinciden con el pinout del launchpad F28027

	Fecha	Nombre		
Dibujado	18/01/2018	Miguel Valera	UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Trabajo Final de Grado Grado en Ingeniería Electrónica Industrial y Automática
Escala		**Esquema eléctrico**	Escuela Técnica Superior de Ingeniería del Diseño	
			REV 1.0	Nº 1 de 3

TOP

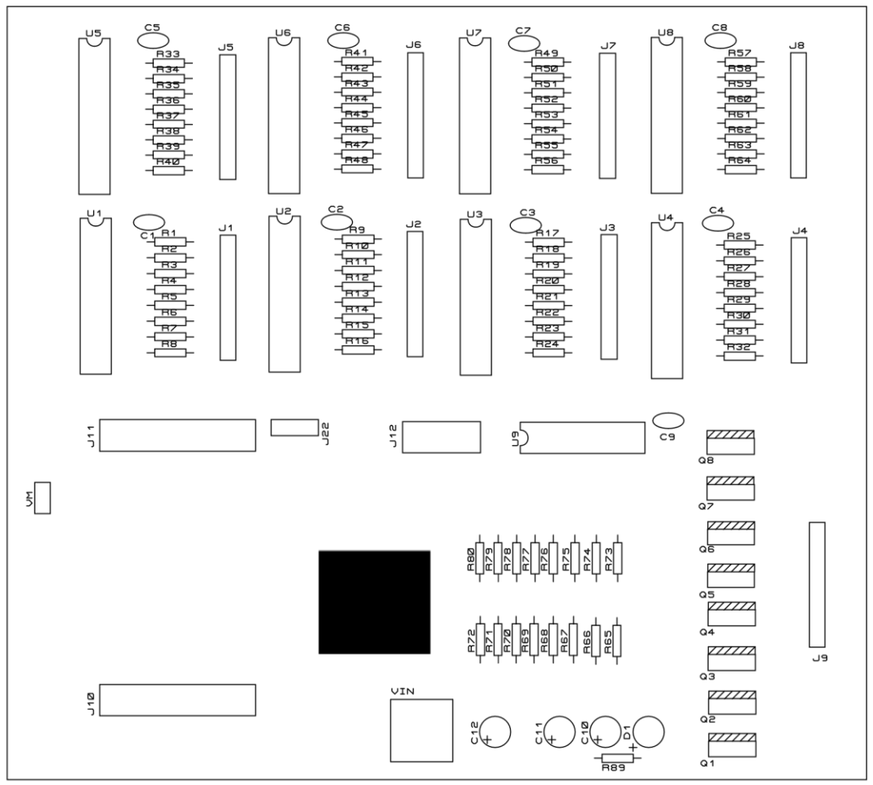


BOTTOM

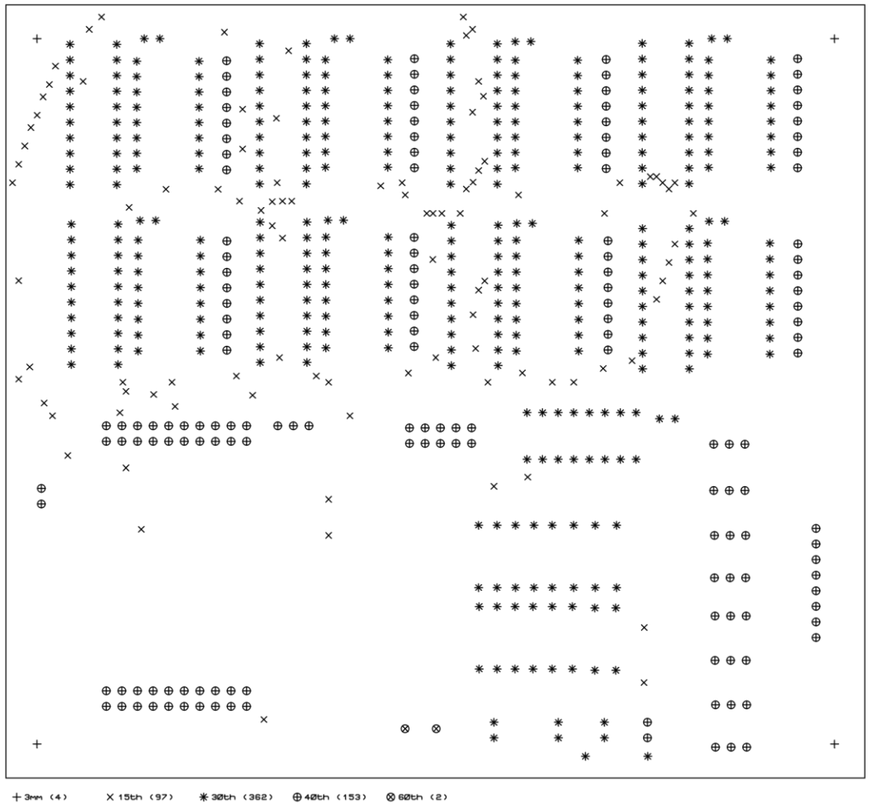


	Fecha	Nombre	UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Trabajo Final de Grado Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	18/01/2018	Miguel Valera		
Escala 1:1	Circuito impreso		Escuela Técnica Superior de Ingeniería del Diseño	
			REV 1.0	Nº 2 de 3

HUELLAS



TALADROS



	Fecha	Nombre	UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Trabajo Final de Grado Grado en Ingeniería Electrónica Industrial y Automática	
Dibujado	18/01/2018	Miguel Valera			
Escala 1:1	Circuito impreso (2)			Escuela Técnica Superior de Ingeniería del Diseño	
				REV 1.0	Nº 3 de 3

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FINAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

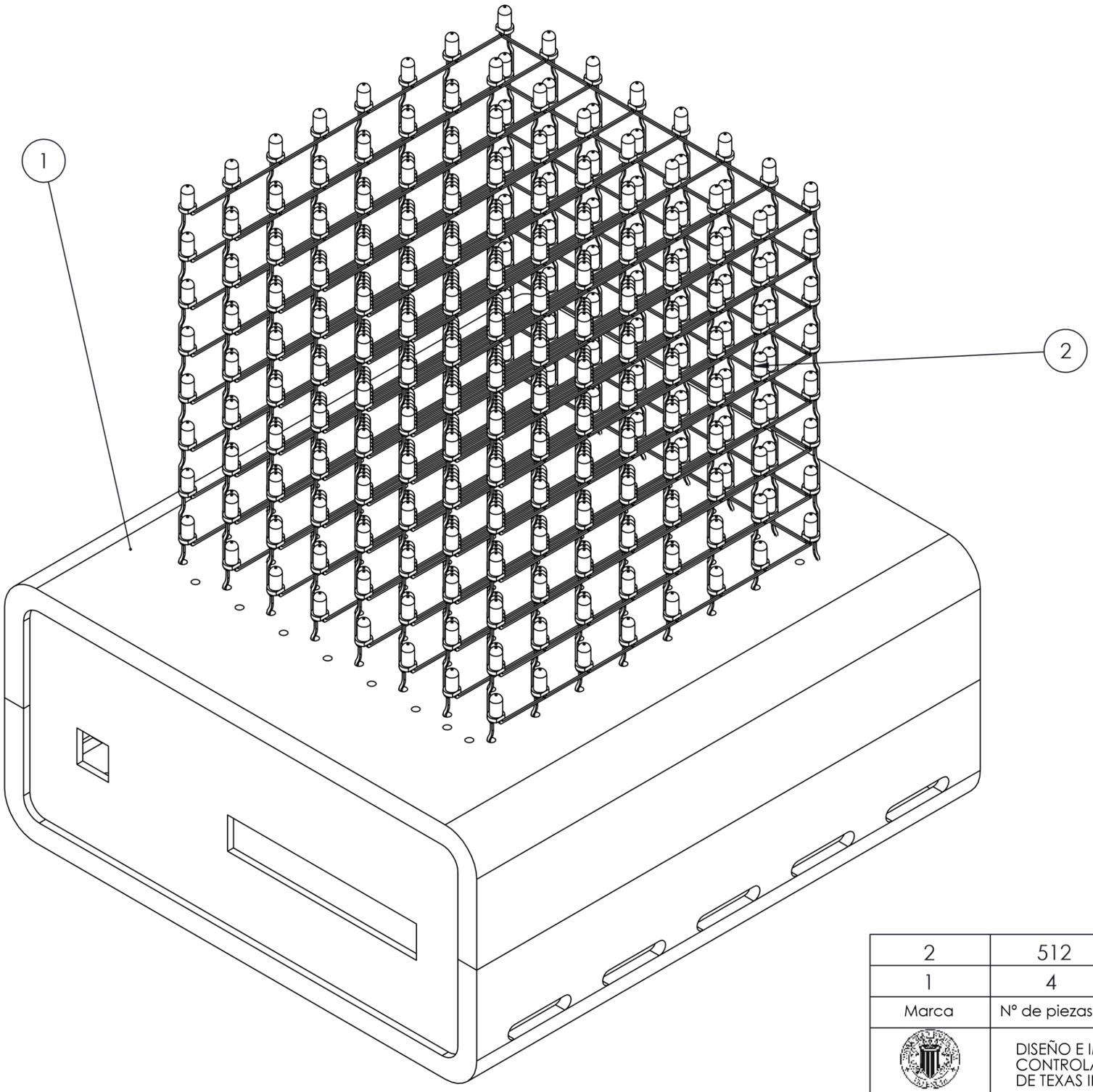
Grado en Ingeniería Electrónica Industrial y Automática

6 - Planimetría

Autor
Miguel Valera Mira
Tutor
Salvador Orts Grau

Valencia - Abril 2018

ÍNDICE



2	512	1.2. Conjunto leds
1	4	1.1. Conjunto carcasa
Marca	N° de piezas	Nomenclatura

Ingeniería electrónica industrial y automática

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Escala: 1:5

N° de plano: 1

Codigo y nombre de pieza:

1. Conjunto Cubo

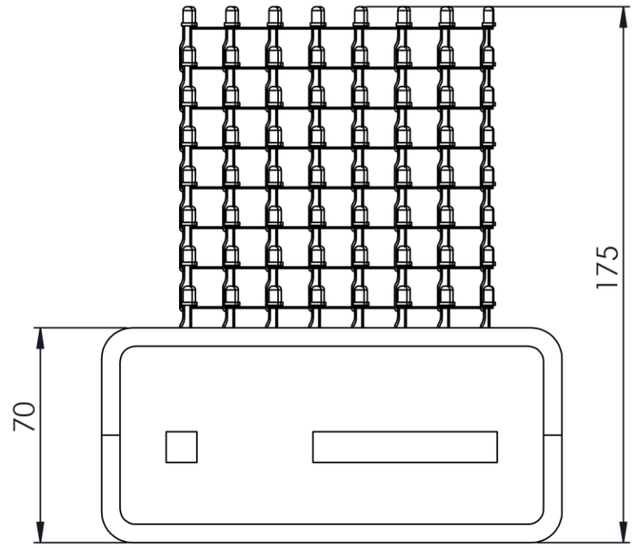
Planimetría

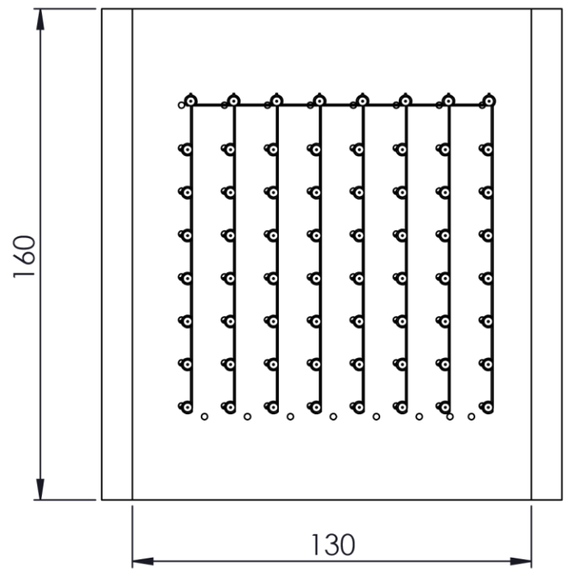
Trabajo Final de Grado

Miguel Valera Mira

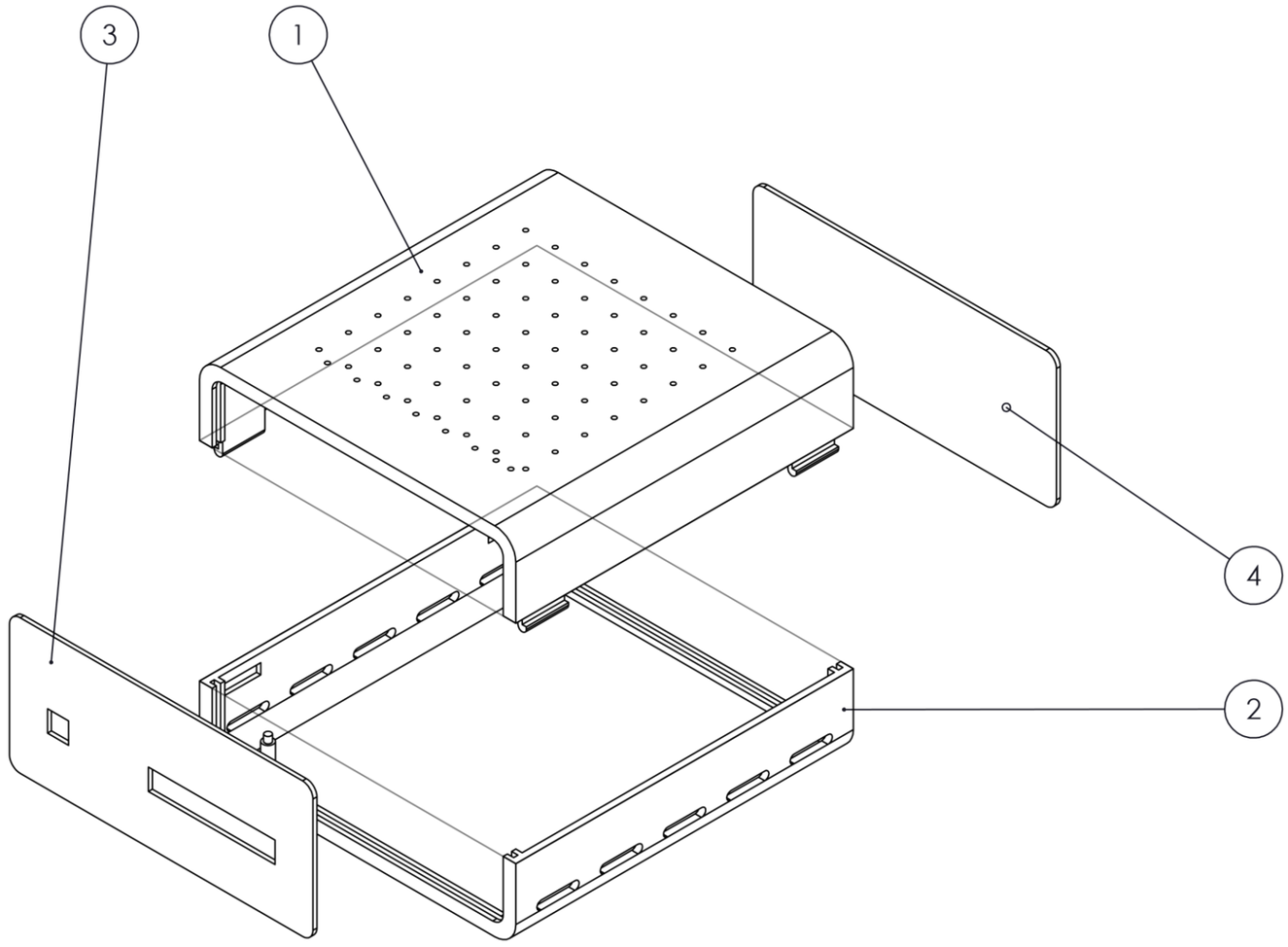
Abril 2018

Formato A3





Ingeniería electrónica industrial y automática	DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS	Escala: 1:5
	Codigo y nombre de pieza: **1.Cotas generales**	Nº de plano: 2
	Planimetría	
	Trabajo Final de Grado	
	Miguel Valera Mira	
	Abril 2018	Formato A3



4	1	1.1.4. Tapa trasera
3	1	1.1.3. Tapa frontal
2	1	1.1.2. Carcasa inferior
1	1	1.1.1. Carcasa superior
Marca	N° de piezas	Nomenclatura

Ingeniería electrónica industrial y automática

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Escala: 1:2

N° de plano: 3

Codigo y nombre de pieza:

1.1. Conjunto carcasa

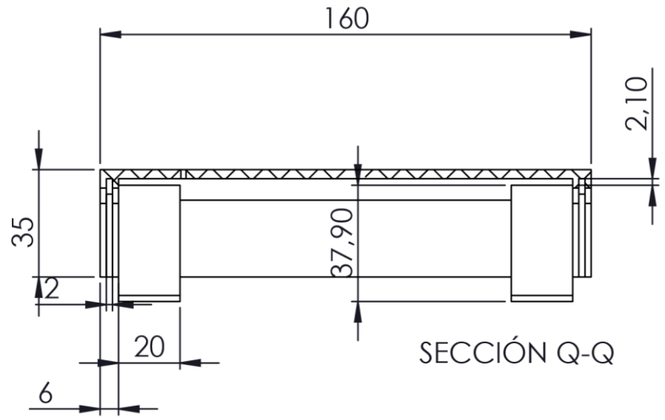
Planimetría

Trabajo Final de Grado

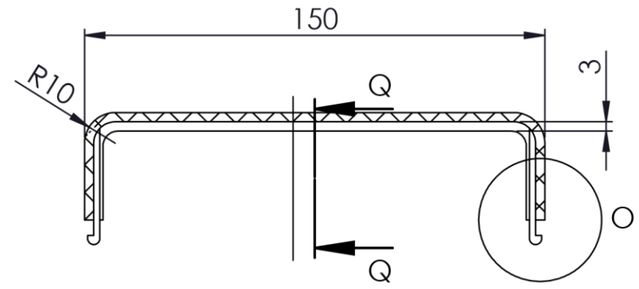
Miguel Valera Mira

Abril 2018

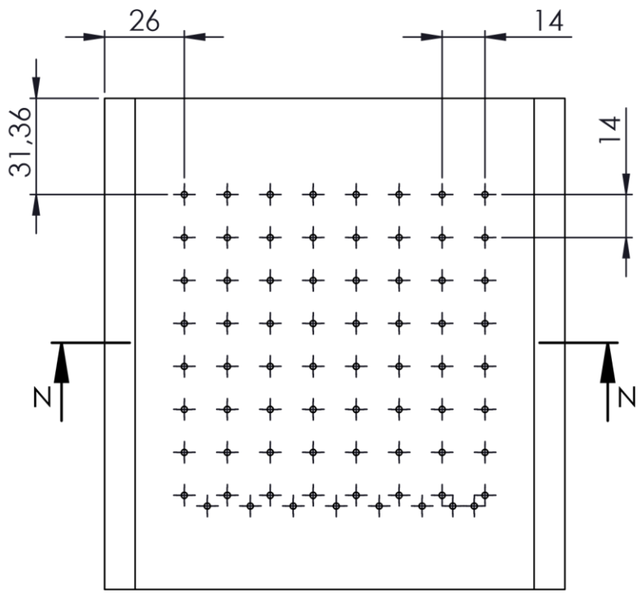
Formato A3

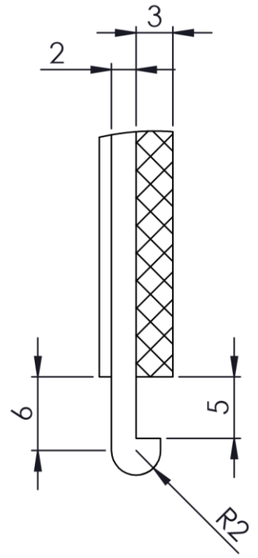


SECCIÓN Q-Q



SECCIÓN N-N





DETALLE O
ESCALA 2 : 1

1	1	Plástico ABS
Marca	Nº de piezas	Material

Ingeniería electrónica industrial y automática

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Escala: 1:2

Nº de plano: 4

Codigo y nombre de pieza:

1.1.1.Carcasa superior

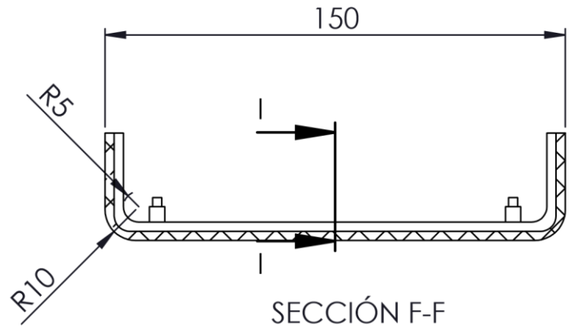
Planimetría

Trabajo Final de Grado

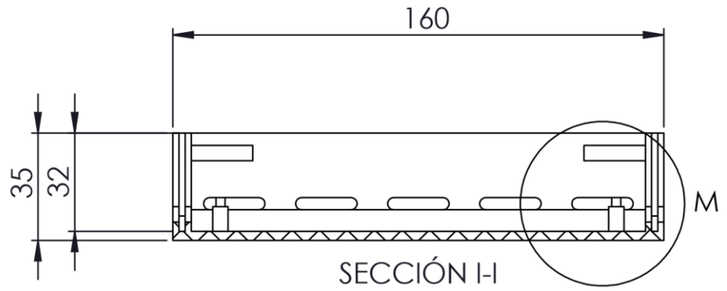
Miguel Valera Mira

Abril 2018

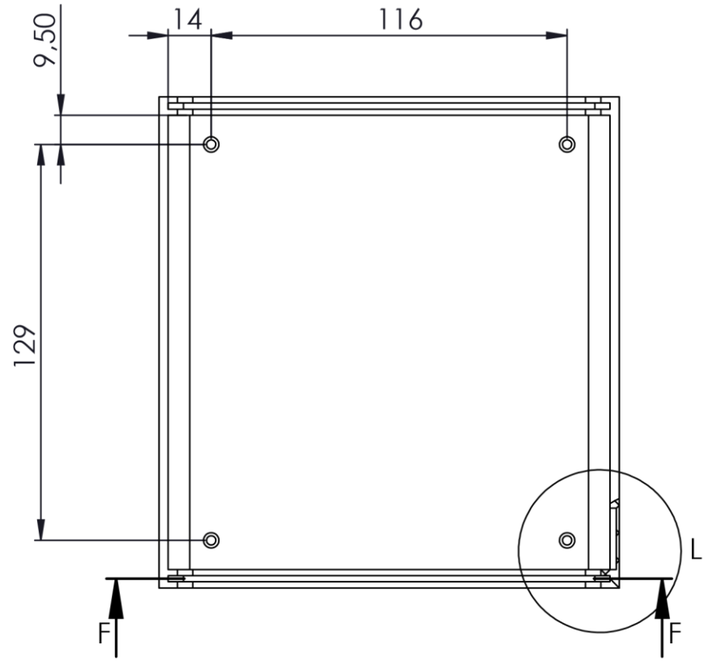
Formato A3

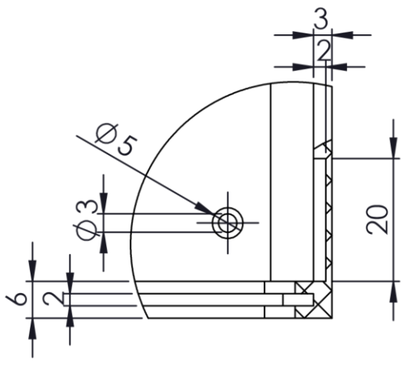


SECCIÓN F-F

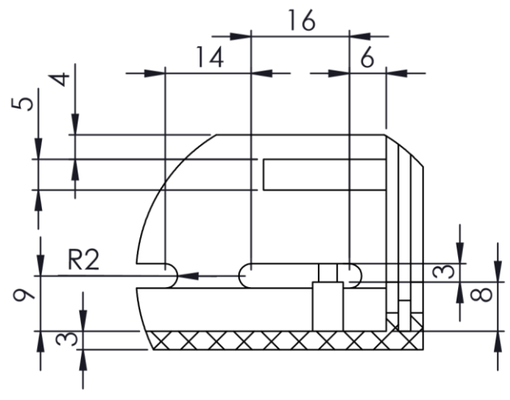


SECCIÓN I-I





DETALLE L
ESCALA 1 : 1



DETALLE M
ESCALA 1 : 1

2	1	Plástico ABS
Marca	Nº de piezas	Material

Ingeniería electrónica industrial y automática

DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS

Escala: 1:1

Nº de plano: 5

Codigo y nombre de pieza:

1.1.2. Carcasa inferior

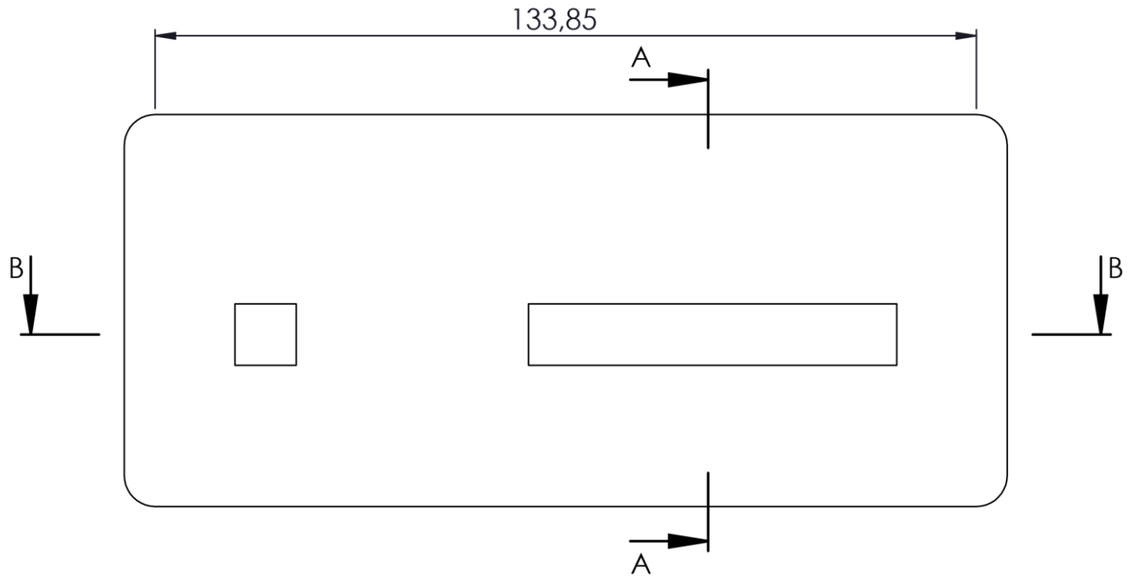
Planimetría

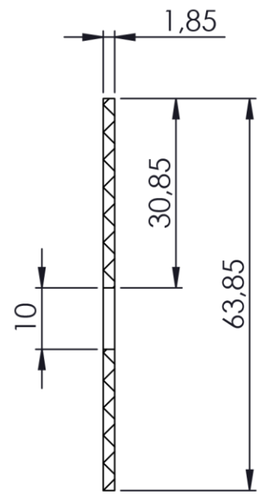
Trabajo Final de Grado

Miguel Valera Mira

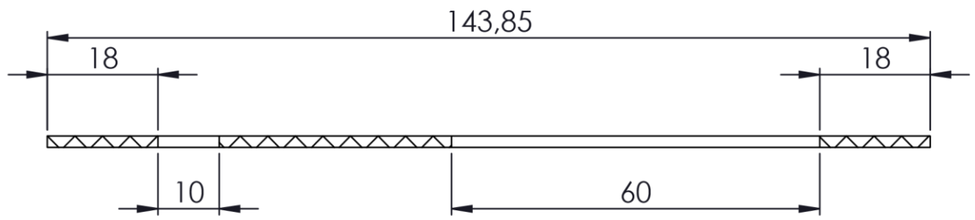
Abril 2018

Formato A3



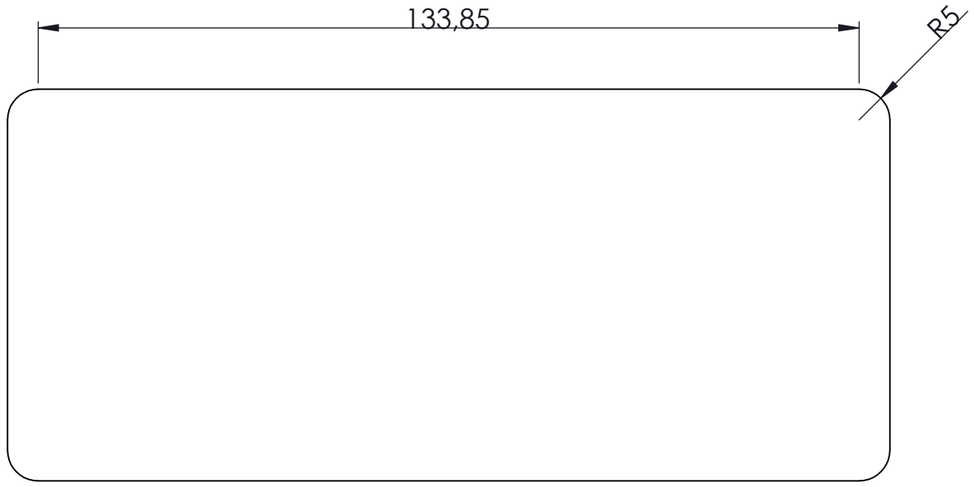


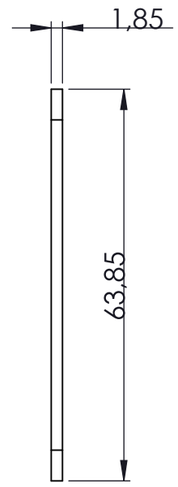
SECCIÓN A-A
ESCALA 1 : 1



SECCIÓN B-B
ESCALA 1 : 1

3	1	Plástico ABS	
Marca	N° de piezas	Material	
Ingeniería electrónica industrial y automática	DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS		Escala: 1:1
	Codigo y nombre de pieza: **1.1.3. Tapa frontal**	Planimetría	N° de plano: 6
		Trabajo Final de Grado	
		Miguel Valera Mira	
		Abril 2018	Formato A3





4	1	Plástico ABS	
Marca	Nº de piezas	Material	
Ingeniería electrónica industrial y automática	DISEÑO E IMPLEMENTACIÓN DE UNA MATRIZ DE LEDS TRIDIMENSIONAL CONTROLADA MEDIANTE EL DSC TMS320F28027 DE LA FAMILIA C2000 DE TEXAS INSTRUMENTS		Escala: 1:1
	Codigo y nombre de pieza: **1.1.4. Tapa trasera**	Planimetría	Nº de plano: 7
		Trabajo Final de Grado	
		Miguel Valera Mira	
		Abril 2018	Formato A3